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 v2.0.0, last revised 2015/06/28.

[†] `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 XXII); 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 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

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

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
\Rcolwidth

The lengths `\Lcolwidth` and `\Rcolwidth` are the widths of the left and right columns, respectively. By default, these are:

```
\setlength{\Lcolwidth}{0.45\textwidth}
\setlength{\Rcolwidth}{0.45\textwidth}
```

They may be adjusted if one text tends to be 'bulkier' than the other.

3.2.2 Column's separator

\columnrulewidth
\columnseparator

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

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

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

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 `\aftercolumnseparator` 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 horizontally 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 reledpar.

`\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
\Rcolwidth

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

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.

sameparallelpagenumbertrue
sameparallelpagenumbertrue

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.

You can also switch the two system using `\sameparallelpagenumbertrue` and `\sameparallelpagenumberfalse`.

4.2.3 Page breaking

\setgoalfraction

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

4.3 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.

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

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 `\Xonlyside[`⟨*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
 \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`
`Rightside`

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

5.2 Numbering text lines and paragraphs

`\beginnumbering`
`\endnumbering`

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

```
\beginnumbering
⟨text⟩
\endnumbering
```

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

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

`\memorydump`

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

```
\begin{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}
```

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

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

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

Following `\firstlinenum{`⟨*num*⟩`}` the first line number will be ⟨*num*⟩, and following `\linenumincrement{`⟨*num*⟩`}` only every ⟨*num*⟩th line will have a printed number. Using these macros inside the `Leftside` and `Rightside` environments gives you independent control over the left and right numbering schemes. The `\firstsublinenum` and `\sublinenumincrement` macros correspondingly set the numbering scheme for sublines. The starred versions change both left and right numbering schemes.

Generally speaking, controls like `\firstlinenum` or `\linenummargin` apply to sequential and left texts. To effect right texts only, they have to be within a `Rightside` environment.

`\lineationR`
`\lineation*`

`\lineationR` macro is the equivalent of reledmac `\lineation` macro for 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`
`\pend`

In a serial (non-parallel) mode, each numbered paragraph, or chunk, is contained between the `\pstart` and `\pend` macros, and the paragraph is output when the `\pend` macro occurs. The situation is somewhat different with parallel typesetting as the left text (contained within `\pstart` and `\pend` groups within the `Leftside` environment) has to be set in parallel with the right text (contained within its own `\pstart` and `\pend` groups within the corresponding `Rightside` environment) the `\pend` macros cannot immediately initiate any typesetting — this has to be controlled by the `\Columns` or `\Pages` macros. Several chunks may 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...\pends. Please read reledmac's handbook (**??** p. **??**).

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 **??** p. **??**) 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, **??** p. **??**). 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:

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

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.

```
⟨*code⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{reledpar}[2015/06/28 v2.0.0 reledmac extension for parallel texts]%

```

II.2 Package's requirement

Few commands use `\xspace` command.

```
\RequirePackage{xspace}%
```

II.3 Package's options

With the option 'shiftedpstarts' a long pstart one the left side (or in 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

```
\newif\ifshiftedpstarts
\DeclareOption{shiftedpstarts}{\shiftedpstartstrue}
```

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

```
\DeclareOption{parledgroup}{\parledgrouptrue}
```

`\ifwidthliketwocolumns`

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

```
\DeclareOption{widthliketwocolumns}{\widthliketwocolumnstrue}%
```

The `sameparallelpagenumber` enable to have the same page number in left and right side.

`\ifsameparallelpagenumber`

```
\newif\ifsameparallelpagenumber%
\DeclareOption{sameparallelpagenumber}{\sameparallelpagenumbertrue}%
```

```
\ProcessOptions%
```

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@dpaging`
`\ifledRcol`

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

II.5 Text’s width

`\Lcolwidth`
`\Rcolwidth`

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

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

II.6 Messages

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

`\eledpar@error`

```
\newcommand{\eledpar@error}[2]{\PackageError{eledpar}{#1}{#2}}
```

`\led@err@TooManyPstarts`

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

r@BadLeftRightPstarts

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

l@err@LeftOnRightPage
l@err@RightOnLeftPage

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

le@PreviousNotPrinted
le@PreviousNotPrinted

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

l@err@Pages@InsideEnv
err@Columns@InsideEnv

```
\newcommand*{\led@err@Pages@InsideEnv}{%
   \eledpar@error{\string\Pages\space must be called *outside* of the `pages` environment}{\@ehc}}
\newcommand*{\led@err@Columns@InsideEnv}{%
   \eledpar@error{\string\Columns\space must be called *outside* of the `pairs` environment}{\@ehc}}
```

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.

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

\ifpst@rtedL
\ifpst@rtedR

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

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

```

\beginnumberingR

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

```
\newcommand*{\beginnumberingR}{%
```

```
  \ifnumberingR
    \led@err@NumberingStarted
    \endnumberingR
  \fi
  \global\l@dnumpstartsR \z@
  \global\pst@rtedRfalse
  \global\numberingRtrue
  \global\advance\section@numR \@ne
  \global\absline@numR \z@
  \gdef\normal@page@breakR{}
  \gdef\l@prev@pbR{}
  \gdef\l@prev@nopbR{}
  \global\line@numR \z@
  \global\@lockR \z@
  \global\sub@lockR \z@
  \global\sublines@false
  \global\let\next@page@numR\relax
  \global\let\sub@change\relax
  \message{Section \the\section@numR R }%
 \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
  \l@dend@stuff
  \setcounter{pstartR}{1}
  \begingroup
  \initnumbering@sectcountR
  \gdef\eled@sectionsR@@{}%
  \if@noeled@sec\else%
   \makeatletter\InputIfFileExists{\jobname.eledsec\the\section@numR R}{}{
   \immediate\openout\eled@sectioningR@out=\jobname.eledsec\the\section@nu
  \fi%
}
```

\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.

