

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

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

Abstract

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

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

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 v1.12.1, last revised 2015/02/27.

[†]herries dot press at earthlink dot net

[‡]maieul at maieul dot net

5 Left and right texts	7
6 Numbering text lines and paragraphs	9
7 Verse	10
8 Side notes	12
9 Parallel ledgroups	12
9.1 Parallel ledgroups and <code>setspace</code> package	14
10 Sectioning commands	14
11 Implementation overview	14
12 Preliminaries	14
12.1 Messages	15
13 Sectioning commands	16
14 Line counting	19
14.1 Choosing the system of lineation	19
14.2 Line-number counters and lists	22
14.3 Reading the line-list file	23
14.4 Commands within the line-list file	24
14.5 Writing to the line-list file	32
15 Marking text for notes	34
16 Pstart numbers dumping and restoration	35
17 Parallel environments	36
18 Paragraph decomposition and reassembly	38
18.1 Boxes, counters, <code>\pstart</code> and <code>\pend</code>	38
18.2 Processing one line	42
18.3 Line and page number computation	45
18.4 Line number printing	48
18.5 Pstart number printing in side	50
18.6 Add insertions to the vertical list	52
18.7 Penalties	52
18.8 Printing leftover notes	53
19 Footnotes	54
19.1 Normal footnote formatting	54
20 Cross referencing	54

	3
21 Side notes	55
22 Familiar footnotes	56
23 Verse	57
24 Naming macros	59
25 Counts and boxes for parallel texts	60
26 Fixing babel	61
27 Parallel columns	64
28 Parallel pages	70
29 Sections' titles' commands	81
30 Page break/no page break, depending on the specific line	81
31 Parallel ledgroup	82
32 The End	85
Appendix A Some things to do when changing version	86
Appendix A.1 Migration to eldpar 1.4.3	86
References	86
Index	86
Change History	97

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 L^AT_EX. 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 T_EX 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 32); 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 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}
...
\end{pairs}
\Columns
```

Keep in mind that the `\Columns` **must be** outside of the **pairs** environment.

\AtBeginPairs You can use the macro `\AtBeginPairs` to insert a code at the beginning of each **pairs** environments. That could be useful to add the `\sloppy` macro to prevent overfull hboxes in two columns.

```
\AtBeginPairs{\sloppy}
```

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.

`\columnspanposition`

By default, columns are positioned to the right of the page. However, you use `\columnspanposition{L}` to align them to the left, or `\columnspanposition{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{\beforecolumnseparator}{length}
```

```
\setlength{\aftercolumnseparator}{length}
```

`\widthliketwocolumns`

If you want to come back to the previous behavior, just set them with a negative value. If you want to mix texts in columns and text without columns, you can horizontally align text in one column to text in two columns with `\widthliketwocolumnstrue`. To reset this feature, just use `\widthliketwocolumnsfalse`.

`\Xnoteswidthliketwocolumns`

`\notesXwidthliketwocolumns`

In most case, you should use `\widthliketwocolumns` in combination with `\Xnoteswidthliketwocolumns` and `\notesXwidthliketwocolumns` to align the critical/familiar footnotes with the two columns. `eledmac` handbook for more details.

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}
\begin{Leftside} ... \end{Leftside}
...
\end{pages}
\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. Note that the **\Pages** **must be** outside of the **pages** environment.

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

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

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

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

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

5 Left and right texts

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

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

Within these environments you can designate the line numbering scheme(s) to be used. The **eledmac** package originally used counters for specifying the numbering scheme; now both **eledmac**¹ and the **eledpar** package use macros instead. 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

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

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

numbering scheme for sublines. The starred versions change both left and right numbering schemes.

`\pstart` In a serial (non-parallel) mode, each numbered paragraph, or chunk, is contained between the `\pstart` and `\pend` macros, and the paragraph is output when the `\pend` macro occurs. The situation is somewhat different with parallel typesetting as the left text (contained within `\pstart` and `\pend` groups within the `Leftside` environment) has to be set in parallel with the right text (contained within its own `\pstart` and `\pend` groups within the corresponding `Rightside` environment) the `\pend` macros cannot immediately initiate any typesetting — this has to be controlled by the `\Columns` or `\Pages` macros. Several chunks may be specified within a `Leftside` or `Rightside` environment. A multi-chunk text then looks like:

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

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

`\lineationR` 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*` `\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.

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

However, sometime if the left `pstarts` are much greater than right `pstarts`, or *vice-versa*, you can decide to shift the `pstarts` on the left and right side. That

means the start of pstarts are not aligned horizontally on the page, the shift is offset at the end of each double pages. To enable this function, load eledpar with the option `shiftedpstarts`.

6 Numbering text lines and paragraphs

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

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

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

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

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

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

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

`\newcommand*{\Rlineflag}{R}`
 This may be useful for parallel columns but for parallel pages it might be more appropriate to redefine it as:

`\printlinesR` `\renewcommand*{\Rlineflag}{}`. The `\printlines` macro is ordinarily used to print the line number references for critical footnotes. For footnotes from right side texts a special version is supplied, called `\printlinesR`, which incorporates `\Rlineflag`. (The macro `\ledsavedprintlines` is a copy of the original `\printlines`, just in case ...). As provided, the package makes no use of `\printlinesR` but you may find it useful. For example, if you only use the B footnote series in righthand texts then you may wish to flag any line numbers in those footnotes with the value of `\Rlineflag`. You could do this by putting the following code in your preamble:

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

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

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

7 Verse

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

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

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

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

The `astanza` environment forms a chunk but you may want to have more than one stanza within the chunk. Here are a couple of ways of doing that with a blank

line between each internal stanza, and with each stanza numbered. First some preliminary definitions:

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

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

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

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

  \end{astanza}
  ...
\end{pairs}
```

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

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

  \end{astanza}
\end{pairs}
```

```

                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 hanging line. If you use it, you must run `LATEX` two time. Example for the French typography

```
\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 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).

By default, the L^AT_EX 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 LaTeX2_ε. The package also requires the **eledmac** package.

```

1 (*code)
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{eledpar}[2015/02/27 v1.12.1 eledmac extension for parallel texts]%
4

```

With the option ‘shiftedpstarts’ a long pstart on the left side (or in the right side) doesn’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 shift stops at every end of pages. The `\shiftedverses` is kept for backward compatibility.

`\ifshiftedpstarts`

```

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

```

`\ifwidthliketwocolumns` The `\widthliketwocolumns` option can be called both in `eledpar` and `eledmac`.

```

11 \DeclareOption{widthliketwocolumns}{\widthliketwocolumnstrue}%

```

```

12 \ProcessOptions%

```

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

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

```

13 \l@dpairingfalse
14 \l@dpagingfalse
15 \ledRcolfalse

```

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

```

\Lcolwidth
\Rcolwidth
16 \newdimen\Lcolwidth
17 \Lcolwidth=0.45\textwidth
18 \newdimen\Rcolwidth
19 \Rcolwidth=0.45\textwidth
20

```

12.1 Messages

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

`\eledpar@error`

```
21 \newcommand{\eledpar@error}[2]{\PackageError{eledpar}{#1}{#2}}
22 % \end{macrocode}
23 % \end{macro}
24 % \begin{macro}{\led@err@TooManyPstarts}
25 % \begin{macrocode}
26 \newcommand*{\led@err@TooManyPstarts}{%
27 \eledpar@error{Too many \string\pstart\space without printing.
28 \qquad Some text will be lost}{\@ehc}}
```

`\led@err@BadLeftRightPstarts`

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

`\led@err@LeftOnRightPage`

`\led@err@RightOnLeftPage`

```
32 \newcommand*{\led@err@LeftOnRightPage}{%
33 \eledpar@error{The left page has ended on a right page}{\@ehc}}
34 \newcommand*{\led@err@RightOnLeftPage}{%
35 \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.

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

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

```
38 \pst@rtedLfalse
39 \newif\ifpst@rtedR
40 \pst@rtedRfalse
41
```

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

```
42 \newcommand*{\beginnumberingR}{%
43 \ifnumberingR
44 \led@err@NumberingStarted
45 \endnumberingR
46 \fi
```



```

47 \global\l@dnumstartsR \z@
48 \global\pst@rtedRfalse
49 \global\numberingRtrue
50 \global\advance\section@numR \@ne
51 \global\absline@numR \z@
52 \gdef\normal@page@breakR{ }
53 \gdef\l@prev@pbR{ }
54 \gdef\l@prev@nopbR{ }
55 \global\line@numR \z@
56 \global\@lockR \z@
57 \global\sub@lockR \z@
58 \global\sublines@false
59 \global\let\next@page@numR\relax
60 \global\let\sub@change\relax
61 \message{Section \the\section@numR R }%
62 \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
63 \l@dend@stuff
64 \setcounter{pstartR}{1}
65 \begingroup
66 \initnumbering@sectcountR
67 \gdef\eled@sectionsR@{ }%
68 \if@noeled@sec\else%
69 \makeatletter\inputIfFileExists{\jobname.eledsec\the\section@numR R}{ }{\makeatother}%
70 \immediate\openout\eled@sectioningR@out=\jobname.eledsec\the\section@numR R\relax%
71 \fi%
72 }

```

\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.

```

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

```

```

90 \else
91   \led@err@NumberingNotStarted
92 \fi
93 \endgroup
94 \if@noeled@sec\else%
95   \immediate\closeout\eled@sectioningR@out%
96 \fi%
97 }
98

```

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

```

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

```

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

```

109 \newcommand*{\pausenumberingR}{%
110   \endnumberingR\global\numberingRtrue}
111 \newcommand*{\resumenumberingR}{%
112   \ifnumberingR
113     \global\pst@rtedRtrue
114     \global\advance\section@numR \@ne
115     \led@mess@SectionContinued{\the\section@numR R}%
116     \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
117     \l@dend@stuff
118     \begingroup%
119     \initnumbering@sectcountR%
120   \else
121     \led@err@numberingShouldHaveStarted
122     \endnumberingR
123     \beginnumberingR
124   \fi}
125

```

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

```

126 \newcommand*{\memorydumpL}{%
127   \endnumbering
128   \numberingtrue

```

```

129 \global\pst@rtedLtrue
130 \global\advance\section@num \@ne
131 \led@mess@SectionContinued{\the\section@num}%
132 \line@list@stuff{\jobname.\extensionchars\the\section@num}%
133 \l@dend@stuff}
134 \newcommand*{\memorydumpR}{%
135 \endnumberingR
136 \numberingRtrue
137 \global\pst@rtedRtrue
138 \global\advance\section@numR \@ne
139 \led@mess@SectionContinued{\the\section@numR R}%
140 \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
141 \l@dend@stuff}
142

```

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

`\bypstart@Rtrue`

`\bypstart@Rfalse`

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

`\ifbypage@R`

`\bypage@Rtrue`

`\bypage@Rfalse` `eledpar` will use the line-of-section system unless instructed otherwise.

```

143 \newif\ifbypage@R
144 \newif\ifbypstart@R
145 \bypage@Rfalse
146 \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`.

```

147 \newcommand*{\lineationR}[1]{%
148 \ifnumbering
149 \led@err@LineationInNumbered
150 \else
151 \def\@tempa{#1}\def\@tempb{page}%
152 \ifx\@tempa\@tempb
153 \global\bypage@Rtrue
154 \global\bypstart@Rfalse
155 \else
156 \def\@tempb{pstart}%
157 \ifx\@tempa\@tempb

```

```

158         \global\bypage@Rfalse
159         \global\bystart@Rtrue
160     \else
161         \def@tempb{section}
162         \ifx\@tempa\@tempb
163             \global\bypage@Rfalse
164             \global\bystart@Rfalse
165         \else
166             \led@warn@BadLineation
167         \fi
168     \fi
169 \fi
170 \fi}}

```

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

```

171 \WithSuffix\newcommand\lineation*[1]{%
172   \lineation{#1}%
173   \lineationR{#1}%
174 }%

```

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

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

```

175 \newcount\line@marginR
176 \renewcommand*{\linenummargin}[1]{%
177   \l@getline@margin{#1}%
178   \ifnum\@l@dtmpcntb>\m@ne
179     \ifledRcol
180       \global\line@marginR=\@l@dtmpcntb
181     \else
182       \global\line@margin=\@l@dtmpcntb
183     \fi
184   \fi}}

```

By default put right text numbers at the right.

```

185 \line@marginR=\@ne
186

```

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

```

187 \newcounter{firstlinenumR}

```

```

188 \setcounter{firstlinenumR}{5}
189 \newcounter{linenumincrementR}
190 \setcounter{linenumincrementR}{5}

```

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

```

191 \newcounter{firstsublinenumR}
192 \setcounter{firstsublinenumR}{5}
193 \newcounter{sublinenumincrementR}
194 \setcounter{sublinenumincrementR}{5}
195

```

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

```

\sublinenumincrement 196 \providecommand*\firstlinenum{}
\firstlinenum*       197 \providecommand*\linenumincrement{}
\linenumincrement*   198 \providecommand*\firstsublinenum{}
\firstsublinenum*    199 \providecommand*\sublinenumincrement{}
\sublinenumincrement* 200 \renewcommand*\firstlinenum[1]{%
201   \ifledRcol \setcounter{firstlinenumR}{#1}%
202   \else      \setcounter{firstlinenumR}{#1}%
203   \fi}
204 \renewcommand*\linenumincrement[1]{%
205   \ifledRcol \setcounter{linenumincrementR}{#1}%
206   \else      \setcounter{linenumincrementR}{#1}%
207   \fi}
208 \renewcommand*\firstsublinenum[1]{%
209   \ifledRcol \setcounter{firstsublinenumR}{#1}%
210   \else      \setcounter{firstsublinenumR}{#1}%
211   \fi}
212 \renewcommand*\sublinenumincrement[1]{%
213   \ifledRcol \setcounter{sublinenumincrementR}{#1}%
214   \else      \setcounter{sublinenumincrementR}{#1}%
215   \fi}
216 \WithSuffix\newcommand\firstlinenum*[1]{\setcounter{firstlinenumR}{#1}\setcounter{firstlinenum}{#1}}
217 \WithSuffix\newcommand\linenumincrement*[1]{\setcounter{linenumincrementR}{#1}\setcounter{linenumincr
218 \WithSuffix\newcommand\firstsublinenum*[1]{\setcounter{subfirstlinenumR}{#1}\setcounter{subfirstlinen
219 \WithSuffix\newcommand\sublinenumincrement*[1]{\setcounter{sublinenumincrementR}{#1}\setcounter{subli

```

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

```

220 \newcommand*\Rlineflag{R}
221

```

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

```

222 \newcommand*\linenumrepR[1]{\@arabic{#1}}
223 \newcommand*\sublinenumrepR[1]{\@arabic{#1}}
224

```

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

```

225 \newcommand*{\leftlinenumR}{%
226   \l@dlinenumR
227   \kern\linenumsep}
228 \newcommand*{\rightlinenumR}{%
229   \kern\linenumsep
230   \l@dlinenumR}
231 \newcommand*{\l@dlinenumR}{%
232   \numlabfont\linenumrepR{\line@numR}\Rlineflag%
233   \ifsublines@
234     \ifnum\subline@num>\z@
235       \unskip\fullstop\sublinenumrepR{\subline@numR}%
236     \fi
237   \fi}
238
```

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` 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.

```

239 \newcount\line@numR
240 \newcount\subline@numR
241 \newcount\absline@numR
242
```

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

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

```

\actionlines@listR
\actions@listR
243 \list@create{\line@listR}
244 \list@create{\insertlines@listR}
245 \list@create{\actionlines@listR}
246 \list@create{\actions@listR}
247 \list@create{\sw@listR}%
248 \list@create{\sw@list@inedtextR}%
249
```

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

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

```

250 \list@create{\linesinpar@listL}
251 \list@create{\linesinpar@listR}
252 \list@create{\maxlinesinpar@list}
253

```

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

```

254 \newcount\page@numR
255

```

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.

```

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

```

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

```

257   \ifledRcol
258     \list@clear{\line@listR}%
259     \list@clear{\insertlines@listR}%
260     \list@clear{\actionlines@listR}%
261     \list@clear{\actions@listR}%
262     \list@clear{\linesinpar@listR}%
263     \list@clear{\linesonpage@listR}
264     \list@clear{\sw@listR}%
265     \list@clear{\sw@list@inedtextR}%
266   \else
267     \list@clearing@reg
268     \list@clear{\linesinpar@listL}%
269     \list@clear{\linesonpage@listL}%
270   \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).

```

271   \list@clear{\maxlinesinpar@list}

```

Now get the file and interpret it.

```

272   \get@linelistfile{#1}%
273   \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.

```

274   \ifledRcol
275     \global\page@numR=\m@ne
276     \ifx\actionlines@listR\empty
277       \gdef\next@actionlineR{1000000}%
278     \else
279       \gl@p\actionlines@listR\to\next@actionlineR

```

```

280     \glp\actions@listR\to\next@actionR
281   \fi
282 \else
283   \global\page@num=\m@ne
284   \ifx\actionlines@list\empty
285     \gdef\next@actionline{1000000}%
286   \else
287     \glp\actionlines@list\to\next@actionline
288     \glp\actions@list\to\next@action
289   \fi
290 \fi}
291

```

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` 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.

```

292 \newcommand{\@nl@regR}{%
293   \ifx\l@dchset@num\relax \else
294     \advance\absline@numR \@ne
295     \set@line@action
296     \let\l@dchset@num\relax
297     \advance\absline@numR \m@ne
298     \advance\line@numR \m@ne%    % do we need this?
299   \fi
300   \advance\absline@numR \@ne
301   \ifx\next@page@numR\relax \else
302     \page@action
303     \let\next@page@numR\relax
304   \fi
305   \ifx\sub@change\relax \else
306     \ifnum\sub@change>z@
307       \sublines@true
308     \else
309       \sublines@false
310     \fi
311     \sub@action
312     \let\sub@change\relax
313   \fi

```



```

314 \ifcase\@lockR
315 \or
316   \@lockR \tw@
317 \or\or
318   \@lockR \z@
319 \fi
320 \ifcase\sub@lockR
321 \or
322   \sub@lockR \tw@
323 \or\or
324   \sub@lockR \z@
325 \fi
326 \ifsublines@
327   \ifnum\sub@lockR<\tw@
328     \advance\subline@numR \@ne
329   \fi
330 \else
331   \ifnum\@lockR<\tw@
332     \advance\line@numR \@ne \subline@numR \z@
333   \fi
334 \fi}
335
336 \renewcommand*{\@nl}[2]{%
337   \fix@page{#1}%
338   \ifledRcol
339     \@nl@regR
340   \else
341     \@nl@reg
342   \fi}
343

```

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

```

\fix@page 344 \newcount\last@page@numR
345   \last@page@numR=-10000
346 \renewcommand*{\fix@page}[1]{%
347   \ifledRcol
348     \ifnum #1=\last@page@numR
349     \else
350       \ifbypage@R
351         \line@numR \z@ \subline@numR \z@
352       \fi
353       \page@numR=#1\relax
354       \last@page@numR=#1\relax
355       \def\next@page@numR{#1}%
356     \fi
357   \else
358     \ifnum #1=\last@page@num
359     \else
360       \ifbypage@
361         \line@num \z@ \subline@num \z@

```

```

362     \fi
363     \page@num=#1\relax
364     \last@page@num=#1\relax
365     \def\next@page@num{#1}%
366     \listxadd{\normal@page@break}{\the\absline@num}
367     \fi
368 \fi}
369

```

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

```

370 \renewcommand*{\@adv}[1]{%
371   \ifsublines@
372     \ifledRcol
373       \advance\subline@numR by #1\relax
374       \ifnum\subline@numR<\z@
375         \led@warn@BadAdvancelineSubline
376         \subline@numR \z@
377       \fi
378     \else
379       \advance\subline@num by #1\relax
380       \ifnum\subline@num<\z@
381         \led@warn@BadAdvancelineSubline
382         \subline@num \z@
383       \fi
384     \fi
385   \else
386     \ifledRcol
387       \advance\line@numR by #1\relax
388       \ifnum\line@numR<\z@
389         \led@warn@BadAdvancelineLine
390         \line@numR \z@
391       \fi
392     \else
393       \advance\line@num by #1\relax
394       \ifnum\line@num<\z@
395         \led@warn@BadAdvancelineLine
396         \line@num \z@
397       \fi
398     \fi
399   \fi
400   \set@line@action}
401

```

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

```

402 \renewcommand*{\@set}[1]{%
403   \ifledRcol
404     \ifsublines@

```

```

405     \subline@numR=#1\relax
406   \else
407     \line@numR=#1\relax
408   \fi
409   \set@line@action
410 \else
411   \ifsublines@
412     \subline@num=#1\relax
413   \else
414     \line@num=#1\relax
415   \fi
416   \set@line@action
417 \fi}
418

```

`\l@d@set` The `\l@d@set{<num>}` macro sets the line number for the next `\pstart...` to the value specified as its argument. This is used to implement `\setlinenum`.
`\l@dchset@num` `\l@dchset@num` is a flag to the `\@l` macro. If it is not `\relax` then a linenum change is to be done.

```

419 \renewcommand*{\l@d@set}[1]{%
420   \ifledRcol
421     \line@numR=#1\relax
422     \advance\line@numR \@ne
423     \def\l@dchset@num{#1}
424   \else
425     \line@num=#1\relax
426     \advance\line@num \@ne
427     \def\l@dchset@num{#1}
428   \fi}
429 \let\l@dchset@num\relax
430

