Parallel typesetting for critical editions: the `reledpar` package*

Maïeul Rouquette†based on the original `ledpar` by Peter Wilson
Herries Press‡

Abstract

The `reledmac` package has been used for some time for typesetting critical editions. The `reledpar` package is an extension to `reledmac` which enables texts and their critical apparatus to be typeset in parallel, either in two columns or on pairs of facing pages.

`reledpar` 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). Examples starting by "3-" are for basic uses, those starting by "4-" are for advanced uses.

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 `reledmac` email list in: `http://geekographie.maieul.net/146`

Contents

---

*This file (`reledpar.dtx`) has version number v2.1.0, last revised 2015/07/27.
†`maieul at maieul dot net`
‡`herries dot press at earthlink dot net`

1 Introduction

1.1 Aim of this package

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 `reledpar` package is an extension to `reledmac` that enables two texts and their apparatus to be set in parallel, either in two columns or on pairs of facing pages.

The package has to try and coerce TeX into paths it was not designed for. Use of the package, therefore, may produce some surprising results. In this case, please reports them to the author via github's issues: `https://github.com/maieul/ledmac/issues/`.

This manual contains a general description of how to use `reledpar` starting in section 2; the complete source code for the package, with extensive documentation (in sections I through XXV); and an Index to the source code. As `reledpar` is an adjunct to `reledmac` we assume that you have read the `reledmac` manual. Also `reledpar` requires `reledmac` to be used, in the version distributed with version.

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. The documentation's sections are numbered in roman numeral.

On a first reading, We suggest that you should skip anything after the general documentation in first sections until I, unless you are particularly interested in the innards of `reledpar`.

1.2 Historical overview

Many of the code of this package is based on the `eledpar` package, which was based on the `ledpar`, created as an extension of the `ledmac` package.

Names of the package related to parallel typesetting have moved in parallel of names of the package related to critical edition.

Please read `reledmac`'s handbook in order to understand this evolution.

2 General

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 will want to print the text that you are editing. Unnumbered text is not printed with line numbers, and you can't use `reledmac`'s note commands with it: this is appropriate for introductions and other material added by the editor around the edited text.

The `reledpar` package lets you typeset two *numbered* texts in parallel[1]. 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

---

[1]You can use, anyway, `\numberlinefalse` to disable printing of line numbers.

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.

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

reledpar 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.

If you \maxchunks is too little you can get a reledpar 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 \stanza, 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, reledmac is a TEX resource hog, and reledpar only makes things worse in this respect.

3   Parallel columns

3.1   Basic use

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}
\end{pairs}
\Columns
\begin{pairs}
\begin{Leftside} ... \end{Leftside}
...
\end{pairs}
\Columns
```

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

3.2 Setting

3.2.1 Column's width

\Lcolwidth The lengths \Lcolwidth and \Rcolwidth are the widths of the left and right
\Rcolwidth 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.

3.2.2 Column's separator

\columnrulewidth The macro \columnseparator is called between each left/right pair of lines. By
\columnseparator 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.

3.2.3 Column's positions

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

When you use \stanza, the visible rule may shift when a verse has a hanging indent. To prevent shifting, use \setstanzaindents outside the Leftside or Rightside environment.

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

- On the left of columns, if columns are aligned right.

- On the right of columns, if columns are aligned left.

- On both the left and right columns, if columns are centered.

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

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

If you want to revert to the previous behavior, just set with a negative value.

3.2.4   Mixing two columns and one column texts

\widthliketwocolumns
If you want to mix two-column with single-column text, you can align horizon-
tally single-column text to two-column text with \widthliketwocolumnstrue.
To reset this feature, use \widthliketwocolumnsfalse. You can also call
\widthliketwocolumns as a global option when loading reledmac or reled-
par.

\Xnoteswidthliketwocolumns
\notesXwidthliketwocolumns
In most cases, you should use \widthliketwocolumns in combination with
\Xnoteswidthliketwocolumns and \notesXwidthliketwocolumns
to align the critical/familiar footnotes with the two colums. See reledmac's handbook
for more details.

4   Facing pages

4.1   Basic usage

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.

4.2   Setting

4.2.1   Text width

\Lcolwidth Within the `pages` environment the lengths `\Lcolwidth` and `\Rcolwidth` are the
\Rcolwidth 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.

4.2.2   Page number

By default, `\Pages` use the standard LaTeX page number scheme. This means that pages are numbered continuously following printed-book conventions: from left-hand to right-hand side, left-hand pages having even numbers, right-hand pages having odd numbers.

However, you can use the package option `sameparallelpagenumber` to have the same page number for both left and right side. In this case, this setting will apply only for pages typeset by `\Pages`, not for "normal" pages.

Please also read advising in 10 p. 16.

4.2.3   Page breaking

\setgoalfraction When doing parallel pages `reledpar` has to guess where TeX is going to put page-breaks 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 standard value is 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. You can change it using `\setgoalfraction{`⟨*newvalue*⟩`}`.

4.2.4   Right page before **`\Pages`**

When `\Pages` are called, it starts at a new left page, in order to have parallel pages. Consequently, if it is called on a left page, it clears the current page and then lets a right void page.

`reledpar` provides two options to customize this (eventual) right page.

prevpgstyle=⟨*style*⟩ in order to set the style of this page. A common value of ⟨*style*⟩ is `empty`. Use `prevpgstyle=empty` will suppress header and footer in this page. Please also read advising in 10 p. 16.

prevpgnotnumbered will make this page won't be counted in the page counter.

4.3   Critical and familiar footnotes

Of course, in "Facing pages", the reledmac's both critical and familiar footnotes can be used. However, some specific points must be taken into consideration.

4.3.1   Notes height setting

Since eledpar v1.13.0, long notes in facing pages can flow from left to right pages, and *vice-versa*.

However, the reledmac default setting for the maximum alloted size to notes is greater than \textheight. That makes impossible for long notes to flow across pages. [2] We have not changed this default setting, because we do not want to break compatibility with older version of reledmac and we want to be as close as possible to default LaTeX's feature.

So, you MUST change the default setting via \Xmaxnotes (for critical notes) and \maxhnotesX (for familiar notes). Both commands are explained in reledmac handbook (?? p. ??). As an advisable setting:

```
\maxhXnotes{0.6\textheight}
\maxhnotesX{0.6\textheight}
```

4.3.2   Notes for one side only

\Xonlyside
\onlysideX
You may want to typeset notes on one side only (either left or right). Use \Xonly-side[⟨s⟩]{⟨p⟩} to set critical notes, and \onlysideX[⟨s⟩]{⟨p⟩} to set familiar notes. {⟨p⟩} must be set to L for notes to be confined only on the left side and to R for notes to be confined only on the right side.

4.3.3   Familiar notes called in the right side, but to be printed in the left side

\footnoteXnomk
\footnoteXmk
As often happens, the left side has less room for text. We may want to call familiar notes in the right side while using at the same time the available space in the left side to print them.

To achieve this, we call \footnoteXnomk{⟨notecontent⟩} in the left side. X is to be replaced by the series letter. We do this call in the left side after the word which matches up to the one in the right side after which we want to insert the actual footnote mark.

In the right side, we call \footnoteXmk at the place we want to have the footnote mark. X is to be replaced by the series letter. For example:

```
\begin{Leftside}
 \beginnumbering
  \pstart
   A little cat\footnoteAnomk{A note.}. And so one ...
  \pend
```

---

[2] The same applies to LaTeX normal notes. Read http://tex.stackexchange.com/a/228283/7712 for technical informations.

```
        \endnumbering
      \end{Leftside}
      \begin{Rightside}
       \beginnumbering
        \pstart
         Un petit chat\footnotemk. And so one ...
        \pend
       \endnumbering
      \end{Rightside}
```

5   Left and right texts

5.1   Environments

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
Rightside   in the Rightside environment. The number of Leftside and Rightside environments must be the same.

5.2   Numbering text lines and paragraphs

\beginnumbering   Each section of numbered text must be preceded by \beginnumbering and fol-
\endnumbering   lowed by \endnumbering, like:
\beginnumbering
⟨text⟩
\endnumbering
These have to be separately specified within Leftside and Rightside environments.

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

\memorydump   The command \memorydump effectively performs an \endumbering imme-
diately 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{pages}
  \begin{Leftside}
    \beginnumbering
```

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

It is possible to insert a number at every \pstart command. You must use the
\numberpstarttrue \numberpstarttrue command to have it. You can stop the numbering with
\numberpstartfalse \numberpstartfalse. You can redefine the commands \thepstartL and
\thepstartL \thepstartR to change style. The numbering restarts on each \beginnumber-
\thepstartR ing.
\skipnumbering The command \skipnumbering when inserted in a line of parallel text causes
the numbering of that particular line to be skipped. This can useful if you are putting
some kind of marker (even if it is only a blank line) between stanzas. Remember, parallel
texts must be numbered and this provides a way to slip in an "unnumbered" line.

5.3 Line numbering scheme

Within these environments you can designate the line numbering scheme(s) to be used.

5.4 Lineation system

\firstlinenum Following \firstlinenum{⟨*num*⟩} the first line number will be ⟨*num*⟩, and fol-
\linenumincrement lowing \linenumincrement{⟨*num*⟩} only every ⟨*num*⟩th line will have a printed
\firstsublinenum number. Using these macros inside the Leftside and Rightside environ-
\sublinenumincrement ments gives you independent control over the left and right numbering schemes. The
\firstlinenum* \firstsublinenum and \sublinenumincrement macros correspondingly
\linenumincrement* set the numbering scheme for sublines. The starred versions change both left and right
\firstsublinenum* numbering schemes.
\sublinenumincrement* Generally speaking, controls like \firstlinenum or \linenummargin ap-
ply to sequential and left texts. To effect right texts only, they have to be within a
Rightside environment.
\lineationR \lineationR macro is the equivalent of reledmac \lineation macro for
\lineation* the right side.
```

`\lineation*` macro is the equivalent of `reledmac` `\lineation` macro for both sides.

\setRlineflag    A "R" is appended to the line numbers of the right texts. This may be useful for parallel columns but for parallel pages it might be more appropriate to redefine it using `\setRlineflag{`⟨*flag*⟩`}`. Use `\setRlineflag{}` to empty it.

## 5.5 Chunks

\pstart    In a serial (non-parallel) mode, each numbered paragraph, or chunk, is contained be-
\pend    tween 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` environ- ment) 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.

\autopar   The `\autopar` macro can be used, instead of manually inserting `\pstart`...`\pend`s. Please read `reledmac`'s handbook (4.2.2 p. 15).

5.6 **\AtEveryPstart** and **\AtEveryPstartCall**

In general, remember that the moment where a `\pstart` is called is different from the moment when the `\pstart`...`\pend` content is printed, which is when `\Pages` or `\Columns` is processed.

Consequently:

- The argument of `\AtEveryPstart` (see 4.2.4 p. 16) is called before every chunk is printed, except if you used an optional argument for the `\pstart`.

- The argument of `\AtEveryPstartCall` is called before every `\pstart`.

5.7   Language setting

If you are using the babel package or the polyglossia 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* language setting 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.

5.8   Shifting

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 reledpar with the option shiftedpstarts.

6   Verse

If you are typesetting verse with reledmac 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   reledpar provides an astanza environment which you can use instead of \stanza. To use it, imply replace \stanza by \begin{astanza} and add \end{astanza} after the ending \&. Within the astanza environment each verse line is treated as a paragraph, so there must be no blank lines in the environment otherwise there will be some extraneous vertical spacing.

If you get an error message along the lines of 'Missing number, treated as zero \sza@0@' it is because you have forgotten to use \setstanzaindents to set the stanza indents.

\sethangingsymbol   Like in reledmac, you could use the \sethangingsymbol command to insert a character in each hanging line. If you use it, you must run LaTeX two time. Example for the French typography

\sethangingsymbol{[\,}

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

```
\sethangingsymbol{\protect\hfill}
```

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

7 Side notes

As in `reledmac`, 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.

8 Parallel ledgroups

8.1 General

You can also make parallel ledgroups (see the documentation of `reledmac` about ledgroups, 9 p. 42). To do it you have:

- To load `reledpar` 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}
  \end{pages}
  \Pages
```

8.2   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:

```
 \setparledgroupnotespacing{\singlespacing}
```

In effect, to have correct spacing, do not change the font size of your notes.

9   Sectioning commands

The standard sectioning commands of `reledmac` are available, and provide parallel sectioning, for both two-column and two-page layout.

\eledsectnotoc   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).

\eledsectmark   By default, the headers are tokens from the left side. You can change them, using \eledsectmark{⟨*arg*⟩}, where ⟨*arg*⟩ could be L (for left side) or R (for right side).

10   Notes about page number

If you use `sameparallelpagenumber` option (4.2.2 p. 9 or `prevpgnotnumbered` option (4.2.4 p. 9), please read the following paragraph if you want to manipulate page numbers manually.

In order to implement these two options, `reledpar` uses its own page counter, called `par@page`. Consequently, if you use at least one of these options:

1. If you modify \thepage command, use the value of `par@page` counter inside and not the value of `page` counter.

2. If you want to modify a page number, modify the value of `page` counter AND the value `par@page` counter.

Notes that `reledpar` automatically do it when you use `\frontmatter` and `\mainmatter` commands.

I   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, TEXessentially 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.

reledmac 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 reledmac code is concerned with handling this box and its contents.

reledpar'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.

II   Preliminaries

II.1   Package's meta-data

Announce the name and version of the package, which is targeted for LATEX2e. The package also requires the reledmac package, however we do not load it automatically, because we prefer users to know it.

```
1 %<*code>
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{reledpar}[2015/07/27 v2.1.0 reledmac
  extension for parallel texts]%
4
5 %
```

II.2   Package's requirement

Few commands use \xspace command.

```
6 \RequirePackage{xspace}%
7 %
```

II.3   Package's options

We use xkeyval in order to manage options with arguments.

```
8 \RequirePackage{xkeyval}
9 %
```

With the option 'shiftedpstarts' a long pstart on the left side (or on the right side) does not 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.

`\ifshiftedpstarts`
```
10 \newif\ifshiftedpstarts
11 \DeclareOptionX{shiftedpstarts}{\shiftedpstartstrue}
12 %
```

The `parledgroup` can be called either on `reledmac` or `reledpar`.

```
13 \DeclareOptionX{parledgroup}{\parledgrouptrue}
14 %
```

`fwidthliketwocolumns`   The `\widthliketwocolumns` option can be called either on `reledmac` or `reledpar`.

```
15 \DeclareOptionX{widthliketwocolumns}{\widthliketwocolumnstrue
   }%
16 %
```

`ameparallelpagenumber`   Options related to page numbering
`\ifprevpgnotnumbered`
```
17 \newif\ifsameparallelpagenumber
18 \newif\ifprevpgnotnumbered
19 \DeclareOptionX{sameparallelpagenumber}{\
   sameparallelpagenumbertrue}
20 \DeclareOptionX{prevpgnotnumbered}{\prevpgnotnumberedtrue}
21 %
```

`\prevpgstyle`   We store on `\prevpgstyle` the argument of the option `prevpgstyle`.

```
22 \DeclareOptionX{prevpgstyle}{\gdef\prevpgstyle{#1}}%
23 %
```

```
24 \ProcessOptionsX%
25 %
```

II.4   Determining side and category of parallel processing

As noted above, much of the code is a duplication of the original `reledmac` 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 we use 'R' or 'L' in the names of macros for the right and left code. The specifics of 'L' and 'R' are normally hidden from the user by letting the `Leftside` and `Rightside` environments set things up appropriately.

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

II.5   Text's width

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

```
26 \newdimen\Lcolwidth
27   \Lcolwidth=0.45\textwidth
28 \newdimen\Rcolwidth
29   \Rcolwidth=0.45\textwidth
30 %
```

II.6   Messages

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

`\eledpar@error`
```
31 \newcommand{\reledpar@error}[2]{\PackageError{reledpar
   }{#1}{#2}}
32 %
```

`\led@err@TooManyPstarts`
```
33 \newcommand*{\led@err@TooManyPstarts}{%
34   \reledpar@error{Too many \string\pstart\space without
   printing.
35                 Some text will be lost}{\@ehc}}
36 %
```

`\led@err@BadLeftRightPstarts`
```
37 \newcommand*{\led@err@BadLeftRightPstarts}[2]{%
38   \reledpar@error{The numbers of left (#1) and right (#2)
39                 \string\pstart s do not match}{\@ehc}}
40 %
```

`\led@err@LeftOnRightPage`
`\led@err@RightOnLeftPage`
```
41 \newcommand*{\led@err@LeftOnRightPage}{%
42   \reledpar@error{The left page has ended on a right page}{\
   @ehc}}
43 \newcommand*{\led@err@RightOnLeftPage}{%
44   \reledpar@error{The right page has ended on a left page}{\
   @ehc}}
45 %
```

`err@Leftside@PreviousNotPrinted`
`r@Rightside@PreviousNotPrinted`
```
46 \newcommand*{\led@err@Leftside@PreviousNotPrinted}{%
47   \reledpar@error{You call a new Leftside environment while
   the previous one has not been typeset by \string\Pages\space
   or \string\Columns}{\@ehc}}
48 \newcommand*{\led@err@Rightside@PreviousNotPrinted}{%
49   \reledpar@error{You call a new Rightside environment while
   the previous one has not been typeset by \string\Pages\space
   or \string\Columns}{\@ehc}}
50 %
```

```
@err@Pages@InsideEnv 51  \newcommand*{\led@err@Pages@InsideEnv}{%
err@Columns@InsideEnv 52    \reledpar@error{\string\Pages\space must be called *outside
                        * of the `pages` environment}{\@ehc}}
                     53  \newcommand*{\led@err@Columns@InsideEnv}{%
                     54    \reledpar@error{\string\Columns\space must be called *
                        outside* of the `pairs` environment}{\@ehc}}
                     55  %
```

```
or@fail@patch@thepage 56  \newcommand{\led@error@fail@patch@thepage}{%
                     57    \reledpar@error{Fail to patch \string\@thepage\space
                        command.}{\@ehc}%
                     58  }%
                     59  %
```

```
@patch@pagenumbering 60  \newcommand{\led@error@fail@patch@pagenumbering}{%
                     61    \reledpar@error{Fail to patch \string\pagenumbering\space
                        command.}{\@ehc}%
                     62  }%
                     63  %
```

```
@fail@patch@@mempnum 64  \newcommand{\led@error@fail@patch@@mempnum}{%
                     65    \reledpar@error{Fail to patch \string\@mempnum\space
                        command.}{\@ehc}%
                     66  }%
                     67  %
```

```
il@patch@@outputpage 68  \newcommand{\led@error@fail@patch@@outputpage}{%
                     69    \reledpar@error{Fail to patch \string\@outputpage\space
                        command.}{\@ehc}%
                     70  }%
                     71  %
```

III   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.

```
72  \newcount\section@numR
73    \section@numR=\z@
74  %
```

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

```
75    \pst@rtedLfalse
76  \newif\ifpst@rtedR
77
78  %
```

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

```
79  \newcommand*{\beginnumberingR}{%
80    \ifnumberingR
81      \led@err@NumberingStarted
82      \endnumberingR
83    \fi
84    \global\l@dnumpstartsR \z@
85    \global\pst@rtedRfalse
86    \global\numberingRtrue
87    \global\advance\section@numR \@ne
88    \global\absline@numR \z@
89    \gdef\normal@page@breakR{}
90    \gdef\l@prev@pbR{}
91    \gdef\l@prev@nopbR{}
92    \global\line@numR \z@
93    \global\@lockR \z@
94    \global\sub@lockR \z@
95    \global\sublines@false
96    \global\let\next@page@numR\relax
97    \global\let\sub@change\relax
98    \message{Section \the\section@numR R }%
99    \line@list@stuffR{\jobname.\extensionchars\the\section@numR
  R}%
100   \l@dend@stuff
101   \setcounter{pstartR}{1}
102   \begingroup
103   \initnumbering@sectcountR
104   \gdef\eled@sectionsR@@{}%
105   \if@noeled@sec\else%
106     \makeatletter\InputIfFileExists{\jobname.eledsec\the\
  section@numR R}{}{}\makeatother%
107     \immediate\openout\eled@sectioningR@out=\jobname.eledsec\
  the\section@numR R\relax%
108   \fi%
109  }
110  %
```

\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 reledmac.