```
\def\endnumberingR{%
  \ifnumberingR
    \global\numberingRfalse
    \normal@pars
    \ifnum\l@dnumpstartsR=0%
      \led@err@NumberingWithoutPstart%
    \fi%
    \ifl@dpairing
      \global\pst@rtedRfalse
    \else
      \ifx\insertlines@listR\empty\else
        \global\noteschanged@true
      \fi
      \ifx\line@listR\empty\else
         \global\noteschanged@true
      \fi
    \fi
    \ifnoteschanged@
      \led@mess@NotesChanged
    \fi
  \else
    \led@err@NumberingNotStarted
  \fi
  \endgroup
  \if@noeled@sec\else%
    \immediate\closeout\eled@sectioningR@out%
  \fi%
  }
```

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.

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

\pausenumberingR
\resumenumberingR

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

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

\memorydumpL
\memorydumpR

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

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

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

```

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.

\ifbypstart@R
\bypstart@Rtrue
\bypstart@Rfalse
\ifbypage@R
\bypage@Rtrue
\bypage@Rfalse

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

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

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

```
\newif\ifbypage@R
\newif\ifbypstart@R
```

\lineationR

\lineationR{⟨*word*⟩} is the macro used to select the lineation system for right texts. Its argument is a string: either page, pstart or section.

```
\newcommand*{\lineationR}[1]{{%
  \ifnumbering
    \led@err@LineationInNumbered
  \else
    \def\@tempa{#1}\def\@tempb{page}%
    \ifx\@tempa\@tempb
```

```
        \global\bypage@Rtrue
        \global\bypstart@Rfalse
         \unless\ifnocritical@%
           \pstartinfootnote[][false]%
         \fi%
    \else
       \def\@tempb{pstart}%
       \ifx\@tempa\@tempb
           \global\bypage@Rfalse
           \global\bypstart@Rtrue
           \unless\ifnocritical@%
             \pstartinfootnote%
           \fi%
       \else
           \def@tempb{section}
           \ifx\@tempa\@tempb
             \global\bypage@Rfalse%
             \global\bypstart@Rfalse%
             \unless\ifnocritical@%
               \pstartinfootnote[][false]%
             \fi%
           \else
             \led@warn@BadLineation
           \fi%
       \fi
    \fi
  \fi}}
```

\lineation*

\lineation* change the lineation system for both sides.

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

IV.2 Setting line number margin

\linenummargin
\line@marginR

You call \linenummargin{⟨*word*⟩} to specify which margin you want your right text's line numbers in; it takes one argument, a string. You can put the line numbers in the same margin on every page using left or right; or you can use inner or outer to get them in the inner or outer margins. You can change this within a numbered section, but the change may not take effect just when you 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.

```
\newcount\line@marginR
```

IV.3 Setting lineation start and step

\c@firstlinenumR
\c@linenumincrementR

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

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

\c@firstsublinenumR
\c@sublinenumincrementR

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

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

```

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

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

```
\WithSuffix\newcommand\firstlinenum*[1]{%
  \setcounter{firstlinenumR}{#1}%
  \setcounter{firstlinenum}{#1}%
}
\WithSuffix\newcommand\linenumincrement*[1]{%
  \setcounter{linenumincrementR}{#1}%
  \setcounter{linenumincrement}{#1}%
}
\WithSuffix\newcommand\firstsublinenum*[1]{%
  \setcounter{subfirstlinenumR}{#1}%
  \setcounter{subfirstlinenum}{#1}%
}
\WithSuffix\newcommand\sublinenumincrement*[1]{%
  \setcounter{sublinenumincrementR}{#1}%
  \setcounter{sublinenumincrement}{#1}%
}
```

IV.4 Setting line flag

\Rlineflag

This is appended to the line numbers of right text.

```
\newcommand{\setRlineflag}[1]{%
  \gdef\@Rlineflag{#1}%
}
\setRlineflag{R}
```

IV.5 Setting line number style

\linenumrepR
\sublinenumrepR

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

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

```

IV.6 Print marginal line number

\leftlinenumR
\rightlinenumR
\l@dlinenumR

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

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

```

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
\subline@numR
\absline@numR

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

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

```

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

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

Here are the commands to create these lists:

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

```

\page@numR

The right text page number.

```
\newcount\page@numR

```

IV.7.2 Specific to reledpar

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

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

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

```

IV.8 Reading the line-list file

\list@clearing@regR

`\Clear the right lines for \read@linelist`

```
\newcommand{\list@clearing@regR}{%
    \list@clear{\line@listR}%
    \list@clear{\insertlines@listR}%
    \list@clear{\actionlines@listR}%
    \list@clear{\actions@listR}%
    \list@clear{\linesinpar@listR}%
    \list@clear{\linesonpage@listR}
}
```

\read@linelist

`\read@linelist{`⟨*file*⟩`}` is the control sequence that is called by `\beginnumbering` (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 associated with.