```

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

```

431 \renewcommand*{\page@action}{%
432   \ifledRcol
433     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
434     \xright@appenditem{\next@page@numR}\to\actions@listR
435   \else
436     \xright@appenditem{\the\absline@num}\to\actionlines@list
437     \xright@appenditem{\next@page@num}\to\actions@list
438   \fi}

```

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

```

439 \renewcommand*{\set@line@action}{%
440   \ifledRcol
441     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
442   \ifsublines@
443     \l@dttempcnta=-\subline@numR

```

```

444 \else
445 \l@ldtempcnta=-\line@numR
446 \fi
447 \advance\l@ldtempcnta by -5000\relax
448 \xright@appenditem{\the\l@ldtempcnta}\to\actions@listR
449 \else
450 \xright@appenditem{\the\absline@num}\to\actionlines@list
451 \ifsublines@
452 \l@ldtempcnta=-\subline@num
453 \else
454 \l@ldtempcnta=-\line@num
455 \fi
456 \advance\l@ldtempcnta by -5000\relax
457 \xright@appenditem{\the\l@ldtempcnta}\to\actions@list
458 \fi}
459

```

`\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.

```

460 \renewcommand*{\sub@action}{%
461 \ifledRcol
462 \xright@appenditem{\the\absline@numR}\to\actionlines@listR
463 \ifsublines@
464 \xright@appenditem{-1001}\to\actions@listR
465 \else
466 \xright@appenditem{-1002}\to\actions@listR
467 \fi
468 \else
469 \xright@appenditem{\the\absline@num}\to\actionlines@list
470 \ifsublines@
471 \xright@appenditem{-1001}\to\actions@list
472 \else
473 \xright@appenditem{-1002}\to\actions@list
474 \fi
475 \fi}
476

```

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

```

477 \newcount\@lockR
478 \newcount\sub@lockR
479
480 \newcommand*{\do@lockonR}{%
481 \xright@appenditem{\the\absline@numR}\to\actionlines@listR
482 \ifsublines@
483 \xright@appenditem{-1005}\to\actions@listR
484 \ifnum\sub@lockR=\z@
485 \sub@lockR \@ne

```

```

486     \else
487         \ifnum\sub@lockR=\thr@@
488             \sub@lockR \@ne
489         \fi
490     \fi
491 \else
492     \xright@appenditem{-1003}\to\actions@listR
493     \ifnum\@lockR=\z@
494         \@lockR \@ne
495     \else
496         \ifnum\@lockR=\thr@@
497             \@lockR \@ne
498         \fi
499     \fi
500 \fi}
501
502 \renewcommand*{\do@lockon}{%
503     \ifx\next\lock@off
504         \global\let\lock@off=\skip@lockoff
505     \else
506         \ifledRcol
507             \do@lockonR
508         \else
509             \do@lockonL
510         \fi
511     \fi}

```

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

```

\do@lockoff 512
\do@lockoffR 513
\skip@lockoff 514 \newcommand{\do@lockoffR}{%
515     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
516     \ifsublines@
517         \xright@appenditem{-1006}\to\actions@listR
518         \ifnum\sub@lockR=\tw@
519             \sub@lockR \thr@@
520         \else
521             \sub@lockR \z@
522         \fi
523     \else
524         \xright@appenditem{-1004}\to\actions@listR
525         \ifnum\@lockR=\tw@
526             \@lockR \thr@@
527         \else
528             \@lockR \z@
529         \fi
530     \fi}
531
532 \renewcommand*{\do@lockoff}{%
533     \ifledRcol

```

```

534 \do@lockoffR
535 \else
536 \do@lockoffL
537 \fi}
538 \global\let\lock@off=\do@lockoff
539

```

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

```

540 \providecommand*{\n@num}{%
541 \renewcommand*{\n@num}{%
542 \ifledRcol
543 \xright@appenditem{\the\absline@numR}\to\actionlines@listR
544 \xright@appenditem{-1007}\to\actions@listR
545 \else
546 \n@num@reg
547 \fi}
548

```

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

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

```

549 \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.

```

550 \renewcommand*{\@ref}[2]{%
551 \ifledRcol
552 \global\insert@countR=#1\relax
553 \loop\ifnum\insert@countR>\z@
554 \xright@appenditem{\the\absline@numR}\to\insertlines@listR
555 \global\advance\insert@countR \m@ne
556 \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.

```

557 \begingroup
558 \let\@ref=\dummy@ref

```

```

559 \let\@lopR\@gobble
560 \let\page@action=\relax
561 \let\sub@action=\relax
562 \let\set@line@action=\relax
563 \let\@lab=\relax
564 \let\@sw\@gobbletwo%
565 #2
566 \global\endpage@num=\page@numR
567 \global\endline@num=\line@numR
568 \global\endsubline@num=\subline@numR
569 \endgroup

```

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

```

570 \xright@appenditem%
571 {\the\page@numR|\the\line@numR|}%
572 \ifsublines@ \the\subline@numR \else 0\fi}%
573 \the\endpage@num|\the\endline@num|}%
574 \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.

```

575 #2
576 \else

```

And when not in right text

```

577 \@ref@reg{#1}{#2}%
578 \fi}

```

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

```

579 \providecommand*\@pend}[1]{}
580 \renewcommand*\@pend}[1]{%
581 \ifbypstart@\global\line@num=0\fi%
582 \xright@appenditem{#1}\to\linesinpar@listL}
583 \providecommand*\@pendR}[1]{}
584 \renewcommand*\@pendR}[1]{%
585 \ifbypstart@R\global\line@numR=0\fi
586 \xright@appenditem{#1}\to\linesinpar@listR}
587

```

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

```

588 \providecommand*\@lopL}[1]{}
589 \renewcommand*\@lopL}[1]{%
590 \xright@appenditem{#1}\to\linesonpage@listL}
591 \providecommand*\@lopR}[1]{}

```

```

592 \renewcommand*{\@lopR}[1]{%
593   \xright@appenditem{#1}\to\linesonpage@listR}
594

```

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`.

```

595 \newwrite\linenum@outR

```

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

```

\first@linenum@out@Rtrue
\first@linenum@out@Rfalse 596 \newif\iffirst@linenum@out@R
597   \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.

```

598 \newcommand*{\line@list@stuffR}[1]{%
599   \read@linelist{#1}%
600   \iffirst@linenum@out@R
601     \immediate\closeout\linenum@outR
602     \global\first@linenum@out@Rfalse
603     \immediate\openout\linenum@outR=#1%
604   \else
605     \if@minipage%
606       \leavevmode%
607     \fi%
608     \closeout\linenum@outR%
609     \openout\linenum@outR=#1%
610   \fi}
611

```

`\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.

```

612 \newcommand*{\new@lineL}{%
613   \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.

```

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

```

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

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

```

616 \renewcommand*{\startsub}{\dimen0\lastskip
617 \ifdim\dimen0>0pt \unskip \fi
618 \ifledRcol \write\linenum@outR{\string\sub@on}%
619 \else \write\linenum@out{\string\sub@on}%
620 \fi
621 \ifdim\dimen0>0pt \hskip\dimen0 \fi}
622 \def\endsub{\dimen0\lastskip
623 \ifdim\dimen0>0pt \unskip \fi
624 \ifledRcol \write\linenum@outR{\string\sub@off}%
625 \else \write\linenum@out{\string\sub@off}%
626 \fi
627 \ifdim\dimen0>0pt \hskip\dimen0 \fi}
628

```

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

```

629 \renewcommand*{\advanceline}[1]{%
630 \ifledRcol \write\linenum@outR{\string\@adv[#1]}%
631 \else \write\linenum@out{\string\@adv[#1]}%
632 \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.

```

633 \renewcommand*{\setline}[1]{%
634 \ifnum#1<\z@
635 \led@warn@BadSetline
636 \else
637 \ifledRcol \write\linenum@outR{\string\@set[#1]}%
638 \else \write\linenum@out{\string\@set[#1]}%
639 \fi
640 \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.

```

641 \renewcommand*{\setlinenum}[1]{%
642 \ifnum#1<\z@
643 \led@warn@BadSetlinenum
644 \else
645 \ifledRcol \write\linenum@outR{\string\l@d@set[#1]}
646 \else \write\linenum@out{\string\l@d@set[#1]} \fi
647 \fi}
648

```

\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.

```

649 \renewcommand*{\startlock}{%
650   \ifledRcol \write\linenum@outR{\string\lock@on}%
651   \else      \write\linenum@out{\string\lock@on}%
652   \fi}
653 \def\endlock{%
654   \ifledRcol \write\linenum@outR{\string\lock@off}%
655   \else      \write\linenum@out{\string\lock@off}%
656   \fi}
657

```

`\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.

```

658 \renewcommand*{\skipnumbering}{%
659   \ifledRcol \write\linenum@outR{\string\n@num}%
660   \advanceline{-1}%
661   \else
662     \skipnumbering@reg
663   \fi}
664

```

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` And similarly for `\edtext`.

`\edtext`
`\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`.

```

665 \renewcommand*{\set@line}{%
666   \ifledRcol
667     \ifx\line@listR\empty
668       \global\noteschanged@true
669       \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
670     \else
671       \glp\line@listR\to\@tempb

```

```

672     \xdef\l@d@nums{\@tempb|\edfont@info}%
673     \global\let\@tempb=\undefined
674   \fi
675 \else
676   \ifx\line@list\empty
677     \global\noteschanged@true
678     \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
679   \else
680     \gl@p\line@list\to\@tempb
681     \xdef\l@d@nums{\@tempb|\edfont@info}%
682     \global\let\@tempb=\undefined
683   \fi
684 \fi}
685

```

16 Pstart numbers dumping and restoration

While in `eledmac` the footnotes are inserted in the same time as the `\pstart` ... `\pend` are read, in `eledpar` they are inserted when the `\Columns` or `\Pages` commands are called. Consequently, if we do nothing, the value of the `PstartL` and `PstartR` counters are not the same in the main text and in the notes. To solve this problem, we dump the values in two list (one by side) when processing `\pstart` and restore these at each `\pstart` when calling `\Columns` or `\Pages`. We also dump and restore the value of the boolean `\ifnumberpstart`.

So, first step, creating the lists. Here, “pc” means “public counters”.

```

\list@pstartL@pc
\list@pstartR@pc 686 \list@create{\list@pstartL@pc}%
687 \list@create{\list@pstartR@pc}%

```

Two commands to dump current pstarts. We prefer two commands to one with argument indicating the side, because the commands are short, and so we save one test (or a `\csname` construction).

```

\dump@pstartL@pc
\dump@pstartR@pc 688 \def\dump@pstartL@pc{%
689   \xright@appenditem{\the\c@pstartL}\to\list@pstartL@pc%
690   \global\cslet{numberpstart@L\the\l@dnumpstartsL}{\ifnumberpstart}%
691 }%
692
693 \def\dump@pstartR@pc{%
694   \xright@appenditem{\the\c@pstartR}\to\list@pstartR@pc%
695   \global\cslet{numberpstart@R\the\l@dnumpstartsR}{\ifnumberpstart}%
696 }%
697

```

```

\restore@pstartL@pc And so, the commands to restore them
\restore@pstartR@pc 698 \def\restore@pstartL@pc{%

```

```

699 \ifx\list@pstartL@pc\empty\else%
700   \gl@p\list@pstartL@pc\to\@temp%
701   \global\c@pstartL=\@temp%
702 \fi%
703 }%
704 \def\restore@pstartR@pc{%
705   \ifx\list@pstartR@pc\empty\else%
706     \gl@p\list@pstartR@pc\to\@temp%
707     \global\c@pstartR=\@temp%
708   \fi%
709 }%

```

17 Parallel environments

The initial set up for parallel processing is deceptively simple.

```

pairs The pairs environment is for parallel columns and the pages environment for
pages parallel pages.
chapterinpages 710 \newenvironment{pairs}{%}
711   \l@dpairingtrue
712   \l@dpagingfalse
713   \initnumbering@sectcmd
714   \at@begin@pairs%
715 }{%
716   \l@dpairingfalse
717 }
718

\AtBeginPairs The \AtBeginPairs macro just define a \at@begin@pairs macro, called at the
begining of each pairs environments.
719 \newcommand{\AtBeginPairs}[1]{\xdef\at@begin@pairs{#1}}%
720 \def\at@begin@pairs{}%
721

```

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.

```

722 \newenvironment{pages}{%
723   \let\oldchapter\chapter
724   \let\chapter\chapterinpages
725   \l@dpairingtrue
726   \l@dpagingtrue
727   \initnumbering@sectcmd
728   \setlength{\Lcolwidth}{\textwidth}%
729   \setlength{\Rcolwidth}{\textwidth}%
730 }{%
731   \l@dpairingfalse

```

```

732 \l@dpagingfalse
733 \let\chapter\oldchapter
734 }
735 \newcommand{\chapterinpages}{\thispagestyle{plain}%
736 \global\@topnum\z@
737 \afterindentfalse
738 \secdef\@chapter\@schapter}
739

```

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

```

740 \newif\ifinstanzaL
741 \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.

```

742 \newenvironment{Leftside}{%
743 \ledRcolfalse
744 \setcounter{pstartL}{1}
745 \let\pstart\pstartL
746 \let\thepstart\thepstartL
747 \let\pend\pendL
748 \let\memorydump\memorydumpL
749 \Leftsidehook
750 \let\old@startstanza\@startstanza
751 \def\@startstanza[##1]{\global\instanzaLtrue\old@startstanza[##1]}
752 }{
753 \Leftsidehookend}

```

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

```

\Leftsidehookend
\Rightsidehook 754 \newcommand*\Leftsidehook{}
\Rightsidehookend 755 \newcommand*\Leftsidehookend{}
756 \newcommand*\Rightsidehook{}
757 \newcommand*\Rightsidehookend{}
758

```

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.

```

759 \newenvironment{Rightside}{%
760 \ledRcoltrue
761 \let\beginnumbering\beginnumberingR
762 \let\endnumbering\endnumberingR
763 \let\pausenumbering\pausenumberingR
764 \let\resumenumbering\resumenumberingR
765 \let\memorydump\memorydumpR

```

```

766 \let\thepstart\thepstartR
767 \let\pstart\pstartR
768 \let\pend\pendR
769 \let\ledpb\ledpbR
770 \let\lednopb\lednopbR
771 \let\lineation\lineationR
772 \Rightsidehook
773 \let\old@startstanza\@startstanza
774 \def\@startstanza[##1]{\global\instanzaRtrue\old@startstanza[##1]}
775 }{f%
776 \ledRcolfalse
777 \Rightsidehookend
778 }
779

```

18 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.

18.1 Boxes, counters, \pstart and \pend

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

\one@lineR

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

```

780 \newcount\num@linesR
781 \newbox\one@lineR
782 \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.

```

783
784 \newcounter{pstartL}
785 \renewcommand{\thepstartL}{\bfseries\@arabic\c@pstartL}. }
786 \newcounter{pstartR}
787 \renewcommand{\thepstartR}{\bfseries\@arabic\c@pstartR}. }
788
789 \newcommandx*{\pstartL}[1][1]{%
790   \if@nobreak%
791     \let\@oldnobreak\@nobreaktrue%
792   \else%
793     \let\@oldnobreak\@nobreakfalse%
794   \fi%
795   \@nobreaktrue%
796   \ifluatex%
797     \xdef\l@luatextextdir@L{\the\luatextextdir}%
798     \xdef\l@luatexpardir@L{\the\luatexpardir}%
799     \xdef\l@luatexbodydir@L{\the\luatexbodydir}%
800   \fi%
801   \ifnumbering \else%
802     \led@err@PstartNotNumbered%
803     \beginnumbering%
804   \fi%
805   \ifnumberedpar%
806     \led@err@PstartInPstart%
807   \pend%
808 \fi%
```

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

```

809 \ifpst@rtedL\else%
810   \list@clear{\inserts@list}%
811   \global\let\next@insert=\empty%
812   \global\pst@rtedLtrue%
813 \fi%
814 \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.

```

815 \global\advance\l@dnumpstartsL \@ne%
816 \ifnum\l@dnumpstartsL>\l@dc@maxchunks%
817   \led@err@TooManyPstarts%
818   \global\l@dnumpstartsL=\l@dc@maxchunks%
819 \fi%
820 \global\setnamebox{\l@dLcolrawbox\the\l@dnumpstartsL}=\vbox\bgroup%
821 \ifautopar\else%
822   \ifnumberpstart%
823     \ifsidepstartnum%
824     \else%
```

```

825     \thepstartL%
826     \fi%
827     \fi%
828     \fi%
829     \hsize=\Lcolwidth%
830     \numberedpar@true%
831     \iflabelpstart\protected@edef\@currentlabel%
832         {\p@pstartL\thepstartL}\fi%

Dump the optional arguments
833     \ifstrempy{#1}%
834         {\csgdef{before@pstartL@the\l@dnumpstartsL}{\at@every@pstart}}%
835         {\csgdef{before@pstartL@the\l@dnumpstartsL}{\noindent#1}}%
836     }

837 \newcommandx*{\pstartR}[1][1]{%
838     \if@nbreak%
839         \let\@oldnbreak\@nbreaktrue%
840     \else%
841         \let\@oldnbreak\@nbreakfalse%
842     \fi%
843     \@nbreaktrue%
844     \ifluatex%
845         \xdef\l@luatextextdir@R{\the\luatextextdir}%
846         \xdef\l@luatexpardir@R{\the\luatexpardir}%
847         \xdef\l@luatexbodydir@R{\the\luatexbodydir}%
848     \fi%
849     \ifnumberingR \else%
850         \led@err@PstartNotNumbered%
851         \beginnumberingR%
852     \fi%
853     \ifnumberedpar@%
854         \led@err@PstartInPstart%
855         \pendR%
856     \fi%
857     \ifpst@rtedR\else%
858         \list@clear{\inserts@listR}%
859         \global\let\next@insertR=\empty%
860         \global\pst@rtedRtrue%
861     \fi%
862     \begingroup\normal@pars%
863     \global\advance\l@dnumpstartsR \@ne%
864     \ifnum\l@dnumpstartsR>\l@dc@maxchunks%
865         \led@err@TooManyPstarts%
866         \global\l@dnumpstartsR=\l@dc@maxchunks%
867     \fi%
868     \global\setnamebox{l@dRcolrawbox\the\l@dnumpstartsR}=\vbox\bgroup%
869     \ifautopar\else%
870         \ifnumberpstart%
871         \ifsidepstartnum\else%
872             \thepstartR%

```



```

873         \fi%
874         \fi%
875         \fi%
876 \hsize=\Rcolwidth%
877 \numberedpar@true%
878 \iflabelpstart\protected@edef\@currentlabel%
879     {\p@pstartR\thepstartR}\fi%
880 \ifstrempy{#1}%
881     {\csgdef{before@pstartR@the\l@dnumpstartsR}{\at@every@pstart}}%
882     {\csgdef{before@pstartR@the\l@dnumpstartsR}{\noindent#1}}%
883 }

```

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

```

884 \newcommandx*{\pendL}[1][1]{%
885     \ifnumbering \else%
886         \led@err@PendNotNumbered%
887     \fi%
888     \ifnumberedpar@ \else%
889         \led@err@PendNoPstart%
890     \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.

```

891 \l@dzeropenalties%
892 \endgraf\global\num@lines=\prevgraf\egroup%
893 \global\par@line=0%

```

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

```

894 \endgroup%
895 \ignorespaces%
896 \@oldnobreak%
897 \dump@pstartL@pc%
898 \ifnumberpstart%
899     \addtocounter{pstartL}{1}%
900 \fi
901 \parledgroup@beforenotes@save{L}%

```

Dump content of the optional argument.

```

902 \ifstrempy{#1}%
903     {\csgdef{after@pendL@the\l@dnumpstartsL}{\at@every@pend}}%
904     {\csgdef{after@pendL@the\l@dnumpstartsL}{\noindent#1}}%
905 }

```

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

```

906 \newcommandx*{\pendR}[1][1]{%
907     \ifnumberingR \else%
908         \led@err@PendNotNumbered%

```

```

909 \fi%
910 \ifnumberedpar@ \else%
911   \led@err@PendNoPstart%
912 \fi%
913 \l@dzeropenalties%
914 \endgraf\global\num@linesR=\prevgraf\egroup%
915 \global\par@lineR=0%
916 \endgroup%
917 \ignorespaces%
918 \@oldnobreak%
919 \dump@pstartR@pc%
920 \ifnumberpstart%
921   \addtocounter{pstartR}{1}%
922 \fi%
923 \parledgroup@beforenotes@save{R}%
924 \ifstrempy{#1}%
925   {\csgdef{after@pendR@the\l@dnumpstartsR}{\at@every@pend}}%
926   {\csgdef{after@pendR@the\l@dnumpstartsR}{\noindent#1}}%
927 }
928

```

`\ifprint@last@after@pendL` Two booleans set to true, when the time is to print the last optional argument of a `\pend`.

```

929 \newif\ifprint@last@after@pendL%
930 \newif\ifprint@last@after@pendR%

```

18.2 Processing one line

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

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

```

931 \newbox\l@dleftbox
932 \newbox\l@drightbox
933

```

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

```

\countRline 934 \newcount\countLline
935   \countLline \z@
936 \newcount\countRline
937   \countRline \z@
938

```

`\@donereallinesL` 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).