\endnumberingR This is the right text equivalent of \endnumbering and must follow the last text for a right text numbered section.

```
111 \def\endnumberingR{%
112   \ifnumberingR
113     \global\numberingRfalse
114     \normal@pars
115     \ifnum\l@dnumpstartsR=0%
116       \led@err@NumberingWithoutPstart%
117     \fi%
118     \ifl@dpairing
119       \global\pst@rtedRfalse
120     \else
121       \ifx\insertlines@listR\empty\else
122         \global\noteschanged@true
123       \fi
124       \ifx\line@listR\empty\else
125         \global\noteschanged@true
126       \fi
127     \fi
128     \ifnoteschanged@
129       \led@mess@NotesChanged
130     \fi
131   \else
132     \led@err@NumberingNotStarted
133   \fi
134   \endgroup
135   \if@noeled@sec\else%
136     \immediate\closeout\eled@sectioningR@out%
137   \fi%
138   }
139
140 %
```

\numbering@sectcountR We do not want the numbering of the right-side section commands to be continuous with the numbering of the left side, we switch the LATEX counter in \numberingR.

```
141 \newcounter{chapterR}
142 \newcounter{sectionR}
143 \newcounter{subsectionR}
144 \newcounter{subsubsectionR}
145 \newcommand{\initnumbering@sectcountR}{
146     \let\c@chapter\c@chapterR
147     \let\c@section\c@sectionR
148     \let\c@subsection\c@subsectionR
149     \let\c@subsubsection\c@subsubsectionR
150 }
151 %
```

\pausenumberingR These are the right text equivalents of \pausenumbering and \resumenum-
\resumenumberingR bering.

```
152 \newcommand*{\pausenumberingR}{%
153   \endnumberingR\global\numberingRtrue}
154 \newcommand*{\resumenumberingR}{%
155   \ifnumberingR
156      \global\pst@rtedRtrue
157      \global\advance\section@numR \@ne
158      \led@mess@SectionContinued{\the\section@numR R}%
159      \line@list@stuffR{\jobname.\extensionchars\the\
    section@numR R}%
160      \l@dend@stuff
161      \begingroup%
162      \initnumbering@sectcountR%
163   \else
164     \led@err@numberingShouldHaveStarted
165     \endnumberingR
166     \beginnumberingR
167   \fi}
168
169 %
```

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

```
170 \newcommand*{\memorydumpL}{%
171   \endnumbering
172   \numberingtrue
173   \global\pst@rtedLtrue
174   \global\advance\section@num \@ne
175      \led@mess@SectionContinued{\the\section@num}%
176   \line@list@stuff{\jobname.\extensionchars\the\section@num}%
177   \l@dend@stuff}
178
179 \newcommand*{\memorydumpR}{%
180   \endnumberingR
181   \numberingRtrue
182   \global\pst@rtedRtrue
183   \global\advance\section@numR \@ne
184      \led@mess@SectionContinued{\the\section@numR R}%
185   \line@list@stuffR{\jobname.\extensionchars\the\section@numR
    R}%
186   \l@dend@stuff}
187
188 %
```

IV   Line counting

IV.1   Setting lineation reset

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. reledpar lets you choose different schemes for the left and right texts.

\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.

```
189 \newcommand*{\lineationR}[1]{{%
190   \ifnumbering
191     \led@err@LineationInNumbered
192   \else
193     \def\@tempa{#1}\def\@tempb{page}%
194     \ifx\@tempa\@tempb
195         \global\bypage@Rtrue
196         \global\bypstart@Rfalse
197         \unless\ifnocritical@%
198           \Xpstart[][false]%
199         \fi%
200     \else
201         \def\@tempb{pstart}%
202         \ifx\@tempa\@tempb
203             \global\bypage@Rfalse
204             \global\bypstart@Rtrue
205             \unless\ifnocritical@%
206               \Xpstart%
207             \fi%
208         \else
209             \def@tempb{section}
210             \ifx\@tempa\@tempb
211               \global\bypage@Rfalse%
212               \global\bypstart@Rfalse%
213               \unless\ifnocritical@%
214                 \Xpstart[][false]%
215               \fi%
216             \else
217                 \led@warn@BadLineation
218             \fi%
219         \fi
220     \fi
221   \fi}}
222 %
```

\lineation*   \lineation* change the lineation system for both sides.

```
223 \WithSuffix\newcommand\lineation*[1]{%
```

```
224     \lineation{#1}%
225     \lineationR{#1}%
226 }%
227 %
```

IV.2   Setting line number margin

\linenummargin   You call \linenummargin{⟨*word*⟩} to specify which margin you want your right
\line@marginR   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 would like; if it is
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.

It is defined only once time, in reledmac.

```
228 \newcount\line@marginR
229 %
```

By default put right text numbers at the right.

```
230 \line@marginR=\@ne
231
232 %
```

IV.3   Setting lineation start and step

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

```
233 \newcounter{firstlinenumR}
234     \setcounter{firstlinenumR}{5}
235 \newcounter{linenumincrementR}
236     \setcounter{linenumincrementR}{5}
237 %
```

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

```
238 \newcounter{firstsublinenumR}
239     \setcounter{firstsublinenumR}{5}
240 \newcounter{sublinenumincrementR}
241     \setcounter{sublinenumincrementR}{5}
242
243 %
```

\firstlinenum These are the user's macros for changing (sub) line numbers. They are defined in
\linenumincrement reledmac. The starred versions are specific to eledpar.
\firstsublinenum
\sublinenumincrement
\firstlinenum*
\linenumincrement*
\firstsublinenum*
\sublinenumincrement*

```
244 \WithSuffix\newcommand\firstlinenum*[1]{%
245   \setcounter{firstlinenumR}{#1}%
246   \setcounter{firstlinenum}{#1}%
247 }
248 \WithSuffix\newcommand\linenumincrement*[1]{%
249   \setcounter{linenumincrementR}{#1}%
250   \setcounter{linenumincrement}{#1}%
251 }
252 \WithSuffix\newcommand\firstsublinenum*[1]{%
253   \setcounter{subfirstlinenumR}{#1}%
254   \setcounter{subfirstlinenum}{#1}%
255 }
256 \WithSuffix\newcommand\sublinenumincrement*[1]{%
257   \setcounter{sublinenumincrementR}{#1}%
258   \setcounter{sublinenumincrement}{#1}%
259 }
260 %
```

IV.4   Setting line flag

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

```
261 \newcommand{\setRlineflag}[1]{%
262   \gdef\@Rlineflag{#1}%
263 }
264 \setRlineflag{R}
265 %
```

IV.5   Setting line number style

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

```
266 \newcommand*{\linenumrepR}[1]{\@arabic{#1}}
267 \newcommand*{\sublinenumrepR}[1]{\@arabic{#1}}
268
269 %
```

IV.6   Print marginal line number

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

```
270 \newcommand*{\leftlinenumR}{%
271   \l@dlinenumR
```

```
272    \kern\linenumsep}
273 \newcommand*{\rightlinenumR}{%
274    \kern\linenumsep
275    \l@dlinenumR}
276 \newcommand*{\l@dlinenumR}{%
277    \numlabfont\linenumrepR{\line@numR}\@Rlineflag%
278    \ifsublines@
279       \ifnum\subline@num>\z@
280          \unskip\fullstop\sublinenumrepR{\subline@numR}%
281       \fi
282    \fi}
283
284 %
```

IV.7   Line-number counters and lists

IV.7.1   Correspond to those in `reledmac` for regular or left text

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

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

```
285 \newcount\line@numR
286 \newcount\subline@numR
287 \newcount\absline@numR
288
289 %
```

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

```
290 \list@create{\line@listR}
291 \list@create{\insertlines@listR}
292 \list@create{\actionlines@listR}
293 \list@create{\actions@listR}
294
295 %
```

\page@numR   The right text page number.

```
296 \newcount\page@numR
297
298 %
```

IV.7.2   Specific to `reledpar`

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

```
299 \list@create{\linesinpar@listL}
300 \list@create{\linesinpar@listR}
301 \list@create{\maxlinesinpar@list}
302
303 %
```

IV.8   Reading the line-list file

\list@clearing@regR   \Clear the right lines for \read@linelist

```
304 \newcommand{\list@clearing@regR}{%
305     \list@clear{\line@listR}%
306     \list@clear{\insertlines@listR}%
307     \list@clear{\actionlines@listR}%
308     \list@clear{\actions@listR}%
309     \list@clear{\linesinpar@listR}%
310     \list@clear{\linesonpage@listR}
311 }
312 %
```

\read@linelist   \read@linelist{⟨*file*⟩} is the control sequence that is called by \beginnum-
bering (via \line@list@stuff) to open and process a line-list file; its argument
is the name of the file. . It is defined only once time in reledmac.

IV.9   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 associ-
ated with.

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

\@nl@regR   \@nl@regR is called by \@nl if we are on a right side. It does everything related to
\@nl   the start of a new line of numbered text on a right side.

```
313 \newcommand{\@nl@regR}{%
314   \ifx\l@dchset@num\relax \else
315     \advance\absline@numR \@ne
316     \set@line@action
317     \let\l@dchset@num\relax
318     \advance\absline@numR \m@ne
319     \advance\line@numR \m@ne%    % do we need this?
320   \fi
321   \advance\absline@numR \@ne
```

```
322    \ifx\next@page@numR\relax \else
323      \page@action
324      \let\next@page@numR\relax
325    \fi
326    \ifx\sub@change\relax \else
327      \ifnum\sub@change>\z@
328        \sublines@true
329      \else
330        \sublines@false
331      \fi
332      \sub@action
333      \let\sub@change\relax
334    \fi
335    \ifcase\@lockR
336    \or
337      \@lockR \tw@
338    \or\or
339      \@lockR \z@
340    \fi
341    \ifcase\sub@lockR
342    \or
343      \sub@lockR \tw@
344    \or\or
345      \sub@lockR \z@
346    \fi
347    \ifsublines@
348      \ifnum\sub@lockR<\tw@
349        \advance\subline@numR \@ne
350      \fi
351    \else
352      \ifnum\@lockR<\tw@
353        \advance\line@numR \@ne \subline@numR \z@
354      \fi
355    \fi}
356
357
358  %
```

\last@page@numR \last@page@numR store the page number of the last right page. It is modified by
\fix@page \fix@page, defined by reledmac.

```
359  \newcount\last@page@numR
360    \last@page@numR=-10000
361
362  %
```

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

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

\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. It is defined in reledmac.

\page@action   \page@action adds an entry to the action-code list to change the page number. It is defined in reledmac.

\set@line@action   \set@line@action adds an entry to the action-code list to change the visible line number. It is defined in reledmac.

\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. It is defined in reledmac.

\do@lockon   \lock@on adds an entry to the action-code list to turn line number locking on. The
\do@lockonR   current setting of the sub-lineation flag tells us whether this applies to line numbers or sub-line numbers. It is defined in reledmac, however the code specific to right side is defined here, in \do@lockonR.

```
363 \newcount\@lockR
364 \newcount\sub@lockR
365
366 \newcommand*{\do@lockonR}{%
367   \xright@appenditem{\the\absline@numR}\to\actionlines@listR
368   \ifsublines@
369     \xright@appenditem{-1005}\to\actions@listR
370     \ifnum\sub@lockR=\z@
371       \sub@lockR \@ne
372     \else
373       \ifnum\sub@lockR=\thr@@
374         \sub@lockR \@ne
375       \fi
376     \fi
377   \else
378     \xright@appenditem{-1003}\to\actions@listR
379     \ifnum\@lockR=\z@
380       \@lockR \@ne
381     \else
382       \ifnum\@lockR=\thr@@
383         \@lockR \@ne
384       \fi
385     \fi
386   \fi}
387
388 %
```

\lock@off   \lock@off adds an entry to the action-code list to turn line number locking off. It
\do@lockoff   is defined in reledmac, however the code specific to right side is defined here, in
\do@lockoffR   \do@lockoffR.
\skip@lockoff

```
389

390

391  \newcommand{\do@lockoffR}{%
392    \xright@appenditem{\the\absline@numR}\to\actionlines@listR
393    \ifsublines@
394      \xright@appenditem{-1006}\to\actions@listR
395      \ifnum\sub@lockR=\tw@
396        \sub@lockR \thr@@
397      \else
398        \sub@lockR \z@
399      \fi
400    \else
401      \xright@appenditem{-1004}\to\actions@listR
402      \ifnum\@lockR=\tw@
403        \@lockR \thr@@
404      \else
405        \@lockR \z@
406      \fi
407    \fi}

408

409

410  %
```

\n@num

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

```
411  \newcount\insert@countR
412  %
```

- #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.

\@ref itself is defined in reledmac. It calls \ref@reg or \ref@regR,
depending weither we are in left or right side. Here, we define only \ref@regR,
\ref@reg is already defined in reledmac.

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

```
413  \newcommand*{\@ref@regR}[2]{%
414      \global\advance\@edtext@level by 1%
415      \global\insert@countR=#1\relax
416      \loop\ifnum\insert@countR>\z@
```

```
417        \xright@appenditem{\the\absline@numR}\to\
  insertlines@listR
418        \global\advance\insert@countR \m@ne
419      \repeat
420 %
```

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.

```
421   \begingroup
422     \let\@ref=\dummy@ref
423     \let\@lopR\@gobble
424     \let\page@action=\relax
425     \let\sub@action=\relax
426     \let\set@line@action=\relax
427     \let\@lab=\relax
428     \let\@lemma=\relax
429     \let\@sw\@gobblethree%
430     #2
431     \global\endpage@num=\page@numR
432     \global\endline@num=\line@numR
433     \global\endsubline@num=\subline@numR
434   \endgroup
435 %
```

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

```
436      \xright@appenditem%
437        {\the\page@numR|\the\line@numR|%
438         \ifsublines@ \the\subline@numR \else 0\fi|%
439         \the\endpage@num|\the\endline@num|%
440         \ifsublines@ \the\endsubline@num \else 0\fi}\to\
  line@listR
441 %
```

Create a list which will store all the second argument of each `\@sw` in this lemma, at this level.

```
442      \expandafter\list@create\expandafter{\csname
  sw@list@edtext@tmp@\the\@edtext@level\endcsname}%
443 %
```

Declare and init boolean for lemma in this level.

```
444      \providebool{lemmacommand@\the\@edtext@level}%
445      \boolfalse{lemmacommand@\the\@edtext@level}%
446 %
```

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

```
447       #2
448 % Now, we store the list of \protect\cs{@sw} of this current
    \protect\cs{edtext} as an element of
449 % the global list of list of \protect\cs{@sw} for a \protect\
    cs{edtext} depth.
450 %      \begin{macrocode}
451      \ifnum\@edtext@level>0%
452        \def\create@this@edtext@level{\expandafter\list@create\
    expandafter{\csname sw@list@edtextR@\the\@edtext@level\
    endcsname}}%
453        \ifcsundef{sw@list@edtextR@\the\@edtext@level}{\
    create@this@edtext@level}{}%
454        \letcs{\@tmp}{sw@list@edtextR@\the\@edtext@level}%
455        \letcs{\@tmpp}{sw@list@edtext@tmp@\the\@edtext@level}%
456        \xright@appenditem{\expandonce\@tmpp}\to\@tmp%
457        \global\cslet{sw@list@edtextR@\the\@edtext@level}{\@tmp
    }%
458      \fi%
459 %
```

Decrease edtext level counter.

```
460      \global\advance\@edtext@level by -1%
461 }
462 %
```

\@pend   \@pend{⟨*num*⟩} adds its argument to the \linesinpar@listL list, and analagously
\@pendR   for \@pendR. If needed, it resets line number. Both are defined in reledmac, but
            they are really defined only in reledpar.

```
463 \renewcommand*{\@pend}[1]{%
464   \ifbypstart@\global\line@num=0\fi%
465   \xright@appenditem{#1}\to\linesinpar@listL}
466 \renewcommand*{\@pendR}[1]{%
467   \ifbypstart@R\global\line@numR=0\fi
468   \xright@appenditem{#1}\to\linesinpar@listR}
469
470 %
```

\@lopL   \@lopL{⟨*num*⟩} adds its argument to the \linesonpage@listL list, and analagously
\@lopR   for \@lopR. Both are defined in reledmac, but they are empty. They are really de-
            fined only in reledpar.

```
471 \renewcommand*{\@lopL}[1]{%
472   \xright@appenditem{#1}\to\linesonpage@listL}
473 \renewcommand*{\@lopR}[1]{%
474   \xright@appenditem{#1}\to\linesonpage@listR}
475
476 %
```

IV.10   Writing to the line-list file

We have now defined all the counters, lists, and commands involved in reading the line-list file at the start of a section. Now we will cover the commands that `reledmac` 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`.

```
477 \newwrite\linenum@outR
478 %
```

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

```
479 \newif\iffirst@linenum@out@R
480   \first@linenum@out@Rtrue
481 %
```

\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.

```
482 \newcommand*{\line@list@stuffR}[1]{%
483   \read@linelist{#1}%
484   \iffirst@linenum@out@R
485     \immediate\closeout\linenum@outR
486     \global\first@linenum@out@Rfalse
487     \immediate\openout\linenum@outR=#1
488     \immediate\write\linenum@outR{\string\line@list@version
   {\this@line@list@version}}%
489   \else
490     \if@minipage%
491       \leavevmode%
492     \fi%
493     \closeout\linenum@outR%
494     \openout\linenum@outR=#1%
495   \fi}
496
497 %
```

\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.

```
498 \newcommand*{\new@lineL}{%
499   \write\linenum@out{\string\@nl[\the\c@page][\thepage]}}
500 %
```

\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.

```
501 \newcommand*{\new@lineR}{%
502   \write\linenum@outR{\string\@nl[\the\c@page][\thepage]}}
503 %
```

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

\startsub   \startsub and \endsub turn sub-lineation on and off, by writing appropriate in-
\endsub   structions to the line-list file. There are both defined in reledmac.