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

\@nl@regR
\@nl

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

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

\renewcommand*{\@nl}[2]{%
  \fix@page{#1}%
  \ifledRcol
    \@nl@regR
  \else
    \@nl@reg
  \fi}

```

\last@page@numR
\fix@page

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

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

```

\@adv

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

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

```
    \ifledRcol
      \advance\line@numR by #1\relax
      \ifnum\line@numR<\z@
        \led@warn@BadAdvancelineLine
        \line@numR \z@
      \fi
    \else
      \advance\line@num by #1\relax
      \ifnum\line@num<\z@
        \led@warn@BadAdvancelineLine
        \line@num \z@
      \fi
    \fi
  \fi
  \set@line@action}

```

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

```
\renewcommand*{\@set}[1]{%
  \ifledRcol
    \ifsublines@
      \subline@numR=#1\relax
    \else
      \line@numR=#1\relax
    \fi
    \set@line@action
  \else
    \ifsublines@
      \subline@num=#1\relax
    \else
      \line@num=#1\relax
    \fi
    \set@line@action
  \fi}

```

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

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

```
\renewcommand*{\l@d@set}[1]{%
  \ifledRcol
    \line@numR=#1\relax
    \advance\line@numR \@ne
    \def\l@dchset@num{#1}
  \else
    \line@num=#1\relax
    \advance\line@num \@ne
```

```
   \def\l@dchset@num{#1}
  \fi}
\let\l@dchset@num\relax

```

\page@action

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

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

\set@line@action

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

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

```

\sub@action

\sub@action adds an entry to the action-code list to turn sub-lineation on or off, according to the current value of the \ifsublines@ flag.

```
\renewcommand*{\sub@action}{%
  \ifledRcol
   \xright@appenditem{\the\absline@numR}\to\actionlines@listR
    \ifsublines@
      \xright@appenditem{-1001}\to\actions@listR
    \else
      \xright@appenditem{-1002}\to\actions@listR
    \fi
  \else
    \xright@appenditem{\the\absline@num}\to\actionlines@list
    \ifsublines@
      \xright@appenditem{-1001}\to\actions@list
    \else
      \xright@appenditem{-1002}\to\actions@list
    \fi
  \fi}

```

\do@lockon
\do@lockonR

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

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

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

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

\lock@off
\do@lockoff
\do@lockoffR
\skip@lockoff

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

```


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

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

```

\n@num

\@ref
\insert@countR

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

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

```
    \newcount\insert@countR
```

- #2, a sequence of other line-list-file commands, executed to determine the ending line-number. (This may also include other \@ref commands, corresponding to uses of \edtext within the first argument of another instance of \edtext.)

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

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

Next, process the second argument to determine the page and line numbers for the end of this lemma. We temporarily equate `\@ref` to a different macro that just executes its argument, so that nested `\@ref` commands are just skipped this time. Some other macros need to be temporarily redefined to suppress their action.

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

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

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

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

```
   \expandafter\list@create\expandafter{\csname sw@list@edtext@tmp@\the\@edtext@lev
```

Declare and init boolean for lemma in this level.

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

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

```
    #2
% Now, we store the list of \cs{@sw} of this current \cs{edtext} as an element of
% the global list of list of \cs{@sw} for a \cs{edtext} depth.
%    \begin{macrocode}
    \ifnum\@edtext@level>0%
    \def\create@this@edtext@level{\expandafter\list@create\expandafter{\csname sw@l
    \ifcsundef{sw@list@edtextR@\the\@edtext@level}{\create@this@edtext@level}{}%
```

```
      \letcs{\@tmp}{sw@list@edtextR@\the\@edtext@level}%
      \letcs{\@tmpp}{sw@list@edtext@tmp@\the\@edtext@level}%
      \xright@appenditem{\expandonce\@tmpp}\to\@tmp%
    \global\cslet{sw@list@edtextR@\the\@edtext@level}{\@tmp}%
    \fi%
```

Decrease edtext level counter.

```
    \global\advance\@edtext@level by -1%
  \else
```

And when not in right text

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

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

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

```

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

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

```

IV.10 Writing to the line-list file

We've now defined all the counters, lists, and commands involved in reading the line-list file at the start of a section. Now we'll cover the commands that Eledmac uses within the text of a section to write commands out to the line-list.

\linenum@outR The file for right texts will be opened on output stream `\linenum@outR`.

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

ffirst@linenum@out@R
rst@linenum@out@Rtrue
st@linenum@out@Rfalse

Once any file is opened on this stream, we keep it open forever, or else switch to another file that we keep open.

```
\newif\iffirst@linenum@out@R
  \first@linenum@out@Rtrue
```

\line@list@stuffR

This is the right text version of the `\line@list@stuff{`⟨*file*⟩`}` macro. It is called by `\beginnumberingR` and performs all the line-list operations needed at the start of a section. Its argument is the name of the line-list file.

```
\newcommand*{\line@list@stuffR}[1]{%
  \read@linelist{#1}%
  \iffirst@linenum@out@R
     \immediate\closeout\linenum@outR
     \global\first@linenum@out@Rfalse
     \immediate\openout\linenum@outR=#1
    \immediate\write\linenum@outR{\string\line@list@version{\this@line@list@version
  \else
     \if@minipage%
       \leavevmode%
     \fi%
     \closeout\linenum@outR%
     \openout\linenum@outR=#1%
  \fi}

```

\new@lineL

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

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

\new@lineR

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

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

\flag@start
\flag@end

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

\startsub
\endsub

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

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

```

\advanceline

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

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

\setline

You can use \setline{⟨*num*⟩} in running text (i.e., within \pstart...\pend) to set the current visible line-number to a specified positive value.

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

\setlinenum

You can use \setlinenum{⟨*num*⟩} before a \pstart to set the visible line-number to a specified positive value. It writes a \l@d@set command to the line-list file.

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

```

\startlock
\endlock

You can use \startlock or \endlock in running text to start or end line number locking at the current line. They decide whether line numbers or sub-line numbers are affected, depending on the current state of the sub-lineation flags.

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

```

\skipnumbering

V Marking text for notes

The \edtext (or \critext) macro is used to create all footnotes and endnotes, as well as to print the portion of the main text to which a given note or notes is keyed. The idea is to have that lemma appear only once in the .tex file: all instances of it in the main text and in the notes are copied from that one appearance.