```

\@donetotallinesL 939 \newcount\@donereallinesL
\@donetotallinesR 940 \newcount\@donetotallinesL

```

```

941 \newcount\@donereallinesR
942 \newcount\@donetotallinesR
943

```

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

```

944 \newcommand*{\do@lineL}{%
945   \letcs{\ifnumberpstart}{numberpstart@L\the\l@dpscl}%
946   \advance\countLline \@ne
947   \ifvbox\namebox{1@dLcolrawbox\the\l@dpscl}%
948   {\vbadness=10000
949    \splittopskip=\z@
950    \do@lineLhook
951    \l@emptyd@ta
952    \global\setbox\one@line=\vsplit\namebox{1@dLcolrawbox\the\l@dpscl}
953      to\baselineskip}%
954   \IfStrEq{\splitfirstmarks\parledgroup@}{begin}{\parledgroup@notes@startL}{%
955     \unvbox\one@line \global\setbox\one@line=\lastbox
956     \getline@numL
957     \ifnum\@lock>\@ne%
958       \inserthangingsymboltrue%
959     \else%
960       \inserthangingsymbolfalse%
961     \fi
962     \setbox\l@leftbox
963     \hb@xt@ \Lcolwidth{%
964       \affixline@num
965       \xifinlist{\the\l@dpscl}{\eled@sections@@}%
966       {\add@inserts\affixside@note}%
967       {\print@lineL}}%
968     \add@penaltiesL
969     \global\advance\@donereallinesL\@ne
970     \global\advance\@donetotallinesL\@ne
971   \else
972     \setbox\l@leftbox \hb@xt@ \Lcolwidth{\hspace*\Lcolwidth}%
973     \global\advance\@donetotallinesL\@ne
974   \fi}
975
976

```

`\print@lineL` `\print@lineL` is for lines without a sectioning command. See `eledmac` definition of `\print@line` for handbook.

```

977 \def\print@lineL{%
978   \affixpstart@numL%
979   \l@dld@ta %space kept for backward compatibility
980   \add@inserts\affixside@note%
981   \l@dlsn@te %space kept for backward compatibility
982   {\ledllfill\hb@xt@ \wd\one@line{%
983     \do@insidelineLhook%

```

```

984      \ifluatex%
985        \luatextextdir\l@luatextextdir@L%
986      \fi%
987      \new@lineL%
988      \inserthangingsymbolL%
989      \l@dunhbox@line{\one@line}}\correctchangelL\ledrlfill\l@drd@ta%
990      \l@drsn@te}}
991

```

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

```

992 \def\print@eledsectionL{%
993   \addtocounter{pstartL}{-1}%
994   \ifdefstring{\@eledsectnotoc}{L}{\ledsectnotoc}{%
995     \ifdefstring{\@eledsectmark}{L}{\ledsectnomark}%
996     \numdef{\temp@}{\l@dpscl-1}%
997     \xifinlist{\temp@}{\eled@sections@}{\@nobreaktrue}{\@nobreakfalse}%
998     \@eled@sectioningtrue%
999   \bgroup%
1000     \ifluatex%
1001       \luatextextdir\l@luatextextdir@L%
1002       \luatexpardir\l@luatexpardir@L%
1003       \luatexbodydir\l@luatexbodydir@L%
1004     \fi%
1005     \csuse{eled@sectioning@the\l@dpscl}%
1006   \egroup%
1007   \@eled@sectioningfalse%
1008   \global\csundef{eled@sectioning@the\l@dpscl}%
1009   \if@RTL%
1010     \hspace{-3\paperwidth}%
1011     {\hbox{\l@dunhbox@line{\one@line}} \new@line}%
1012   \else%
1013     \hspace{3\paperwidth}%
1014     {\new@line \hbox{\l@dunhbox@line{\one@line}}}%
1015   \fi%
1016   \vskip\eledsection@correcting@skip%
1017 }
1018

```

\dolineLhook These high-level commands just redefine the low-level commands. They have to be used by user, without \makeatletter.

```

\doininsidelineLhook 1019 \newcommand*{\dolineLhook}[1]{\gdef\dolineLhook{#1}}%
\doininsidelineRhook 1020 \newcommand*{\dolineRhook}[1]{\gdef\dolineRhook{#1}}%
1021 \newcommand*{\doininsidelineLhook}[1]{\gdef\doininsidelineLhook{#1}}%
1022 \newcommand*{\doininsidelineRhook}[1]{\gdef\doininsidelineRhook{#1}}%
1023

```

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

```

\dolineRhook 1024 \newcommand*{\do@lineLhook}{}
\doininsidelineLhook 1025 \newcommand*{\do@lineRhook}{}
\doininsidelineRhook

```

```

1026 \newcommand*{\do@insidelineLhook}{}
1027 \newcommand*{\do@insidelineRhook}{}
1028

```

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

```

1029 \newcommand*{\do@lineR}{%
1030   \letcs{\ifnumberpstart}{numberpstart@R\the\l@dpscR}%
1031   \ledRcol@true%
1032   \advance\countRline \@ne
1033   \ifvbox\namebox{\l@dRcolrawbox\the\l@dpscR}%
1034   {\vbadness=10000
1035    \splittopskip=\z@
1036    \do@lineRhook
1037    \l@emptyd@ta
1038    \global\setbox\one@lineR=\vsplit\namebox{\l@dRcolrawbox\the\l@dpscR}
1039                                to\baselineskip}%
1040    \IfStrEq{\splitfirstmarks\parledgroup@}{begin}{\parledgroup@notes@startR}{}
1041    \unvbox\one@lineR \global\setbox\one@lineR=\lastbox
1042    \getline@numR
1043    \ifnum\@lockR>\@ne%
1044      \inserthangingsymbolRtrue
1045    \else%
1046      \inserthangingsymbolRfalse%
1047    \fi%
1048    \setbox\l@dtrightbox
1049    \hb@xt@ \Rcolwidth{%
1050      \affixline@numR%
1051      \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}%
1052      {\add@insertsR\affixside@noteR}%
1053      {\print@lineR}%
1054    }%
1055    \add@penaltiesR
1056    \global\advance\@donereallinesR\@ne
1057    \global\advance\@donetotallinesR\@ne
1058  \else
1059    \setbox\l@dtrightbox \hb@xt@ \Rcolwidth{\hspace*\Rcolwidth}
1060    \global\advance\@donetotallinesR\@ne
1061  \fi
1062  \ledRcol@false%
1063 }
1064
1065

```

```

\print@lineR
\print@eledsectionR

```

18.3 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.

```

1066 \newcommand*{\getline@numR}{%
1067   \global\advance\absline@numR \@ne
1068   \do@actionsR
1069   \do@ballastR
1070 \ifledgroupnotesR@\else\ifnumberline
1071   \ifsublines@
1072     \ifnum\sub@lockR<\tw@
1073       \global\advance\subline@numR \@ne
1074     \fi
1075   \else
1076     \ifnum\@lockR<\tw@
1077       \global\advance\line@numR \@ne
1078       \global\subline@numR \z@
1079     \fi
1080   \fi
1081 \fi
1082 \fi
1083 }
1084 \newcommand*{\getline@numL}{%
1085   \global\advance\absline@num \@ne
1086   \do@actions
1087   \do@ballast
1088 \ifledgroupnotesL@\else\ifnumberline
1089   \ifsublines@
1090     \ifnum\sub@lock<\tw@
1091       \global\advance\subline@num \@ne
1092     \fi
1093   \else
1094     \ifnum\@lock<\tw@
1095       \global\advance\line@num \@ne
1096       \global\subline@num \z@
1097     \fi
1098   \fi
1099 \fi
1100 \fi
1101 }
1102
1103

```

`\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.

```

1104 \newcommand*{\do@ballastR}{\global\ballast@count=\z@
1105   \begingroup
1106     \advance\absline@numR \@ne
1107     \ifnum\next@actionlineR=\absline@numR
1108       \ifnum\next@actionR>-1001
1109         \global\advance\ballast@count by -\c@ballast
1110       \fi
1111     \fi
1112   \endgroup}

```

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

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

```

1113 \newcommand*{\do@actions@fixedcodeR}{%
1114   \ifcase\@l@tempcnta%
1115   \or%                % 1001
1116     \global\sublines@true
1117   \or%                % 1002
1118     \global\sublines@false
1119   \or%                % 1003
1120     \global\@lockR=\@ne
1121   \or%                % 1004
1122     \ifnum\@lockR=\tw@
1123       \global\@lockR=\thr@@
1124     \else
1125       \global\@lockR=\z@
1126     \fi
1127   \or%                % 1005
1128     \global\sub@lockR=\@ne
1129   \or%                % 1006
1130     \ifnum\sub@lockR=\tw@
1131       \global\sub@lockR=\thr@@
1132     \else
1133       \global\sub@lockR=\z@
1134     \fi
1135   \or%                % 1007
1136     \l@dskipnumbertrue
1137   \else
1138     \led@warn@BadAction
1139   \fi}
1140
1141
1142 \newcommand*{\do@actionsR}{%
1143   \global\let\do@actions@nextR=\relax
1144   \@l@tempcntb=\absline@numR
1145   \ifnum\@l@tempcntb<\next@actionlineR\else
1146     \ifnum\next@actionR>-1001\relax
1147       \global\page@numR=\next@actionR
1148       \ifbypage@R
1149         \global\line@numR \z@ \global\subline@numR \z@
1150       \fi
1151     \else
1152       \ifnum\next@actionR<-4999\relax % 9/05 added relax here
1153         \@l@tempcnta=-\next@actionR
1154         \advance\@l@tempcnta by -5001\relax
1155         \ifsublines@
1156           \global\subline@numR=\@l@tempcnta

```

```

1157         \else
1158             \global\line@numR=\@l@dttempcnta
1159         \fi
1160     \else
1161         \@l@dttempcnta=-\next@actionR
1162         \advance\@l@dttempcnta by -1000\relax
1163         \do@actions@fixedcodeR
1164     \fi
1165 \fi
1166 \ifx\actionlines@listR\empty
1167     \gdef\next@actionlineR{1000000}%
1168 \else
1169     \gl@p\actionlines@listR\to\next@actionlineR
1170     \gl@p\actions@listR\to\next@actionR
1171     \global\let\do@actions@nextR=\do@actionsR
1172 \fi
1173 \fi
1174 \do@actions@nextR}
1175

```

18.4 Line number printing

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

```

\ch@cksub@l@ckR 1176
\ch@ck@l@ckR 1177 \providecommand*\l@dcalcnun}[3]{%
\fx@l@cksR 1178 \ifnum #1 > #2\relax
\affixline@numR 1179 \l@dttempcnta = #1\relax
1180 \advance\@l@dttempcnta by -#2\relax
1181 \divide\@l@dttempcnta by #3\relax
1182 \multiply\@l@dttempcnta by #3\relax
1183 \advance\@l@dttempcnta by #2\relax
1184 \else
1185 \l@dttempcnta=#2\relax
1186 \fi}
1187
1188 \newcommand*\ch@cksub@l@ckR}{%
1189 \ifcase\sub@lockR
1190 \or
1191 \ifnum\sublock@disp=\@ne
1192 \l@dttempcntb \z@ \l@dttempcnta \@ne
1193 \fi
1194 \or
1195 \ifnum\sublock@disp=\tw@
1196 \else
1197 \l@dttempcntb \z@ \l@dttempcnta \@ne
1198 \fi
1199 \or
1200 \ifnum\sublock@disp=\z@
1201 \l@dttempcntb \z@ \l@dttempcnta \@ne

```



```

1202   \fi
1203 \fi}
1204
1205 \newcommand*{\ch@ck@l@ckR}{%
1206   \ifcase\@lockR
1207   \or
1208     \ifnum\lock@disp=\@ne
1209       \@l@tempcntb \z@ \l@tempcnta \@ne
1210     \fi
1211   \or
1212     \ifnum\lock@disp=\tw@
1213     \else
1214       \@l@tempcntb \z@ \l@tempcnta \@ne
1215     \fi
1216   \or
1217     \ifnum\lock@disp=\z@
1218       \@l@tempcntb \z@ \l@tempcnta \@ne
1219     \fi
1220 \fi}
1221
1222 \newcommand*{\f@x@l@cksR}{%
1223   \ifcase\@lockR
1224   \or
1225     \global\@lockR \tw@
1226   \or \or
1227     \global\@lockR \z@
1228   \fi
1229   \ifcase\sub@lockR
1230   \or
1231     \global\sub@lockR \tw@
1232   \or \or
1233     \global\sub@lockR \z@
1234   \fi}
1235
1236
1237 \newcommand*{\affixline@numR}{%
1238   \ifledgroupnotesR\else\ifnumberline
1239     \ifl@dskipnumber
1240       \global\l@dskipnumberfalse
1241     \else
1242       \ifsublines@
1243         \@l@tempcntb=\subline@numR
1244         \l@dcalcnm{\subline@numR}{\c@firstsublinenumR}{\c@sublinenumincrementR}%
1245         \ch@cksub@lockR
1246       \else
1247         \@l@tempcntb=\line@numR
1248         \ifx\linenumberlist\empty
1249           \l@dcalcnm{\line@numR}{\c@firstlinenumR}{\c@linenumincrementR}%
1250         \else
1251           \@l@tempcnta=\line@numR

```

```

1252 \edef\rem@inder{\linenumberlist,\number\line@numR,}%
1253 \edef\sc@n@list{\def\noexpand\sc@n@list
1254   ###1,\number\@l@tempcnta,###2|{\def\noexpand\rem@inder{###2}}}%
1255 \sc@n@list\expandafter\sc@n@list\rem@inder|
1256 \ifx\rem@inder\empty\advance\@l@tempcnta\@ne\fi
1257 \fi
1258 \ch@ck@l@ckR
1259 \fi
1260 \ifnum\@l@tempcnta=\@l@tempcntb
1261 \if@twocolumn
1262 \if@firstcolumn
1263 \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1264 \else
1265 \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1266 \fi
1267 \else
1268 \l@tempcntb=\line@marginR
1269 \ifnum\@l@tempcntb>\@ne
1270 \advance\@l@tempcntb by\page@numR
1271 \fi
1272 \ifodd\@l@tempcntb
1273 \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1274 \else
1275 \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1276 \fi
1277 \fi
1278 \fi
1279 \f@x@l@cksR
1280 \fi
1281 \fi
1282 \fi}

```

18.5 Pstart number printing in side

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

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

```

\affixpstart@numL
\affixpstart@numR 1283
\leftpstartnumR 1284 \newcommand*{\affixpstart@numL}{%
\rightpstartnumR 1285 \ifsidepstartnum
\leftpstartnumL 1286 \if@twocolumn
\rightpstartnumL 1287 \if@firstcolumn
\ifpstartnumR 1288 \gdef\l@dld@ta{\llap{\leftpstartnumL}}}%
1289 \else

```

```

1290     \gdef\l@drd@ta{\rlap{{\rightpstartnumL}}}%
1291     \fi
1292     \else
1293       \@l@tempcntb=\line@margin
1294       \ifnum\@l@tempcntb>\@ne
1295         \advance\@l@tempcntb \page@num
1296       \fi
1297       \ifodd\@l@tempcntb
1298         \gdef\l@drd@ta{\rlap{{\rightpstartnumL}}}%
1299       \else
1300         \gdef\l@dld@ta{\llap{{\leftpstartnumL}}}%
1301       \fi
1302     \fi
1303 \fi
1304 }
1305 \newcommand*{\affixpstart@numR}{%
1306 \ifsidepstartnum
1307 \if@twocolumn
1308   \if@firstcolumn
1309     \gdef\l@dld@ta{\llap{{\leftpstartnumR}}}%
1310   \else
1311     \gdef\l@drd@ta{\rlap{{\rightpstartnumR}}}%
1312   \fi
1313 \else
1314   \@l@tempcntb=\line@marginR
1315   \ifnum\@l@tempcntb>\@ne
1316     \advance\@l@tempcntb \page@numR
1317   \fi
1318   \ifodd\@l@tempcntb
1319     \gdef\l@drd@ta{\rlap{{\rightpstartnumR}}}%
1320   \else
1321     \gdef\l@dld@ta{\llap{{\leftpstartnumR}}}%
1322   \fi
1323 \fi
1324 \fi
1325 }
1326
1327 \newcommand*{\leftpstartnumL}{%
1328 \ifpstartnum
1329 \thepstartL
1330 \kern\linenumsep\global\pstartnumfalse\fi
1331 }
1332 \newcommand*{\rightpstartnumL}{%
1333 \ifpstartnum\kern\linenumsep
1334 \thepstartL
1335 \global\pstartnumfalse\fi
1336 }
1337 \newif\ifpstartnumR
1338 \pstartnumRtrue
1339 \newcommand*{\leftpstartnumR}{%

```

```

1340 \ifpstartnumR
1341 \thepstartR
1342 \kern\linenumsep\global\pstartnumRfalse\fi
1343 }
1344 \newcommand*{\rightpstartnumR}{
1345 \ifpstartnumR\kern\linenumsep
1346 \thepstartR
1347 \global\pstartnumRfalse\fi
1348 }

```

18.6 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.

```
1349 \list@create{\inserts@listR}
```

`\add@insertsR` The right text version.

```

\add@inserts@nextR 1350 \newcommand*{\add@insertsR}{%
1351   \global\let\add@inserts@nextR=\relax
1352   \ifx\inserts@listR\empty \else
1353     \ifx\next@insertR\empty
1354       \ifx\insertlines@listR\empty
1355         \global\noteschanged@true
1356         \gdef\next@insertR{100000}%
1357       \else
1358         \glp\insertlines@listR\to\next@insertR
1359       \fi
1360     \fi
1361     \ifnum\next@insertR=\absline@numR
1362       \glp\inserts@listR\to\@insertR
1363       \@insertR
1364       \global\let\@insertR=\undefined
1365       \global\let\next@insertR=\empty
1366       \global\let\add@inserts@nextR=\add@insertsR
1367     \fi
1368   \fi
1369   \add@inserts@nextR}
1370

```

18.7 Penalties

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

In the code below, which is a virtual copy of the original `\add@penalties`, `\num@lines` is the number of lines in the whole paragraph, and `\par@line` is the

line we’re working on at the moment. The count `\@l@dttempcnta` is used to calculate and accumulate the penalty; it is initially set to the value of `\ballast@count`, which has been worked out in `\do@ballast`. Finally, the penalty is checked to see that it doesn’t go below -10000 .

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

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

```
1371 \newcommand*{\add@penaltiesL}{\}
1372 \newcommand*{\add@penaltiesR}{\}
1373
```

18.8 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.

```
1374 \newcommand*{\flush@notesR}{%
1375 \xloop
1376 \ifx\inserts@listR\empty \else
1377 \glp\inserts@listR\to\@insertR
1378 \@insertR
1379 \global\let\@insertR=\undefined
1380 \repeat}
```

1381

19 Footnotes

19.1 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
1382 \def\printlinesR#1|#2|#3|#4|#5|#6|#7|{\begingroup
1383   \setprintlines{#1}{#2}{#3}{#4}{#5}{#6}%
1384   \ifl@d@pnum #1\fullstop\fi
1385   \ifledplinenum \linenumr@p{#2}\Rlineflag\else \symplinenum\fi
1386   \ifl@d@ssub \fullstop \sublinenumr@p{#3}\fi
1387   \ifl@d@dash \endashchar\fi
1388   \ifl@d@pnum #4\fullstop\fi
1389   \ifl@d@elin \linenumr@p{#5}\Rlineflag\fi
1390   \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumr@p{#6}\fi
1391 \endgroup}
1392
1393 \let\ledsavedprintlines\printlines
1394

```

20 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.

```

1395 \list@create{\labelref@listR}
1396

```

`\edlabel` Since version 1.18.0, this command is defined only one time in `eledmac`, including features for `eledpar`.