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

\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. It is defined in
               reledmac.

\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. It is defined in reledmac.

\startlock   You can use \startlock or \endlock in running text to start or end line number
\endlock   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. They are defined in
               reledmac.

\skipnumbering

V   Marking text for notes

The \edtext 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
\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. It is defined in reledmac.

V.1   Specific hooks and commands for notes

The reledmac \newseries@ initializes commands which are linked to notes se-
ries. However, to keep reledmac as light as possible, it does not define commands
which are specific to reledpar. This is what does \newseries@par. The specific
hooks are also defined here.

```
\newseries@par 504 \newcommand{\newseries@par}[1]{%
            505 %
```

V.1.1 Notes to be printed on one side only

`reledpar` allows notes to be printed on one side only. We need to declare these options. We also need boolean flags, and to set them to true when a note series is not printed on one side. We check the `nofamiliar` and `nocritical` Eledmac options.

```
506         \unless\ifnofamiliar@%
507           \csgdef{onlysideX@#1}{}%
508           \global\newbool{keepforsideX@#1}%
509         \fi%
510         \unless\ifnocritical@%
511           \global\newbool{keepforXside@#1}%
512           \csgdef{Xonlyside@#1}{}%
513         \fi%
514 %
```

V.1.2 Familiar footnotes without marks

The `\footnoteXnomk` commands are for notes which are printed on the left side, while they are called in the right side. Basically, they set first toggle `\nomark@` to true, then call the `\footnoteX`. and finally add the footnote counter in the footnote counter list.

First, check the `nofamiliar` option of `reledmac`.

```
515     \unless\ifnofamiliar@%
516 % So declare the list.
517 %     \begin{macrocode}
518         \expandafter\list@create\csname footnote#1@mk\endcsname
    %
519 %
```

Then, declare the `\footnoteXnomk` command.

```
520         \expandafter\newcommand\csname footnote#1nomk\endcsname
    [1]{%
521 %
```

First step: just call the normal `\footnoteX`, saying that we do not want to print the mark.

```
522           \toggletrue{nomk@}%
523           \csuse{footnote#1}{##1}%
524           \togglefalse{nomk@}%
525 %
```

Second, and last, step: store the footnote counter in the footnote counters list. We use some `\let`, because `\xright@appenditem` is difficult to use with `\expandafter`.

```
526              \letcs{\@tmp}{footnote#1@mk}%
527              \numdef\@tmpa{\csuse{c@footnote#1}}%
528              \global\xright@appenditem{\@tmpa}\to\@tmp%
529              \global\cslet{footnote#1@mk}{\@tmp}%
530          }%
531  %
```

Then, declare the command which inserts the footnotemark in the right side.

```
532          \expandafter\newcommand\csname footnote#1mk\endcsname{%
533  %
```

Get the first element of the footnote mark list. As `\gl@p` is difficult to use with dynamic name macro, we use `\let` commands.

```
534              \letcs{\@tmp}{footnote#1@mk}%
535              \gl@p\@tmp\to\@tmpa%
536              \global\cslet{footnote#1@mk}{\@tmp}%
537  %
```

Set the footnotecounter with it. For the sake of security, we make a backup of the previous value.

```
538              \letcs{\old@footnote}{c@footnote#1}%
539              \setcounter{footnote#1}{\@tmpa}%
540  %
```

Define the footnote mark and print it

```
541              \protected@csxdef{@thefnmark#1}{\csuse{thefootnote
     #1}}%
542              \csuse{@footnotemark#1}%
543  %
```

Restore previous footnote counter and finally add space.

```
544              \setcounter{footnote#1}{\old@footnote}%
545              \xspace%
546          }%
547  %
```

End of tools for familiar notes without marks

```
548      \fi
549  %
```

End of `\newseries@par`.

```
550  }%
551  %
```

V.1.3   Create hooks

Read the `reledmac` code handbook about `\newhookcommand@series`. Here, we create hooks which are specific to `reledpar`.

```
552  \unless\ifnocritical@%
553    \newhookcommand@series{Xonlyside}%
554  \fi%
555  \unless\ifnofamiliar@%
556    \newhookcommand@series{onlysideX}%
557  \fi
558
559
560  %
```

V.1.4   Init standards series (A,B,C,D,E,Z)

\init@series@par \newseries@par is called by \newseries. However, this last command is
called before reledpar is loaded. Thus, we need to initiate a specific series hook
for reledpar.

```
561  \newcommand{\init@series@par}{%
562    \def\do##1{\newseries@par{##1}}%
563    \dolistloop{\@series}%
564  }%
565  \init@series@par%
566  %
```

VI   Pstart numbers dumping and restoration

While in reledmac the footnotes are inserted in the same time as the \pstart…\pend
are read, in reledpar 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 `567  \list@create{\list@pstartL@pc}%`
\list@pstartR@pc `568  \list@create{\list@pstartR@pc}%`
`569  %`

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

\dump@pstartL@pc `570  \def\dump@pstartL@pc{%`
\dump@pstartR@pc `571    \xright@appenditem{\the\c@pstartL}\to\list@pstartL@pc%`
`572    \global\cslet{numberpstart@L\the\l@dnumpstartsL}{\`
`ifnumberpstart}%`

```
573 }%
574
575 \def\dump@pstartR@pc{%
576   \xright@appenditem{\the\c@pstartR}\to\list@pstartR@pc%
577   \global\cslet{numberpstart@R\the\l@dnumpstartsR}{\
    ifnumberpstart}%
578 }%
579
580 %
```

\restore@pstartL@pc   And so, the commands to restore them.
\restore@pstartR@pc
```
581 \def\restore@pstartL@pc{%
582   \ifx\list@pstartL@pc\empty\else%
583     \gl@p\list@pstartL@pc\to\@temp%
584     \global\c@pstartL=\@temp%
585   \fi%
586 }%
587 \def\restore@pstartR@pc{%
588   \ifx\list@pstartR@pc\empty\else%
589     \gl@p\list@pstartR@pc\to\@temp%
590     \global\c@pstartR=\@temp%
591   \fi%
592 }%
593 %
```

VII   Parallel environments

The initial set up for parallel processing is deceptively simple.
      pairs pages

chapterinpages   The `pairs` environment is for parallel columns and the `pages` environment for par-
allel pages.

```
594 \newenvironment{pairs}{%}
595   \l@dpairingtrue
596   \l@dpagingfalse
597   \initnumbering@quote
598   \at@begin@pairs%
599 }{%
600   \l@dpairingfalse
601 }
602
603 %
```

\AtBeginPairs   The \AtBeginPairs macro just define a \at@begin@pairs macro, called at the
beginning of each `pairs` environments.

```
604 \newcommand{\AtBeginPairs}[1]{\xdef\at@begin@pairs{#1}}%
605 \def\at@begin@pairs{}%

606
607 %
```

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.

```
608 \newenvironment{pages}{%
609   \l@dpairingtrue
610   \l@dpagingtrue
611   \initnumbering@quote
612   \setlength{\Lcolwidth}{\textwidth}%
613   \setlength{\Rcolwidth}{\textwidth}%
614 }{%
615   \l@dpairingfalse
616   \l@dpagingfalse
617 }

618
619 %
```

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

```
620   \newif\ifinstanzaL
621   \newif\ifinstanzaR
622 %
```

`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.

```
623 \newenvironment{Leftside}{%
624   \expandafter\ifvoid\csname l@dLcolrawbox1\endcsname\else%
625    \led@err@Leftside@PreviousNotPrinted%
626   \fi%
627   \ledRcolfalse
628   \setcounter{pstartL}{1}
629   \let\pstart\pstartL
630   \let\thepstart\thepstartL
631   \let\pend\pendL
632   \let\memorydump\memorydumpL
633   \Leftsidehook
634   \let\old@startstanza\@startstanza
635   \def\@startstanza[##1]{\global\instanzaLtrue\
   old@startstanza[##1]}
636 }{
637     \Leftsidehookend}
638 %
```

`\Leftsidehook`
`\Leftsidehookend`
`\Rightsidehook`
`\Rightsidehookend`

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

```
639 \newcommand*{\Leftsidehook}{}
640 \newcommand*{\Leftsidehookend}{}
641 \newcommand*{\Rightsidehook}{}
642 \newcommand*{\Rightsidehookend}{}
643
644 %
```

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.

```
645 \newenvironment{Rightside}{%
646   \expandafter\ifvoid\csname l@dRcolrawbox1\endcsname\else%
647     \led@err@Rightside@PreviousNotPrinted%
648   \fi%
649   \ledRcoltrue
650   \let\beginnumbering\beginnumberingR
651   \let\endnumbering\endnumberingR
652   \let\pausenumbering\pausenumberingR
653   \let\resumenumbering\resumenumberingR
654   \let\memorydump\memorydumpR
655   \let\thepstart\thepstartR
656   \let\pstart\pstartR
657   \let\pend\pendR
658   \let\ledpb\ledpbR
659   \let\lednopb\lednopbR
660   \let\lineation\lineationR
661   \Rightsidehook
662   \let\old@startstanza\@startstanza
663   \def\@startstanza[##1]{\global\instanzaRtrue\
   old@startstanza[##1]}
664 }{%
665   \ledRcolfalse
666   \Rightsidehookend
667 }
668
669 %
```

VIII   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.

VIII.1   Boxes, counters, **\pstart** and **\pend**

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

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

When we first form the paragraph, it goes into a box register, \l@dLcolrawbox or \l@dRcolrawbox for right text, instead of onto the current vertical list. The \ifnumberedpar@ flag will be true while a paragraph is being processed in that way. \num@lines(R) will store the number of lines in the paragraph when it is 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.

```
670  \newcount\num@linesR
671  \newbox\one@lineR
672  \newcount\par@lineR
673  %
```

\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 is 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.

```
674
675  \newcounter{pstartL}
676  \renewcommand{\thepstartL}{{\bfseries\@arabic\c@pstartL}. }
677  \newcounter{pstartR}
678  \renewcommand{\thepstartR}{{\bfseries\@arabic\c@pstartR}. }
679
680  \newcommandx*{\pstartL}[1][1]{%
681    \if@nobreak%
682      \let\@oldnobreak\@nobreaktrue%
683    \else%
684      \let\@oldnobreak\@nobreakfalse%
685    \fi%
686      \@nobreaktrue%
687    \ifluatex%
688      \xdef\l@luatextextdir@L{\the\luatextextdir}%
689      \xdef\l@luatexpardir@L{\the\luatexpardir}%
690      \xdef\l@luatexbodydir@L{\the\luatexbodydir}%
691    \fi%
692    \ifnumbering \else%
693      \led@err@PstartNotNumbered%
694      \beginnumbering%
695    \fi%
696    \ifnumberedpar@%
```

```
697    \led@err@PstartInPstart%
698    \pend%
699  \fi%
700 %
```

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

```
701  \ifpst@rtedL\else%
702    \list@clear{\inserts@list}%
703    \global\let\next@insert=\empty%
704    \global\pst@rtedLtrue%
705  \fi%
706  \begingroup\normal@pars%
707 %
```

When parallel processing we check that we have not exceeded the maximum number of chunks. In any event we grab a box for the forthcoming text.

```
708  \global\advance\l@dnumpstartsL \@ne%
709  \ifnum\l@dnumpstartsL>\l@dc@maxchunks%
710    \led@err@TooManyPstarts%
711    \global\l@dnumpstartsL=\l@dc@maxchunks%
712  \fi%
713  \global\setnamebox{l@dLcolrawbox\the\l@dnumpstartsL}=\vbox\bgroup%
714 %
```

We set all the usual interline penalties to zero; this ensures that there will 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.

```
715  \l@dzeropenalties%
716  \ifautopar\else%
717    \ifnumberpstart%
718      \ifsidepstartnum%
719        \else%
720          \thepstartL%
721        \fi%
722      \fi%
723    \fi%
724  \hsize=\Lcolwidth%
725  \numberedpar@true%
726  \iflabelpstart\protected@edef\@currentlabel%
727      {\p@pstartL\thepstartL}\fi%
728 %
```

Dump the optional arguments

```
729  \ifstrempty{#1}%
730    {\csgdef{before@pstartL@\the\l@dnumpstartsL}{\at@every@pstart}}%
```

```
731    {\csgdef{before@pstartL@\the\l@dnumpstartsL}{\noindent
   #1}}%
732    \at@every@pstart@call%
733 %
```

Gobble following space (automatically done if there is no optional argument)

```
734    \ignorespaces%
735 %


736    }
737 %
```

The same for right side.

```
738 \newcommandx*{\pstartR}[1][1]{%
739    \if@nobreak%
740      \let\@oldnobreak\@nobreaktrue%
741    \else%
742      \let\@oldnobreak\@nobreakfalse%
743    \fi%
744      \@nobreaktrue%
745    \ifluatex%
746      \xdef\l@luatextextdir@R{\the\luatextextdir}%
747      \xdef\l@luatexpardir@R{\the\luatexpardir}%
748      \xdef\l@luatexbodydir@R{\the\luatexbodydir}%
749    \fi%
750    \ifnumberingR \else%
751      \led@err@PstartNotNumbered%
752      \beginnumberingR%
753    \fi%
754    \ifnumberedpar@%
755      \led@err@PstartInPstart%
756      \pendR%
757    \fi%
758    \ifpst@rtedR\else%
759      \list@clear{\inserts@listR}%
760      \global\let\next@insertR=\empty%
761      \global\pst@rtedRtrue%
762    \fi%
763    \begingroup\normal@pars%
764    \global\advance\l@dnumpstartsR \@ne%
765    \ifnum\l@dnumpstartsR>\l@dc@maxchunks%
766      \led@err@TooManyPstarts%
767      \global\l@dnumpstartsR=\l@dc@maxchunks%
768    \fi%
769    \global\setnamebox{l@dRcolrawbox\the\l@dnumpstartsR}=\vbox\
   bgroup%
770      \l@dzeropenalties%
771      \ifautopar\else%
772        \ifnumberpstart%
773          \ifsidepstartnum\else%
```

```
774            \thepstartR%
775          \fi%
776        \fi%
777      \fi%
778    \hsize=\Rcolwidth%
779    \numberedpar@true%
780    \iflabelpstart\protected@edef\@currentlabel%
781        {\p@pstartR\thepstartR}\fi%
782    \ifstrempty{#1}%
783        {\csgdef{before@pstartR@\the\l@dnumpstartsR}{\
   at@every@pstart}}%
784        {\csgdef{before@pstartR@\the\l@dnumpstartsR}{\noindent
   #1}}%
785    \at@every@pstart@call%
786    \ignorespaces%
787    }
788 %
```

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

```
789 \newcommandx*{\pendL}[1][1]{%
790    \ifnumbering \else%
791        \led@err@PendNotNumbered%
792    \fi%
793    \ifnumberedpar@ \else%
794        \led@err@PendNoPstart%
795    \fi%
796 %
```

We immediately call \endgraf to end the paragraph; this ensures that there will be
no large interline penalties to prevent us from slicing the paragraph into pieces.

```
797    \endgraf\global\num@lines=\prevgraf\egroup%
798    \global\par@line=0%
799 %
```

End the group that was begun in the \pstart.

```
800    \endgroup%
801    \ignorespaces%
802    \@oldnobreak%
803    \dump@pstartL@pc%
804    \ifnumberpstart%
805        \addtocounter{pstartL}{1}%
806    \fi
807    \parledgroup@beforenotes@save{L}%
808 %
```

Dump content of the optional argument.

```
809    \ifstrempty{#1}%
```

```
810     {\csgdef{after@pendL@\the\l@dnumpstartsL}{\at@every@pend
    }}%
811     {\csgdef{after@pendL@\the\l@dnumpstartsL}{\noindent#1}}%
812   }
813 %
```

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

```
814 \newcommandx*{\pendR}[1][1]{%
815   \ifnumberingR \else%
816     \led@err@PendNotNumbered%
817   \fi%
818   \ifnumberedpar@ \else%
819     \led@err@PendNoPstart%
820   \fi%
821   \endgraf\global\num@linesR=\prevgraf\egroup%
822   \global\par@lineR=0%
823   \endgroup%
824   \ignorespaces%
825   \@oldnobreak%
826   \dump@pstartR@pc%
827   \ifnumberpstart%
828     \addtocounter{pstartR}{1}%
829   \fi%
830   \parledgroup@beforenotes@save{R}%
831   \ifstrempty{#1}%
832     {\csgdef{after@pendR@\the\l@dnumpstartsR}{\at@every@pend
    }}%
833     {\csgdef{after@pendR@\the\l@dnumpstartsR}{\noindent#1}}%
834 }
835
836 %
```

\AtEveryPstartCall   The \AtEveryPstartCall argument is called when the \pstartL or \pstartR is called. That is different of \AtEveryPstart the argument of which is called when the \pstarts are printed.

```
837 \newcommand{\AtEveryPstartCall}[1]{\gdef\at@every@pstart@call
    {#1}}%
838 \gdef\at@every@pstart@call{}%
839 %
```

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

```
840 \newif\ifprint@last@after@pendL%
841 \newif\ifprint@last@after@pendR%
842 %
```

VIII.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
\l@drightbox   right text.

```
843  \newbox\l@dleftbox
844  \newbox\l@drightbox
845
846  %
```

\countLline   We need to know the number of lines processed.
\countRline
```
847  \newcount\countLline
848    \countLline \z@
849  \newcount\countRline
850    \countRline \z@
851
852  %
```

\@donereallinesL   We need to know the number of 'real' lines output (i.e., those that have been input by
\@donetotallinesL   the user), and the total lines output (which includes any blank lines output for synchro-
\@donereallinesR   nisation).
\@donetotallinesR
```
853  \newcount\@donereallinesL
854  \newcount\@donetotallinesL
855  \newcount\@donereallinesR
856  \newcount\@donetotallinesR
857
858  %
```