\critext requires two arguments. At any point within numbered text, you use it by saying:

```
\critext{#1}#2/
```

Similarly \edtext requires the same two arguments but you use it by saying:

```
\edtext{#1}{#2}
```

\critext

\edtext

And similarly for \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.

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

```

V.1 Specific hooks and commands for notes

The reledmac \newseries@ initializes commands which are linked to notes series. 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

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

V.1.1 Notes to be printed on one side only

eledpar 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.

```
    \unless\ifnofamiliar@%
      \csgdef{onlysideX@#1}{}%
      \global\newbool{keepforsideX@#1}%
    \fi%
    \unless\ifnocritical@%
      \global\newbool{keepforXside@#1}%
      \csgdef{Xonlyside@#1}{}%
    \fi%
```

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 Eledmac

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

Then, declare the \footnoteXnomk command.

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

First step: just call the normal \footnoteX, saying that we don't want to print the mark.

```
         \toggletrue{nomk@}%
         \csuse{footnote#1}{##1}%
         \togglefalse{nomk@}%
```

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.

```
         \letcs{\@tmp}{footnote#1@mk}%
         \numdef\@tmpa{\csuse{c@footnote#1}}%
         \global\xright@appenditem{\@tmpa}\to\@tmp%
         \global\cslet{footnote#1@mk}{\@tmp}%
     }%
```

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

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

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

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

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

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

Define the footnote mark and print it

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

Restore previous footnote counter and finally add space.

```
         \setcounter{footnote#1}{\old@footnote}%
         \xspace%
      }%
```

End of tools for familiar notes without marks

```
  \fi
```

End of \newseries@eledpar.

```
}%
```

V.1.3 Create hooks

Read the eledmac code handbook about \newhookcommand@series. Here, we create hooks which are specific to eledpar.

```
\unless\ifnocritical@%
 \newhookcommand@series{Xonlyside}%
\fi%
\unless\ifnofamiliar@%
 \newhookcommand@series{onlysideX}%
\fi


```

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

\init@series@eledpar

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

```
\newcommand{\init@series@eledpar}{%
 \def\do##1{\newseries@par{##1}}%
 \dolistloop{\@series}%
}%
\init@series@eledpar%
```

VI Pstart numbers dumping and restoration

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

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

\list@pstartL@pc
\list@pstartR@pc

```
\list@create{\list@pstartL@pc}%
\list@create{\list@pstartR@pc}%
```

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

\dump@pstartL@pc
\dump@pstartR@pc

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

\def\dump@pstartR@pc{%
  \xright@appenditem{\the\c@pstartR}\to\list@pstartR@pc%
  \global\cslet{numberpstart@R\the\l@dnumpstartsR}{\ifnumberpstart}%
}%

```

\restore@pstartL@pc
\restore@pstartR@pc

And so, the commands to restore them

```
\def\restore@pstartL@pc{%
  \ifx\list@pstartL@pc\empty\else%
    \gl@p\list@pstartL@pc\to\@temp%
    \global\c@pstartL=\@temp%
  \fi%
}%
\def\restore@pstartR@pc{%
  \ifx\list@pstartR@pc\empty\else%
    \gl@p\list@pstartR@pc\to\@temp%
    \global\c@pstartR=\@temp%
  \fi%
}%
```

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 parallel pages.

```
\newenvironment{pairs}{%}
  \l@dpairingtrue
  \l@dpagingfalse
  \initnumbering@quote
  \at@begin@pairs%
}{%
  \l@dpairingfalse
}

```

\AtBeginPairs

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

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

```

The pages environment additionally sets the 'column' widths to the \textwidth (as known at the time the package is called). In this environment, there are two text in parallel on 2 pages. To prevent chapters starting on a lefthand page, the \chapter command is redefined to not clear pages.

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

```

ifinstanzaL
ifinstanzaR

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

```
 \newif\ifinstanzaL
 \newif\ifinstanzaR
```

Leftside

Within the pairs and pages environments the left and right hand texts are within Leftside and Rightside environments, respectively. The Leftside environment is simple, indicating that right text is not within its purview and using some particular macros.

```
\newenvironment{Leftside}{%
  \expandafter\ifvoid\csname l@dLcolrawbox1\endcsname\else%
   \led@err@Leftside@PreviousNotPrinted%
  \fi%
  \ledRcolfalse
  \setcounter{pstartL}{1}
  \let\pstart\pstartL
  \let\thepstart\thepstartL
  \let\pend\pendL
  \let\memorydump\memorydumpL
  \Leftsidehook
  \let\old@startstanza\@startstanza
 \def\@startstanza[##1]{\global\instanzaLtrue\old@startstanza[##1]}
}{
    \Leftsidehookend}
```

\Leftsidehook
\Leftsidehookend
\Rightsidehook
\Rightsidehookend

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

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

```

Rightside

The Rightside environment is only slightly more complicated than the Leftside. Apart from indicating that right text is being provided it ensures that the right right text code will be used.