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

```

1397 \def\l@dmake@labelsR#1|#2|#3|#4|#5{%
1398   \expandafter\ifx\csname the@label#5\endcsname \relax\else
1399     \led@warn@DuplicateLabel{#4}%
1400   \fi

```

```

1401 \expandafter\gdef\csname the@label#5\endcsname{#1|#2\Rlineflag|#3|#4}%
1402 \ignorespaces}
1403 \AtBeginDocument{%
1404 \def\l@dmake@labelsR#1|#2|#3|#4|#5{%
1405 }
1406

```

\@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.

```

1407 \renewcommand*{\@lab}{%
1408 \ifledRcol
1409 \xright@appenditem{\linenumr@p{\line@numR}}{|%
1410 \ifsublines@ \sublinenumr@p{\subline@numR}\else 0\fi}%
1411 \to\labelref@listR
1412 \else
1413 \xright@appenditem{\linenumr@p{\line@num}}{|%
1414 \ifsublines@ \sublinenumr@p{\subline@num}\else 0\fi}%
1415 \to\labelref@list
1416 \fi}
1417

```

21 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* 1418 \WithSuffix\newcommand\sidenotemargin*[1]{%
1419 \l@dgetsidenote@margin{#1}
1420 \global\sidenote@marginR=\@l@tempcntb
1421 \global\sidenote@margin=\@l@tempcntb
1422 }
1423 \newcount\sidenote@marginR
1424 \global\sidenote@margin=\@ne
1425

```

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

```

1426 \newcommand*{\affixside@noteR}{%
1427 \def\sidenotecontent@{%
1428 \numgdef{\itemcount@}{0}%
1429 \def\do##1{%
1430 \ifnumequal{\itemcount@}{0}%
1431 {%
1432 \appto\sidenotecontent@{##1}}% Not print not separator before the 1st note
1433 {\appto\sidenotecontent@{\sidenotessep ##1}%

```

```

1434         }%
1435         \numgdef{\itemcount@}{\itemcount@+1}%
1436     }%
1437     \dolistloop{\l@dcstotetext}%
1438     \ifnumgreater{\itemcount@}{1}{\led@err@ManySidenotes}{}%
1439     \gdef\@templ@d{%
1440     \gdef\@templ@n{\l@dcstotetext\l@dcstotetext@l\l@dcstotetext@r}%
1441     \ifx\@templ@d\@templ@n \else%
1442     \if@twocolumn%
1443     \if@firstcolumn%
1444     \setl@dlp@rbox{##1}{\sidenotecontent@}%
1445     \else%
1446     \setl@drp@rbox{\sidenotecontent@}%
1447     \fi%
1448     \else%
1449     \l@dttempcntb=\sidenote@marginR%
1450     \ifnum\l@dttempcntb>\@ne%
1451     \advance\l@dttempcntb by\page@numR%
1452     \fi%
1453     \ifodd\l@dttempcntb%
1454     \setl@drp@rbox{\sidenotecontent@}%
1455     \gdef\sidenotecontent@{%
1456     \numdef{\itemcount@}{0}%
1457     \dolistloop{\l@dcstotetext@l}%
1458     \ifnumgreater{\itemcount@}{1}{\led@err@ManyLeftnotes}{}%
1459     \setl@dlp@rbox{\sidenotecontent@}%
1460     \else%
1461     \setl@dlp@rbox{\sidenotecontent@}%
1462     \gdef\sidenotecontent@{%
1463     \numdef{\itemcount@}{0}%
1464     \dolistloop{\l@dcstotetext@r}%
1465     \ifnumgreater{\itemcount@}{1}{\led@err@ManyRightnotes}{}%
1466     \setl@drp@rbox{\sidenotecontent@}%
1467     \fi%
1468     \fi%
1469     \fi%
1470 }
1471

```

22 Familiar footnotes

`\l@dbfnote` `\l@dbfnote` adds the footnote to the insert list, and `\v1@dbfnote` calls the original `\@footnotetext`.

```

1472 \renewcommand{\l@dbfnote}[1]{%
1473   \ifnumberedpar@
1474   \gdef\@tag{#1\relax}%
1475   \ifledRcol%
1476     \xright@appenditem{\noexpand\v1@dbfnote{\@expandonce\@tag}}{\@thefnmark}}%

```



```

1477             \to\inserts@listR
1478     \global\advance\insert@countR \@ne%
1479     \else%
1480     \xright@appenditem{\noexpand\vl@dbfnote{\@expandonce\@tag}\@thefnmark}}%
1481             \to\inserts@list
1482     \global\advance\insert@count \@ne%
1483     \fi
1484 \fi\ignorespaces}
1485

```

\normalbfnoteX

```

1486 \renewcommand{\normalbfnoteX}[2]{%
1487   \ifnumberedpar@
1488     \ifledRcol%
1489       \ifluatex
1490         \footnotelang@lua[R]%
1491       \fi
1492       \@ifundefined{xpg@main@language}%if polyglossia
1493       {}%
1494       {\footnotelang@poly[R]}%
1495       \protected@xdef\thisfootnote{\csuse{thefootnote#1}}%
1496       \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\expandonce\thisfootnote}}%
1497               \to\inserts@listR
1498       \global\advance\insert@countR \@ne%
1499     \else%
1500       \ifluatex
1501         \footnotelang@lua%
1502       \fi
1503       \@ifundefined{xpg@main@language}%if polyglossia
1504       {}%
1505       {\footnotelang@poly}%
1506       \protected@xdef\thisfootnote{\csuse{thefootnote#1}}%
1507       \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\expandonce\thisfootnote}}%
1508               \to\inserts@list
1509       \global\advance\insert@count \@ne%
1510     \fi
1511 \fi\ignorespaces}
1512

```

23 Verse

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

```

\inserthangingsymbolL
\inserthangingsymbolR 1513 \newif\ifinserthangingsymbolR
1514 \newcommand{\inserthangingsymbolL}{%
1515 \ifinserthangingsymbol%
1516   \ifinstanzaL%

```

```

1517         \hangingsymbol%
1518     \fi%
1519 \fi}
1520 \newcommand{\inserthangingsymbolR}{%
1521 \ifinserthangingsymbolR%
1522     \ifinstanzaR%
1523         \hangingsymbol%
1524     \fi%
1525 \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 1526 \newcommand{\correcthangingL}{%
1527 \ifl@dpadding\else%
1528     \ifinstanzaL%
1529         \ifinserthangingsymbol%
1530             \hskip \ifundefined{sza@00}{0}{\expandafter%
1531                 \noexpand\csname sza@00\endcsname}\stanzaindentbase%
1532         \fi%
1533     \fi%
1534 \fi}
1535
1536 \newcommand{\correcthangingR}{%
1537 \ifl@dpadding\else%
1538     \ifinstanzaR%
1539         \ifinserthangingsymbolR%
1540             \hskip \ifundefined{sza@00}{0}{\expandafter%
1541                 \noexpand\csname sza@00\endcsname}\stanzaindentbase%
1542         \fi%
1543     \fi%
1544 \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.

```

1545 \chardef\next=\catcode'\&
1546 \catcode'\&=\active
1547

```

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

```

1548 \newenvironment{astanza}{%
1549     \startstanzahook
1550     \catcode'\&\active
1551     \global\stanza@count\@ne\stanza@modulo\@ne

```

```

1552 \ifnum\usernamecount{sza@0@}=\z@
1553   \let\stanza@hang\relax
1554   \let\endlock\relax
1555 \else
1556 %%    \interlinepenalty\@M % this screws things up, but I don't know why
1557   \rightskip\z@ plus 1fil\relax
1558 \fi
1559 \ifnum\usernamecount{szp@0@}=\z@
1560   \let\sza@penalty\relax
1561 \fi
1562 \def&{%
1563   \endlock\mbox{}}%
1564   \sza@penalty
1565   \global\advance\stanza@count\@ne
1566   \@astanza@line}%
1567 \def\&{%
1568   \endlock\mbox{}}
1569 \pend
1570 \endstanzaextra}%
1571 \pstart
1572 \@astanza@line
1573 }{}
1574

```

`\@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.

```

1575 \newcommand*{\@astanza@line}{%
1576   \ifnum\value{stanzaindentrepetition}=0
1577     \parindent=\csname sza@\number\stanza@count
1578       @\endcsname\stanzaindentbase
1579   \else
1580     \parindent=\csname sza@\number\stanza@modulo
1581       @\endcsname\stanzaindentbase
1582     \managestanza@modulo
1583   \fi
1584   \par
1585   \stanza@hang%\mbox{}}%
1586   \ignorespaces}
1587

```

Lastly reset the modified category codes.

```

1588 \catcode'\&=\next
1589

```

24 Naming macros

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

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

```

\unhnamebox 1590 \providecommand*\newnamebox}[1]{%
\unvnamebox 1591 \expandafter\newbox\csname #1\endcsname}
\namebox 1592 \providecommand*\setnamebox}[1]{%
1593 \expandafter\setbox\csname #1\endcsname}
1594 \providecommand*\unhnamebox}[1]{%
1595 \expandafter\unhbox\csname #1\endcsname}
1596 \providecommand*\unvnamebox}[1]{%
1597 \expandafter\unvbox\csname #1\endcsname}
1598 \providecommand*\namebox}[1]{%
1599 \csname #1\endcsname}
1600

\newnamecount Macros for creating and using ‘named’ counts.
\usenamecount 1601 \providecommand*\newnamecount}[1]{%
1602 \expandafter\newcount\csname #1\endcsname}
1603 \providecommand*\usenamecount}[1]{%
1604 \csname #1\endcsname}
1605

```

25 Counts and boxes for parallel texts

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

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

```

1606 \newcount\l@dc@maxchunks
1607 \newcommand*\maxchunks}[1]{\l@dc@maxchunks=#1}
1608 \maxchunks{5120}
1609

\l@dnumpsL The numbers of left and right chunks. \l@dnumpsL is defined in eledmac.
\l@dnumpsR 1610 \newcount\l@dnumpsL
1611

\l@psL A couple of scratch counts for use in left and right texts, respectively.
\l@psR 1612 \newcount\l@dpsL
1613 \newcount\l@dpsR
1614

\l@dsetuprawboxes This macro creates \maxchunks pairs of boxes for left and right chunks. The boxes  

are called \l@dLcolrawbox1, \l@dLcolrawbox2, etc.
1615 \newcommand*\l@dsetuprawboxes}{%

```

```

1616 \l@tempcntb=\l@dc@maxchunks
1617 \loop\ifnum\l@tempcntb>\z@
1618   \newnamebox{\l@Lcolrawbox\the\l@tempcntb}
1619   \newnamebox{\l@Rcolrawbox\the\l@tempcntb}
1620   \advance\l@tempcntb \m@ne
1621 \repeat}
1622

```

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

```

1623 \newcommand*{\l@dssetupmaxlinecounts}{%
1624   \l@tempcntb=\l@dc@maxchunks
1625   \loop\ifnum\l@tempcntb>\z@
1626     \newnamecount{\l@dmaxlinesinpar\the\l@tempcntb}
1627     \advance\l@tempcntb \m@ne
1628   \repeat}
1629 \newcommand*{\l@dzeromaxlinecounts}{%
1630   \begingroup
1631   \l@tempcntb=\l@dc@maxchunks
1632   \loop\ifnum\l@tempcntb>\z@
1633     \global\usenamecount{\l@dmaxlinesinpar\the\l@tempcntb}=\z@
1634     \advance\l@tempcntb \m@ne
1635   \repeat
1636   \endgroup}
1637

```

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

```

1638 \AtBeginDocument{%
1639   \l@dssetuprawboxes
1640   \l@dssetupmaxlinecounts
1641   \l@dzeromaxlinecounts
1642   \l@dnumpstartsL=\z@
1643   \l@dnumpstartsR=\z@
1644   \l@dpscL=\z@
1645   \l@dpscR=\z@}
1646

```

26 Fixing babel

With parallel texts there is the possibility that the two sides might use different languages via `babel`. On the other hand, `babel` might not be called at all (even though it might be already built into the format).

With the normal sequential text each line is initially typeset in the current language environment, and then it is output at which time its attachments are typeset (in the same language environment. In the parallel case lines are typeset

in their current language but an attachment might be typeset outside the language environment of its line if the left and right side languages are different. To counter this, we have to make sure that the correct language is used at the proper times.

```

\ifl@dusedbabel A flag for checking if babel has been used as a package.
\l@dusedbabelfalse 1647 \newif\ifl@dusedbabel
\l@dusedbabeltrue 1648 \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 \.
1649 \newcommand*{\l@dbbl@set@language}[1]{%
1650 \edef\language{#1}%
1651 \select@language{\language}%
1652 \if@files
1653 \protected@write\@auxout{}\string\select@language{\language}%
1654 \addtocontents{toc}{\string\select@language{\language}%
1655 \addtocontents{lof}{\string\select@language{\language}%
1656 \addtocontents{lot}{\string\select@language{\language}%
1657 \fi}
1658

```

The rest of the setup has to be postponed until the end of the preamble when we know if `babel` has been used or not. However, for now assume that it has not been used.

```

\selectlanguage \selectlanguage is a babel command. \theledlanguageL and \theledlanguageR
\l@duselanguage are the names of the languages of the left and right texts. \l@duselanguage is
\theledlanguageL similar to \selectlanguage.
\theledlanguageR 1659 \providecommand{\selectlanguage}[1]{
1660 \newcommand*{\l@duselanguage}[1]{
1661 \gdef\theledlanguageL{
1662 \gdef\theledlanguageR{
1663

```

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

```

1664 \AtBeginDocument{%
1665 \ifundefined{xpg@main@language}{%
1666 \ifundefined{bbl@main@language}{%
1667 \l@dusedbabelfalse
1668 \renewcommand*{\selectlanguage}[1]{}}}%

```

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

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.

```

1669   \l@dusedbabeltrue
1670   \let\l@doldselectlanguage\selectlanguage
1671   \let\l@doldbbl@set@language\bbl@set@language
1672   \let\bbl@set@language\l@dbbl@set@language
1673   \renewcommand{\selectlanguage}[1]{%
1674     \l@doldselectlanguage{#1}%
1675     \ifledRcol \gdef\theledlanguageR{#1}%
1676     \else      \gdef\theledlanguageL{#1}%
1677     \fi}

```

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

```

1678   \renewcommand*{\l@duselanguage}[1]{%
1679     \l@doldselectlanguage{#1}}

```

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

```

1680   \gdef\theledlanguageL{\bbl@main@language}%
1681   \gdef\theledlanguageR{\bbl@main@language}%
1682   }%
1683   }

```

If on Polyglossia

```

1684   {   \let\old@otherlanguage\otherlanguage%
1685       \renewcommand{\otherlanguage}[2][]{%
1686         \selectlanguage[#1]{#2}%
1687         \ifledRcol \gdef\theledlanguageR{#2}%
1688         \else      \gdef\theledlanguageL{#2}%
1689         \fi}%
1690       \let\l@duselanguage\select@language%
1691       \gdef\theledlanguageL{\xpg@main@language}%
1692       \gdef\theledlanguageR{\xpg@main@language}%

```

That's it.

```

1693 }}

```

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

```

\@pstartstrue 1694 \newif\if@pstarts
\@pstartsfalse 1695 \newcommand*\check@pstarts{%
\check@pstarts 1696   \@pstartsfalse
1697   \ifnum\l@dnumpstartsL>\l@dpscL
1698     \@pstartstrue
1699   \else
1700     \ifnum\l@dnumpstartsR>\l@dpscR
1701       \@pstartstrue
1702     \fi
1703   \fi
1704 }
1705

```

```

\ifaraw@text \checkraw@text checks whether the current Left or Right box is void or not. If
\araw@texttrue one or other is not void it sets \araw@texttrue, otherwise both are void and it
\araw@textfalse sets \araw@textfalse.
\checkraw@text
1706 \newif\ifaraw@text
1707 \araw@textfalse
1708 \newcommand*\checkraw@text}{%
1709 \araw@textfalse
1710 \ifvbox\namebox{l@dLcolrawbox\the\l@dpscl}
1711 \araw@texttrue
1712 \else
1713 \ifvbox\namebox{l@dRcolrawbox\the\l@dpscl}
1714 \araw@texttrue
1715 \fi
1716 \fi
1717 }
1718

\@writelinesinparL These write the number of text lines in a chunk to the section files, and then
\@writelinesinparR afterwards zero the counter.
1719 \newcommand*\@writelinesinparL}{%
1720 \edef\next{%
1721 \write\linenum@out{\string\@pend[\the\@donereallinesL]}}%
1722 \next
1723 \global\@donereallinesL \z@}
1724 \newcommand*\@writelinesinparR}{%
1725 \edef\next{%
1726 \write\linenum@outR{\string\@pendR[\the\@donereallinesR]}}%
1727 \next
1728 \global\@donereallinesR \z@}
1729

```

27 Parallel columns

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

```

\@eledsectionR
1730 \newsavebox{\@eledsectionL}%
1731 \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.

```

1732 \newcommand*\Columns}{%
1733 \l@dprintingcolumnstrue%
1734 \eledsection@correcting@skip=-\baselineskip% Correction for sections' titles
1735 \ifnum\l@dnumstartsL=\l@dnumstartsR\else
1736 \led@err@BadLeftRightPstarts{\the\l@dnumstartsL}{\the\l@dnumstartsR}%
1737 \fi

```


Start a group and zero counters, etc.

```

1738 \begingroup
1739 \l@dzeropenalties
1740 \endgraf\global\l@num@lines=\prevgraf
1741 \global\l@num@linesR=\prevgraf
1742 \global\l@par@line=\z@
1743 \global\l@par@lineR=\z@
1744 \global\l@dpscL=\z@
1745 \global\l@dpscR=\z@

```

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

```

1746 \check@pstarts
1747 \loop\if@pstarts
1748 \global\l@pstartnumtrue
1749 \global\l@pstartnumRtrue

```

Increment `\l@dpscL` and `\l@dpscR` which here count the numbers of left and right chunks. Also restore the value of the public `pstart` counters.

```

1750 \global\advance\l@dpscL \@ne
1751 \global\advance\l@dpscR \@ne
1752 \restore@pstartL@pc%
1753 \restore@pstartR@pc%

```

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

```

1754 \checkraw@text
1755 { \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.

```

1756 \l@duselanguage{\theledlanguageL}%
1757 \do@lineL
1758 \xifinlist{\the\l@dpscL}{\eled@sections@@}
1759 {%
1760 \ifdefstring{\@eledsectmark}{L}%
1761 {\csuse{eled@sectmark@\the\l@dpscL}%
1762 }{}}%
1763 \global\csundef{eled@sectmark@\the\l@dpscL}%
1764 \savebox{\@eledsectionL}{\parbox[t][t]{\Lcolwidth}{\vbox{\print@eledsectionL}}}%\vb
1765 }%
1766 {}%
1767 \l@duselanguage{\theledlanguageR}%
1768 \do@lineR
1769 \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}
1770 {%
1771 \ifdefstring{\@eledsectmark}{R}%
1772 {\csuse{eled@sectmark@\the\l@dpscR}%
1773 }{}}%
1774 \global\csundef{eled@sectmark@\the\l@dpscR}%
1775 \savebox{\@eledsectionR}{\parbox[t][t]{\Rcolwidth}{\vbox{\print@eledsectionR}}}%\vb

```

```

1776      {}%
1777      \hb@xt@ \hsize{%
1778      \ifdefstring{\columns@position}{L}{\}{\hfill }%
1779      \unhbox\l@dleftbox%
1780      \ifhbox\@eledsectionL%
1781      \usebox{\@eledsectionL}%
1782      \fi%
1783      \print@columnseparator%
1784      \unhbox\l@drightbox%
1785      \ifhbox\@eledsectionR%
1786      \usebox{\@eledsectionR}%
1787      \fi%
1788      \ifdefstring{\columns@position}{R}{\}{\hfill}%
1789      }%
1790      \checkraw@text
1791      \checkverseL
1792      \checkverseR
1793      \checkpb@columns
1794      \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.