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

```
859  \newcommand*{\do@lineL}{%
860    \letcs{\ifnumberpstart}{numberpstart@L\the\l@dpscL}%
861    \advance\countLline \@ne%
862    \ifvbox\namebox{l@dLcolrawbox\the\l@dpscL}%
863    {\vbadness=10000%
864      \splittopskip=\z@%
865      \do@lineLhook%
866      \l@demptyd@ta%
867      \global\setbox\one@line=\vsplit\namebox{l@dLcolrawbox\the\
     l@dpscL}%
868                                  to\baselineskip}%
869    \IfStrEq{\splitfirstmarks\parledgroup@}{begin}{\
     parledgroup@notes@startL}{}%
870    \unvbox\one@line \global\setbox\one@line=\lastbox%
```

```
871    \getline@numL%
872    \ifnum\@lock>\@ne%
873      \inserthangingsymboltrue%
874    \else%
875      \inserthangingsymbolfalse%
876    \fi%
877    \setbox\l@dleftbox%
878    \hb@xt@ \Lcolwidth{%
879      \ifl@dhidenumber%
880        \global\l@dhidenumberfalse%
881        \f@x@l@cks%
882      \else%
883        \affixline@num%
884      \fi%
885      \xifinlist{\the\l@dpscL}{\eled@sections@@}%
886        {\add@inserts\affixside@note}%
887        {\print@lineL}}%
888    \add@penaltiesL%
889    \global\advance\@donereallinesL\@ne%
890    \global\advance\@donetotallinesL\@ne%
891  \else%
892    \setbox\l@dleftbox \hb@xt@ \Lcolwidth{\hspace*{\Lcolwidth}}
    %
893    \global\advance\@donetotallinesL\@ne%
894  \fi}
895
896
897  %
```

\print@lineL   \print@lineL is for lines without a sectioning command. See reledmac defini-
tion of \print@line for handbook.

```
898  \def\print@lineL{%
899      \affixpstart@numL%
900      \l@dld@ta %space kept for backward compatibility
901      \add@inserts\affixside@note%
902      \l@dlsn@te %space kept for backward compatibility
903      {\ledllfill\hb@xt@ \Lcolwidth{%
904              \do@insidelineLhook%
905              \ifluatex%
906                \luatextextdir\l@luatextdir@L%
907              \fi%
908              \new@lineL%
909              \inserthangingsymbolL%
910              \l@dunhbox@line{\one@line}}\ledrlfill\l@drd@ta%
911      \l@drsn@te}}
912
913  %
```

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

```
914  \def\print@eledsectionL{%%
915      \addtocounter{pstartL}{-1}%
916      \ifdefstring{\@eledsectnotoc}{L}{\ledsectnotoc}{}
917      \ifdefstring{\@eledsectmark}{L}{}{\ledsectnomark}
918      \numdef{\temp@}{\l@dpscL-1}%
919      \xifinlist{\temp@}{\eled@sections@@}{\@nobreaktrue}{\
     @nobreakfalse}%
920      \@eled@sectioningtrue%
921      \bgroup%
922        \ifluatex%
923          \luatextextdir\l@luatextextdir@L%
924          \luatexpardir\l@luatexpardir@L%
925          \luatexbodydir\l@luatexbodydir@L%
926          \ifdefstring{\l@luatextextdir@L}{TRT}{\@RTLtrue}{}%
927        \fi%
928        \csuse{eled@sectioning@\the\l@dpscL}%
929      \egroup%
930      \@eled@sectioningfalse%
931      \global\csundef{eled@sectioning@\the\l@dpscL}%
932      \if@RTL%
933        \hspace{-3\paperwidth}%
934      {\hbox{\l@dunhbox@line{\one@line}}  \new@line}%
935      \else%
936        \hspace{3\paperwidth}%
937      {\new@line \hbox{\l@dunhbox@line{\one@line}}}%
938      \fi%
939      \vskip\eledsection@correcting@skip%
940  }
941
942  %
```

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

```
943  \newcommand*{\dolineLhook}[1]{\gdef\do@lineLhook{#1}}%
944  \newcommand*{\dolineRhook}[1]{\gdef\do@lineRhook{#1}}%
945  \newcommand*{\doinsidelineLhook}[1]{\gdef\do@insidelineLhook
     {#1}}%
946  \newcommand*{\doinsidelineRhook}[1]{\gdef\do@insidelineRhook
     {#1}}%
947
948  %
```

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

```
949  \newcommand*{\do@lineLhook}{}
950  \newcommand*{\do@lineRhook}{}
951  \newcommand*{\do@insidelineLhook}{}
952  \newcommand*{\do@insidelineRhook}{}
```

```
953
954 %
```

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

```
955 \newcommand*{\do@lineR}{%
956   \letcs{\ifnumberpstart}{numberpstart@R\the\l@dpscR}%
957   \ledRcol@true%
958   \advance\countRline \@ne%
959   \ifvbox\namebox{l@dRcolrawbox\the\l@dpscR}%
960   {\vbadness=10000%
961    \splittopskip=\z@%
962    \do@lineRhook%
963    \l@demptyd@ta%
964    \global\setbox\one@lineR=\vsplit\namebox{l@dRcolrawbox\the
   \l@dpscR}%
965                            to\baselineskip}%
966   \IfStrEq{\splitfirstmarks\parledgroup@}{begin}{\
   parledgroup@notes@startR}{}%
967   \unvbox\one@lineR \global\setbox\one@lineR=\lastbox%
968   \getline@numR%
969   \ifnum\@lockR>\@ne%
970     \inserthangingsymbolRtrue%
971   \else%
972     \inserthangingsymbolRfalse%
973   \fi%
974   \setbox\l@drightbox%
975   \hb@xt@ \Rcolwidth{%
976     \ifl@dhidenumber%
977       \global\l@dhidenumberfalse%
978       \f@x@l@cksR%
979     \else%
980       \affixline@numR%
981     \fi%
982     \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}%
983      {\add@insertsR\affixside@noteR}%
984      {\print@lineR}%
985   }%
986   \add@penaltiesR%
987   \global\advance\@donereallinesR\@ne%
988   \global\advance\@donetotallinesR\@ne%
989 \else%
990   \setbox\l@drightbox \hb@xt@ \Rcolwidth{\hspace*{\Rcolwidth
   }}%
991   \global\advance\@donetotallinesR\@ne%
992 \fi%
993 \ledRcol@false%
994 }
995
```

```
996
997 %
```

\print@lineR
\print@eledsectionR

VIII.3   Line and page number computation

\getline@numR The \getline@numR macro determines the page and line numbers for the right text
line we are about to send to the vertical list. The \getline@numL is the same for left
text.

```
998  \newcommand*{\getline@numR}{%
999    \global\advance\absline@numR \@ne
1000   \do@actionsR
1001   \do@ballastR
1002   \ifledgroupnotesR@\else
1003        \ifnumberline
1004          \ifsublines@
1005            \ifnum\sub@lockR<\tw@
1006              \global\advance\subline@numR \@ne
1007            \fi
1008          \else
1009            \ifnum\@lockR<\tw@
1010              \global\advance\line@numR \@ne
1011              \global\subline@numR \z@
1012            \fi
1013          \fi
1014        \fi
1015   \fi
1016 }
1017 \newcommand*{\getline@numL}{%
1018   \global\advance\absline@num \@ne
1019   \do@actions
1020   \do@ballast
1021        \ifledgroupnotesL@\else
1022          \ifnumberline
1023            \ifsublines@
1024              \ifnum\sub@lock<\tw@
1025                \global\advance\subline@num \@ne
1026              \fi
1027            \else
1028              \ifnum\@lock<\tw@
1029                \global\advance\line@num \@ne
1030                \global\subline@num \z@
1031              \fi
1032            \fi
1033        \fi
1034   \fi
1035 }
1036
```

```
1037
1038 %
```

\do@ballastR   The real work in the line macros above is done in \do@actions, but before we plunge
into that, let is get \do@ballastR out of the way.

```
1039 \newcommand*{\do@ballastR}{\global\ballast@count=\z@
1040   \begingroup
1041     \advance\absline@numR \@ne
1042     \ifnum\next@actionlineR=\absline@numR
1043       \ifnum\next@actionR>-1001
1044         \global\advance\ballast@count by -\c@ballast
1045       \fi
1046     \fi
1047   \endgroup}
1048 %
```

\l@dskipversenumberR   The \do@actionsR macro looks at the list of actions to take at particular right text
\do@actionsR   absolute line numbers, and does everything that is specified for the current line.
\do@actions@fixedcodeR   It may call itself recursively and we use tail recursion, via \do@actions@nextR
\do@actions@nextR   for this.

```
1049
1050 \newif\ifl@dskipversenumberR
1051 \newcommand*{\do@actions@fixedcodeR}{%
1052   \ifcase\@l@dtempcnta%
1053   \or%                        % 1001
1054     \global\sublines@true
1055   \or%                        % 1002
1056     \global\sublines@false
1057   \or%                        % 1003
1058     \global\@lockR=\@ne
1059   \or%                        % 1004%
1060     \ifnum\@lockR=\tw@
1061       \global\@lockR=\thr@@
1062     \else
1063       \global\@lockR=\z@
1064     \fi
1065   \or%                        % 1005
1066     \global\sub@lockR=\@ne
1067   \or%                        % 1006
1068     \ifnum\sub@lockR=\tw@
1069       \global\sub@lockR=\thr@@
1070     \else
1071       \global\sub@lockR=\z@
1072     \fi
1073   \or%                        % 1007
1074     \l@dskipnumbertrue
1075   \or%                        % 1008
1076     \l@dskipversenumberRtrue%
```

```
1077  \or%                                      % 1009
1078      \l@dhidenumbertrue%
1079  \else%
1080      \led@warn@BadAction
1081  \fi%
1082 }


1085 \newcommand*{\do@actionsR}{%
1086   \global\let\do@actions@nextR=\relax
1087   \@l@dtempcntb=\absline@numR
1088   \ifnum\@l@dtempcntb<\next@actionlineR\else
1089     \ifnum\next@actionR>-1001\relax
1090       \global\page@numR=\next@actionR
1091       \ifbypage@R
1092         \global\line@numR \z@  \global\subline@numR \z@
1093       \fi
1094     \else
1095       \ifnum\next@actionR<-4999\relax    % 9/05 added relax
    here
1096         \@l@dtempcnta=-\next@actionR
1097         \advance\@l@dtempcnta by -5001\relax
1098         \ifsublines@
1099           \global\subline@numR=\@l@dtempcnta
1100         \else
1101           \global\line@numR=\@l@dtempcnta
1102         \fi
1103       \else
1104         \@l@dtempcnta=-\next@actionR
1105         \advance\@l@dtempcnta by -1000\relax
1106         \do@actions@fixedcodeR
1107       \fi
1108     \fi
1109     \ifx\actionlines@listR\empty
1110       \gdef\next@actionlineR{1000000}%
1111     \else
1112       \gl@p\actionlines@listR\to\next@actionlineR
1113       \gl@p\actions@listR\to\next@actionR
1114       \global\let\do@actions@nextR=\do@actionsR
1115     \fi
1116   \fi
1117   \do@actions@nextR}

1119 %
```

VIII.4   Line number printing

\l@dcalcnum   \affixline@numR is the right text version of the \affixline@num macro.
\ch@cksub@l@ckR
\ch@ck@l@ckR
\f@x@l@cksR
\affixline@numR

```
1121 \newcommand*{\l@dcalcnum}[3]{%
1122    \ifnum #1 > #2\relax
1123      \@l@dtempcnta = #1\relax
1124      \advance\@l@dtempcnta by -#2\relax
1125      \divide\@l@dtempcnta by #3\relax
1126      \multiply\@l@dtempcnta by #3\relax
1127      \advance\@l@dtempcnta by #2\relax
1128    \else
1129      \@l@dtempcnta=#2\relax
1130    \fi}
1131
1132 \newcommand*{\ch@cksub@l@ckR}{%
1133    \ifcase\sub@lockR
1134    \or
1135      \ifnum\sublock@disp=\@ne
1136        \@l@dtempcntb \z@  \@l@dtempcnta \@ne
1137      \fi
1138    \or
1139      \ifnum\sublock@disp=\tw@
1140      \else
1141        \@l@dtempcntb \z@  \@l@dtempcnta \@ne
1142      \fi
1143    \or
1144      \ifnum\sublock@disp=\z@
1145        \@l@dtempcntb \z@  \@l@dtempcnta \@ne
1146      \fi
1147    \fi}
1148
1149 \newcommand*{\ch@ck@l@ckR}{%
1150    \ifcase\@lockR
1151    \or
1152      \ifnum\lock@disp=\@ne
1153        \@l@dtempcntb \z@  \@l@dtempcnta \@ne
1154      \fi
1155    \or
1156      \ifnum\lock@disp=\tw@
1157      \else
1158        \@l@dtempcntb \z@  \@l@dtempcnta \@ne
1159      \fi
1160    \or
1161      \ifnum\lock@disp=\z@
1162        \@l@dtempcntb \z@  \@l@dtempcnta \@ne
1163      \fi
1164    \fi}
1165
1166 \newcommand*{\f@x@l@cksR}{%
1167    \ifcase\@lockR
1168    \or
1169      \global\@lockR \tw@
1170    \or \or
```

```
1171      \global\@lockR \z@
1172    \fi
1173    \ifcase\sub@lockR
1174    \or
1175      \global\sub@lockR \tw@
1176    \or \or
1177      \global\sub@lockR \z@
1178    \fi}
1179
1180
1181 \newcommand*{\affixline@numR}{%
1182 \ifledgroupnotesR@\else\ifnumberline
1183 \ifl@dskipnumber
1184    \global\l@dskipnumberfalse
1185 \else
1186    \ifsublines@
1187      \@l@dtempcntb=\subline@numR
1188      \l@dcalcnum{\subline@numR}{\c@firstsublinenumR}{\
   c@sublinenumincrementR}%
1189      \ch@cksub@lockR
1190    \else
1191      \@l@dtempcntb=\line@numR
1192      \ifx\linenumberlist\empty
1193        \l@dcalcnum{\line@numR}{\c@firstlinenumR}{\
   c@linenumincrementR}%
1194      \else
1195        \@l@dtempcnta=\line@numR
1196        \edef\rem@inder{,\linenumberlist,\number\line@numR,}%
1197          \edef\sc@n@list{\def\noexpand\sc@n@list
1198          ####1,\number\@l@dtempcnta,####2|{\def\noexpand\
   rem@inder{####2}}}%
1199        \sc@n@list\expandafter\sc@n@list\rem@inder|%
1200          \ifx\rem@inder\empty\advance\@l@dtempcnta\@ne\fi
1201      \fi
1202      \ch@ck@l@ckR
1203    \fi
1204    \ifnum\@l@dtempcnta=\@l@dtempcntb
1205      \ifl@dskipversenumberR\else
1206        \if@twocolumn
1207          \if@firstcolumn
1208            \gdef\l@dld@ta{\llap{{\leftlinenumR}}}%
1209          \else
1210            \gdef\l@drd@ta{\rlap{{\rightlinenumR}}}%
1211          \fi
1212        \else
1213          \@l@dtempcntb=\line@marginR
1214          \ifnum\@l@dtempcntb>\@ne
1215            \advance\@l@dtempcntb by\page@numR
1216          \fi
1217          \ifodd\@l@dtempcntb
```

```
1218            \gdef\l@drd@ta{\rlap{{\rightlinenumR}}}%
1219          \else
1220            \gdef\l@dld@ta{\llap{{\leftlinenumR}}}%
1221          \fi
1222        \fi
1223      \fi
1224    \fi
1225    \f@x@l@cksR
1226  \fi
1227  \fi
1228  \fi}
1229  %
```

VIII.5   Pstart number printing in side

The printing of the pstart number is like in reledmac, 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 1230
\affixpstart@numR 1231  \newcommand*{\affixpstart@numL}{%
  \leftpstartnumR 1232  \ifsidepstartnum
 \rightpstartnumR 1233  \if@twocolumn
  \leftpstartnumL 1234      \if@firstcolumn
 \rightpstartnumL 1235        \gdef\l@dld@ta{\llap{{\leftpstartnumL}}}%
     \ifpstartnumR 1236      \else
                  1237        \gdef\l@drd@ta{\rlap{{\rightpstartnumL}}}%
                  1238      \fi
                  1239    \else
                  1240      \@l@dtempcntb=\line@margin
                  1241      \ifnum\@l@dtempcntb>\@ne
                  1242        \advance\@l@dtempcntb \page@num
                  1243      \fi
                  1244      \ifodd\@l@dtempcntb
                  1245        \gdef\l@drd@ta{\rlap{{\rightpstartnumL}}}%
                  1246      \else
                  1247        \gdef\l@dld@ta{\llap{{\leftpstartnumL}}}%
                  1248      \fi
                  1249    \fi
                  1250  \fi
                  1251  }
                  1252  \newcommand*{\affixpstart@numR}{%
                  1253  \ifsidepstartnum
                  1254  \if@twocolumn
                  1255      \if@firstcolumn
```

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

VIII.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.

```
1297  \list@create{\inserts@listR}
1298  %
```

\add@insertsR   The right text version.
\add@inserts@nextR

```
1299 \newcommand*{\add@insertsR}{%
1300   \global\let\add@inserts@nextR=\relax
1301   \ifx\inserts@listR\empty \else
1302     \ifx\next@insertR\empty
1303       \ifx\insertlines@listR\empty
1304         \global\noteschanged@true
1305         \gdef\next@insertR{100000}%
1306       \else
1307         \gl@p\insertlines@listR\to\next@insertR
1308       \fi
1309     \fi
1310     \ifnum\next@insertR=\absline@numR
1311       \gl@p\inserts@listR\to\@insertR
1312       \@insertR
1313       \global\let\@insertR=\undefined
1314       \global\let\next@insertR=\empty
1315       \global\let\add@inserts@nextR=\add@insertsR
1316     \fi
1317   \fi
1318   \add@inserts@nextR}
1319
1320 %
```

VIII.7   Penalties

\add@penaltiesL   \add@penaltiesL is the last macro used by \do@lineL. It adds up the club,
\add@penaltiesR   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 are working on at the moment. The count \@l@dtempcnta is used to calculate and accumulate the penalty; it is initially set to the value of \ballast@count,
which has been worked out in \do@ballast. Finally, the penalty is checked to see
that it does not go below −10000.

```
\newcommand*{\add@penaltiesR}{\@l@dtempcnta=\ballast@count
  \ifnum\num@linesR>\@ne
    \global\advance\par@lineR \@ne
    \ifnum\par@lineR=\@ne
      \advance\@l@dtempcnta by \clubpenalty
    \fi
    \@l@dtempcntb=\par@lineR \advance\@l@dtempcntb \@ne
    \ifnum\@l@dtempcntb=\num@linesR
      \advance\@l@dtempcnta by \widowpenalty
    \fi
```

```
      \ifnum\par@lineR<\num@linesR
        \advance\@l@dtempcnta by \interlinepenalty
      \fi
  \fi
    \ifnum\@l@dtempcnta=\z@
      \relax
    \else
      \ifnum\@l@dtempcnta>-10000
        \penalty\@l@dtempcnta
      \else
        \penalty -10000
      \fi
  \fi}
```

This is for a single chunk. However, as we are probably dealing with several chunks at a time, the above is nor really relevant. Peter Wilson thinks 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, Peter Wilson ends up with the following.

```
1321 \newcommand*{\add@penaltiesL}{}
1322 \newcommand*{\add@penaltiesR}{}
1323
1324 %
```

VIII.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.