```
\newenvironment{Rightside}{%
  \expandafter\ifvoid\csname l@dRcolrawbox1\endcsname\else%
   \led@err@Rightside@PreviousNotPrinted%
  \fi%
  \ledRcoltrue
  \let\beginnumbering\beginnumberingR
  \let\endnumbering\endnumberingR
  \let\pausenumbering\pausenumberingR
  \let\resumenumbering\resumenumberingR
  \let\memorydump\memorydumpR
  \let\thepstart\thepstartR
  \let\pstart\pstartR
  \let\pend\pendR
  \let\ledpb\ledpbR
  \let\lednopb\lednopbR
  \let\lineation\lineationR
  \Rightsidehook
  \let\old@startstanza\@startstanza
 \def\@startstanza[##1]{\global\instanzaRtrue\old@startstanza[##1]}
}{%
```

```
  \ledRcolfalse
  \Rightsidehookend
}

```

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's complete. When we chop it up into lines, each line in turn goes into the \one@line or \one@lineR register, and \par@line(R) will be the number of that line within the paragraph.

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

\pstartL
\pstartR

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

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

We have to have specific left and right \pstart when parallel processing; among other things because of potential changes in the linewidth.

```

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

\newcommandx*{\pstartL}[1][1]{%
  \if@nobreak%
    \let\@oldnobreak\@nobreaktrue%
```

```
\else%
  \let\@oldnobreak\@nobreakfalse%
\fi%
  \@nobreaktrue%
\ifluatex%
  \xdef\l@luatextextdir@L{\the\luatextextdir}%
  \xdef\l@luatexpardir@L{\the\luatexpardir}%
  \xdef\l@luatexbodydir@L{\the\luatexbodydir}%
\fi%
\ifnumbering \else%
  \led@err@PstartNotNumbered%
  \beginnumbering%
\fi%
\ifnumberedpar@%
  \led@err@PstartInPstart%
  \pend%
\fi%
```

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

```
\ifpst@rtedL\else%
  \list@clear{\inserts@list}%
  \global\let\next@insert=\empty%
   \global\pst@rtedLtrue%
\fi%
\begingroup\normal@pars%
```

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

```
\global\advance\l@dnumpstartsL \@ne%
\ifnum\l@dnumpstartsL>\l@dc@maxchunks%
  \led@err@TooManyPstarts%
  \global\l@dnumpstartsL=\l@dc@maxchunks%
\fi%
\global\setnamebox{l@dLcolrawbox\the\l@dnumpstartsL}=\vbox\bgroup%
```

We set all the usual interline penalties to zero; this ensures that there'll be no large interline penalties to prevent us from slicing the paragraph into pieces. These penalties revert to the values that you set when the group for the \vbox ends.

```
\l@dzeropenalties%
\ifautopar\else%
  \ifnumberpstart%
    \ifsidepstartnum%
      \else%
       \thepstartL%
      \fi%
     \fi%
    \fi%
\hsize=\Lcolwidth%
\numberedpar@true%
```

```
  \iflabelpstart\protected@edef\@currentlabel%
       {\p@pstartL\thepstartL}\fi%
```

Dump the optional arguments

```
  \ifstrempty{#1}%
   {\csgdef{before@pstartL@\the\l@dnumpstartsL}{\at@every@pstart}}%
   {\csgdef{before@pstartL@\the\l@dnumpstartsL}{\noindent#1}}%
   \at@every@pstart@call%
  }
\newcommandx*{\pstartR}[1][1]{%
  \if@nobreak%
    \let\@oldnobreak\@nobreaktrue%
  \else%
    \let\@oldnobreak\@nobreakfalse%
  \fi%
    \@nobreaktrue%
  \ifluatex%
    \xdef\l@luatextextdir@R{\the\luatextextdir}%
    \xdef\l@luatexpardir@R{\the\luatexpardir}%
    \xdef\l@luatexbodydir@R{\the\luatexbodydir}%
  \fi%
  \ifnumberingR \else%
    \led@err@PstartNotNumbered%
    \beginnumberingR%
  \fi%
  \ifnumberedpar@%
    \led@err@PstartInPstart%
    \pendR%
  \fi%
  \ifpst@rtedR\else%
    \list@clear{\inserts@listR}%
    \global\let\next@insertR=\empty%
    \global\pst@rtedRtrue%
  \fi%
  \begingroup\normal@pars%
  \global\advance\l@dnumpstartsR \@ne%
  \ifnum\l@dnumpstartsR>\l@dc@maxchunks%
    \led@err@TooManyPstarts%
    \global\l@dnumpstartsR=\l@dc@maxchunks%
  \fi%
 \global\setnamebox{l@dRcolrawbox\the\l@dnumpstartsR}=\vbox\bgroup%
    \l@dzeropenalties%
    \ifautopar\else%
      \ifnumberpstart%
        \ifsidepstartnum\else%
          \thepstartR%
         \fi%
        \fi%
       \fi%
  \hsize=\Rcolwidth%
  \numberedpar@true%
   \iflabelpstart\protected@edef\@currentlabel%
       {\p@pstartR\thepstartR}\fi%
   \ifstrempty{#1}%
   {\csgdef{before@pstartR@\the\l@dnumpstartsR}{\at@every@pstart}}%
   {\csgdef{before@pstartR@\the\l@dnumpstartsR}{\noindent#1}}%
   \at@every@pstart@call%
  }
```

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

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

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

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

End the group that was begun in the \pstart.

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

Dump content of the optional argument.

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

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

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

```

\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.

```
\newcommand{\AtEveryPstartCall}[1]{\xdef\at@every@pstart@call{\unexpanded{#1}}}%
\gdef\at@every@pstart@call{}%
```

rint@last@after@pendL
rint@last@after@pendR

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

```
\newif\ifprint@last@after@pendL%
\newif\ifprint@last@after@pendR%
```

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
\l@drightbox

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

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

```

\countLline
\countRline

We need to know the number of lines processed.

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

```

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

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

```
\newcount\@donereallinesL
\newcount\@donetotallinesL
```

```
\newcount\@donereallinesR
\newcount\@donetotallinesR