```

1795      \@writelinesinparL
1796      \@writelinesinparR
1797      \check@pstarts
1798      \ifbypstart@
1799      \write\linenum@out{\string\@set[1]}
1800      \resetprevline@
1801      \fi
1802      \ifbypstart@R
1803      \write\linenum@outR{\string\@set[1]}
1804      \resetprevline@
1805      \fi
1806      \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.

```

1807      \flush@notes
1808      \flush@notesR
1809      \endgroup
1810      \global\l@dpscL=\z@
1811      \global\l@dpscR=\z@
1812      \global\l@dnumpstartsL=\z@
1813      \global\l@dnumpstartsR=\z@
1814      \l@dprintingcolumnsfalse%
1815      \ignorespaces
1816      \global\instanzaLfalse
1817      \global\instanzaRfalse}
1818

```

`\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).

```

1819 \def\print@columnseparator{%
1820   \ifdim\beforecolumnseparator<0pt%
1821     \hfill%
1822   \else%
1823     \hspace{\beforecolumnseparator}%
1824   \fi%
1825   \columnseparator%
1826   \ifdim\aftercolumnseparator<0pt%
1827     \hfill%
1828   \else%
1829     \hspace{\beforecolumnseparator}%
1830   \fi%
1831 }%
1832 %\end{macrocode}
1833 % \end{macro}
1834 % \begin{macro}{\checkpb@columns}
1835 % \cs{checkpb@columns} prevent or make pagebreaking in columns, depending of the use of \cs{ledpb} or
1836 % \begin{macrocode}
1837
1838 \newcommand{\checkpb@columns}{%
1839   \newif\if@pb
1840   \newif\if@nopb
1841   \IfStrEq{\led@pb@setting}{before}{%
1842     \numdef{\next@absline}{\the\absline@num+1}%
1843     \numdef{\next@abslineR}{\the\absline@numR+1}%
1844     \xifinlistcs{\next@absline}{l@prev@pb}{\@pbtrue}{}%
1845     \xifinlistcs{\next@abslineR}{l@prev@pbR}{\@pbtrue}{%
1846       \xifinlistcs{\next@absline}{l@prev@nopb}{\@nopbtrue}{}%
1847       \xifinlistcs{\next@abslineR}{l@prev@nopbR}{\@nopbtrue}{%
1848         }{}
1849       \IfStrEq{\led@pb@setting}{after}{%
1850         \xifinlistcs{\the\absline@num}{l@prev@pb}{\@pbtrue}{}%
1851         \xifinlistcs{\the\absline@numR}{l@prev@pbR}{\@pbtrue}{%
1852           \xifinlistcs{\the\absline@num}{l@prev@nopb}{\@nopbtrue}{}%
1853           \xifinlistcs{\the\absline@numR}{l@prev@nopbR}{\@nopbtrue}{%
1854             }{}
1855           \if@nopb\nopagebreak[4]\enlargethispage{\baselineskip}\fi
1856           \if@pb\pagebreak[4]\fi
1857         }

```

`\columnseparator` The separator between line pairs in parallel columns is in the form of a vertical rule extending a little below the baseline and with a height slightly greater than the `\baselineskip`. The width of the rule is `\columnrulewidth` (initially 0pt so the rule is invisible).

```

1858 \newcommand*{\columnseparator}{%
1859   \smash{\rule[-0.2\baselineskip]{\columnrulewidth}{1.05\baselineskip}}}

```

```

1860 \newdimen\columnrulewidth
1861 \columnrulewidth=\z@
1862

```

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

```

1863 \newcommand*{\columnspan}[1]{%
1864   \xdef\columns@position{#1}%
1865 }%
1866 \xdef\columns@position{R}%

```

`\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`.

```

1867 \newlength{\beforecolumnseparator}%
1868 \setlength{\beforecolumnseparator}{-2pt}%
1869
1870 \newlength{\aftercolumnseparator}%
1871 \setlength{\aftercolumnseparator}{-2pt}%
1872

```

`setwidthliketwocolumns@L` The `\setwidth...` macros are called in `\beginnumbering` in a **non-parallel** typesetting context, to fix the width of the lines to be vertically aligned with parallel columns. They are also called at the beginning of a note's group, if some options are enabled. The `\setposition...` macros are called in `\beginnumbering` in a **non- parallel** typesetting context to fix the position of the lines. The `\setnoteposition...` macros are called in `\xxxfootstart` in a **non- parallel** typesetting context to fix the position of notes block.

```

1873 \newcommand{\setwidthliketwocolumns@L}{%
1874 % Temporary dimension, initially equal to the standard hsize, i.e. text width
1875 %   \begin{macrocode}
1876 \newdimen\temp%
1877 \temp=\hsize%

```

Hsize : Left + Right width

```

1878 \hsize=\Lcolwidth%
1879 \advance\hsize\Rcolwidth%

```

Now, calculating the remaining space

```

1880 \advance\temp-\hsize%

```

And multiply the hsize by 2/3 of this space

```

1881 \multiply\temp by 2%
1882 \divide\temp by 3%
1883 \advance\hsize\temp%
1884 }%
1885
1886 \newcommand{\setpositionliketwocolumns@L}{%
1887 \renewcommand{\ledrlfill}{\hfill}%
1888 }%

```

```

1889
1890 \newcommand{\setnotespositionliketwocolumns@L}{%
1891 }%
1892
1893
1894 \newcommand{\setwidthliketwocolumns@C}{%
1895 % Temporary dimension, initially equal to the standard hsize, i.e. text width
1896   \newdimen\temp%
1897   \temp=\hsize%
1898 % Hsize : Left + Right width
1899   \hsize=\Lcolwidth%
1900   \advance\hsize\Rcolwidth%
1901 % Now, calculating the remaining space
1902   \advance\temp-\hsize%
1903
1904   And multiply the hsize by 1/2 of this space
1905   \divide\temp by 2%
1906   \advance\hsize\temp%
1907 }%
1908
1909 \newcommand{\setpositionliketwocolumns@C}{%
1910 \doinsidelinehook{\hfill}%
1911 \renewcommand{\ledrlfill}{\hfill}%
1912 }%
1913
1914 \newcommand{\setnotespositionliketwocolumns@C}{%
1915   \newdimen\temp%
1916   \newdimen\tempa%
1917   \temp=\hsize%
1918   \tempa=\Lcolwidth%
1919   \advance\tempa\Rcolwidth%
1920   \advance\temp-\tempa%
1921   \divide\temp by 2%
1922   \leftskip=\temp%
1923   \rightskip=-\temp%
1924 }%
1925
1926 \newcommand{\setwidthliketwocolumns@R}{%
1927 % Temporary dimension, initially equal to the standard hsize, i.e. text width
1928   \newdimen\temp%
1929   \temp=\hsize%
1930
1931   Hsize : Left + Right width
1932   \hsize=\Lcolwidth%
1933   \advance\hsize\Rcolwidth%
1934
1935   Now, calculating the remaining space
1936   \advance\temp-\hsize%

```

And multiply the hsize by 2/3 of this space

```

1930 \multiply\temp by 2%
1931 \divide\temp by 3%
1932 \advance\hsize\temp%
1933 }%
1934
1935 \newcommand{\setpositionliketwocolumns@R}{%
1936 \doinsidelinehook{\hfill}%
1937 }%
1938
1939 \newcommand{\setnotespositionliketwocolumns@R}{%
1940 \newdimen\temp%
1941 \newdimen\tempa%
1942 \temp=\hsize%
1943 \tempa=Lcolwidth%
1944 \advance\tempa\Rcolwidth%
1945 \advance\temp-\tempa%
1946 \divide\temp by 2%
1947 \leftskip=\temp%
1948 \rightskip=-\temp%
1949 }%
1950
```

28 Parallel pages

This is considerably more complicated than parallel columns.

`\numpagelinesL` Counts for the number of lines on a left or right page, and the smaller of the
`\numpagelinesR` number of lines on a pair of facing pages.
`\l@dminpagelines` 1951 `\newcount\numpagelinesL`
 1952 `\newcount\numpagelinesR`
 1953 `\newcount\l@dminpagelines`
 1954

`\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.

```

1955 \newcommand*{\Pages}{%
1956 \l@dprintingpagetrue%
1957 \eledsection@correcting@skip=-2\baselineskip% line correcting for section titles.
1958 \parledgroup@notespacing@set@correction
1959 \typeout{}
1960 \typeout{***** PAGES *****}
1961 \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
1962 \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
1963 \fi
```

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

```

1964 \cleartol@devenpage
1965 \begingroup
1966 \l@dzeroopenalties
1967 \endgraf\global\num@lines=\prevgraf
1968 \global\num@linesR=\prevgraf
1969 \global\par@line=\z@
1970 \global\par@lineR=\z@
1971 \global\l@dpscL=\z@
1972 \global\l@dpscR=\z@
1973 \writtenlinesLfalse
1974 \writtenlinesRfalse

```

Check if there are chunks to be processed.

```

1975 \check@pstarts
1976 \loop\if@pstarts

```

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

```

1977 \global\advance\l@dpscL \@ne
1978 \global\advance\l@dpscR \@ne

```

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

```

1979 \getlinesfromparlistL
1980 \getlinesfromparlistR
1981 \l@dcalc@maxoftwo{\@cs@linesinparL}{\@cs@linesinparR}%
1982 {\usernamecount{l@dmxlinesinpar\the\l@dpscL}}%
1983 \check@pstarts
1984 \repeat

```

Zero the counts again, ready for the next bit.

```

1985 \global\l@dpscL=\z@
1986 \global\l@dpscR=\z@

```

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

```

1987 \getlinesfrompagelistL
1988 \getlinesfrompagelistR
1989 \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
1990 {\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.

```

1991 \check@pstarts
1992 \if@pstarts

```

Increment the chunk counts to get the first pair. Restore also the value of public pstart counters.

```

1993 \global\advance\l@dpscL \@ne
1994 \global\advance\l@dpscR \@ne
1995 \restore@pstartL@pc%
1996 \restore@pstartR@pc%

```

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

```
1997      \global\@donereallinesL=\z@
1998      \global\@donetotallinesL=\z@
1999      \global\@donereallinesR=\z@
2000      \global\@donetotallinesR=\z@
```

Start a loop over the boxes (chunks).

```
2001      \checkraw@text
2002 %      \begingroup
2003 {      \loop\ifaraw@text
```

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

```
2004      \checkpageL
2005      \l@duselanguage{\theledlanguageL}%
2006 %%%      \begingroup
2007 {      \loop\ifl@dsamepage%
```

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

```
2008      \ifdefstring{\@eledsectnotoc}{L}{\ledsectnotoc}{%
2009      \csuse{before@pstartL@the\l@dpscl}%
2010      \global\csundef{before@pstartL@the\l@dpscl}
2011      \do@lineL
2012      \xifinlist{\the\l@dpscl}{\eled@sections@}%
2013      {\print@eledsectionL}%
2014      {}%
2015      \advance\numpagelinesL \@ne
2016      \ifshiftedpstarts
2017          \ifdim\ht\l@dleftbox>0pt\hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
2018      \else%
2019          \parledgroup@correction@notespacing{L}
2020          \hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
2021      \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. Check if we have to print the optional argument of the last pend. Check if the page is full. Check if the verse is split in two subsequent pages. Check there is any forced page breaks.

```
2022      \get@nextboxL%
2023      \ifprint@last@after@pendL%
2024      \csuse{after@pendL@the\l@dpscl}%
2025      \global\csundef{after@pendL@the\l@dpscl}%
2026      \fi%
2027      \checkpageL%
2028      \checkverseL
2029      \checkpbl
2030      \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.

```

2031      \ifl@dpagfull
2032      \@writelinesonpageL{\the\numpagelinesL}%
2033      \else
2034      \@writelinesonpageL{1000}%
2035      \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.

```

2036      \numpagelinesL \z@
2037      \parledgroup@correction@notespacing@init
2038      \clearl@dleftpage }%

```

Now do the same for the right text.

```

2039      \checkpageR%
2040      \l@duselanguage{\theledlanguageR}%
2041 {
2042      \loop\ifl@dsamepage%
2043      \initnumbering@sectcountR
2044      \ifdefstring{\@eledsectnotoc}{R}{\ledsectnotoc}{%
2045      \csuse{before@pstartR@the\l@dpscR}%
2046      \global\csundef{before@pstartR@the\l@dpscR}
2047      \do@lineR
2048      \xifinlist{\the\l@dpscR}{\eled@sectionsR@@@}
2049      {\print@eledsectionR}%
2050      {}%
2051      \advance\numpagelinesR \@ne
2052      \ifshiftedpstarts
2053      \ifdim\ht\l@drightbox>0pt\hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}\fi%
2054      \else%
2055      \parledgroup@correction@notespacing{R}
2056      \hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}%
2057      \fi
2058      \get@nextboxR%
2059      \ifprint@last@after@pendR%
2060      \csuse{after@pendR@the\l@dpscR}%
2061      \global\csundef{after@pendR@the\l@dpscR}%
2062      \fi%
2063      \checkpageR%
2064      \checkverseR
2065      \checkpbR
2066      \repeat
2067      \ifl@dpagfull
2068      \@writelinesonpageR{\the\numpagelinesR}%
2069      \else
2070      \@writelinesonpageR{1000}%
2071      \fi

```

```

2071      \numpagelinesR=\z@
2072      \parledgroup@correction@notespacing@init
    The page is full, so move onto the next (left, odd) page and repeat left text
    processing.
2073      \clearl@drihtpage}
    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.
2074      \checkraw@text
2075      \ifaraw@text
2076      \getlinesfrompagelistL
2077      \getlinesfrompagelistR
2078      \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
2079                      {\l@dminpagelines}%
2080      \fi
2081      \repeat}
    We have now output the text from all the chunks.
2082      \fi
    Make sure that there are no inserts hanging around.
2083      \flush@notes
2084      \flush@notesR
2085      \endgroup
    Zero counts ready for the next set of left/right text chunks. The boolean tests for
    stanza are switched to false.
2086      \global\l@dpscL=\z@
2087      \global\l@dpscR=\z@
2088      \global\l@dnumstartsL=\z@
2089      \global\l@dnumstartsR=\z@
2090      \global\instanzaLfalse
2091      \global\instanzaRfalse
2092      \l@dprintingpagesfalse%
2093      \ignorespaces}
2094
\ledstrutL  Struts inserted into leftand right text lines.
\ledstrutR 2095 \newcommand*{\ledstrutL}{\strut}
2096 \newcommand*{\ledstrutR}{\strut}
2097

\cleartoevenpage \cleartoevenpage, which is defined in the memoir class, is like \clear(double)page
\cleartol@devenpage except that we end up on an even page. \cleartol@devenpage is similar except
\clearl@dleftpage that it first checks to see if it is already on an empty page. \clearl@dleftpage
\clearl@drihtpage and \clearl@drihtpage get us onto an odd and even page, respectively, checking
that we end up on the immediately next page.
2098 \providecommand{\cleartoevenpage}[1][\@empty]{%
2099   \clearpage
2100   \ifodd\c@page\hbox{ }#1\clearpage\fi}

```

```

2101 \newcommand*{\cleartol@devenpage}{%
2102   \ifdim\pagetotal<\topskip% on an empty page
2103   \else
2104     \clearpage
2105   \fi
2106   \ifodd\c@page\hbox{}\clearpage\fi}
2107 \newcommand*{\clearl@dleftpage}{%
2108   \clearpage
2109   \ifodd\c@page\else
2110     \led@err@LeftOnRightPage
2111     \hbox{}%
2112     \cleardoublepage
2113   \fi}
2114 \newcommand*{\clearl@drightpage}{%
2115   \clearpage
2116   \ifodd\c@page
2117     \led@err@RightOnLeftPage
2118     \hbox{}%
2119     \cleartoevenpage
2120   \fi}
2121

```

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

```

\cs@linesinparR 2122 \newcommand*{\getlinesfromparlistL}{%
2123   \ifx\linesinpar@listL\empty
2124     \gdef\cs@linesinparL{0}%
2125   \else
2126     \gl@p\linesinpar@listL\to\cs@linesinparL
2127   \fi}
2128 \newcommand*{\getlinesfromparlistR}{%
2129   \ifx\linesinpar@listR\empty
2130     \gdef\cs@linesinparR{0}%
2131   \else
2132     \gl@p\linesinpar@listR\to\cs@linesinparR
2133   \fi}
2134

```

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

```

\cs@linesonpageR 2135 \newcommand*{\getlinesfrompagelistL}{%
2136   \ifx\linesonpage@listL\empty
2137     \gdef\cs@linesonpageL{1000}%
2138   \else
2139     \gl@p\linesonpage@listL\to\cs@linesonpageL
2140   \fi}
2141 \newcommand*{\getlinesfrompagelistR}{%

```

```

2142 \ifx\linesonpage@listR\empty
2143   \gdef\@cs@linesonpageR{1000}%
2144 \else
2145   \gl@p\linesonpage@listR\to\@cs@linesonpageR
2146 \fi}
2147

```

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

```

2148 \newcommand*{\@writelinesonpageL}[1]{%
2149   \edef\next{\write\linenum@out{\string\@lopL{#1}}}%
2150   \next}
2151 \newcommand*{\@writelinesonpageR}[1]{%
2152   \edef\next{\write\linenum@outR{\string\@lopR{#1}}}%
2153   \next}
2154

```

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

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

```

2155 \newcommand*{\l@dcalc@maxoftwo}[3]{%
2156   \ifnum #2>#1\relax
2157     #3=#2\relax
2158   \else
2159     #3=#1\relax
2160   \fi}
2161 \newcommand*{\l@dcalc@minoftwo}[3]{%
2162   \ifnum #2<#1\relax
2163     #3=#2\relax
2164   \else
2165     #3=#1\relax
2166   \fi}
2167

```

`\ifl@dsamepage` `\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@dpagfull` is set FALSE and `\l@dsamepagetrue` is set TRUE. If the page is spatially full then `\ifl@dpagfull` 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@dpagfull` and `\l@dpagfulltrue` are set FALSE.

```

\checkpageL 2168 \newif\ifl@dsamepage
\checkpageR 2169   \l@dsamepagetrue
              2170 \newif\ifl@dpagfull
              2171
2172 \newcommand*{\checkpageL}{%
2173   \l@dpagfulltrue
2174   \l@dsamepagetrue

```

```

2175 \check@goal
2176 \ifdim\pagetotal<\ledthegoal
2177   \ifnum\numpagelinesL<\l@dminpagelines
2178     \else
2179       \l@dsamepagefalse
2180       \l@dpagfullfalse
2181     \fi
2182   \else
2183     \l@dsamepagefalse
2184     \l@dpagfulltrue
2185   \fi%
2186 \ifprint@last@after@pendL%
2187   \l@dpagfullfalse%
2188   \l@dsamepagefalse%
2189   \print@last@after@pendLfalse%
2190 \fi%
2191 }%
2192
2193 \newcommand*{\checkpageR}{%
2194   \l@dpagfulltrue
2195   \l@dsamepagetrue
2196   \check@goal
2197   \ifdim\pagetotal<\ledthegoal
2198     \ifnum\numpagelinesR<\l@dminpagelines
2199       \else
2200         \l@dsamepagefalse
2201         \l@dpagfullfalse
2202       \fi
2203     \else
2204       \l@dsamepagefalse
2205       \l@dpagfulltrue
2206     \fi%
2207 \ifprint@last@after@pendR%
2208   \l@dpagfullfalse%
2209   \l@dsamepagefalse%
2210   \print@last@after@pendRfalse%
2211 \fi%
2212 }%
2213

```

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

```

2214 \newcommand{\checkpbL}{
2215   \IfStrEq{\led@pb@setting}{after}{
2216     \xifinlistcs{\the\absline@num}{\l@prev@pb}{\l@dpagfulltrue\l@dsamepagefalse}{
2217       \xifinlistcs{\the\absline@num}{\l@prev@nopb}{\l@dpagfullfalse\l@dsamepagetrue}{
2218     }}
2219   \IfStrEq{\led@pb@setting}{before}{
2220     \numdef{\next@absline}{\the\absline@num+1}

```

```

2221 \xifinlistcs{\next@absline}{\l@prev@pb}{\l@dpagfulltrue\l@dsamepagefalse}{}
2222 \xifinlistcs{\next@absline}{\l@prev@nopb}{\l@dpagfullfalse\l@dsamepagetrue}{}
2223 }{}
2224 }
2225
2226 \newcommand{\checkpbR}{
2227 \IfStrEq{\led@pb@setting}{after}{
2228 \xifinlistcs{\the\absline@numR}{\l@prev@pbR}{\l@dpagfulltrue\l@dsamepagefalse}{}
2229 \xifinlistcs{\the\absline@numR}{\l@prev@nopbR}{\l@dpagfullfalse\l@dsamepagetrue}{}
2230 }{}
2231 \IfStrEq{\led@pb@setting}{before}{
2232 \numdef{\next@abslineR}{\the\absline@numR+1}
2233 \xifinlistcs{\next@abslineR}{\l@prev@pbR}{\l@dpagfulltrue\l@dsamepagefalse}{}
2234 \xifinlistcs{\next@abslineR}{\l@prev@nopbR}{\l@dpagfullfalse\l@dsamepagetrue}{}
2235 }{}
2236 }

```

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

```

2237 \newcommand{\checkverseL}{
2238 \ifinstanzaL
2239 \iflednopbinverse
2240 \ifinserthangingsymbol
2241 \numgdef{\prev@abslineverse}{\the\absline@num-1}
2242 \IfStrEq{\led@pb@setting}{after}{\lednopbnum{\prev@abslineverse}}{}
2243 \IfStrEq{\led@pb@setting}{before}{\ifnum\numpagelinesL<3\ledpbnum{\prev@abslineverse}}{}
2244 \fi
2245 \fi
2246 \fi
2247 }
2248 \newcommand{\checkverseR}{
2249 \ifinstanzaR
2250 \iflednopbinverse
2251 \ifinserthangingsymbolR
2252 \numgdef{\prev@abslineverse}{\the\absline@numR-1}
2253 \IfStrEq{\led@pb@setting}{after}{\lednopbnumR{\prev@abslineverse}}{}
2254 \IfStrEq{\led@pb@setting}{before}{\ifnum\numpagelinesR<3\ledpbnumR{\prev@abslineverse}}{}
2255 \fi
2256 \fi
2257 \fi
2258 }

```

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

```

2259 \newdimen\ledthegoal
2260 \ifshiftedpstarts
2261 \newcommand*{\goalfraction}{0.95}
2262 \else

```

```

2263      \newcommand*{\goalfraction}{0.9}
2264 \fi
2265
2266 \newcommand*{\check@goal}{%
2267   \ledthegoal=\goalfraction\pagegoal}
2268

```

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

```

\ifwrittenlinesL 2269 \newif\ifwrittenlinesL
                  2270 \newif\ifwrittenlinesR
                  2271

```

`\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.

```

2272 \newcommand*{\get@nextboxL}{%
2273   \ifvbox\namebox{1@dLcolrawbox\the\l@dpscL}% box is not empty

   The current box is not empty; do nothing.
2274   \else%                                box is empty

   The box is empty. Check if enough lines (real and blank) have been output.
2275   \ifnum\usenamecount{1@dmaxlinesinpar\the\l@dpscL}>\@donetotallinesL
2276     \parledgroup@notes@endL
2277   \else

   Sufficient lines have been output.
2278   \ifnum\usenamecount{1@dmaxlinesinpar\the\l@dpscL}=\@donetotallinesL
2279     \parledgroup@notes@endL
2280   \fi
2281   \ifwrittenlinesL\else

   Write out the number of lines done, and set the boolean so this is only done once.
2282   \@writelinesinparL
2283   \writtenlinesLtrue
2284   \fi
2285   \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.