```
1325 \newcommand*{\flush@notesR}{%
1326   \@xloop
1327     \ifx\inserts@listR\empty \else
1328       \gl@p\inserts@listR\to\@insertR
1329       \@insertR
1330       \global\let\@insertR=\undefined
1331   \repeat}
1332
1333 %
```

IX   Footnotes

IX.1   Line number printing

The \printlinesR 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 V.9 p. 80 of Eledmac'

handbook: 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. Just a reminder of the arguments:

```
\printlinesR   #1      | #2 | #3     | #4     | #5 | #6      | #7
\printlinesR start-page | line | subline | end-page | line | sub-
line | font
```

```
1334 \def\printlinesR#1|#2|#3|#4|#5|#6|#7|{\begingroup
1335   \setprintlines{#1}{#2}{#3}{#4}{#5}{#6}%
1336   \ifl@d@pnum #1\fullstop\fi
1337   \ifledplinenum \linenumr@p{#2}\@Rlineflag\else \symplinenum
\fi
1338   \ifl@d@ssub \fullstop \sublinenumr@p{#3}\fi
1339   \ifl@d@dash \endashchar\fi
1340   \ifl@d@pnum #4\fullstop\fi
1341   \ifl@d@elin \linenumr@p{#5}\@Rlineflag\fi
1342   \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumr@p{#6}\fi
1343 \endgroup}
1344
1345
1346 %
```

IX.2 Footnotes output specific to **\Pages**

print@Xnotes@forpages   The \Xonlyside and \onlysideX hooks for \Pages allow notes to be printed
correct@Xfootins@box   either in left or right pages only.   The implementation of such features is dele-
print@notesX@forpages   gated to \print@Xnotes@forpages, which replaces \print@Xnotes inside
correct@footinsX@box   \Pages. Here is how we proceed[3]:

- If notes are to be printed in both sides, we just proceed the usual way: print the foot starts for the series, then the foot group.

- If notes are to be printed in the left side, we do these prints only for even pages ; if notes are to be printed in the right side, we do these prints only for odd pages.

- However, that is not enough. Because the problem does not only consists in printing notes in any particular page. It is also not to put aside room for notes in the pages where we do not want to print them. To take an example: if some note in the left side is too long by 160pt to be printed in full in the left page, we do not want to put aside 160pt a space for it in the following right page.

- To solve this problem, we change the magnification factor associated with notes before going to the next page. If we start a page where no notes are supposed to be printed, the magnification counter is set to 0. We also set the note skip to 0pt. Before starting a new page where these notes are supposed to be printed, we reset

---

[3]See http://tex.stackexchange.com/a/230332/7712.

these counter and skip to their default values. (About these counter and skip, read *The TeXbook* p. 122-125).

- There still remains a last problem. This problem is quite complex to understand, so an example will speak for itself. Suppose we allow 10 lines of notes by page. Suppose a long note, be it 25 lines, which needs three pages to be printed. Suppose it must be printed only on left pages, namely odd pages.

  On p. 2, the first 10 lines of the notes are printed. On p. 3, the box associated to the notes contains 10 lines. However, as we are in a right page, we do not void this box. So TEX will keep its content for the pages to come. However, on p. 4 it will also add one line in the footnote box, because in any case, TEXadds some content in the box when preparing the output routines, even if there is some content left in this box from the previous pages. So the lines in the note box at p. 4 will be $10 + 1 = 11$. There is one line which should not be there. Furthermore, as the box size is for 10 lines and not for 11 lines, this last line will be glued to the previous one.

  To fix this double issue:

  – For the pages where notes must be NOT printed, we allow to every note box one line less than it ought to be. In our example, that means that we allow TEX to add only $10 - 1 = 9$ line in the note box on p. 3. Before shifting to the pages where notes must be printed, we allow to every notes the expected number of lines. In our example, that means that we allow TEXto add 10 lines in the note box on p. 4. As on p. 3 only 9 lines were allowed, that means note box of p. 4 will contain $9 + 1 = 10$ lines. So the "one line too many" problem is solved.

  – Still remains the "glue" problem. We solve it by recreating a clean note box. We split the one which is created by TEX to get the next line printed. Then, we create the new box, by bringing together the first part and the last part of the split box, adding some skip between them. That is achieved by `\correct@Xfootins@box` (or `\correct@footinsX@box` for familiar notes).

The code to print critical notes, when processing `\Pages`.

```
1347  \newcommand\print@Xnotes@forpages[1]{%
1348  %
```

First case: notes are for both sides. Just print the note start and the note group

```
1349      \ifcsempty{Xonlyside@#1}{%
1350        \csuse{#1footstart}{#1}%
1351        \csuse{#1footgroup}{#1}%
1352      }%
1353  %
```

Second case: notes are for one side only. First test if we are in a page where they must be printed.

```
1354     {%
1355       \ifboolexpr{%
1356         ((test {\ifcsstring{Xonlyside@#1}{L}} and not test{\
      ifnumodd{\c@page}})%
1357         or%
1358         (test {\ifcsstring{Xonlyside@#1}{R}} and test{\
      ifnumodd{\c@page}}))%
1359         }%
1360 %
```

If we are in a page where notes must be printed, print the notes, after having made the corrections which are needed for boxes.

```
1361         {%
1362           \correct@Xfootins@box{#1}%
1363           \csuse{#1footstart}{#1}%
1364           \csuse{#1footgroup}{#1}%
1365 %
```

Then, say not to keep room for notes in the next page.

```
1366           \global\count\csuse{#1footins}=0%
1367           \global\skip\csuse{#1footins}=0pt%
1368 %
```

And also, allow one line less for notes in the next page.

```
1369           \csuse{Xnotefontsize@#1}%
1370           \global\advance\dimen\csuse{#1footins} by -\
      baselineskip%
1371 %
```

Now we have printed the notes. So we put aside this fact.

```
1372           \global\boolfalse{keepforXside@#1}%
1373         }%
1374 %
```

In case we are on a page where notes must NOT be printed. First, memorize that we have not printed the notes, despite having some to print.

```
1375         {%
1376           \global\booltrue{keepforXside@#1}%
1377 %
```

Then restore expected rooms for notes on the next page.

```
1378           \global\count\csuse{#1footins}=\csuse{default@#1
      footins}%
1379           \global\skip\csuse{#1footins}=\csuse{Xbeforenotes@
      #1}%
1380 %
```

Last but not least, restore the normal line number allowed to notes for the following page.

```
1381              \bgroup%
1382                  \csuse{Xnotefontsize@#1}%
1383                  \global\advance\dimen\csuse{#1footins} by \
      baselineskip%
1384              \egroup%
1385 %
```

```
1386 % End of \protect\cs{print@Xnotes@forpages}.
1387          }%
1388        }%
1389 }%
1390 %
```

Now, `\correct@Xfootins@box`, to fix problem of last line being glued to the previous one.

```
1391 \newcommand{\correct@Xfootins@box}[1]{%
1392 %
```

We need to make correction only in case we have not printed any note in the previous page, although there was to be "normally" printed.

```
1393    \ifbool{keepforXside@#1}{%
1394 %
```

Some setting needed to do the right splitting.

```
1395          \csuse{Xnotefontsize@#1}%
1396          \splittopskip=0pt%
1397 %
```

And now, split the last line, and push in the right place.

```
1398          \global\setbox\csuse{#1footins}=\vbox{%
1399            \vsplit\csuse{#1footins} to \dimexpr\ht\csuse{#1
      footins}-1pt\relax%
1400            \vskip \dimexpr-0.5\baselineskip-0.5\lineskip-0.5pt\
      relax%
1401            \unvbox\csuse{#1footins}%
1402          }%
1403 %
```

End of the macro.

```
1404    }{}%
1405 }%
1406 %
```

And now, the same for familiar footnotes.

```
1407 \newcommand\print@notesX@forpages[1]{%
1408    \ifcsempty{onlysideX@#1}{%
1409      \csuse{footstart#1}{#1}%
1410      \csuse{footgroup#1}{#1}%
1411    }%
```

```
1412      {%
1413        \ifboolexpr{%
1414          ((test {\ifcsstring{onlysideX@#1}{L}} and not test{\
      ifnumodd{\c@page}})%
1415          or%
1416          (test {\ifcsstring{onlysideX@#1}{R}} and test{\
      ifnumodd{\c@page}}))%
1417          }%
1418          {%
1419            \correct@footinsX@box{#1}%
1420            \csuse{footstart#1}{#1}%
1421            \csuse{footgroup#1}{#1}%
1422            \global\count\csuse{footins#1}=0%
1423            \global\skip\csuse{footins#1}=0pt%
1424            \csuse{notefontsizeX@#1}%
1425            \global\advance\dimen\csuse{footins#1} by -\
      baselineskip%
1426          \global\boolfalse{keepforsideX@#1}%
1427          }%
1428          {%
1429            \global\booltrue{keepforsideX@#1}%
1430            \global\count\csuse{footins#1}=\csuse{
      default@footins#1}%
1431            \global\skip\csuse{footins#1}=\csuse{beforenotesX@
      #1}%
1432            \bgroup%
1433              \csuse{notefontsizeX@#1}%
1434              \global\advance\dimen\csuse{footins#1} by \
      baselineskip%
1435            \egroup%
1436          }%
1437        }%
1438  }%
1439  \newcommand{\correct@footinsX@box}[1]{%
1440    \ifbool{keepforsideX@#1}{%
1441        \csuse{notefontsizeX@#1}%
1442        \splittopskip=0pt%
1443        \global\setbox\csuse{footins#1}=\vbox{%
1444          \vsplit\csuse{footins#1} to \dimexpr\ht\csuse{footins
      #1}-1pt\relax%
1445          \vskip \dimexpr-0.5\baselineskip-0.5\lineskip-0.5pt\
      relax%
1446          \unvbox\csuse{footins#1}%
1447        }%
1448    }{}%
1449  }%
1450  %
```

X 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.

```
1451 \list@create{\labelref@listR}
1452
1453 %
```

\edlabel   This command is defined only one time in reledmac, including features for reled-par.

\l@dmake@labelsR   This is the right text version of \l@dmake@labels, taking account of \@Rline-flag.

```
1454 \def\l@dmake@labelsR#1|#2|#3|#4|#5{%
1455   \expandafter\ifx\csname the@label#5\endcsname \relax\else
1456     \led@warn@DuplicateLabel{#4}%
1457   \fi
1458   \expandafter\gdef\csname the@label#5\endcsname{#1|#2\
@Rlineflag|#3|#4}%
1459   \ignorespaces}
1460 \AtBeginDocument{%
1461   \def\l@dmake@labelsR#1|#2|#3|#4|#5{}%
1462 }
1463
1464 %
```

\@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.

     It is defined on reledmac.

XI Side notes

Regular \marginpars do not work inside numbered text — they do not produce any note but do put an extra unnumbered blank line into the text.

\sidenote@marginR   Specifies which margin sidenotes can be in.

\sidenotemargin*

```
1465 \WithSuffix\newcommand\sidenotemargin*[1]{%
1466   \l@dgetsidenote@margin{#1}
1467   \global\sidenote@marginR=\@l@dtempcntb
1468   \global\sidenote@margin=\@l@dtempcntb
1469 }
1470 \newcount\sidenote@marginR
1471 \global\sidenote@margin=\@ne
1472
1473 %
```

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

```
1474 \newcommand*{\affixside@noteR}{%
1475     \def\sidenotecontent@{}%
1476     \numgdef{\itemcount@}{0}%
1477     \def\do##1{%
1478         \ifnumequal{\itemcount@}{0}%
1479             {%
1480             \appto\sidenotecontent@{##1}}% Not print not
     separator before the 1st note
1481             {\appto\sidenotecontent@{\sidenotesep ##1}%
1482             }%
1483             \numgdef{\itemcount@}{\itemcount@+1}%
1484     }%
1485     \dolistloop{\l@dcsnotetext}%
1486     \ifnumgreater{\itemcount@}{1}{\led@err@ManySidenotes}{}%
1487   \gdef\@templ@d{}%
1488   \gdef\@templ@n{\l@dcsnotetext\l@dcsnotetext@l\
   l@dcsnotetext@r}%
1489   \ifx\@templ@d\@templ@n \else%
1490     \if@twocolumn%
1491       \if@firstcolumn%
1492         \setl@dlp@rbox{##1}{\sidenotecontent@}%
1493       \else%
1494         \setl@drp@rbox{\sidenotecontent@}%
1495       \fi%
1496     \else%
1497       \@l@dtempcntb=\sidenote@marginR%
1498       \ifnum\@l@dtempcntb>\@ne%
1499         \advance\@l@dtempcntb by\page@numR%
1500       \fi%
1501       \ifodd\@l@dtempcntb%
1502         \setl@drp@rbox{\sidenotecontent@}%
1503         \gdef\sidenotecontent@{}%
1504         \numdef{\itemcount@}{0}%
1505         \dolistloop{\l@dcsnotetext@l}%
1506         \ifnumgreater{\itemcount@}{1}{\led@err@ManyLeftnotes
   }{}%
1507         \setl@dlp@rbox{\sidenotecontent@}%
1508       \else%
1509         \setl@dlp@rbox{\sidenotecontent@}%
1510         \gdef\sidenotecontent@{}%
1511         \numdef{\itemcount@}{0}%
1512         \dolistloop{\l@dcsnotetext@r}%
1513         \ifnumgreater{\itemcount@}{1}{\led@err@ManyRightnotes
   }{}%
1514         \setl@drp@rbox{\sidenotecontent@}%
1515       \fi%
1516     \fi%
1517   \fi%
```

```
1518 }
1519
1520 %
```

XII   Familiar footnotes

`\l@dbfnote`   `\l@dbfnote` adds the footnote to the insert list, and `\vl@dbfnote` calls the original `\@footnotetext`.   There are both defined in `reledmac`.

`\normalbfnoteX`

XIII   Verse

Like in `reledmac`, the insertion of hangingsymbol is base on `\ifinserthang-ingsymbol`, and, for the right side, on `\ifinserthangingsymbolR`. Both commands also include the hanging space, to be sure the `\one@line` of hanging lines has the same width that the `\one@line` of normal lines and to prevent the column separator from shifting.

`\inserthangingsymbolL`
`\inserthangingsymbolR`

```
1521 \newif\ifinserthangingsymbolR
1522 \newcommand{\inserthangingsymbolL}{%
1523   \ifinserthangingsymbol%
1524     \ifinstanzaL%
1525         \hskip \@ifundefined{sza@0@}{0}{\expandafter%
1526           \noexpand\csname sza@0@\endcsname}\stanzaindentbase
%
1527         \@hangingsymbol%
1528     \fi%
1529   \fi%
1530 }%
1531 \newcommand{\inserthangingsymbolR}{%
1532   \ifinserthangingsymbolR%
1533     \ifinstanzaR%
1534         \hskip \@ifundefined{sza@0@}{0}{\expandafter%
1535           \noexpand\csname sza@0@\endcsname}\
stanzaindentbase%
1536         \@hangingsymbol%
1537     \fi%
1538   \fi%
1539 }%
1540 %
```

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.

```
1541   \chardef\next=\catcode`\&
1542   \catcode`\&=\active
```

```
1543
1544 %
```

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

```
1545 \newenvironment{astanza}{%
1546   \catcode`\&\active
1547   \global\stanza@count\@ne\stanza@modulo\@ne
1548   \ifnum\usenamecount{sza@0@}=\z@
1549     \let\stanza@hang\relax
1550     \let\endlock\relax
1551   \else
1552     \rightskip\z@ plus 1fil\relax
1553   \fi
1554   \ifnum\usenamecount{szp@0@}=\z@
1555     \let\sza@penalty\relax
1556   \fi
1557   \def&{%
1558     \endlock\mbox{}%
1559     \sza@penalty
1560     \global\advance\stanza@count\@ne
1561     \@astanza@line}%
1562   \def\&{\@stopastanza}%
1563   \pstart
1564   \@astanza@line
1565 }{}
1566
1567 %
```

\@stopastanza   This command is called by \& in astanza environment. It allows optional arguments.

```
1568 \newcommandx{\@stopastanza}[1][1,usedefault]{%
1569     \endlock\mbox{}%
1570     \pend[#1]%
1571 }%
1572 %
```

\@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.

```
1573 \newcommand*{\@astanza@line}{%
1574   \ifnum\value{stanzaindentsrepetition}=0
1575     \parindent=\csname sza@\number\stanza@count
1576               @\endcsname\stanzaindentbase
1577   \else
1578     \parindent=\csname sza@\number\stanza@modulo
1579               @\endcsname\stanzaindentbase
1580     \managestanza@modulo
1581   \fi
1582   \par
```

```
1583      \stanza@hang%\mbox{}%
1584      \ignorespaces}
1585
1586 %
```

Lastly reset the modified category codes.

```
1587      \catcode`\&=\next
1588
1589 %
```

XIV   Naming macros

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

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

```
1590 \providecommand*{\newnamebox}[1]{%
1591    \expandafter\newbox\csname #1\endcsname}
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
1601 %
```

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

```
1602 \providecommand*{\newnamecount}[1]{%
1603    \expandafter\newcount\csname #1\endcsname}
1604 \providecommand*{\usenamecount}[1]{%
1605                      \csname #1\endcsname}
1606
1607 %
```

XV   Fixing babel and polyglossia

With parallel texts there is the possibility that the two sides might use different languages via `babel`. On the other hand, nor `babel` nor `polyglossia` might not be called at all (even though it might be already built into the format).

With the normal sequential text each line is initially typeset in the current language environment, and then it is output at which time its attachments are typeset (in the same language environment. In the parallel case lines are typeset in their current language but an attachment might be typeset outside the language environment of its line if the left and right side languages are different. To counter this, we have to make sure that the correct language is used at the proper times.

`\ifl@dusedbabel`
`\l@dusedbabelfalse`
`\l@dusedbabeltrue`

A flag for checking if `babel` has been used as a package.

```
1608  \newif\ifl@dusedbabel
1609  %
```

`\l@dchecklang`

`\bbl@set@language`   In `babel` the macro `\bbl@set@language{`⟨*lang*⟩`}` does the work when the language ⟨*lang*⟩ is changed via `\selectlanguage`. Unfortunately for us, if it is given an argument in the form of a control sequence it strips off the `\` character rather than expanding the command. We need a version that accepts an argument in the form `\lang` without it stripping the `\`.

```
1610  \patchcmd{\bbl@set@language}%
1611    {\select@language{\languagename}}%
1612    {\edef\languagename{#1}\select@language{\languagename}}%
1613    {}%
1614    {}%
1615
1616  %
```

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

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

```
1617  \newcommand*{\l@duselanguage}[1]{}
1618  \gdef\theledlanguageL{}
1619  \gdef\theledlanguageR{}
1620
1621  %
```

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

```
1622  \AtBeginDocument{%
1623    \@ifundefined{xpg@main@language}{%
1624      \@ifundefined{bbl@main@language}{%
1625  %
```

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

```
1626        \l@dusedbabelfalse
1627        }{%
1628 %
```

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.