```

\do@lineL

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

```
\newcommand*{\do@lineL}{%
  \letcs{\ifnumberpstart}{numberpstart@L\the\l@dpscL}%
  \advance\countLline \@ne%
  \ifvbox\namebox{l@dLcolrawbox\the\l@dpscL}%
  {\vbadness=10000%
   \splittopskip=\z@%
   \do@lineLhook%
   \l@demptyd@ta%
  \global\setbox\one@line=\vsplit\namebox{l@dLcolrawbox\the\l@dpscL}%
                          to\baselineskip}%
 \IfStrEq{\splitfirstmarks\parledgroup@}{begin}{\parledgroup@notes@startL
  \unvbox\one@line \global\setbox\one@line=\lastbox%
  \getline@numL%
  \ifnum\@lock>\@ne%
    \inserthangingsymboltrue%
  \else%
    \inserthangingsymbolfalse%
  \fi%
  \setbox\l@dleftbox%
  \hb@xt@ \Lcolwidth{%
    \ifl@dhidenumber%
      \global\l@dhidenumberfalse%
      \f@x@l@cks%
    \else%
      \affixline@num%
    \fi%
    \xifinlist{\the\l@dpscL}{\eled@sections@@}%
      {\add@inserts\affixside@note}%
      {\print@lineL}}%
  \add@penaltiesL%
  \global\advance\@donereallinesL\@ne%
  \global\advance\@donetotallinesL\@ne%
\else%
 \setbox\l@dleftbox \hb@xt@ \Lcolwidth{\hspace*{\Lcolwidth}}%
  \global\advance\@donetotallinesL\@ne%
\fi}


```

\print@lineL

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

```
\def\print@lineL{%
   \affixpstart@numL%
```

```
    \l@dld@ta %space kept for backward compatibility
    \add@inserts\affixside@note%
    \l@dlsn@te %space kept for backward compatibility
    {\ledllfill\hb@xt@ \Lcolwidth{%
            \do@insidelineLhook%
            \ifluatex%
              \luatextextdir\l@luatextextdir@L%
            \fi%
            \new@lineL%
            \inserthangingsymbolL%
            \l@dunhbox@line{\one@line}}\ledrlfill\l@drd@ta%
     \l@drsn@te}}

```

`\print@eledsectionL`

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

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

```

`\dolineLhook`
`\dolineRhook`
`\doinsidelineLhook`
`\doinsidelineRhook`

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

```
\newcommand*{\dolineLhook}[1]{\gdef\do@lineLhook{#1}}%
\newcommand*{\dolineRhook}[1]{\gdef\do@lineRhook{#1}}%
\newcommand*{\doinsidelineLhook}[1]{\gdef\do@insidelineLhook{#1}}%
\newcommand*{\doinsidelineRhook}[1]{\gdef\do@insidelineRhook{#1}}%
```

```

```

\do@lineLhook
\do@lineRhook
\do@insidelineLhook
\do@insidelineRhook

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

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

```

\do@lineR

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

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


```

\print@lineR
\print@eledsectionR

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're about to send to the vertical list.

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

\do@ballastR

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

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

\l@dskipversenumberR
\do@actionsR
\do@actions@fixedcodeR
\do@actions@nextR

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

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

```

\newif\ifl@dskipversenumberR
\newcommand*{\do@actions@fixedcodeR}{%
  \ifcase\@l@dtempcnta%
  \or%                     % 1001
    \global\sublines@true
  \or%                     % 1002
    \global\sublines@false
  \or%                     % 1003
    \global\@lockR=\@ne
  \or%                     % 1004%
    \ifnum\@lockR=\tw@
      \global\@lockR=\thr@@
    \else
      \global\@lockR=\z@
    \fi
  \or%                     % 1005
     \global\sub@lockR=\@ne
  \or%                     % 1006
    \ifnum\sub@lockR=\tw@
      \global\sub@lockR=\thr@@
    \else
      \global\sub@lockR=\z@
    \fi
  \or%                     % 1007
    \l@dskipnumbertrue
 \or%                      % 1008
    \l@dskipversenumberRtrue%
 \or%                     % 1009
    \l@dhidenumbertrue%
 \else%
    \led@warn@BadAction
 \fi%
}


\newcommand*{\do@actionsR}{%
  \global\let\do@actions@nextR=\relax
  \@l@dtempcntb=\absline@numR
  \ifnum\@l@dtempcntb<\next@actionlineR\else
   \ifnum\next@actionR>-1001\relax
      \global\page@numR=\next@actionR
      \ifbypage@R
        \global\line@numR \z@  \global\subline@numR \z@
      \fi
    \else
    \ifnum\next@actionR<-4999\relax  %  9/05 added relax here
        \@l@dtempcnta=-\next@actionR
        \advance\@l@dtempcnta by -5001\relax
        \ifsublines@
          \global\subline@numR=\@l@dtempcnta
        \else
          \global\line@numR=\@l@dtempcnta
        \fi
      \else
        \@l@dtempcnta=-\next@actionR
        \advance\@l@dtempcnta by -1000\relax
        \do@actions@fixedcodeR
      \fi
    \fi
    \ifx\actionlines@listR\empty
      \gdef\next@actionlineR{1000000}%
    \else
      \gl@p\actionlines@listR\to\next@actionlineR
      \gl@p\actions@listR\to\next@actionR
      \global\let\do@actions@nextR=\do@actionsR
    \fi
  \fi
  \do@actions@nextR}

```

VIII.4 Line number printing

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

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