```

2286   \writtenlinesLfalse
2287   \ifbypstart@
2288     \global\line@num=0%
2289     \resetprevline@%
2290   \fi
2291 % Add the content of the optional argument of the previous \cs{pend}.
2292 %   \begin{macrocode}
2293     \csuse{after@pendL@\the\l@dpscL}%
2294     \global\csundef{after@pendL@\the\l@dpscL}%
2295 %   \end{macrocode}
2296 % Check the number of lines

```

```

2297 %      \begin{macrocode}
2298         \l@dcalc@maxoftwo{\the\usernamecount{1@dmaxlinesinpar\the\l@dpscL}}%
2299                     {\the\@donetotallinesL}%
2300                     {\usernamecount{1@dmaxlinesinpar\the\l@dpscL}}%
2301         \global\@donetotallinesL \z@
2302 %      \end{macrocode}
2303 % Go to the next pstart
2304 %      \begin{macrocode}
2305         \global\advance\l@dpscL \@ne
2306         \global\pstartnumtrue%
2307         \restore@pstartL@pc%
2308
2309         Add notes of parallel ledgroup.
2310
2311         \parledgroup@notes@endL
2312         \parledgroup@correction@notes@spacing@final{L}
2313         \else
2314 % Add the content of the optional argument of the last \cs{pend}.
2315 %      \begin{macrocode}
2316         \print@last@after@pendLtrue%
2317         \end{macrocode}
2318         \begin{macrocode}
2319         \fi
2320         \fi}
2321
2322 \newcommand*{\get@nextboxR}{%
2323   \ifvbox\namebox{1@dRcolrawbox\the\l@dpscR}% box is not empty
2324   \else% box is empty
2325     \ifnum\usernamecount{1@dmaxlinesinpar\the\l@dpscR}>\@donetotallinesR
2326       \parledgroup@notes@endR
2327     \else
2328       \ifnum\usernamecount{1@dmaxlinesinpar\the\l@dpscR}=\@donetotallinesR
2329         \parledgroup@notes@endR
2330       \fi
2331       \ifwrittenlinesR\else
2332         \@writelinesinparR
2333         \writtenlinesRtrue
2334       \fi
2335       \ifnum\l@dnumpstartsR>\l@dpscR
2336         \writtenlinesRfalse
2337         \ifbypstart@R
2338           \global\line@numR=0%
2339           \resetprevline@%
2340         \fi
2341         \csuse{after@pendR@the\l@dpscR}%
2342         \global\csundef{after@pendR@the\l@dpscR}%
2343         \l@dcalc@maxoftwo{\the\usernamecount{1@dmaxlinesinpar\the\l@dpscR}}%
2344                     {\the\@donetotallinesR}%
2345                     {\usernamecount{1@dmaxlinesinpar\the\l@dpscR}}%
2346         \global\@donetotallinesR \z@
2347         \global\advance\l@dpscR \@ne

```



```

2345      \global\pstartnumRtrue%
2346      \restore@pstartR@pc%
2347      \parledgroup@notes@endR
2348      \parledgroup@correction@notespadding@final{R}
2349      \else
2350      \print@last@after@pendRtrue%
2351      \fi
2352      \fi
2353      \fi}
2354

```

29 Sections' titles' commands

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

```

2355 \newcommand{\eledsectnotoc}[1]{\xdef\@eledsectnotoc{#1}}
2356 \eledsectnotoc{R}

```

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

```

2357 \newcommand{\eledsectmark}[1]{\xdef\@eledsectmark{#1}}
2358 \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`.

```

2359 \newskip\eledsection@correcting@skip

```

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

```

2360 \newwrite\eled@sectioningR@out

```

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

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

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

```

2361 \def\l@prev@pbR{}
2362 \def\l@prev@nopbR{}

```

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

the call to `\led@nopbR` in line-list file. The `\lednopbnumR` macro writes the call to `\led@nopbnumR` in line-list file.

```
2363 \newcommand{\ledpbR}{\write\linenum@outR{\string\led@pbR}}
2364 \newcommand{\ledpbnumR}[1]{\write\linenum@outR{\string\led@pbnumR{#1}}}
2365 \newcommand{\lednopbR}{\write\linenum@outR{\string\led@nopbR}}
2366 \newcommand{\lednopbnumR}[1]{\write\linenum@outR{\string\led@nopbnumR{#1}}}
```

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

```
2367 \newcommand{\led@pbR}{\listxadd{\l@prev@pbR}{\the\absline@numR}}
2368 \newcommand{\led@pbnumR}[1]{\listxadd{\l@prev@pbR}{#1}}
2369 \newcommand{\led@nopbR}{\listxadd{\l@prev@nopbR}{\the\absline@numR}}
2370 \newcommand{\led@nopbnumR}[1]{\listxadd{\l@prev@nopbR}{#1}}
```

31 Parallel ledgroup

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

```
2371 \newmarks\parledgroup@
2372 \newmarks\parledgroup@series
2373 \newmarks\parledgroup@type
```

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

```
2374 \newcommand{\parledgroup@notes@startL}{%
2375   \ifnum\usenamecount{\l@maxlinesinpar\the\l@dpscL}>0%
2376     \IfStrEq{\splitfirstmarks\parledgroup@type}{footnoteX}{\csuse{bhooknoteX@splitfirstmarks}}
2377     \IfStrEq{\splitfirstmarks\parledgroup@type}{Xfootnote}{\csuse{bhookXnote@splitfirstmarks}}
2378   \fi%
2379   \global\ledgroupnotesL@true%
2380   \insert@noterule@ledgroup{L}%
2381 }
2382 \newcommand{\parledgroup@notes@startR}{%
2383   \ifnum\usenamecount{\l@maxlinesinpar\the\l@dpscR}>0%
2384     \IfStrEq{\splitfirstmarks\parledgroup@type}{footnoteX}{\csuse{bhooknoteX@splitfirstmarks}}
2385     \IfStrEq{\splitfirstmarks\parledgroup@type}{Xfootnote}{\csuse{bhookXnote@splitfirstmarks}}
2386   \fi%
2387   \global\ledgroupnotesR@true%
2388   \insert@noterule@ledgroup{R}%
2389 }
```

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

```

2390 \newcommand{\parledgroup@notes@endL}{\%
2391   \global\ledgroupnotesL=false%
2392 }
2393 \newcommand{\parledgroup@notes@endR}{\%
2394   \global\ledgroupnotesR=false%
2395 }

```

`\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`

```

2396 \newcommand{\insert@noterule@ledgroup}[1]{
2397   \IfStrEq{\splitbotmarks\parledgroup@}{begin}{\%
2398     \IfStrEq{\splitbotmarks\parledgroup@type}{Xfootnote}{
2399       \csuse{ifledgroupnotes#1@}
2400       \vskip\skip\csuse{mp\splitbotmarks\parledgroup@series footins}
2401       \csuse{\splitbotmarks\parledgroup@series footnoterule}
2402       \fi
2403     }
2404     {}
2405     \IfStrEq{\splitbotmarks\parledgroup@type}{footnoteX}{
2406       \csuse{ifledgroupnotes#1@}
2407       \vskip\skip\csuse{mpfootins\splitbotmarks\parledgroup@series}
2408       \csuse{footnoterule\splitbotmarks\parledgroup@series}
2409       \fi
2410     }{}
2411   }
2412   {}
2413 }

```

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

```

2414 \newcommand{\parledgroupnotespacing}{}

```

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

```

2415 \dimdef{\parledgroup@notespacing@correction}{0pt}
2416 \newcommand{\parledgroup@notespacing@set@correction}{\%
2417   {\notefontsetup\parledgroupnotespacing\dimgdef{\temp@spacing}{\baselineskip}}\%
2418   \dimgdef{\parledgroup@notespacing@correction}{\baselineskip-\temp@spacing}\%
2419 }

```

`rrrection@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.

```

2420 \newcommand{\parledgroup@correction@notespacing@init}{
2421   \dimdef{\parledgroup@notespacing@correction@accumulated}{0pt}
2422   \dimdef{\parledgroup@notespacing@correction@modulo}{0pt}
2423 }

```

2424 \parledgroup@correction@notespacing@init

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

```

2425 \newcommand{\parledgroup@correction@notespacing@final}[1]{
2426   \ifparledgroup
2427   \vspace{\parledgroup@notespacing@correction@accumulated}
2428   \parledgroup@correction@notespacing@init%
2429   \ifstrequal{#1}{L}{
2430     \numdef{\@checking}{\the\l@dpscL-1}
2431   }{
2432     \numdef{\@checking}{\the\l@dpscR-1}
2433   }
2434   \dimdef{\@beforenotes@current@diff}{\csuse{@parledgroup@beforenotes@\@checking L}-\cs
2435   \ifstrequal{#1}{L}%
2436   {% Left
2437     \ifdimgreater{\@beforenotes@current@diff}{0pt}{\vspace{-\@beforenotes@current@diff}
2438   }%
2439   {% Right
2440     \ifdimgreater{\@beforenotes@current@diff}{0pt}{\vspace{\@beforenotes@current@diff}}
2441   }%
2442   \fi
2443 }
```

\parledgroup@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.

```

2444 {}
2445 \newcommand{\parledgroup@correction@notespacing}[1]{%
2446   \csuse{ifledgroupnotes#1@}%
2447   \vspace{-\parledgroup@notespacing@correction}%
2448   \dimdef{\parledgroup@notespacing@correction@accumulated}{\parledgroup@notespacing@co
2449   \dimdef{\parledgroup@notespacing@correction@modulo}{\parledgroup@notespacing@correct
2450   \ifdimless{\parledgroup@notespacing@correction@modulo}{\baselineskip}{\advance\nun
2451   \dimdef{\parledgroup@notespacing@correction@modulo}{\parledgroup@notespacing@correct
2452   }% mean greater than equal
```

```

2453     \fi%
2454 }

```

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

```

2455 \dimdef\parledgroup@beforenotesL{Opt}
2456 \dimdef\parledgroup@beforenotesR{Opt}

```

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

```

2457 \newcommand{\parledgroup@beforenotes@save}[1]{
2458   \ifparledgroup
2459     \csdimgdef{@parledgroup@beforenotes@the\csuse{1@dnumstarts#1}#1}{\csuse{parledgroup@beforenotes
2460     \csdimgdef{parledgroup@beforenotes#1}{Opt}
2461   \fi
2462 }

```

32 The End

i/code_i

Appendix A Some things to do when changing version

Appendix A.1 Migration to eledpar 1.4.3

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

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

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

See the two following examples:

With standard `\hangingsymbol`:

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

With the modification of `\hangingsymbol`:

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

References

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

Index

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

Symbols	<code>\@adv</code>	<u>370</u> , 630, 631
<code>\&</code>	1545, 1546, 1550, 1567, 1588	<code>\@afterindentfalse</code> 737
<code>\@M</code>	1556	<code>\@arabic</code> 222, 223, 785, 787

- \@astanza@line 1566, 1572, 1575
 - \@auxout 1653
 - \@beforenotes@current@diff 2434, 2437, 2440
 - \@chapter 738
 - \@checking 2430, 2432, 2434
 - \@cs@linesinparL 1981, 2122
 - \@cs@linesinparR 1981, 2122
 - \@cs@linesonpageL 1989, 2078, 2135
 - \@cs@linesonpageR 1989, 2078, 2135
 - \@currentlabel 831, 878
 - \@donereallinesL 939, 969, 1721, 1723, 1997
 - \@donereallinesR 939, 1056, 1726, 1728, 1999
 - \@donetotallinesL 939, 970, 973, 1998, 2275, 2278, 2299, 2301
 - \@donetotallinesR 939, 1057, 1060, 2000, 2322, 2325, 2341, 2343
 - \@eled@sectioningfalse 1007
 - \@eled@sectioningtrue 998
 - \@eledsectionL 1730, 1764, 1780, 1781
 - \@eledsectionR 1730, 1775, 1785, 1786
 - \@eledsectmark 995, 1760, 1771, 2357
 - \@eledsectnotoc 994, 2008, 2043, 2355
 - \@gobble 559
 - \@gobbletwo 564
 - \@insertR 1362–1364, 1377–1379
 - \@l@dttempcnta 443, 445, 447, 448, 452, 454, 456, 457, 1114, 1153, 1154, 1156, 1158, 1161, 1162, 1179–1183, 1185, 1192, 1197, 1201, 1209, 1214, 1218, 1251, 1254, 1256, 1260
 - \@l@dttempcntb 178, 180, 182, 1144, 1145, 1192, 1197, 1201, 1209, 1214, 1218, 1243, 1247, 1260, 1268–1270, 1272, 1293–1295, 1297, 1314–1316, 1318, 1420, 1421, 1449–1451, 1453, 1616–1620, 1624–1627, 1631–1634
 - \@lab 563, 1407
 - \@lock 957, 1094
 - \@lockR 56, 314, 316, 318, 331, 477, 493, 494, 496, 497, 525, 526, 528, 1043, 1076, 1120, 1122, 1123, 1125, 1206, 1223, 1225, 1227
 - \@lopL 588, 2149
 - \@lopR 559, 588, 2152
 - \@nl 292, 613, 615
 - \@nl@reg 341
 - \@nl@regR 292
 - \@nobreakfalse 793, 841, 997
 - \@nobreaktrue 791, 795, 839, 843, 997
 - \@noprtrue 1846, 1847, 1852, 1853
 - \@oldnobreak 791, 793, 839, 841, 896, 918
 - \@pbtrue 1844, 1845, 1850, 1851
 - \@pend 579, 1721
 - \@pendR 579, 1726
 - \@pstartfalse 1694
 - \@pstarttrue 1694
 - \@ref 549
 - \@ref@reg 577
 - \@schapter 738
 - \@set 402, 637, 638, 1799, 1803
 - \@startstanza 750, 751, 773, 774
 - \@sw 564
 - \@tag 1474, 1476, 1480
 - \@temp 700, 701, 706, 707
 - \@templ@d 1439, 1441
 - \@templ@n 1440, 1441
 - \@writelinesinparL 1719, 1795, 2282
 - \@writelinesinparR 1719, 1796, 2329
 - \@writelinesonpageL 2032, 2034, 2148
 - \@writelinesonpageR 2067, 2069, 2148
 - \@xloop 1375
- A**
- \absline@num 366, 436, 450, 469, 1085, 1842, 1850, 1852, 2216, 2217, 2220, 2241
 - \absline@numR 51, 239, 294, 297, 300, 433, 441, 462, 481, 515, 543, 554, 1067, 1106, 1107, 1144, 1361, 1843, 1851, 1853, 2228, 2229, 2232, 2252, 2367, 2369
 - \actionlines@list 284, 287, 436, 450, 469
 - \actionlines@listR 243, 260, 276, 279, 433, 441, 462, 481, 515, 543, 1166, 1169
 - \actions@list 288, 437, 457, 471, 473
 - \actions@listR 243, 261, 280, 434, 448, 464, 466, 483, 492, 517, 524, 544, 1170
 - \add@inserts 966, 980
 - \add@inserts@nextR 1350
 - \add@insertsR 1052, 1350
 - \add@penaltiesL 968, 1371
 - \add@penaltiesR 1055, 1371

- \addtocontents 1654–1656
 - \addtocounter 899, 921, 993
 - \advanceline 629, 660
 - \affixline@num 964
 - \affixline@numR 1050, 1176
 - \affixpstart@numL 978, 1283
 - \affixpstart@numR 1283
 - \affixside@note 966, 980
 - \affixside@noteR 1052, 1426
 - \aftercolumnseparator . . . 4, 1826, 1867
 - \appto 1432, 1433
 - \araw@textfalse 1706
 - \araw@texttrue 1706
 - astanza (environment) 8, 1548
 - \at@begin@pairs 714, 719, 720
 - \at@every@pend 903, 925
 - \at@every@pstart 834, 881
 - \AtBeginDocument . . . 1403, 1638, 1664
 - \AtBeginPairs 4, 719
- B**
- \ballast@count 1104, 1109
 - \bbl@main@language 1680, 1681
 - \bbl@set@language 1671, 1672
 - \beforecolumnseparator
..... 4, 1820, 1823, 1829, 1867
 - \beginnumbering 7, 761, 803
 - \beginnumberingR . . . 42, 123, 761, 851
 - \bfseries 785, 787
 - \bypage@Rfalse 143, 158, 163
 - \bypage@Rtrue 143, 153
 - \bypstart@Rfalse 143, 154, 164
 - \bypstart@Rtrue 143, 159
- C**
- \c@ballast 1109
 - \c@chapter 104
 - \c@chapterR 104
 - \c@firstlinenumR 187, 1249
 - \c@firstsublinenumR 191, 1244
 - \c@linenumincrementR 187, 1249
 - \c@page 613, 615, 2100, 2106, 2109, 2116
 - \c@pstartL 689, 701, 785
 - \c@pstartR 694, 707, 787
 - \c@section 105
 - \c@sectionR 105
 - \c@sublinenumincrementR . . 191, 1244
 - \c@subsection 106
 - \c@subsectionR 106
 - \c@subsubsection 107
 - \c@subsubsectionR 107
 - \ch@ck@l@ckR 1176
 - \ch@cksub@l@ckR 1176
 - \ch@cksub@lockR 1245
 - \chapter 723, 724, 733
 - \chapterinpages 710, 724, 735
 - \chardef 1545
 - \check@goal 2175, 2196, 2259
 - \check@pstarts
..... 1694, 1746, 1797, 1975, 1983, 1991
 - \checkpageL 2004, 2027, 2168
 - \checkpageR 2039, 2062, 2168
 - \checkpb@columns . . . 1793, 1834, 1838
 - \checkpbL 2029, 2214
 - \checkpbR 2064, 2214
 - \checkraw@text
..... 1706, 1754, 1790, 2001, 2074
 - \checkverseL 1791, 2028, 2237
 - \checkverseR 1792, 2063, 2237
 - \cleardoublepage 2112
 - \clearl@dleftpage 2038, 2098
 - \clearl@drighthpage 2073, 2098
 - \cleartoevenpage 2098
 - \cleartol@devenpage 1964, 2098
 - \closeout 95, 601, 608
 - \columnrulewidth 4, 1858
 - \Columns 3, 1732
 - \columns@position . . 1778, 1788, 1863
 - \columnseparator 4, 1825, 1858
 - \columnspostion 4, 1863
 - \correcthangingL 989, 1526
 - \correcthangingR 1526
 - \countLline 934, 946
 - \countRline 934, 1032
 - \critext 665
 - \cs 1835, 2291, 2311
 - \csdingdef 2459, 2460
 - \csgdef 834,
835, 881, 882, 903, 904, 925, 926
 - \cslet 690, 695
 - \csundef 1008, 1763, 1774,
2010, 2025, 2045, 2060, 2294, 2339
 - \csuse 1005,
1495, 1506, 1761, 1772, 2009,
2024, 2044, 2059, 2293, 2338,
2376, 2377, 2384, 2385, 2399–
2401, 2406–2408, 2434, 2446, 2459
- D**
- \DeclareOption 8–11

- \def@tempb 161
 - \dimdef 2415, 2421, 2422,
2434, 2448, 2449, 2451, 2455, 2456
 - \dimen 616, 617, 621–623, 627
 - \dingdef 2417, 2418
 - \divide 1181, 1882, 1903, 1919, 1931, 1946
 - \do@actions 1086
 - \do@actions@fixedcodeR 1113
 - \do@actions@nextR 1113
 - \do@actionsR 1068, 1113
 - \do@ballast 1087
 - \do@ballastR 1069, 1104
 - \do@insidelineLhook .. 983, 1021, 1024
 - \do@insidelineRhook 1022, 1024
 - \do@lineL 944, 1757, 2011
 - \do@lineLhook 950, 1019, 1024
 - \do@lineR 1029, 1768, 2046
 - \do@lineRhook 1020, 1024, 1036
 - \do@lockoff 512
 - \do@lockoffL 536
 - \do@lockoffR 512
 - \do@lockon 477
 - \do@lockonL 509
 - \do@lockonR 477
 - \doinsidelinehook 1908, 1936
 - \doinsidelineLhook 1019
 - \doinsidelineRhook 1019
 - \dolineLhook 1019
 - \dolineRhook 1019
 - \dolistloop 1437, 1457, 1464
 - \dummy@ref 558
 - \dump@pstartL@pc 688, 897
 - \dump@pstartR@pc 688, 919
- E**
- \edfont@info 669, 672, 678, 681
 - \edlabel 1397
 - \edtext 665
 - \eled@sectioningR@out .. 70, 95, 2360
 - \eled@sections@@ 965, 997, 1758, 2012
 - \eled@sectionsR@@ 67, 1051, 1769, 2047
 - \eledpar@error 21, 30, 33, 35
 - \eledsection@correcting@skip ...
..... 1016, 1734, 1957, 2359
 - \eledsectmark 12, 2357
 - \eledsectnotoc 12, 2355
 - \empty 80, 83, 276, 284,
667, 676, 699, 705, 811, 859,
1166, 1248, 1256, 1352–1354,
1365, 1376, 2123, 2129, 2136, 2142
 - \endashchar 1387
 - \endgraf 892, 914, 1740, 1967
 - \endline@num 567, 573
 - \endlock 649, 1554, 1563, 1568
 - \endnumbering 7, 73, 127, 762
 - \endnumberingR 45, 73, 110, 122, 135, 762
 - \endpage@num 566, 573
 - \endstanzaextra 1570
 - \endsub 616
 - \endsubline@num 568, 574
 - \enlargethispage 1855
- environments:
- astanza 8, 1548
 - Leftside 5, 742
 - pages 5, 710
 - pairs 3, 710
 - Rightside 5, 759
- \expandonce 1476, 1480, 1496, 1507
 - \extensionchars 62, 116, 132, 140
- F**
- \f@x@l@cksR 1176
 - \first@linenum@out@Rfalse .. 596, 602
 - \first@linenum@out@Rtrue 596
 - \firstlinenum 5, 196
 - \firstlinenum* 5, 196
 - \firstsublinenum 5, 196
 - \firstsublinenum* 5, 196
 - \fix@page 337, 344
 - \flag@end 616
 - \flag@start 616
 - \flush@notes 1807, 2083
 - \flush@notesR 1374, 1808, 2084
 - \footnotelang@lua 1490, 1501
 - \footnotelang@poly 1494, 1505
 - \fullstop . 235, 1384, 1386, 1388, 1390
- G**
- \get@linelistfile 272
 - \get@nextboxL 2022, 2272
 - \get@nextboxR 2057, 2272
 - \getline@numL 956, 1084
 - \getline@numR 1042, 1066
 - \getlinesfrompagelistL
..... 1987, 2076, 2135
 - \getlinesfrompagelistR
..... 1988, 2077, 2135
 - \getlinesfromparlistL ... 1979, 2122
 - \getlinesfromparlistR ... 1980, 2122