```
1629        \l@dusedbabeltrue
1630        \let\l@doldselectlanguage\selectlanguage
1631        \let\l@doldbbl@set@language\bbl@set@language
1632        \renewcommand{\selectlanguage}[1]{%
1633          \l@doldselectlanguage{#1}%
1634          \ifledRcol \gdef\theledlanguageR{#1}%
1635          \else       \gdef\theledlanguageL{#1}%
1636          \fi}
1637 %
```

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

```
1638        \renewcommand*{\l@duselanguage}[1]{%
1639          \l@doldselectlanguage{#1}}
1640 %
```

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

```
1641        \gdef\theledlanguageL{\bbl@main@language}%
1642        \gdef\theledlanguageR{\bbl@main@language}%
1643        }%
1644      }
1645 %
```

If use `polyglossia`

```
1646    {      \let\old@otherlanguage\otherlanguage%
1647        \renewcommand{\otherlanguage}[2][]{%
1648          \selectlanguage[#1]{#2}%
1649          \ifledRcol  \gdef\theledlanguageR{#2}%
1650          \else        \gdef\theledlanguageL{#2}%
1651          \fi}%
1652        \let\l@duselanguage\select@language%
1653        \gdef\theledlanguageL{\xpg@main@language}%
1654        \gdef\theledlanguageR{\xpg@main@language}%
1655 %
```

That is it.

```
1656 }}
1657 %
```

XVI   Counts and boxes for parallel texts

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

`\maxchunks`
`\l@dc@maxchunks`

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

```
1658 \newcount\l@dc@maxchunks
1659 \newcommand{\maxchunks}[1]{\l@dc@maxchunks=#1}
1660   \maxchunks{5120}
1661
1662 %
```

`\l@dnumpstartsL`
`\l@dnumpstartsR`

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

```
1663 \newcount\l@dnumpstartsR
1664
1665 %
```

`\l@pscL`
`\l@pscR`

A couple of scratch counts for use in left and right texts, respectively.

```
1666 \newcount\l@dpscL
1667 \newcount\l@dpscR
1668
1669 %
```

`\l@dsetuprawboxes`

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

```
1670 \newcommand*{\l@dsetuprawboxes}{%
1671   \@l@dtempcntb=\l@dc@maxchunks
1672   \loop\ifnum\@l@dtempcntb>\z@
1673     \newnamebox{l@dLcolrawbox\the\@l@dtempcntb}
1674     \newnamebox{l@dRcolrawbox\the\@l@dtempcntb}
1675     \advance\@l@dtempcntb \m@ne
1676   \repeat}
1677
1678 %
```

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

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

```
1679 \newcommand*{\l@dsetupmaxlinecounts}{%
1680   \@l@dtempcntb=\l@dc@maxchunks
1681   \loop\ifnum\@l@dtempcntb>\z@
1682     \newnamecount{l@dmaxlinesinpar\the\@l@dtempcntb}
1683     \advance\@l@dtempcntb \m@ne
1684   \repeat}
1685 \newcommand*{\l@dzeromaxlinecounts}{%
1686   \begingroup
1687   \@l@dtempcntb=\l@dc@maxchunks
1688   \loop\ifnum\@l@dtempcntb>\z@
1689     \global\usenamecount{l@dmaxlinesinpar\the\@l@dtempcntb}=\
      z@
1690     \advance\@l@dtempcntb \m@ne
1691   \repeat
1692   \endgroup}
1693
1694 %
```

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

```
1695 \AtBeginDocument{%
1696   \l@dsetuprawboxes
1697   \l@dsetupmaxlinecounts
1698   \l@dzeromaxlinecounts
1699   \l@dnumpstartsL=\z@
1700   \l@dnumpstartsR=\z@
1701   \l@dpscL=\z@
1702   \l@dpscR=\z@}
1703
1704 %
```

XVII   Checking text to be processed

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

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

```
1705 \newif\if@pstarts
1706 \newcommand*{\check@pstarts}{%
1707   \@pstartsfalse
1708   \ifnum\l@dnumpstartsL>\l@dpscL
1709     \@pstartstrue
1710   \else
1711     \ifnum\l@dnumpstartsR>\l@dpscR
1712       \@pstartstrue
1713     \fi
1714   \fi
1715 }
1716
1717 %
```

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

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

```
1718 \newif\ifaraw@text
1719 \newcommand*{\checkraw@text}{%
1720   \araw@textfalse
1721   \ifvbox\namebox{l@dLcolrawbox\the\l@dpscL}
1722     \araw@texttrue
1723   \else
1724     \ifvbox\namebox{l@dRcolrawbox\the\l@dpscR}
1725       \araw@texttrue
1726     \fi
1727   \fi
1728 }
1729
1730 %
```

\@writelinesinparL
\@writelinesinparR

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

```
1731 \newcommand*{\@writelinesinparL}{%
1732   \edef\next{%
1733     \write\linenum@out{\string\@pend[\the\@donereallinesL]}}%
1734   \next
1735   \global\@donereallinesL \z@}
1736 \newcommand*{\@writelinesinparR}{%
1737   \edef\next{%
1738     \write\linenum@outR{\string\@pendR[\the\@donereallinesR
]}}%
1739   \next
1740   \global\@donereallinesR \z@}
1741
1742 %
```

XVIII   Parallel columns

\@eledsectionL
\@eledsectionR

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

```
1743 \newsavebox{\@eledsectionL}%
1744 \newsavebox{\@eledsectionR}%
1745 %
```

\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.

```
1746 \newcommand*{\Columns}{%
1747   \ifl@dpairing%
1748     \led@err@Columns@InsideEnv%
```

```
1749    \fi%
1750    \l@dprintingcolumnstrue%
1751    \eledsection@correcting@skip=-\baselineskip% Correction for
     sections' titles
1752    \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
1753      \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\
     l@dnumpstartsR}%
1754    \fi
1755 %
```

Start a group and zero counters, etc.

```
1756    \begingroup
1757      \l@dzeropenalties
1758      \endgraf\global\num@lines=\prevgraf
1759              \global\num@linesR=\prevgraf
1760      \global\par@line=\z@
1761      \global\par@lineR=\z@
1762      \global\l@dpscL=\z@
1763      \global\l@dpscR=\z@
1764 %
```

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

```
1765      \check@pstarts
1766      \loop\if@pstarts
1767          \global\pstartnumtrue
1768          \global\pstartnumRtrue
1769 %
```

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.

```
1770          \global\advance\l@dpscL \@ne
1771          \global\advance\l@dpscR \@ne
1772          \restore@pstartL@pc%
1773          \restore@pstartR@pc%
1774 %
```

We print the optional argument of \pstart or the argument of \AtEveryPstart.

```
1775          \Columns@print@before@pstart%
1776 %
```

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

```
1777          \checkraw@text
1778 {          \loop\ifaraw@text
1779 %
```

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

```
1780              \l@duselanguage{\theledlanguageL}%
1781              \do@lineL
1782              \xifinlist{\the\l@dpscL}{\eled@sections@@}
1783                  {%
1784                  \ifdefstring{\@eledsectmark}{L}%
1785                    {\csuse{eled@sectmark@\the\l@dpscL}%
1786                    }{}%
1787                  \global\csundef{eled@sectmark@\the\l@dpscL}%
1788                  \savebox{\@eledsectionL}{\parbox[t][][t]{\
   Lcolwidth}{\vbox{}\print@eledsectionL}}%\vbox{}-> prevent
   alignment troubles with RTL language
1789                  }%
1790                  {}%
1791              \l@duselanguage{\theledlanguageR}%
1792              \do@lineR
1793              \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}
1794                  {%
1795                  \ifdefstring{\@eledsectmark}{R}%
1796                    {\csuse{eled@sectmark@\the\l@dpscR R}%
1797                    }{}%
1798                  \global\csundef{eled@sectmark@\the\l@dpscR R}
   %
1799                  \savebox{\@eledsectionR}{\parbox[t][][t]{\
   Rcolwidth}{\vbox{}\print@eledsectionR}}}%\vbox{}-> prevent
   alignment troubles with RTL language
1800                  {}%
1801          \hb@xt@ \hsize{%
1802            \ifdefstring{\columns@position}{L}{}{\hfill }%
1803            \unhbox\l@dleftbox%
1804            \ifhbox\@eledsectionL%
1805                \usebox{\@eledsectionL}%
1806            \fi%
1807            \print@columnseparator%
1808            \unhbox\l@drightbox%
1809            \ifhbox\@eledsectionR%
1810              \usebox{\@eledsectionR}%
1811            \fi%
1812            \ifdefstring{\columns@position}{R}{}{\hfill}%
1813          }%
1814          \checkraw@text
1815          \checkverseL
1816          \checkverseR
1817          \checkpb@columns
1818       \repeat}
1819 %
```

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 is by pstart.

```
1820        \@writelinesinparL
```

```
1821          \@writelinesinparR
1822          \check@pstarts
1823            \ifbypstart@%
1824              \write\linenum@out{\string\@set[1]}
1825              \resetprevline@
1826            \fi
1827          \ifbypstart@R
1828              \write\linenum@outR{\string\@set[1]}
1829              \resetprevline@
1830            \fi
1831          \Columns@print@after@pend%
1832      \repeat
1833 %
```

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.

```
1834        \flush@notes
1835        \flush@notesR
1836    \endgroup
1837 %

1838    \global\l@dpscL=\z@
1839    \global\l@dpscR=\z@
1840    \global\l@dnumpstartsL=\z@
1841    \global\l@dnumpstartsR=\z@
1842    \l@dprintingcolumnsfalse%
1843    \ignorespaces
1844      \global\instanzaLfalse
1845      \global\instanzaRfalse}
1846
1847 %
```

\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).

```
1848 \def\print@columnseparator{%
1849    \ifdim\beforecolumnseparator<0pt%
1850      \hfill%
1851    \else%
1852      \hspace{\beforecolumnseparator}%
1853    \fi%
1854    \columnseparator%
1855    \ifdim\aftercolumnseparator<0pt%
1856      \hfill%
1857    \else%
1858      \hspace{\beforecolumnseparator}%
1859    \fi%
1860 }%
1861 %
```

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

```
1862
1863 \newcommand{\checkpb@columns}{%
1864     \newif\if@pb
1865     \newif\if@nopb
1866     \IfStrEq{\led@pb@setting}{before}{
1867     \numdef{\next@absline}{\the\absline@num+1}%
1868     \numdef{\next@abslineR}{\the\absline@numR+1}%
1869   \xifinlistcs{\next@absline}{l@prev@pb}{\@pbtrue}{}%
1870   \xifinlistcs{\next@abslineR}{l@prev@pbR}{\@pbtrue}{}
1871   \xifinlistcs{\next@absline}{l@prev@nopb}{\@nopbtrue}{}%
1872   \xifinlistcs{\next@abslineR}{l@prev@nopbR}{\@nopbtrue}{}
1873   }{}
1874     \IfStrEq{\led@pb@setting}{after}{
1875   \xifinlistcs{\the\absline@num}{l@prev@pb}{\@pbtrue}{}%
1876   \xifinlistcs{\the\absline@numR}{l@prev@pbR}{\@pbtrue}{}
1877   \xifinlistcs{\the\absline@num}{l@prev@nopb}{\@nopbtrue}{}%
1878   \xifinlistcs{\the\absline@numR}{l@prev@nopbR}{\@nopbtrue}{}
1879   }{}
1880 \if@nopb\nopagebreak[4]\enlargethispage{\baselineskip}\fi
1881 \if@pb\pagebreak[4]\fi
1882 }
1883 %
```

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

```
1884 \newcommand*{\columnseparator}{%
1885   \smash{\rule[-0.2\baselineskip]{\columnrulewidth}{1.05\
baselineskip}}}
1886 \newdimen\columnrulewidth
1887   \columnrulewidth=\z@
1888
1889 %
```

`\columnsposition` The position of the `\Columns` in a page. Default value is R. Stored in `\columns@position`.
`\columns@position`
```
1890 \newcommand*{\columnsposition}[1]{%
1891   \xdef\columns@position{#1}%
1892   }%
1893 \xdef\columns@position{R}%
1894 %
```

`\beforecolumnseparator` `\beforecolumnseparator` and `\aftercolumnseparator` lengths are de-
`\aftercolumnseparator` fined 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`.

```
1895 \newlength{\beforecolumnseparator}%
1896 \setlength{\beforecolumnseparator}{-2pt}%
1897
1898 \newlength{\aftercolumnseparator}%
1899 \setlength{\aftercolumnseparator}{-2pt}%
1900
1901 %
```

setwidthliketwocolumns@L
setpositionliketwocolumns@L
setnotepositionliketwocolumns@L
setwidthliketwocolumns@C
setpositionliketwocolumns@C
setnotepositionliketwocolumns@C
setwidthliketwocolumns@R
setpositionliketwocolumns@R
setnotepositionliketwocolumns@R

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.

```
1902 \newcommand{\setwidthliketwocolumns@L}{%
1903 % Temporary dimension, initially equal to the standard hsize,
     i.e. text width
1904 %      \begin{macrocode}
1905   \newdimen\temp%
1906   \temp=\hsize%
1907 %
```

Hsize : Left + Right width

```
1908   \hsize=\Lcolwidth%
1909   \advance\hsize\Rcolwidth%
1910 %
```

Now, calculating the remaining space

```
1911   \advance\temp-\hsize%
1912 %
```

And multiply the hsize by 2/3 of this space

```
1913   \multiply\temp by 2%
1914   \divide\temp by 3%
1915   \advance\hsize\temp%
1916 }%
1917
1918 \newcommand{\setpositionliketwocolumns@L}{%
1919   \renewcommand{\ledrlfill}{\hfill}%
1920 }%
1921
1922 \newcommand{\setnotespositionliketwocolumns@L}{%
1923 }%
1924
1925
1926 %
```

```
1927 \newcommand{\setwidthliketwocolumns@C}{%
1928 % Temporary dimension, initially equal to the standard hsize,
     i.e. text width
1929 %
```

```
1930   \newdimen\temp%
1931   \temp=\hsize%
1932 % Hsize : Left + Right width
1933 %
```

```
1934   \hsize=\Lcolwidth%
1935   \advance\hsize\Rcolwidth%
1936 % Now, calculating the remaining space
1937 %
```

```
1938   \advance\temp-\hsize%
1939 %
```

And multiply the hsize by 1/2 of this space

```
1940   \divide\temp by 2%
1941   \advance\hsize\temp%
1942 }%
1943
1944 \newcommand{\setpositionliketwocolumns@C}{%
1945   \doinsidelinehook{\hfill}%
1946   \renewcommand{\ledrlfill}{\hfill}%
1947 }%
1948
1949 \newcommand{\setnotespositionliketwocolumns@C}{%
1950   \newdimen\temp%
1951   \newdimen\tempa%
1952   \temp=\hsize%
1953   \tempa=\Lcolwidth%
1954   \advance\tempa\Rcolwidth%
1955   \advance\temp-\tempa%
1956   \divide\temp by 2%
1957   \leftskip=\temp%
1958   \rightskip=-\temp%
1959 }%
1960
1961 \newcommand{\setwidthliketwocolumns@R}{%
1962 %
```

Temporary dimension, initially equal to the standard hsize, i.e. text width

```
1963   \newdimen\temp%
1964   \temp=\hsize%
1965 %
```

Hsize : Left + Right width

```
1966    \hsize=\Lcolwidth%
1967    \advance\hsize\Rcolwidth%
1968 %
```

Now, calculating the remaining space

```
1969    \advance\temp-\hsize%
1970 %
```

And multiply the hsize by 2/3 of this space

```
1971    \multiply\temp by 2%
1972    \divide\temp by 3%
1973    \advance\hsize\temp%
1974 }%
1975
1976 \newcommand{\setpositionliketwocolumns@R}{%
1977    \doinsidelinehook{\hfill}%
1978 }%
1979
1980 \newcommand{\setnotespositionliketwocolumns@R}{%
1981    \newdimen\temp%
1982    \newdimen\tempa%
1983    \temp=\hsize%
1984    \tempa=\Lcolwidth%
1985    \advance\tempa\Rcolwidth%
1986    \advance\temp-\tempa%
1987    \divide\temp by 2%
1988    \leftskip=\temp%
1989    \rightskip=-\temp%
1990 }%
1991
1992 %
```

\Columns@print@before@pstart   The \Columns@print@before@pstart and \Columns@print@after@pend
\Columns@print@after@pend   print the content of the optional argument of \pstart / \pend. If this content is not
empty, it also print the separator.

```
1993 \newcommand{\Columns@print@before@pstart}{%
1994    \ifboolexpr{%
1995       test{\ifcsstring{before@pstartL@\the\l@dpscL}{\
at@every@pstart}}%
1996       and test {\ifcsstring{before@pstartR@\the\l@dpscR}{\
at@every@pstart}}%
1997       and test {\ifdefempty{\at@every@pstart}}}%
1998          {}%
1999          {%
2000             \hb@xt@ \hsize{%
2001                \ifdefstring{\columns@position}{L}{}{\hfill }%
2002                \par\parbox[t][][t]{\Lcolwidth}{%
2003                   \csuse{before@pstartL@\the\l@dpscL}%
2004                }%
```

```
2005            \print@columnseparator%
2006            \parbox[t][][t]{\Rcolwidth}{%
2007              \csuse{before@pstartR@\the\l@dpscR}%
2008            }%
2009            \ifdefstring{\columns@position}{R}{}{\hfill}%
2010          }%
2011        }%
2012      \global\csundef{before@pstartL@\the\l@dpscL}%
2013      \global\csundef{before@pstartR@\the\l@dpscR}%
2014  }%
2015  \newcommand{\Columns@print@after@pend}{%
2016    \ifboolexpr{%
2017      test{\ifcsstring{after@pendL@\the\l@dpscL}{\at@every@pend
      }}%
2018      and test {\ifcsstring{after@pendR@\the\l@dpscR}{\
      at@every@pend}}%
2019      and test {\ifdefempty{\at@every@pend}}}%
2020        {}%
2021        {%
2022          \hb@xt@ \hsize{%
2023            \ifdefstring{\columns@position}{L}{}{\hfill }%
2024            \parbox[t][][t]{\Lcolwidth}{%
2025              \csuse{after@pendL@\the\l@dpscL}%
2026            }%
2027            \print@columnseparator%
2028            \parbox[t][][t]{\Rcolwidth}{%
2029              \csuse{after@pendR@\the\l@dpscR}%
2030            }%
2031            \ifdefstring{\columns@position}{R}{}{\hfill}%
2032          }%
2033        }%
2034      \global\csundef{after@pendL@\the\l@dpscL}%
2035      \global\csundef{after@pendR@\the\l@dpscR}%
2036  }%
2037  %
```

XIX   Parallel pages

This is considerably more complicated than parallel columns.

XIX.1   Specific counters

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

```
2038  \newcount\numpagelinesL
2039  \newcount\numpagelinesR
2040  \newcount\l@dminpagelines
```

```
2041
2042 %
```

XIX.2   Main macro

\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.