```

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

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

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

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

```
    \global\sub@lockR \tw@
  \or \or
    \global\sub@lockR \z@
  \fi}


\newcommand*{\affixline@numR}{%
\ifledgroupnotesR@\else\ifnumberline
\ifl@dskipnumber
  \global\l@dskipnumberfalse
\else
  \ifsublines@
    \@l@dtempcntb=\subline@numR
   \l@dcalcnum{\subline@numR}{\c@firstsublinenumR}{\c@sublinenumincrementR}%
     \ch@cksub@lockR
  \else
    \@l@dtempcntb=\line@numR
    \ifx\linenumberlist\empty
    \l@dcalcnum{\line@numR}{\c@firstlinenumR}{\c@linenumincrementR}%
    \else
      \@l@dtempcnta=\line@numR
      \edef\rem@inder{,\linenumberlist,\number\line@numR,}%
        \edef\sc@n@list{\def\noexpand\sc@n@list
     ####1,\number\@l@dtempcnta,####2|{\def\noexpand\rem@inder{####2}}}%
      \sc@n@list\expandafter\sc@n@list\rem@inder|%
        \ifx\rem@inder\empty\advance\@l@dtempcnta\@ne\fi
     \fi
     \ch@ck@l@ckR
  \fi
  \ifnum\@l@dtempcnta=\@l@dtempcntb
    \ifl@dskipversenumberR\else
      \if@twocolumn
        \if@firstcolumn
          \gdef\l@dld@ta{\llap{{\leftlinenumR}}}%
        \else
          \gdef\l@drd@ta{\rlap{{\rightlinenumR}}}%
        \fi
      \else
        \@l@dtempcntb=\line@marginR
        \ifnum\@l@dtempcntb>\@ne
          \advance\@l@dtempcntb by\page@numR
        \fi
        \ifodd\@l@dtempcntb
          \gdef\l@drd@ta{\rlap{{\rightlinenumR}}}%
        \else
          \gdef\l@dld@ta{\llap{{\leftlinenumR}}}%
        \fi
      \fi
    \fi
  \fi
```

```
 \f@x@l@cksR
\fi
\fi
\fi}
```

VIII.5 Pstart number printing in side

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

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

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

```

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

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

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.

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

\add@insertsR
\add@inserts@nextR

The right text version.

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

VIII.7 Penalties

\add@penaltiesL
\add@penaltiesR

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

In the code below, which is a virtual copy of the original \add@penalties, \num@lines is the number of lines in the whole paragraph, and \par@line is the line we're working on at the moment. The count \@l@dtempcnta is used to calculate and accumulate the penalty; it is initially set to the value of \ballast@count, which has been worked out in \do@ballast. Finally, the penalty is checked to see that it doesn't go below −10000.

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

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

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

```

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.

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

```

IX Footnotes

IX.1 Normal footnote formatting

The \printlines macro prints the line numbers for a note—which, in the general case, is a rather complicated task. The seven parameters of the argument are the line numbers as stored in \l@d@nums, in the form described on ?? p. ?? 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
\ledsavedprintlines

This is the right text version of \printlines and takes account of \@Rlineflag. Just in case, \ledsavedprintlines is a copy of the original \printlines.

Just a reminder of the arguments:

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

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

\let\ledsavedprintlines\printlines

```

IX.2 Footnotes output specific to \Pages

\print@Xnotes@forpages
\correct@Xfootins@box
\print@notesX@forpages
\correct@footinsX@box

The \Xonlyside and \onlysideX hooks for \Pages allow notes to be printed either in left or right pages only. The implementation of such features is delegated to \print@Xnotes@forpages, which replaces \print@Xnotes inside \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 don't 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 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 don't 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.

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

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 splitted 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`

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

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

```
   \ifcsempty{Xonlyside@#1}{%
     \csuse{#1footstart}{#1}%
     \csuse{#1footgroup}{#1}%
   }%
```

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

```
   {%
     \ifboolexpr{%
     ((test {\ifcsstring{Xonlyside@#1}{L}} and not test{\ifnumodd{\c@page}})%
       or%
     (test {\ifcsstring{Xonlyside@#1}{R}} and test{\ifnumodd{\c@page}}))%
       }%
```

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

```
      {%
         \correct@Xfootins@box{#1}%
         \csuse{#1footstart}{#1}%
         \csuse{#1footgroup}{#1}%
```

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

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

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

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

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

```
         \global\boolfalse{keepforXside@#1}%
       }%
```

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.

```
       {%
         \global\booltrue{keepforXside@#1}%
```

Then restore expected rooms for notes on the next page.

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

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

```
         \bgroup%
            \csuse{Xnotefontsize@#1}%
            \global\advance\dimen\csuse{#1footins} by \baselineskip%
         \egroup%
```

```
% End of \cs{print@Xnotes@forpages}.
       }%
     }%
}%
```

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

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

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.

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

Some setting needed to do the right splitting.