\glp 279, 280, 287, 288, 671,
680, 700, 706, 1169, 1170, 1358,
1362, 1377, 2126, 2132, 2139, 2145
\goalfraction 5, 2259

H

\hangingsymbol 10, 1517, 1523
\hb@xt@ 963, 972, 982, 1049,
1059, 1777, 2017, 2020, 2052, 2055
\hsize 829, 876, 1777, 1877–
1880, 1883, 1897, 1899, 1900,
1902, 1904, 1915, 1926–1929,
1932, 1942, 2017, 2020, 2052, 2055

I

\if@filesw 1652
\if@firstcolumn 1262, 1287, 1308, 1443
\if@nobreak 790, 838
\if@noeled@sec 68, 94
\if@nopb 1840, 1855
\if@pb 1839, 1856
\if@pstarts 1694, 1747, 1976, 1992
\if@RTL 1009
\ifaraw@text 1706, 1755, 2003, 2075
\ifautopar 821, 869
\ifbypage@ 360
\ifbypage@R 143, 350, 1148
\ifbypstart@ 581, 1798, 2287
\ifbypstart@R 143, 585, 1802, 2334
\ifdefstring 994, 995,
1760, 1771, 1778, 1788, 2008, 2043
\ifdim 617, 621, 623, 627, 1820,
1826, 2017, 2052, 2102, 2176, 2197
\ifdimgreater 2437, 2440
\ifdimless 2450
\iffirst@linenum@out@R 596, 600
\ifhbox 1780, 1785
\ifinserthangingsymbol
. 1515, 1529, 2240
\ifinserthangingsymbolR
. 1513, 1521, 1539, 2251
\ifinstanzaL 740, 740, 1516, 1528, 2238
\ifinstanzaR 740, 741, 1522, 1538, 2249
\ifl@d@dash 1387
\ifl@d@elin 1389, 1390
\ifl@d@esl 1390
\ifl@d@pnum 1384, 1388
\ifl@d@ssub 1386
\ifl@d@pagefull 2031, 2066, 2168
\ifl@d@paging 13, 1527, 1537

\ifl@d@pairing 13, 77
\ifl@d@samelang 1649
\ifl@d@samepage 2007, 2041, 2168
\ifl@d@skipnumber 1239
\ifl@dusedbabel 1647
\ifl@labelpstart 831, 878
\ifledgroupnotesL@ 1088
\ifledgroupnotesR@ 1070, 1238
\iflednopbinverse 2239, 2250
\ifledplinenum 1385
\ifledRcol 13, 179, 201, 205, 209,
213, 257, 274, 338, 347, 372,
386, 403, 420, 432, 440, 461,
506, 533, 542, 551, 618, 624,
630, 637, 645, 650, 654, 659,
666, 1408, 1475, 1488, 1675, 1687
\ifluatex 796, 844, 984, 1000, 1489, 1500
\ifnoteschanged@ 87
\ifnumberedpar@
. 805, 853, 888, 910, 1473, 1487
\ifnumbering 148, 801, 885
\ifnumberingR 43, 74, 112, 849, 907
\ifnumberline 1070, 1088, 1238
\ifnumberpstart 690,
695, 822, 870, 898, 920, 945, 1030
\ifnumequal 1430
\ifnumgreater 1438, 1458, 1465
\ifodd 1272, 1297,
1318, 1453, 2100, 2106, 2109, 2116
\ifparledgroup 2426, 2458
\ifprint@last@after@pendL
. 929, 2023, 2186
\ifprint@last@after@pendR
. 929, 2058, 2207
\ifpst@rtedL 38, 809
\ifpst@rtedR 38, 857
\ifpstartnum 1328, 1333
\ifpstartnumR 1283
\ifshiftedpstarts 5, 2016, 2051, 2260
\ifsidepstartnum 823, 871, 1285, 1306
\ifstreempty 833, 880, 902, 924
\IfStrEq 954, 1040, 1841,
1849, 2215, 2219, 2227, 2231,
2242, 2243, 2253, 2254, 2376,
2377, 2384, 2385, 2397, 2398, 2405
\ifstrequal 2429, 2435
\ifsublines@ 233,
326, 371, 404, 411, 442, 451,
463, 470, 482, 516, 572, 574,
1071, 1089, 1155, 1242, 1410, 1414

- \ifvbox 947, 1033, 1710, 1713, 2273, 2320
 - \ifwidthliketwocolumns 11
 - \ifwrittenlinesL 2269, 2281
 - \ifwrittenlinesR 2270, 2328
 - \initnumbering@sectcmd 713, 727
 - \initnumbering@sectcountR
 - 66, 99, 119, 2042
 - \InputIfFileExists 69
 - \insert@count 548, 1482, 1509
 - \insert@countR 549, 1478, 1498
 - \insert@noterule@ledgroup
 - 2380, 2388, 2396
 - \inserthangingsymbolfalse 960
 - \inserthangingsymbolL 988, 1513
 - \inserthangingsymbolR 1513
 - \inserthangingsymbolRfalse 1046
 - \inserthangingsymbolRtrue 1044
 - \inserthangingsymboltrue 958
 - \insertlines@listR
 - 80, 243, 259, 554, 1354, 1358
 - \inserts@list 810, 1481, 1508
 - \inserts@listR 858, 1349,
 - 1352, 1362, 1376, 1377, 1477, 1497
 - \instanzaLfalse 1816, 2090
 - \instanzaLtrue 751
 - \instanzaRfalse 1817, 2091
 - \instanzaRtrue 774
 - \interlinepenalty 1556
 - \itemcount@ 1428, 1430,
 - 1435, 1438, 1456, 1458, 1463, 1465
- L**
- \l@d@nums 669, 672, 678, 681
 - \l@d@set 419, 645, 646
 - \l@dbbl@set@language 1649, 1672
 - \l@dbfnote 1472
 - \l@dc@maxchunks 816, 818,
 - 864, 866, 1606, 1616, 1624, 1631
 - \l@dcalc@maxoftwo
 - 1981, 2155, 2298, 2340
 - \l@dcalc@minoftwo .. 1989, 2078, 2155
 - \l@dcalcnun 1176
 - \l@dchecklang 1649
 - \l@dchset@num 293, 296, 419
 - \l@dcsnotetext 1437, 1440
 - \l@dcsnotetext@l 1440, 1457
 - \l@dcsnotetext@r 1440, 1464
 - \l@demptyd@ta 951, 1037
 - \l@dend@stuff 63, 117, 133, 141
 - \l@dgetline@margin 177
 - \l@dgetsidenote@margin 1419
 - \l@dld@ta 979,
 - 1263, 1275, 1288, 1300, 1309, 1321
 - \l@dleftbox
 - .. 931, 962, 972, 1779, 2017, 2020
 - \l@dlinenumR 225
 - \l@dlsn@te 981
 - \l@dmake@labelsR 1397
 - \l@dminpagelines
 - 1951, 1990, 2079, 2177, 2198
 - \l@dnumpstartsL 690, 815,
 - 816, 818, 820, 834, 835, 903,
 - 904, 1610, 1642, 1697, 1735,
 - 1736, 1812, 1961, 1962, 2088, 2285
 - \l@dnumpstartsR 47, 695, 863,
 - 864, 866, 868, 881, 882, 925,
 - 926, 1610, 1643, 1700, 1735,
 - 1736, 1813, 1961, 1962, 2089, 2332
 - \l@doldbbl@set@language 1671
 - \l@doldselectlanguage 1670, 1674, 1679
 - \l@dpagefullfalse
 - 2168, 2217, 2222, 2229, 2234
 - \l@dpagefulltrue
 - 2168, 2216, 2221, 2228, 2233
 - \l@dpagingfalse 14, 712, 732
 - \l@dpagingtrue 726
 - \l@dpairingfalse 13, 716, 731
 - \l@dpairingtrue 711, 725
 - \l@dprintingcolumnsfalse 1814
 - \l@dprintingcolumnstrue 1733
 - \l@dprintingpagesfalse 2092
 - \l@dprintingpagestrue 1956
 - \l@dpscL 945, 947,
 - 952, 965, 996, 1005, 1008, 1612,
 - 1644, 1697, 1710, 1744, 1750,
 - 1758, 1761, 1763, 1810, 1971,
 - 1977, 1982, 1985, 1993, 2009,
 - 2010, 2012, 2024, 2025, 2086,
 - 2273, 2275, 2278, 2285, 2293,
 - 2294, 2298, 2300, 2305, 2375, 2430
 - \l@dpscR 1030, 1033, 1038,
 - 1051, 1613, 1645, 1700, 1713,
 - 1745, 1751, 1769, 1772, 1774,
 - 1811, 1972, 1978, 1986, 1994,
 - 2044, 2045, 2047, 2059, 2060,
 - 2087, 2320, 2322, 2325, 2332,
 - 2338-2340, 2342, 2344, 2383, 2432
 - \l@drd@ta 989,
 - 1265, 1273, 1290, 1298, 1311, 1319

- \l@drightbox 931, 1048, 1059, 1784, 2052, 2055
- \l@drsn@te 990
- \l@dsamepagefalse 2168, 2216, 2221, 2228, 2233
- \l@dsamepagetrue 2168, 2217, 2222, 2229, 2234
- \l@dsetupmaxlinecounts .. 1623, 1640
- \l@dsetupprawboxes 1615, 1639
- \l@dskipnumberfalse 1240
- \l@dskipnumbertrue 1136
- \l@dunhbox@line 989, 1011, 1014
- \l@dusedbabelfalse 1647, 1667
- \l@dusedbabeltrue 1647, 1669
- \l@duselanguage 1659, 1756, 1767, 2005, 2040
- \l@dzeromaxlinecounts ... 1623, 1641
- \l@dzeropenalties 891, 913, 1739, 1966
- \l@luatexbodydir@L 799, 1003
- \l@luatexbodydir@R 847
- \l@luatexpardir@L 798, 1002
- \l@luatexpardir@R 846
- \l@luatextextdir@L 797, 985, 1001
- \l@luatextextdir@R 845
- \l@prev@nopbR ... 54, 2362, 2369, 2370
- \l@prev@pbR 53, 2361, 2367, 2368
- \l@pscl 1612
- \l@pscr 1612
- \labelref@list 1415
- \labelref@listR 1395, 1411
- \languagename .. 1650, 1651, 1653–1656
- \last@page@num 358, 364
- \last@page@numR 344
- \lastbox 955, 1041
- \lastskip 616, 622
- \Lcolwidth 4, 5, 16, 728, 829, 963, 972, 1764, 1878, 1899, 1916, 1927, 1943
- \led@err@BadLeftRightPstarts ... 29, 1736, 1962
- \led@err@LeftOnRightPage ... 32, 2110
- \led@err@LineationInNumbered ... 149
- \led@err@ManyLeftnotes 1458
- \led@err@ManyRightnotes 1465
- \led@err@ManySidenotes 1438
- \led@err@NumberingNotStarted ... 91
- \led@err@numberingShouldHaveStarted 121
- \led@err@NumberingStarted 44
- \led@err@PendNoPstart 889, 911
- \led@err@PendNotNumbered ... 886, 908
- \led@err@PstartInPstart ... 806, 854
- \led@err@PstartNotNumbered . 802, 850
- \led@err@RightOnLeftPage ... 32, 2117
- \led@err@TooManyPstarts 24, 26, 817, 865
- \led@mess@NotesChanged 88
- \led@mess@SectionContinued 115, 131, 139
- \led@nopbnumR 2366, 2367
- \led@nopbR 2365, 2367
- \led@pb@setting 1841, 1849, 2215, 2219, 2227, 2231, 2242, 2243, 2253, 2254
- \led@pbnumR 2364, 2367
- \led@pbR 2363, 2367
- \led@warn@BadAction 1138
- \led@warn@BadAdvancelineLine 389, 395
- \led@warn@BadAdvancelineSubline . 375, 381
- \led@warn@BadLineation 166
- \led@warn@BadSetline 635
- \led@warn@BadSetlinenum 643
- \led@warn@DuplicateLabel 1399
- \ledgroupnotesL@false 2391
- \ledgroupnotesL@true 2379
- \ledgroupnotesR@false 2394
- \ledgroupnotesR@true 2387
- \ledllfill 982
- \lednopb 770
- \lednopbnum 2242, 2363
- \lednopbnumR 2253, 2363
- \lednopbR 770, 2365
- \ledpb 769
- \ledpbnum 2243
- \ledpbnumR 2254, 2363
- \ledpbR 769, 2363
- \ledRcol@false 1062
- \ledRcol@true 1031
- \ledRcolfalse 15, 743, 776
- \ledRcoltrue 760
- \ledrlfill 989, 1887, 1909
- \ledsavedprintlines 8, 1382
- \ledsectnomark 995
- \ledsectnotoc 994, 2008, 2043
- \ledstrutL 2017, 2020, 2095
- \ledstrutR 2052, 2055, 2095
- \ledthegoal 2176, 2197, 2259
- \leftlinenumR 225, 1263, 1275
- \leftpstartnumL 1283

- `\leftstartnumR` 1283
 - `Leftside` (environment) 5, 742
 - `\Leftsidehook` 749, 754
 - `\Leftsidehookend` 753, 754
 - `\letcs` 945, 1030
 - `\line@list` 676, 680
 - `\line@list@stuff` 132
 - `\line@list@stuffR` .. 62, 116, 140, 598
 - `\line@listR` . 83, 243, 258, 574, 667, 671
 - `\line@margin` 182, 1293
 - `\line@marginR` 175, 1268, 1314
 - `\line@num` . 361, 393, 394, 396, 414,
425, 426, 454, 581, 1095, 1413, 2288
 - `\line@numR` 55,
232, 239, 298, 332, 351, 387,
388, 390, 407, 421, 422, 445,
567, 571, 585, 1077, 1149, 1158,
1247, 1249, 1251, 1252, 1409, 2335
 - `\lineation` 171, 172, 771
 - `\lineation*` 6, 171
 - `\lineationR` 6, 147, 173, 771
 - `\linenum@out` ... 613, 619, 625, 631,
638, 646, 651, 655, 1721, 1799, 2149
 - `\linenum@outR` 595,
601, 603, 608, 609, 615, 618,
624, 630, 637, 645, 650, 654,
659, 1726, 1803, 2152, 2363–2366
 - `\linenumberlist` 1248, 1252
 - `\linenumincrement` 5, 196
 - `\linenumincrement*` 5, 196
 - `\linenummargin` 175
 - `\linenumr@p` 1385, 1389, 1409, 1413
 - `\linenumrepR` 222, 232
 - `\linenumsep`
. 227, 229, 1330, 1333, 1342, 1345
 - `\linesinpar@listL`
..... 250, 268, 582, 2123, 2126
 - `\linesinpar@listR`
..... 250, 262, 586, 2129, 2132
 - `\linesonpage@listL` 269, 590, 2136, 2139
 - `\linesonpage@listR` 263, 593, 2142, 2145
 - `\list@clear`
. 258–265, 268, 269, 271, 810, 858
 - `\list@clearing@reg` 267
 - `\list@create` 243–
248, 250–252, 686, 687, 1349, 1395
 - `\list@pstartL@pc` .. 686, 689, 699, 700
 - `\list@pstartR@pc` .. 686, 694, 705, 706
 - `\listxadd` 366, 2367–2370
 - `\lock@disp` 1208, 1212, 1217
 - `\lock@off` 503, 504, 512, 654, 655
 - `\lock@on` 650, 651
 - `\luatexbodydir` 799, 847, 1003
 - `\luatexpardir` 798, 846, 1002
 - `\luatextextdir` ... 797, 845, 985, 1001
- M**
- `\managestanza@modulo` 1582
 - `\maxchunks` 3, 1606
 - `\maxlinesinpar@list` 250, 271
 - `\memorydump` 7, 748, 765
 - `\memorydumpL` 126, 748
 - `\memorydumpR` 126, 765
 - `\message` 61
 - `\multiply` 1182, 1881, 1930
- N**
- `\n@num` 540, 659
 - `\n@num@reg` 546
 - `\namebox` 947, 952, 1033,
1038, 1590, 1710, 1713, 2273, 2320
 - `\NeedsTeXFormat` 2
 - `\new@line` 1011, 1014
 - `\new@lineL` 612, 987
 - `\new@lineR` 614
 - `\newbox` 781, 931, 932, 1591
 - `\newcommandx` 789, 837, 884, 906
 - `\newcounter` 99–
102, 187, 189, 191, 193, 784, 786
 - `\newif` 5, 39, 143,
144, 596, 740, 741, 929, 930,
1337, 1513, 1647, 1694, 1706,
1839, 1840, 2168, 2170, 2269, 2270
 - `\newlength` 1867, 1870
 - `\newmarks` 2371–2373
 - `\newnamebox` 1590, 1618, 1619
 - `\newnamecount` 1601, 1626
 - `\newsavebox` 1730, 1731
 - `\newwrite` 595, 2360
 - `\next@absline`
.... 1842, 1844, 1846, 2220–2222
 - `\next@abslineR`
.... 1843, 1845, 1847, 2232–2234
 - `\next@action` 288
 - `\next@actionline` 285, 287
 - `\next@actionlineR`
. 277, 279, 1107, 1145, 1167, 1169
 - `\next@actionR` 280, 1108,
1146, 1147, 1152, 1153, 1161, 1170
 - `\next@insert` 811

- \next@insertR 859, 1353, 1356, 1358, 1361, 1365
 - \next@page@num 365, 437
 - \next@page@numR 59, 301, 303, 355, 434
 - \noindent 835, 882, 904, 926
 - \normal@page@break 366
 - \normal@page@breakR 52
 - \normal@pars 76, 814, 862
 - \normalbfnoteX 1486
 - \notefontsetup 2417
 - \noteschanged@true 81, 84, 668, 677, 1355
 - \notesXwidthliketwocolumns 4
 - \num@lines 892, 1740, 1967
 - \num@linesR 780, 914, 1741, 1968
 - \numberedpar@true 830, 877
 - \numberingRfalse 75
 - \numberingRtrue 49, 110, 136
 - \numberingtrue 128
 - \numberpstartfalse 8
 - \numberpstarttrue 8
 - \numdef 996, 1456, 1463, 1842, 1843, 2220, 2232, 2430, 2432
 - \numgdef 1428, 1435, 2241, 2252
 - \numlabfont 232
 - \numpagelinesL 1951, 2015, 2032, 2036, 2177, 2243, 2450
 - \numpagelinesR 1951, 2050, 2067, 2071, 2198, 2254
- O**
- \old@otherlanguage 1684
 - \old@startstanza .. 750, 751, 773, 774
 - \oldchapter 723, 733
 - \one@line 952, 955, 982, 989, 1011, 1014
 - \one@lineR 780, 1038, 1041
 - \openout 70, 603, 609
 - \otherlanguage 1684, 1685
- P**
- \p@pstartL 832
 - \p@pstartR 879
 - \PackageError 21
 - \page@action 302, 431, 560
 - \page@num 283, 363, 1295
 - \page@numR 254, 275, 353, 566, 571, 1147, 1270, 1316, 1451
 - \pagebreak 1856
 - \pagegoal 2267
 - \Pages 5, 1955
 - pages (environment) 5, 710
 - \pagetotal 2102, 2176, 2197
 - pairs (environment) 3, 710
 - \paperwidth 1010, 1013
 - \par@line 893, 1742, 1969
 - \par@lineR 780, 915, 1743, 1970
 - \parbox 1764, 1775
 - \parledgroup@ .. 954, 1040, 2371, 2397
 - \parledgroup@beforenotes@save .. 901, 923, 2457
 - \parledgroup@beforenotesL 2455
 - \parledgroup@beforenotesR 2455
 - \parledgroup@correction@notespacing 2019, 2054, 2444
 - \parledgroup@correction@notespacing@final 2309, 2348, 2425
 - \parledgroup@correction@notespacing@init 2037, 2072, 2420, 2428
 - \parledgroup@notes@endL 2276, 2279, 2308, 2390
 - \parledgroup@notes@endR 2323, 2326, 2347, 2393
 - \parledgroup@notes@startL 954, 2374, 2390
 - \parledgroup@notes@startR 1040, 2374, 2390
 - \parledgroup@notespacing@correction 2415, 2447–2449
 - \parledgroup@notespacing@correction@accumulated 2421, 2427, 2448
 - \parledgroup@notespacing@correction@modulo 2422, 2449–2451
 - \parledgroup@notespacing@set@correction 1958, 2415
 - \parledgroup@series 2372, 2376, 2377, 2384, 2385, 2400, 2401, 2407, 2408
 - \parledgroup@type 2373, 2376, 2377, 2384, 2385, 2398, 2405
 - \parledgroupnotespacing . 2414, 2417
 - \parledgroupseries@ 2371
 - \parledgrouptrue 10
 - \parledgrouptype@ 2371
 - \pausenumbering 763
 - \pausenumberingR 109, 763
 - \pend 6, 747, 768, 807, 1569
 - \pendL 747, 884
 - \pendR 768, 855, 906
 - \prev@abslineverse 2241–2243, 2252–2254