```
2043 \newcommand*{\Pages}{%
2044   \ifl@dpairing%
2045     \led@err@Pages@InsideEnv%
2046   \fi%
2047   \eledsection@correcting@skip=-2\baselineskip% line
correcting for section titles.
2048   \parledgroup@notespacing@set@correction%
2049   \typeout{}%
2050   \typeout{************************ PAGES
************************}%
2051   \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else%
2052     \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\
l@dnumpstartsR}%
2053   \fi%
2054 %
```

As \Pages must be called outside of the pages environment, we have to redefine the
\Lcolwidth and \Rcolwidth lengths, to prevent false overfull hboxes.

```
2055   \setlength{\Lcolwidth}{\textwidth}%
2056   \setlength{\Rcolwidth}{\textwidth}%
2057 %
```

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

```
2058   \cleartol@devenpage%
2059   \l@dprintingpagestrue%
2060   \begingroup%
2061     \l@dzeropenalties%
2062     \endgraf\global\num@lines=\prevgraf%
2063             \global\num@linesR=\prevgraf%
2064     \global\par@line=\z@%
2065     \global\par@lineR=\z@%
2066     \global\l@dpscL=\z@%
2067     \global\l@dpscR=\z@%
2068     \writtenlinesLfalse%
2069     \writtenlinesRfalse%
2070 %
```

The footnotes are printed in a different way from expected in reledmac, as we may
want to print the notes on one side only.

```
2071        \let\print@Xnotes\print@Xnotes@forpages%
2072        \let\print@notesX\print@notesX@forpages%
2073 %
```

Check if there are chunks to be processed.

```
2074        \check@pstarts%
2075        \loop\if@pstarts%
2076 %
```

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

```
2077          \global\advance\l@dpscL \@ne%
2078          \global\advance\l@dpscR \@ne%
2079 %
```

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

```
2080          \getlinesfromparlistL%
2081          \getlinesfromparlistR%
2082          \l@dcalc@maxoftwo{\@cs@linesinparL}{\@cs@linesinparR}%
2083              {\usenamecount{l@dmaxlinesinpar\the\l@dpscL}}%
2084          \check@pstarts%
2085        \repeat%
2086 %
```

Zero the counts again, ready for the next bit.

```
2087        \global\l@dpscL=\z@%
2088        \global\l@dpscR=\z@%
2089 %
```

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

```
2090        \getlinesfrompagelistL%
2091        \getlinesfrompagelistR%
2092        \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
2093            {\l@dminpagelines}%
2094 %
```

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.

```
2095        \check@pstarts%
2096        \if@pstarts%
2097 %
```

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

```
2098          \global\advance\l@dpscL \@ne%
2099          \global\advance\l@dpscR \@ne%
2100          \restore@pstartL@pc%
2101          \restore@pstartR@pc%
2102 %
```

We have not processed any lines from these chunks yet, so zero the respective line counts.

```
2103        \global\@donereallinesL=\z@%
2104        \global\@donetotallinesL=\z@%
2105        \global\@donereallinesR=\z@%
2106        \global\@donetotallinesR=\z@%
2107 %
```

Start a loop over the boxes (chunks).

```
2108        \checkraw@text%
2109 %

2110 %       \begingroup
2111 {        \loop\ifaraw@text%
2112 %
```

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

```
2113            \checkpageL%
2114            \l@duselanguage{\theledlanguageL}%
2115 {            \loop\ifl@dsamepage%
2116 %
```

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

```
2117              \ifdefstring{\@eledsectnotoc}{L}{\ledsectnotoc
}{}%
2118              \csuse{before@pstartL@\the\l@dpscL}%
2119              \global\csundef{before@pstartL@\the\l@dpscL}%
2120              \do@lineL%
2121              \xifinlist{\the\l@dpscL}{\eled@sections@@}
2122                {\print@eledsectionL}%
2123                {}%
2124              \advance\numpagelinesL \@ne%
2125 %
```

When using shiftedpstarts option, a \l@dleftbox with a null height is not printed. That means we do not insert blank lines at the end of a left chunk lower than the corresponding right chunk. However, a \l@dleftbox with a null height will advance the \pagetotal in any case. Because if we do not do this, the \checkpageL could let \ifl@pagefull to false, and consequently a \@lopL equal to 1000 could be written in the numbered file, even if all the lines actually needed for the current page have been printed. ł@dleftbox

```
2126              \ifshiftedpstarts%
2127                  \ifdim\ht\l@dleftbox>0pt%
2128                      \parledgroup@correction@notespacing{L
}%
2129                      \hb@xt@ \hsize{\ledstrutL\unhbox\
l@dleftbox}%
```

```
2130                          \else%
2131                            \dimen0=\pagetotal%
2132                            \advance\dimen0 by \baselineskip%
2133                            \global\pagetotal=\dimen0%
2134                          \fi%
2135                        \else%
2136                          \parledgroup@correction@notespacing{L}%
2137                          \hb@xt@ \hsize{\ledstrutL\unhbox\
       l@dleftbox}%
2138                        \fi%
2139  %
```

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. Reset the verse skipnumber boolean

```
2140                        \get@nextboxL%
2141       \global\l@dskipversenumberfalse%
2142                        \ifprint@last@after@pendL%
2143                          \csuse{after@pendL@\the\l@dpscL}%
2144                          \global\csundef{after@pendL@\the\l@dpscL
       }%
2145                          \fi%
2146                        \checkpageL%
2147                        \checkverseL%
2148                        \checkpbL%
2149                      \repeat%
2150  %
```

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.

```
2151                        \ifl@dpagefull%
2152                          \@writelinesonpageL{\the\numpagelinesL}%
2153                        \else%
2154                          \@writelinesonpageL{1000}%
2155                        \fi%
2156  %
```

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.

```
2157                        \numpagelinesL \z@%
2158                        \parledgroup@correction@notespacing@init%
2159                        \clearl@dleftpage }%
2160  %
```

Now do the same for the right text.

```
2161              \checkpageR%
2162              \l@duselanguage{\theledlanguageR}%
2163 {              \loop\ifl@dsamepage%
2164                 \initnumbering@sectcountR%
2165                 \ifdefstring{\@eledsectnotoc}{R}{\ledsectnotoc
}{}%
2166                 \csuse{before@pstartR@\the\l@dpscR}%
2167                 \global\csundef{before@pstartR@\the\l@dpscR}%
2168                 \do@lineR%
2169                 \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}%
2170                   {\print@eledsectionR}%
2171                   {}%
2172                 \advance\numpagelinesR \@ne%
2173                 \ifshiftedpstarts%
2174                       \ifdim\ht\l@drightbox>0pt%
2175                         \parledgroup@correction@notespacing{R
}%
2176                         \hb@xt@ \hsize{\ledstrutR\unhbox\
l@drightbox}%
2177                       \else%
2178                         \dimen0=\pagetotal%
2179                         \advance\dimen0 by \baselineskip%
2180                         \global\pagetotal=\dimen0%
2181                       \fi%
2182                 \else%
2183                       \parledgroup@correction@notespacing{R}%
2184                       \hb@xt@ \hsize{\ledstrutR\unhbox\
l@drightbox}%
2185                 \fi%
2186                 \get@nextboxR%
2187      \global\l@dskipversenumberRfalse%
2188                 \ifprint@last@after@pendR%
2189                   \csuse{after@pendR@\the\l@dpscR}%
2190                   \global\csundef{after@pendR@\the\l@dpscR
}%
2191                 \fi%
2192              \checkpageR%
2193              \checkverseR%
2194              \checkpbR%
2195            \repeat%
2196            \ifl@dpagefull%
2197              \@writelinesonpageR{\the\numpagelinesR}%
2198            \else%
2199              \@writelinesonpageR{1000}%
2200            \fi%
2201            \numpagelinesR=\z@%
2202            \parledgroup@correction@notespacing@init%
2203 %
```

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

```
2204                   \clearl@drightpage}%
2205 %
```

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.

```
2206               \checkraw@text%
2207               \ifaraw@text%
2208                 \getlinesfrompagelistL%
2209                 \getlinesfrompagelistR%
2210                 \l@dcalc@minoftwo{\@cs@linesonpageL}{\
@cs@linesonpageR}%
2211                                     {\l@dminpagelines}%
2212               \fi%
2213           \repeat}%
2214 %
```

We have now output the text from all the chunks.

```
2215       \fi%
2216 %
```

Make sure that there are no inserts hanging around.

```
2217       \flush@notes%
2218       \flush@notesR%
2219    \endgroup%
2220 %
```

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

```
2221    \global\l@dpscL=\z@%
2222    \global\l@dpscR=\z@%
2223    \global\l@dnumpstartsL=\z@%
2224    \global\l@dnumpstartsR=\z@%
2225      \global\instanzaLfalse%
2226      \global\instanzaRfalse%
2227  \l@dprintingpagesfalse%
2228  \finish@Pages@notes%Needed to prevent final notes overlap
line number
2229    \ignorespaces}
2230
2231
2232 %
```

XIX.3   Ensure all notes be printed at the end of parallel pages

\finish@Pages@notes This macro ensures that all long notes are printed at the end of \Pages typesetting, and that there is no more long notes left for the next pages.

```
2233 \newcommand{\finish@Pages@notes}{%
2234    \def\do##1{%
2235 %
```

First, declare footnote box if there was no previous declared. E.g. if familiar or critical notes were disabled by reledmac's options.

```
2236        \ifnocritical@%
2237          \global\newnamebox{##1footins}
2238        \fi
2239        \ifnofamiliar@%
2240          \global\newnamebox{footins##1}
2241        \fi
2242 %
```

And now, add a \newpage if there is no more footnote to print.

```
2243        \ifvoid\csuse{##1footins}%
2244          \ifvoid\csuse{footins##1}\else%
2245            \newpage\null%
2246            \listbreak%
2247          \fi%
2248        \else%
2249          \newpage\null%
2250          \listbreak%
2251        \fi%
2252      }%
2253    \dolistloop{\@series}%
2254 }%
2255 %
```

XIX.4   Struts

\ledstrutL   Struts inserted into leftand right text lines.
\ledstrutR
```
2256 \newcommand*{\ledstrutL}{\strut}
2257 \newcommand*{\ledstrutR}{\strut}
2258
2259 %
```

XIX.5   Page clearing

\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
                    that it first checks to see if it is already on an empty page.

```
2260 \providecommand{\cleartoevenpage}[1][\@empty]{%
2261   \clearpage
2262   \ifodd\c@page\hbox{}#1\clearpage\fi}
2263
2264 \newcommand*{\cleartol@devenpage}{%
2265   \ifdim\pagetotal<\topskip%   on an empty page
2266   \else
2267     \clearpage
```

```
2268    \fi
2269    \ifodd\c@page%
2270      \ifprevpgnotnumbered%
2271        \addtocounter{par@page}{-1}%
2272        \ifdef{\prevpgstyle}{\thispagestyle{\prevpgstyle}}{}%
2273      \fi%
2274      \hbox{}\clearpage%
2275    \fi%
2276 }%
2277 %
```

\clearl@dleftpage   \clearl@dleftpage and \clearl@drightpage get us onto an odd and even
\clearl@drightpage   page, respectively, checking that we end up on the subsquent page. Both commands
use \newpage and not \clearpage. Because \clearpage prints all footnotes
before the next page, even if it has to add new empty pages, while \newpage does not.
And as we want notes started in the left page continue in the right page and *vice-versa*,
we must use \newpage and not \clearpage

```
2278 \newcommand*{\clearl@dleftpage}{%
2279   \ifdim\pagetotal=0pt\hbox{}\fi%
2280   \newpage%
2281   \ifodd\c@page\else
2282     \led@err@LeftOnRightPage
2283     \hbox{}%
2284     \cleardoublepage
2285   \fi}
2286
2287 \newcommand*{\clearl@drightpage}{%
2288   \ifdim\pagetotal=0pt\hbox{}\fi%
2289   \newpage%
2290   \ifodd\c@page
2291     \led@err@RightOnLeftPage
2292     \hbox{}%
2293     \cleartoevenpage
2294   \fi}
2295
2296 %
```

XIX.6   Lines managing

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

```
2297 \newcommand*{\getlinesfromparlistL}{%
2298   \ifx\linesinpar@listL\empty
2299     \gdef\@cs@linesinparL{0}%
2300   \else
2301     \gl@p\linesinpar@listL\to\@cs@linesinparL
```

```
2302      \fi}
2303 \newcommand*{\getlinesfromparlistR}{%
2304    \ifx\linesinpar@listR\empty
2305      \gdef\@cs@linesinparR{0}%
2306    \else
2307      \gl@p\linesinpar@listR\to\@cs@linesinparR
2308    \fi}
2309
2310 %
```

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

```
2311 \newcommand*{\getlinesfrompagelistL}{%
2312    \ifx\linesonpage@listL\empty
2313      \gdef\@cs@linesonpageL{1000}%
2314    \else
2315      \gl@p\linesonpage@listL\to\@cs@linesonpageL
2316    \fi}
2317 \newcommand*{\getlinesfrompagelistR}{%
2318    \ifx\linesonpage@listR\empty
2319      \gdef\@cs@linesonpageR{1000}%
2320    \else
2321      \gl@p\linesonpage@listR\to\@cs@linesonpageR
2322    \fi}
2323
2324 %
```

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

```
2325 \newcommand*{\@writelinesonpageL}[1]{%
2326    \edef\next{\write\linenum@out{\string\@lopL{#1}}}%
2327    \next}
2328 \newcommand*{\@writelinesonpageR}[1]{%
2329    \edef\next{\write\linenum@outR{\string\@lopR{#1}}}%
2330    \next}
2331
2332 %
```

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

```
2333 \newcommand*{\l@dcalc@maxoftwo}[3]{%
2334    \ifnum #2>#1\relax
2335      #3=#2\relax
2336    \else
```

```
2337      #3=#1\relax
2338    \fi}
2339  \newcommand*{\l@dcalc@minoftwo}[3]{%
2340    \ifnum #2<#1\relax
2341      #3=#2\relax
2342    \else
2343      #3=#1\relax
2344    \fi}
2345
2346  %
```

XIX.7   Page break managing

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

\checkpageL tests if the space and lines already taken on the page by text and footnotes is less than the constraints. If so, then \ifl@dpagefull is set FALSE and \ifl@dsamepage is set TRUE. If the page is spatially full then \ifl@dpagefull is set TRUE and \ifl@dsamepage is set FALSE. If it is not spatially full but the maximum number of lines have been output then both \ifl@dpagefull and \ifl@dsamepage are set FALSE.

```
2347  \newif\ifl@dsamepage
2348    \l@dsamepagetrue
2349  \newif\ifl@dpagefull
2350
2351  \newcommand*{\checkpageL}{%
2352    \l@dpagefulltrue
2353    \l@dsamepagetrue
2354    \check@goal
2355    \ifdim\pagetotal<\ledthegoal
2356      \ifnum\numpagelinesL<\l@dminpagelines
2357      \else
2358        \l@dsamepagefalse
2359        \l@dpagefullfalse
2360      \fi
2361    \else
2362      \l@dsamepagefalse
2363      \l@dpagefulltrue
2364    \fi%
2365    \ifprint@last@after@pendL%
2366        \l@dpagefullfalse%
2367        \l@dsamepagefalse%
2368        \print@last@after@pendLfalse%
2369    \fi%
2370    }%
2371
2372  \newcommand*{\checkpageR}{%
2373    \l@dpagefulltrue
2374    \l@dsamepagetrue
2375    \check@goal
```

```
2376    \ifdim\pagetotal<\ledthegoal
2377      \ifnum\numpagelinesR<\l@dminpagelines
2378      \else
2379        \l@dsamepagefalse
2380        \l@dpagefullfalse
2381      \fi
2382    \else
2383      \l@dsamepagefalse
2384      \l@dpagefulltrue
2385    \fi%
2386    \ifprint@last@after@pendR%
2387        \l@dpagefullfalse%
2388        \l@dsamepagefalse%
2389        \print@last@after@pendRfalse%
2390    \fi%
2391    }%
2392
2393  %
```

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

```
2394  \newcommand{\checkpbL}{
2395      \IfStrEq{\led@pb@setting}{after}{
2396        \xifinlistcs{\the\absline@num}{l@prev@pb}{\
      l@dpagefulltrue\l@dsamepagefalse}{}
2397        \xifinlistcs{\the\absline@num}{l@prev@nopb}{\
      l@dpagefullfalse\l@dsamepagetrue}{}
2398      }{}
2399      \IfStrEq{\led@pb@setting}{before}{
2400        \numdef{\next@absline}{\the\absline@num+1}
2401        \xifinlistcs{\next@absline}{l@prev@pb}{\l@dpagefulltrue\
      l@dsamepagefalse}{}
2402        \xifinlistcs{\next@absline}{l@prev@nopb}{\
      l@dpagefullfalse\l@dsamepagetrue}{}
2403      }{}
2404  }
2405
2406  \newcommand{\checkpbR}{
2407      \IfStrEq{\led@pb@setting}{after}{
2408        \xifinlistcs{\the\absline@numR}{l@prev@pbR}{\
      l@dpagefulltrue\l@dsamepagefalse}{}
2409        \xifinlistcs{\the\absline@numR}{l@prev@nopbR}{\
      l@dpagefullfalse\l@dsamepagetrue}{}
2410      }{}
2411      \IfStrEq{\led@pb@setting}{before}{
2412        \numdef{\next@abslineR}{\the\absline@numR+1}
2413        \xifinlistcs{\next@abslineR}{l@prev@pbR}{\
      l@dpagefulltrue\l@dsamepagefalse}{}
```

```
2414        \xifinlistcs{\next@abslineR}{l@prev@nopbR}{\
      l@dpagefullfalse\l@dsamepagetrue}{}
2415    }{}
2416 }
2417 %
```

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

```
2418 \newcommand{\checkverseL}{
2419 \ifinstanzaL
2420   \iflednopbinverse
2421     \ifinserthangingsymbol
2422       \numgdef{\prev@abslineverse}{\the\absline@num-1}
2423       \IfStrEq{\led@pb@setting}{after}{\lednopbnum{\
      prev@abslineverse}}{}
2424       \IfStrEq{\led@pb@setting}{before}{\ifnum\numpagelinesL
      <3\ledpbnum{\prev@abslineverse}\fi}{}
2425     \fi
2426   \fi
2427 \fi
2428 }
2429 \newcommand{\checkverseR}{
2430 \ifinstanzaR
2431   \iflednopbinverse
2432     \ifinserthangingsymbolR
2433       \numgdef{\prev@abslineverse}{\the\absline@numR-1}
2434       \IfStrEq{\led@pb@setting}{after}{\lednopbnumR{\
      prev@abslineverse}}{}
2435       \IfStrEq{\led@pb@setting}{before}{\ifnum\numpagelinesR
      <3\ledpbnumR{\prev@abslineverse}\fi}{}
2436     \fi
2437   \fi
2438 \fi
2439 }
2440 %
```

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

```
2441 \newdimen\ledthegoal
2442 \ifshiftedpstarts
2443        \newcommand*{\@goalfraction}{0.95}
2444 \else
2445        \newcommand*{\@goalfraction}{0.9}
2446 \fi
2447
2448 \newcommand*{\check@goal}{%
2449   \ledthegoal=\@goalfraction\pagegoal}
```

```
2450 \newcommand{\setgoalfraction}[1]{%
2451   \xdef\@goalfraction{#1}%
2452 }
2453 %
```

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

```
2454 \newif\ifwrittenlinesL
2455 \newif\ifwrittenlinesR
2456
2457 %
```

XIX.8   Getting boxes content

\get@nextboxL   If the current box is not empty (i.e., still contains some lines) nothing is done. Otherwise
\get@nextboxR   if and only if a synchronisation point is reached the next box is started.

```
2458 \newcommand*{\get@nextboxL}{%
2459   \ifvbox\namebox{l@dLcolrawbox\the\l@dpscL}% box is not
     empty
2460 %
```

The current box is not empty; do nothing.

```
2461   \else%                                                           box is empty
2462 %
```

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

```
2463     \ifnum\usenamecount{l@dmaxlinesinpar\the\l@dpscL}>\
       @donetotallinesL
2464       \parledgroup@notes@endL
2465     \else
2466 %
```

Sufficient lines have been output.

```
2467     \ifnum\usenamecount{l@dmaxlinesinpar\the\l@dpscL}=\
       @donetotallinesL
2468       \parledgroup@notes@endL
2469     \fi
2470     \ifwrittenlinesL\else
2471 %
```

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

```
2472       \@writelinesinparL
2473       \writtenlinesLtrue
2474     \fi
2475     \ifnum\l@dnumpstartsL>\l@dpscL
2476 %
```

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.

```
2477            \writtenlinesLfalse
2478             \ifbypstart@
2479                \global\line@num=0%
2480                \resetprevline@%
2481            \fi
2482 % Add the content of the optional argument of the previous \
    protect\cs{pend}.
2483 %     \begin{macrocode}
2484            \csuse{after@pendL@\the\l@dpscL}%
2485            \global\csundef{after@pendL@\the\l@dpscL}%
2486 %
```

Check the number of lines

```
2487            \l@dcalc@maxoftwo{\the\usenamecount{l@dmaxlinesinpar\
    the\l@dpscL}}%
2488                               {\the\@donetotallinesL}%
2489                               {\usenamecount{l@dmaxlinesinpar\the\
    l@dpscL}}%
2490            \global\@donetotallinesL \z@
2491 %
```

Go to the next pstart

```
2492            \global\advance\l@dpscL \@ne
2493            \global\pstartnumtrue%
2494            \restore@pstartL@pc%
2495 %
```

Add notes of parallel ledgroup.

```
2496            \parledgroup@notes@endL
2497            \parledgroup@correction@notespacing@final{L}
2498        \else
2499 %

2500          \fi
2501      \fi
2502    \fi}
2503 %

2504 \newcommand*{\get@nextboxR}{%
2505   \ifvbox\namebox{l@dRcolrawbox\the\l@dpscR}% box is not
    empty
2506   \else%                                        box is empty
2507     \ifnum\usenamecount{l@dmaxlinesinpar\the\l@dpscR}>\
    @donetotallinesR
2508       \parledgroup@notes@endR
2509     \else
```

```
2510        \ifnum\usenamecount{l@dmaxlinesinpar\the\l@dpscR}=\
@donetotallinesR
2511          \parledgroup@notes@endR
2512        \fi
2513        \ifwrittenlinesR\else
2514          \@writelinesinparR
2515          \writtenlinesRtrue
2516        \fi
2517        \ifnum\l@dnumpstartsR>\l@dpscR
2518          \writtenlinesRfalse
2519          \ifbypstart@R
2520              \global\line@numR=0%
2521              \resetprevline@%
2522          \fi
2523          \csuse{after@pendR@\the\l@dpscR}%
2524          \global\csundef{after@pendR@\the\l@dpscR}%
2525          \l@dcalc@maxoftwo{\the\usenamecount{l@dmaxlinesinpar\
the\l@dpscR}}%
2526                              {\the\@donetotallinesR}%
2527                              {\usenamecount{l@dmaxlinesinpar\the\
l@dpscR}}%
2528          \global\@donetotallinesR \z@
2529          \global\advance\l@dpscR \@ne
2530          \global\pstartnumRtrue%
2531          \restore@pstartR@pc%
2532          \parledgroup@notes@endR
2533          \parledgroup@correction@notespacing@final{R}
2534        \else
2535            \print@last@after@pendRtrue%
2536        \fi
2537      \fi
2538    \fi}
2539
2540 %
```

XX   Page numbering

The `sameparallelpagenumber` option allows the same page number on both left and right side The `prevpgnotnumbered` option allows an empty (not numbered) right-side page before \Pages.

We cannot implement these two options by changing the value of the `page` counter, since its value is used by many LaTeXfeatures to determine whether a page is left (even-numbered) or right (odd-numbered). Consequently, we have to do it by patching \thepage, in order to use the value of the `par@page` counter instead of value of `page` counter.

This counter will be increased in a patched version of the LaTeX's \@outputpage macro, as is the `page` counter in this macro. However, this increase will take account

of the options.

\par@patch@thepage \par@patch@thepage patches \thepage in order to use the value of par@page
r@patch@pagenumbering counter and not the value of par@page. It must be called after any redefinition of
\thepage. That why we insert it at the end of the LaTeXmacro \pagenumber-
ing, which is called by some \xxxmatter commands. In the case of memoir
class using, we insert it at the end of \@mempnum. When using \pagenumber-
ing, we also need to restart par@page counter. Consequently, we have wrapped
\par@patch@thepage and counter restart in \par@patch@pagenumbering
We also call \par@patch@thepage it at the beginning of the document.

```
2541
2542 \newcommand{\par@patch@thepage}{%
2543   \ifboolexpr{%
2544     bool{sameparallelpagenumber}%
2545     or bool{prevpgnotnumbered}%
2546   }%
2547   {%
2548     \patchcmd{\thepage}%
2549       {page}{par@page}%
2550       {}%
2551       {\led@error@fail@patch@thepage}%
2552   }{}%
2553 }%
2554
2555 \newcommand{\par@patch@pagenumbering}{%
2556   \ifboolexpr{%
2557     bool{sameparallelpagenumber}%
2558     or bool{prevpgnotnumbered}%
2559     }%
2560     {%
2561     \setcounter{par@page}{1}%
2562     }%
2563     {}%
2564   \par@patch@thepage%
2565 }%
2566
2567 \ifl@dmemoir%
2568   \apptocmd{\@mempnum}%
2569     {\par@patch@pagenumbering}%
2570     {}%%
2571     {\led@error@fail@patch@@mempnum}%
2572
2573 \else%
2574   \apptocmd{\pagenumbering}%
2575     {\par@patch@pagenumbering}%
2576     {}%
2577     {\led@error@fail@patch@pagenumbering}%
2578 \fi%
2579
```

```
2580  \AtBeginDocument{\par@patch@thepage}%
2581  %
```

\@outputpage   As its name says, \@outputpage is a LaTeX's macro called in the output routine.
It is this macro which increases the page counter.. We patch it in order to increase,
conditionally, the par@page counter.

```
2582  \AtBeginDocument{%
2583    \apptocmd{\@outputpage}{%
2584      \ifsameparallelpagenumber%
2585        \ifl@dprintingpages%
2586          \ifodd\c@page\else%
2587            \stepcounter{par@page}%
2588          \fi%
2589        \else%
2590          \stepcounter{par@page}%
2591        \fi%
2592      \else%
2593        \stepcounter{par@page}%
2594      \fi%
2595      }%
2596      {}%
2597      {\led@error@fail@patch@@outputpage}%
2598  }
2599  %
```

\thepar@page   And now, initialize par@page counter.

```
2600  \newcounter{par@page}%
2601  \setcounter{par@page}{1}%
2602  %
```

XXI   Sections' titles' commands

As switching from left to right pages does not clear the page since v1.13.0, but only
creates new pages, no \vbox{} is inserted, and consequently parallel chapters are
mis-aligned.

So we patch the \chapter command in order to prevent this problem.

```
\chapter 2603  \pretocmd{\chapter}{%
2604    \ifl@dprintingpages%
2605      \vbox{}%
2606    \fi%
2607    }%
2608    {}%
2609    {}%
2610  %
```

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

```
2611 \newcommand{\eledsectnotoc}[1]{\xdef\@eledsectnotoc{#1}}
2612 \eledsectnotoc{R}
2613 %
```

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

```
2614 \newcommand{\eledsectmark}[1]{\xdef\@eledsectmark{#1}}
2615 \eledsectmark{L}
2616 %
```

ction@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.

```
2617 \newskip\eledsection@correcting@skip
2618 %
```

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

```
2619 \newwrite\eled@sectioningR@out
2620 %
```

XXII   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's list, which contains the lines in which
\prev@nopbR   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).

```
2621 \def\l@prev@pbR{}
2622 \def\l@prev@nopbR{}
2623 %
```

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

```
2624 \newcommand{\ledpbR}{\write\linenum@outR{\string\led@pbR}}
2625 \newcommand{\ledpbnumR}[1]{\write\linenum@outR{\string\
    led@pbnumR{#1}}}
```

```
2626 \newcommand{\lednopbR}{\write\linenum@outR{\string\led@nopbR
     }}
2627 \newcommand{\lednopbnumR}[1]{\write\linenum@outR{\string\
     led@nopbnumR{#1}}}
2628 %
```

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

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

```
2629 \newcommand{\led@pbR}{\listxadd{\l@prev@pbR}{\the\
     absline@numR}}
2630 \newcommand{\led@pbnumR}[1]{\listxadd{\l@prev@pbR}{#1}}
2631 \newcommand{\led@nopbR}{\listxadd{\l@prev@nopbR}{\the\
     absline@numR}}
2632 \newcommand{\led@nopbnumR}[1]{\listxadd{\l@prev@nopbR}{#1}}
2633 %
```

XXIII   Parallel ledgroup

\parledgroup@
\parledgroupseries@
\parledgrouptype@

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

```
2634 \newmarks\parledgroup@
2635 \newmarks\parledgroup@series
2636 \newmarks\parledgroup@type
2637 %
```

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

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

```
2638 \newcommand{\parledgroup@notes@startL}{%
2639    \ifnum\usenamecount{l@dmaxlinesinpar\the\l@dpscL}>0%
2640       \IfStrEq{\splitfirstmarks\parledgroup@type}{footnoteX}{\
     csuse{bhooknoteX@\splitfirstmarks\parledgroup@series}}{}%
2641       \IfStrEq{\splitfirstmarks\parledgroup@type}{Xfootnote}{\
     csuse{bhookXnote@\splitfirstmarks\parledgroup@series}}{}%
2642    \fi%
2643    \global\ledgroupnotesL@true%
2644    \insert@noterule@ledgroup{L}%
2645 }
2646 \newcommand{\parledgroup@notes@startR}{%
2647    \ifnum\usenamecount{l@dmaxlinesinpar\the\l@dpscR}>0%
2648       \IfStrEq{\splitfirstmarks\parledgroup@type}{footnoteX}{\
     csuse{bhooknoteX@\splitfirstmarks\parledgroup@series}}{}%
```

```
2649      \IfStrEq{\splitfirstmarks\parledgroup@type}{Xfootnote}{\
      csuse{bhookXnote@\splitfirstmarks\parledgroup@series}}{}%
2650    \fi%
2651    \global\ledgroupnotesR@true%
2652    \insert@noterule@ledgroup{R}%
2653  }
2654  %
```

`ledgroup@notes@startL`
`ledgroup@notes@startR`

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

```
2655  \newcommand{\parledgroup@notes@endL}{%
2656    \global\ledgroupnotesL@false%
2657  }
2658  \newcommand{\parledgroup@notes@endR}{%
2659    \global\ledgroupnotesR@false%
2660  }
2661  %
```

`ert@noterule@ledgroup`

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

```
2662  \newcommand{\insert@noterule@ledgroup}[1]{
2663      \IfStrEq{\splitbotmarks\parledgroup@}{begin}{%
2664        \IfStrEq{\splitbotmarks\parledgroup@type}{Xfootnote}{
2665          \csuse{ifledgroupnotes#1@}
2666            \vskip\skip\csuse{mp\splitbotmarks\
      parledgroup@series footins}
2667            \csuse{\splitbotmarks\parledgroup@series
      footnoterule}
2668          \fi
2669          }
2670          {}
2671          \IfStrEq{\splitbotmarks\parledgroup@type}{footnoteX}{
2672            \csuse{ifledgroupnotes#1@}
2673            \vskip\skip\csuse{mpfootins\splitbotmarks\
      parledgroup@series}
2674            \csuse{footnoterule\splitbotmarks\
      parledgroup@series}
2675            \fi
2676            }{}
2677          }
2678      {}
2679  }
2680  %
```

`arledgroupnotespacing`

`\@parledgroupnotespacing` can be redefined by the user to change the inter-line spacing of ledgroup notes.

```
2681 \newcommand{\setparledgroupnotespacing}[1]{\gdef\
     @parledgroupnotespacing{#1}}
2682 \newcommand{\@parledgroupnotespacing}{}
2683 %
```

ledgroup@notespacing@correction \parledgroup@notespacing@correction is the difference between a nor-
oup@notespacing@set@correction   mal line skip and a line skip in a note. It is set by \parledgroup@notespacing@set@correction
                                  called at the beginning of \Pages.

```
2684 \dimdef{\parledgroup@notespacing@correction}{0pt}
2685 \newcommand{\parledgroup@notespacing@set@correction}{%
2686   {\@getfirstseries\csuse{Xnotefontsize@\@firstseries}}%We
     suppose all the series has the same footnote size setup
2687   \@parledgroupnotespacing\dimgdef{\temp@spacing}{\
     baselineskip}}%
2688   \dimgdef{\parledgroup@notespacing@correction}{\baselineskip
     -\temp@spacing}%
2689 }
2690 %
```

up@correction@notespacing@init \parledgroup@correction@notespacing@init sets the value of accu-
                                mulated corrections of note spacing to 0 pt. It is called at the beginning of each pages
                                AND at the end of each ledgroup.

```
2691 \newcommand{\parledgroup@correction@notespacing@init}{
2692   \dimdef{\parledgroup@notespacing@correction@accumulated}{0
     pt}
2693   \dimdef{\parledgroup@notespacing@correction@modulo}{0pt}
2694 }
2695 \parledgroup@correction@notespacing@init
2696 %
```

@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 is called after the print
                               of each pstart/pend.

```
2697 \newcommand{\parledgroup@correction@notespacing@final}[1]{
2698     \ifparledgroup
2699     \vspace{\parledgroup@notespacing@correction@accumulated}
2700     \parledgroup@correction@notespacing@init%
2701     \ifstrequal{#1}{L}{
2702         \numdef{\@checking}{\the\l@dpscL-1}
2703     }{
2704         \numdef{\@checking}{\the\l@dpscR-1}
2705     }
2706     \dimdef{\@beforenotes@current@diff}{\csuse{
     @parledgroup@beforenotes@\@checking L}-\csuse{
     @parledgroup@beforenotes@\@checking R}}%
```

```
2707    \ifstrequal{#1}{L}%
2708       {% Left
2709       \ifdimgreater{\@beforenotes@current@diff}{0pt}{}{\
      vspace{-\@beforenotes@current@diff}}%
2710       }%
2711       {% Right
2712       \ifdimgreater{\@beforenotes@current@diff}{0pt}{\vspace
      {\@beforenotes@current@diff}}{}
2713       }%
2714    \fi
2715 }
2716 %
```

orrection@notespacing   \parledgroup@correction@notespacing is used before each printed line.
If it is 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 \parled-
group@notespacing@correction@accumulated and \parledgroup@notespacing@correctio
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.

```
2717 \newcommand{\parledgroup@correction@notespacing}[1]{%
2718     \csuse{ifledgroupnotes#1@}%
2719       \vspace{-\parledgroup@notespacing@correction}%
2720       \dimdef{\parledgroup@notespacing@correction@accumulated
      }{\parledgroup@notespacing@correction@accumulated+\
      parledgroup@notespacing@correction}%
2721       \dimdef{\parledgroup@notespacing@correction@modulo}{\
      parledgroup@notespacing@correction@modulo+\
      parledgroup@notespacing@correction}%
2722       \ifdimless{\parledgroup@notespacing@correction@modulo
      }{\baselineskip}{}{\advance\numpagelinesL -\@ne%
2723       \dimdef{\parledgroup@notespacing@correction@modulo}{\
      parledgroup@notespacing@correction@modulo-\baselineskip}%
2724       }% mean greater than equal
2725     \fi%
2726 }
2727 %
```

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

```
2728  \dimdef\parledgroup@beforenotesL{0pt}
2729  \dimdef\parledgroup@beforenotesR{0pt}
2730  %
```

\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.

```
2731  \newcommand{\parledgroup@beforenotes@save}[1]{
2732    \ifparledgroup
2733      \csdimgdef{@parledgroup@beforenotes@\the\csuse{
      l@dnumpstarts#1}#1}{\csuse{parledgroup@beforenotes#1}}
2734      \csdimgdef{parledgroup@beforenotes#1}{0pt}
2735    \fi
2736  }
2737  %
```

XXIV   Compatibility with **eledmac**

Here, we define some command for the eledmac-compat option.

```
2738  \ifeledmaccompat@%
2739
2740
2741    \unless\ifnocritical@
2742    \let\onlyXside\Xonlyside
2743    \fi
2744  \fi
2745  %
```

XXV   The End

</code>

Appendix A   Some things to do when changing version

Appendix A.1   Migration to **eledpar** 1.4.3

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

However, if you want to return to automatic flushright margins for verses with hanging indents, you have to redefine the `\hangingsymbol` command.

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

See the following two examples:

With standard `\hangingsymbol`:

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

With the modification of the hangingsymbol:

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

Appendix A.2   Migration from **eledpar** to **reledpar**

As for migration from `eledmac` to `reledmac`:

- One option has been removed because it is deprecated.

- Some of the customizations previously made by `\renewcommand` have been replaced with commands.

- Some command names have been changed in order to have a more logical and uniform pattern.

Appendix A.2.1   Deprecated options

The `shiftedverses` option has been removed. Use the general `shiftedpstart` option instead.

Appendix A.2.2   **\renewcommand** replaced with command

Many uses of `\renewcommand` have been replaced with uses of specific commands. Please read the handbook about these particular commands.

Deprecated `\renewcommand`	*Replaced with*
`\goalfraction`	`\setgoalfraction`
`\parledgroupnotespacing`	`\setparledgroupnotespacing`
`\Rlineflag`	`\setRlineflag`

Appendix A.2.3   Commands the names of which have changed

In order to ease the migration from `eledpar` to `reledpar`, you may load `reled-mac` with `eledmac-compat` option. However, it is advised to change the command names.

Old command	New command
`\onlyXside`	`\Xonlyside`

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/ed-mac`)

[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

A

S

T

U

W

X

Change History