- \prev@nopbR 2361
 - \prev@pbR 2361
 - \prevgraf
 - 892, 914, 1740, 1741, 1967, 1968
 - \print@columnseparator .. 1783, 1819
 - \print@eledsectionL .. 992, 1764, 2013
 - \print@eledsectionR .. 1066, 1775, 2048
 - \print@last@after@pendLfalse .. 2189
 - \print@last@after@pendLtrue ... 2313
 - \print@last@after@pendRfalse .. 2210
 - \print@last@after@pendRtrue ... 2350
 - \print@lineL 967, 977
 - \print@lineR 1053, 1066
 - \printlines 1393
 - \printlinesR 8, 1382
 - \ProcessOptions 12
 - \protected@edef 831, 878
 - \protected@write 1653
 - \protected@xdef 1495, 1506
 - \ProvidesPackage 3
 - \pst@rtedLfalse 38
 - \pst@rtedLtrue 129, 812
 - \pst@rtedRfalse 40, 48, 78
 - \pst@rtedRtrue 113, 137, 860
 - \pstart 6, 27, 31, 745, 767, 1571
 - \pstartL 745, 783
 - \pstartnumfalse 1330, 1335
 - \pstartnumRfalse 1342, 1347
 - \pstartnumRtrue 1338, 1749, 2345
 - \pstartnumtrue 1748, 2306
 - \pstartR 767, 783
- R**
- \Rcolwidth
 - 4, 5, 16, 729, 876, 1049, 1059,
 - 1775, 1879, 1900, 1917, 1928, 1944
 - \read@linelist 256, 599
 - \rem@inder 1252, 1254–1256
 - \resetprevline@ 1800, 1804, 2289, 2336
 - \restore@pstartL@pc
 - 698, 1752, 1995, 2307
 - \restore@pstartR@pc
 - 698, 1753, 1996, 2346
 - \resumenumbering 764
 - \resumenumberingR 109, 764
 - \rightlinenumR 225, 1265, 1273
 - \rightpstartnumL 1283
 - \rightpstartnumR 1283
 - Rightside (environment) 5, 759
 - \Rightsidehook 754, 772
 - \Rightsidehookend 754, 777
 - \rlap 1265, 1273, 1290, 1298, 1311, 1319
 - \Rlineflag 8, 220, 232, 1385, 1389, 1401
 - \rule 1859
- S**
- \savebox 1764, 1775
 - \sc@n@list 1253, 1255
 - \secdef 738
 - \section@num 130–132
 - \section@numR 36,
 - 50, 61, 62, 69, 70, 114–116, 138–140
 - \select@language 1651, 1653–1656, 1690
 - \selectlanguage 1659
 - \set@line 665
 - \set@line@action
 - 295, 400, 409, 416, 439, 562
 - \set@dlp@rbox 1444, 1459, 1461
 - \set@drp@rbox 1446, 1454, 1466
 - \setline 633
 - \setlinenum 641
 - \setnamebox 820, 868, 1590
 - \setnotepositionliketwocolumns@C
 - 1873
 - \setnotepositionliketwocolumns@L
 - 1873
 - \setnotepositionliketwocolumns@R
 - 1873
 - \setnotespositionliketwocolumns@C
 - 1912
 - \setnotespositionliketwocolumns@L
 - 1890
 - \setnotespositionliketwocolumns@R
 - 1939
 - \setpositionliketwocolumns@C ...
 - 1873, 1907
 - \setpositionliketwocolumns@L ...
 - 1873, 1886
 - \setpositionliketwocolumns@R ...
 - 1873, 1935
 - \setprintlines 1383
 - \setwidthliketwocolumns@C 1873, 1894
 - \setwidthliketwocolumns@L 1873, 1873
 - \setwidthliketwocolumns@R 1873, 1924
 - \shiftedpstartsfalse 7
 - \shiftedpstartstrue 6, 8, 9
 - \shiftedversesfalse 7
 - \shiftedversestrue 6
 - \sidenote@margin 1421, 1424
 - \sidenote@marginR 1418, 1449

- `\sidenotecontent@`
 . 1427, 1432, 1433, 1444, 1446,
 1454, 1455, 1459, 1461, 1462, 1466
`\sidenotemargin` 1418
`\sidenotemargin*` 1418
`\sidenotesep` 1433
`\skip` 2400, 2407
`\skip@lockoff` 504, 512
`\skipnumbering` 8, 658
`\skipnumbering@reg` 662
`\smash` 1859
`\splitbotmarks` 2397,
 2398, 2400, 2401, 2405, 2407, 2408
`\splitfirstmarks`
 954, 1040, 2376, 2377, 2384, 2385
`\splittopskip` 949, 1035
`\stanza@count` 1551, 1565, 1577
`\stanza@hang` 1553, 1585
`\stanza@modulo` 1551, 1580
`\stanzaindentbase`
 1531, 1541, 1578, 1581
`\startlock` 649
`\startstanzahook` 1549
`\startsub` 616
`\sub@action` 311, 460, 561
`\sub@change` 60, 305, 306, 312
`\sub@lock` 1090
`\sub@lockR` 57, 320, 322, 324,
 327, 478, 484, 485, 487, 488,
 518, 519, 521, 1072, 1128, 1130,
 1131, 1133, 1189, 1229, 1231, 1233
`\sub@off` 624, 625
`\sub@on` 618, 619
`\subline@num` 234, 361, 379,
 380, 382, 412, 452, 1091, 1096, 1414
`\subline@numR` 235,
 239, 328, 332, 351, 373, 374,
 376, 405, 443, 568, 572, 1073,
 1078, 1149, 1156, 1243, 1244, 1410
`\sublinenumincrement` 5, 196
`\sublinenumincrement*` 5, 196
`\sublinenumr@p` . 1386, 1390, 1410, 1414
`\sublinenumrepR` 222, 235
`\sublines@false` 58, 309, 1118
`\sublines@true` 307, 1116
`\sublock@disp` 1191, 1195, 1200
`\sw@list@inedtextR` 248, 265
`\sw@listR` 247, 264
`\symplinenum` 1385
`\sza@penalty` 1560, 1564
- T**
- `\temp` 1876, 1877, 1880–1883,
 1896, 1897, 1902–1904, 1913,
 1915, 1918–1921, 1925, 1926,
 1929–1932, 1940, 1942, 1945–1948
`\temp@` 996, 997
`\temp@spacing` 2417, 2418
`\tempa` 1914, 1916–1918, 1941, 1943–1945
`\textwidth` 17, 19, 728, 729
`\theledlanguageL` ... 1659, 1756, 2005
`\theledlanguageR` ... 1659, 1767, 2040
`\thepage` 613, 615
`\thepstart` 746, 766
`\thepstartL`
 . 8, 746, 785, 825, 832, 1329, 1334
`\thepstartR`
 . 8, 766, 787, 872, 879, 1341, 1346
`\thisfootnote` .. 1495, 1496, 1506, 1507
`\thr@@` .. 487, 496, 519, 526, 1123, 1131
`\topskip` 2102
- U**
- `\unhbox` 1595,
 1779, 1784, 2017, 2020, 2052, 2055
`\unhnamebox` 1590
`\unvbox` 955, 1041, 1597
`\unvnamebox` 1590
`\usebox` 1781, 1786
`\usenamecount` 1552, 1559, 1601, 1633,
 1982, 2275, 2278, 2298, 2300,
 2322, 2325, 2340, 2342, 2375, 2383
- V**
- `\value` 1576
`\vbadness` 948, 1034
`\vbfnoteX` 1496, 1507
`\vbox` 820, 868, 1764, 1775
`\vl@dbfnote` 1476, 1480
`\vsplit` 952, 1038
- W**
- `\wd` 982
`\widthliketwocolumns` 4
`\widthliketwocolumnstrue` 11
`\WithSuffix` 171, 216–219, 1418
`\writtenlinesLfalse` 1973, 2286
`\writtenlinesLtrue` 2283
`\writtenlinesRfalse` 1974, 2333
`\writtenlinesRtrue` 2330

X	
<code>\xifinlist</code>	965,
	997, 1051, 1758, 1769, 2012, 2047
<code>\xifinlistcs</code>	1844–
	1847, 1850–1853, 2216, 2217,
	2221, 2222, 2228, 2229, 2233, 2234
<code>\Xnoteswidthliketwocolumns</code>	4
<code>\xpg@main@language</code>	1691, 1692
<code>\xright@appenditem</code>	
	. 433, 434, 436, 437, 441, 448,
	450, 457, 462, 464, 466, 469,
	471, 473, 481, 483, 492, 515,
	517, 524, 543, 544, 554, 570,
	582, 586, 590, 593, 689, 694,
	1409, 1413, 1476, 1480, 1496, 1507

Change History

v0.1.	
General: First public release	1
v0.2.	
General: Added section of babel re-	
lated code	60
Fix babel problems	1
<code>\Columns</code> : Added <code>\l@dchecklang</code>	
and <code>\l@duselanguage</code> to	
<code>\Columns</code>	63
<code>\Pages</code> : Added <code>\l@duselanguage</code>	
to <code>\Pages</code>	70
v0.3.	
General: Added <code>\do@lineLhook</code>	
and <code>\do@lineRhook</code>	43
Reorganize for ledarab	1
<code>\affixline@numR</code> : Changed	
<code>\affixline@numR</code> to match new	
eledmac	46
<code>\do@actions@nextR</code> : Used	
<code>\do@actions@fixedcode</code> in	
<code>\do@actionsR</code>	45
<code>\do@lineL</code> : Added <code>\do@lineLhook</code>	
to <code>\do@lineL</code>	41
Simplified <code>\do@lineL</code> by using	
macros for some common code	41
<code>\do@lineR</code> : Changed <code>\do@lineR</code>	
similarly to <code>\do@lineL</code>	43
Leftside: Added hooks into Left-	
side environment	35
<code>\flag@end</code> : Removed extraneous	
spaces from <code>\flag@end</code>	31
<code>\ifledRcol</code> : Moved <code>\ifl@dpairing</code>	
to eledmac	13
<code>\ifpst@rtedR</code> : Moved <code>\ifpst@rtedL</code>	
to eledmac	14
<code>\l@dlinenumber</code> : Simplified	
<code>\leftlinenumber</code> and <code>\rightlinenumber</code>	
by introducing <code>\l@dlinenumber</code>	20
<code>\l@dnumpstartsR</code> : Moved	
<code>\l@dnumpstartsL</code> to eledmac	58
<code>\ledsavedprintlines</code> : Simplified	
<code>\printlinesR</code> by using	
<code>\setprintlines</code>	52
<code>\ledstrutR</code> : Added <code>\ledtrutL</code> and	
<code>\ledstrutR</code>	73
<code>\normalbfnoteX</code> : Removed	
extraneous spaces from	
<code>\normalbfnoteX</code>	55
<code>\Pages</code> : Added <code>\ledstrutL</code> to	
<code>\Pages</code>	70
Added <code>\ledstrutR</code> to <code>\Pages</code>	71
<code>\Rightsidehookend</code> : Added	
<code>\Leftsidehook</code> , <code>\Leftsidehookend</code> ,	
<code>\Rightsidehook</code> and <code>\Rightsidehookend</code>	
.....	35
<code>\sublinenumberR</code> : Added	
<code>\linenumberR</code> and <code>\sublinenumberR</code>	
.....	19
v0.3.a.	
General: Minor <code>\linenumbermargin</code>	
fix	1
<code>\line@marginR</code> : Don't just	
set <code>\line@marginR</code> in	
<code>\linenumbermargin</code>	18
v0.3.b.	
General: Improved parallel page	
balancing	1
<code>\Pages</code> : Added <code>\l@dminpagelines</code>	
calculation for succeeding page	
pairs	72

v0.3.c.	General: Compatibilty with Polyglossia	1	v0.10.	General: <code>\edlabel</code> commands on the right side are now correctly indicated.	1
v0.4.	General: No more ledparpatch. All patches are now in the main file.	1		<code>\edlabel</code> commands which start a paragraph are now put in the right place.	1
v0.5.	General: Corrections about <code>\section</code> and other titles in numbered sections	1	v0.11.	General: Change <code>\do@lineL</code> and <code>\do@lineR</code> to allow line numbering by <code>pstart</code> (like in eledmac 0.15).	41
v0.6.	General: Be able to use <code>\chapter</code> in parallel pages.	1		Lineation can be by <code>pstart</code> (like in eledmac 0.15).	17
v0.7.	General: Option ‘shiftedverses’ which make there is no blank between two parallel verses with inequal length.	1		New management of hangingsymbol insertion, preventing undesirable insertions.	56
v0.8.	General: Possibility to have a symbol on each hanging of verses, like in the french typography. Redefine the commande <code>\hangingsymbol</code> to define the character.	1		Prevent shift of column separator when a verse is hanged	56
v0.9.	General: Possibility to number <code>\pstart</code>	8		<code>\affixline@numR</code> : Changed <code>\affixline@numR</code> to allow to disable line numbering (like in eledmac 0.15).	46
	Possibility to number the <code>pstart</code> with the commands <code>\numberpstarttrue</code>	1		<code>\Columns</code> : Line numbering by <code>pstart</code>	64
	<code>\ifledRcol</code> : Moved <code>\iflledRcol</code> and <code>\ifnumberingR</code> to eledmac	13		<code>\get@nextboxR</code> : Change <code>\get@nextboxL</code> and <code>\get@nextboxR</code> to allow to disable line numbering (like in eledmac 0.15).	77
v0.9.1.	General: The numbering of the <code>pstarts</code> restarts on each <code>\beginnumbering</code>	1		<code>Pstart</code> number can be printed in side	78
v0.9.2.	General: Debug : with <code>\Columns</code> , the hanging indentation now runs on the left columns and the hanging symbol is shown only when <code>\stanza</code> is used.	1	v0.12.	General: New new management of hangingsymbol insertion, preventing undesirable insertions.	56
v0.9.3.	General: <code>\thepstartL</code> and <code>\thepstartR</code> use now <code>\bfseries</code> and not <code>\bf</code> , 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 <code>pstart</code>	17
			v1.0.1.	General: Correction on <code>\numberonlyfirstinline</code> with lineation by <code>pstart</code> or by page.	1
			v1.1.	General: Shiftedverses becomes shiftedpstarts.	1
				<code>\pstartR</code> : Add <code>\labelpstarttrue</code> (from eledmac).	36

v1.1.1.	\pstartR: Correct \pstartR bug introduced by 1.1.	36	length of a line.	1
v1.1.2.	\affixside@noteR: Remove spurious space between line number and line content	54	\inserthangingsymbolR: Hang verse is now not automatically flush right.	56
v1.2.	General: Support for \led<section> commands in parallel texts. . . .	1	\pendL: Spurious spaces in \pendL.	39
v1.2.1.	\initnumbering@sectcountR: For the right section, the counter is defined only once.	16	\pendR: Spurious spaces in \pstartR.	40
v1.3.	\edtext: Manage RTL language.	32	\pstartR: Spurious spaces in \pstartL and \pstartR.	36
v1.3.1.	\l@dbfnote: Compatibility of standard footnotes with eledmac when theses footnotes contain any commands.	55	v1.5.0.	
v1.3.2.	General: Debug with some classes.	1	General: Add, as in eledmac, features to manage page breaks. . . .	1
v1.3.3.	General: Debugging the left notes of the right column.	53	\sublinenumincrement*: Add starred version of \firstlinenum, \linenumincrement, \firstsublinenum, \sublinenumincrement to change both Left and Right-side.	19
v1.3.4.	\l@dbfnote: Spurious space with footnote in right column.	55	v1.6.0.	
v1.3.5.	General: Allow use of commands in sidenotes, as introduced by eledmac 1.0.	53	General: Add tool and documentation for parallel ledgroups	10
v1.4.	\normalbfnoteX: Allows one to redefine \thefootnoteX with alph when some packages are loaded.	55	v1.7.0.	
v1.4.1.	General: Added \do@insidelineLhook and \do@insidelineRhook	43	General: Add, as in eledmac, features to make crossrefs with pstart numbers.	1
v1.4.3.	\normalbfnoteX: Fix bug with normal familiar footnotes when mixing RTL and LTR text. . . .	55	v1.8.0.	
	astanza: Enable the use of stanzaindentrepetition within stanza environment.	57	General: \beginnumbering is defined only on eledmac, not on eledpar.	14
			\l@dlsnote, \l@drsnote and \l@dcsnote defined only one time, in eledmac.	53
			Add \beforecolumnseparator and \aftercolumnseparator. . . .	4
			Add \columnspan.	4
			Add, as in eledmac, new system of sectioning commands.	1
			Add, as in eledmac, option to insert something after \pends / verses.	1
			Add, as in eledmac, option to insert something between \pstarts / verse.	1
			Change \do@lineR and \do@lineL to allow new sectioning commands.	41
			Compatibility with musixtex. . . .	1
			Debug eledmac sectioning command after using \resumenummering.	1

New sectioning commands, as in eledmac.	12	one time for eledmac and eled- par	31
\Columns: Modify \Columns to en- able to add section's title. . . .	62	\lineation*: Add \lineation* . .	18
Suppress \l@ddchecklang from \Columns.	63	v1.8.3. General: Add \noeledxxx, as in eledmac	1
\l@ddchecklang: Suppress \l@ddchecklang which didn't work and was not logical, be- cause both columns could have the same language but not the main language of the docu- ment.	60	\doinsidelineRhook: Added \dolineLhook, \dolineRhook, \doinsidelineLhook and \doinsidelineRhook	42
\Pages: Modify \Pages to enable to add section's title.	68	\Pages: Debug blank pages when using optional argument in the last \pend.	68
\pendL: As in eledmac, \pendL can have an optional argument. . . .	39	\resumenumberingR: Debug \resumenumberingR	16
\pendR: As in eledmac, \pendR can have an optional argument. . . .	40	v1.9.0. General: Add \AtBeginPairs macro.	4
\print@columnseparator: Move some code of \Columns to \print@columnseparator. . . .	65	Compatibility with \Xnoteswidthliketwocolumns and \notesXwidthliketwocolumns	1
\pstartR: As in eledmac, \pendL and \pendR can have an op- tional argument.	36	\ifwidthliketwocolumns: Added widthliketwocolumns option . .	13
\sidenotemargin*: \sidenotemargin is now directly defined in eled- mac to be able to manage eled- par.	53	\theledlanguageR: Debug left/right language switching with polyglossia. Don't write in .aux file when setting left/right lines.	61
Add \sidenotemargin*	53	v1.9.1. \ifledRcol: Moved \ifl@dpaging to eledmac	13
\theledlanguageR: Correct left/right language setting with polyglossia.	61	v1.10.0. General: Compatibility with \AtEveryPstart and \AtEveryPend	1
v1.8.1. \do@lineL: Fix a bug with critical notes a the begining of a page, (maybe added by v1.8.0) (?). .	41	Restore critical notes in \eledsection in parallel columns (this bug was added in 1.8.2).	1
\do@lineR: Fix a bug with critical notes a the begining of a page, added by v1.8.0 (?).	43	\Pages: Debug wrong pages split- ting when no optional argument is used in last \pend (bug was added in v.1.8.3).	68
v1.8.2. General: Debug \eledxxx with some paper sizes	1	Debug wrong parallel pages syn- chronization when an \edtext falls accross two pages.	68
Debug left and side note (bugs added by 1.8.0)	1	v1.10.1. \line@list@stuffR: Revert modi- fication of 1.4.2, which makes bugs with numbering. Leave	
\eledpar@error: Errors specific to eledpar send to eledpar hand- book	14		
\flag@end: \flag@start and \flag@end are now defined only			

vertical mode to solve spurious space before minipage.	30	time for both eledmac and eled- par	53
v1.11.0.		<code>\Pages</code> : Add <code>\l@dprintingpagetrue</code>	69
General: Compatibility of standard footnotes with some biblatex styles.	1	<code>\print@eledsectionL</code> : Compati- bility with Lua ^A T _E X RTL lan- guages.	42
<code>\edtext</code> : <code>\critext</code> and <code>\edtext</code> are now defined only in eledmac. . .	32	<code>\print@eledsectionR</code> : Compati- bility with Lua ^A T _E X RTL lan- guages.	44
v1.12.0.		<code>\print@elineL</code> : Compatibility with Lua ^A T _E X RTL languages. . . .	41
General: Compatibility with Lua ^A T _E X RTL languages. . . .	1	v1.12.1.	
<code>\Columns</code> : Add <code>\l@dprintingcolumnstrue</code>	63	<code>\print@eledsectionL</code> : Fixes bug with Lua ^A T _E XRTL <code>\eledsection</code>	42
<code>\edlabel</code> : <code>\edlabel</code> and <code>\edindex</code> works now with hyperref when using eledpar.	53		
<code>\edlabel</code> is now defined only one			