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

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

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

End of the macro.

```
  }{}%
}%
```

And now, the same for familiar footnotes.

```
\newcommand\print@notesX@forpages[1]{%
   \ifcsempty{onlysideX@#1}{%
     \csuse{footstart#1}{#1}%
     \csuse{footgroup#1}{#1}%
   }%
   {%
     \ifboolexpr{%
     ((test {\ifcsstring{onlysideX@#1}{L}} and not test{\ifnumodd{\c@page}})%
       or%
     (test {\ifcsstring{onlysideX@#1}{R}} and test{\ifnumodd{\c@page}}))%
       }%
      {%
         \correct@footinsX@box{#1}%
         \csuse{footstart#1}{#1}%
         \csuse{footgroup#1}{#1}%
         \global\count\csuse{footins#1}=0%
         \global\skip\csuse{footins#1}=0pt%
          \csuse{notefontsizeX@#1}%
                \global\advance\dimen\csuse{footins#1}  by -\baselineskip%
       \global\boolfalse{keepforsideX@#1}%
       }%
       {%
          \global\booltrue{keepforsideX@#1}%
      \global\count\csuse{footins#1}=\csuse{default@footins#1}%
      \global\skip\csuse{footins#1}=\csuse{beforenotesX@#1}%
         \bgroup%
            \csuse{notefontsizeX@#1}%
            \global\advance\dimen\csuse{footins#1} by \baselineskip%
         \egroup%
       }%
     }%
}%
\newcommand{\correct@footinsX@box}[1]{%
  \ifbool{keepforsideX@#1}{%
      \csuse{notefontsizeX@#1}%
      \splittopskip=0pt%
      \global\setbox\csuse{footins#1}=\vbox{%
     \vsplit\csuse{footins#1} to \dimexpr\ht\csuse{footins#1}-1pt\relax%
             \vskip  \dimexpr-0.5\baselineskip-0.5\lineskip-0.5pt\relax%
       \unvbox\csuse{footins#1}%
      }%
  }{}%
}%
```

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.

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

```

\edlabel

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

\l@dmake@labelsR

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

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

```

\@lab

The \@lab command, which appears in the \linenum@out file, appends the current values of page, line and sub-line to the \labelref@list. These values are defined by the earlier \@page, \@nl, and the \sub@on and \sub@off commands appearing in the \linenum@out file.

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

```

XI Side notes

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

\sidenote@marginR
\sidenotemargin*

Specifies which margin sidenotes can be in.

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

```

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

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

```
        \setl@drp@rbox{\sidenotecontent@}%
      \fi%
    \fi%
  \fi%
}

```

XII Familiar footnotes

\l@dbfnote

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

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

```

\normalbfnoteX

```
\renewcommand{\normalbfnoteX}[2]{%
  \ifnumberedpar@
    \ifledRcol%
      \ifluatex
          \footnotelang@lua[R]%
      \fi
      \@ifundefined{xpg@main@language}%if polyglossia
        {}%
        {\footnotelang@poly[R]}%
      \protected@xdef\thisfootnote{\csuse{thefootnote#1}}%
    \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\expandonce\thisfootnote
                         \to\inserts@listR
       \global\advance\insert@countR \@ne%
    \else%
      \ifluatex
          \footnotelang@lua%
      \fi
      \@ifundefined{xpg@main@language}%if polyglossia
        {}%
        {\footnotelang@poly}%
      \protected@xdef\thisfootnote{\csuse{thefootnote#1}}%
    \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\expandonce\thisfootnote}}%
                         \to\inserts@list
     \global\advance\insert@count \@ne%
    \fi
  \fi\ignorespaces}

```

XIII Verse

Like in eledmac, the insertion of hangingsymbol is base on \ifinserthangingsymbol, 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

```
\newif\ifinserthangingsymbolR
\newcommand{\inserthangingsymbolL}{%
  \ifinserthangingsymbol%
    \ifinstanzaL%
        \hskip \@ifundefined{sza@0@}{0}{\expandafter%
        \noexpand\csname sza@0@\endcsname}\stanzaindentbase%
        \@hangingsymbol%
    \fi%
  \fi%
}%
\newcommand{\inserthangingsymbolR}{%
  \ifinserthangingsymbolR%
    \ifinstanzaR%
        \hskip \@ifundefined{sza@0@}{0}{\expandafter%
        \noexpand\csname sza@0@\endcsname}\stanzaindentbase%
        \@hangingsymbol%
    \fi%
  \fi%
}%
```

Before we can define the main stanza macros we need to be able to save and reset the category code for &. To save the current value we use \next from the \loop macro.

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

```

astanza

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

```
\newenvironment{astanza}{%
  \catcode`\&\active
  \global\stanza@count\@ne\stanza@modulo\@ne
  \ifnum\usenamecount{sza@0@}=\z@
```

```
    \let\stanza@hang\relax
    \let\endlock\relax
  \else
    \rightskip\z@ plus 1fil\relax
  \fi
  \ifnum\usenamecount{szp@0@}=\z@
    \let\sza@penalty\relax
  \fi
  \def&{%
    \endlock\mbox{}%
    \sza@penalty
    \global\advance\stanza@count\@ne
    \@astanza@line}%
  \def\&{\@stopastanza}%
  \pstart
  \@astanza@line
}{}

```

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

```
\newcommandx{\@stopastanza}[1][1,usedefault]{%
    \endlock\mbox{}%
    \pend[#1]%
}%
```

\@astanza@line This gets put at the start of each line in the environment. It sets up the paragraph style — each line is treated as a paragraph.

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

```

Lastly reset the modified category codes.

```
  \catcode`\&=\next

```

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 \setnamebox \unhnamebox \unvnamebox \namebox

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

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

```

\newnamecount \usenamecount

Macros for creating and using 'named' counts.

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

```

XV Counts and boxes for parallel texts

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

\maxchunks \l@dc@maxchunks

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

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

```

\l@dnumpstartsL \l@dnumpstartsR

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

```
\newcount\l@dnumpstartsR

```

\l@pscL \l@pscR

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

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

```

\l@dsetuprawboxes

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

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

```

\l@dsetupmaxlinecounts
\l@dzeromaxlinecounts

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

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

```

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

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

```

XVI Fixing babel

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

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

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

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

```
\newif\ifl@dusedbabel
```

\ifl@dsamelang

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

\l@dchecklang

\l@dbbl@set@language

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

```
\newcommand*{\l@dbbl@set@language}[1]{%
  \edef\languagename{#1}%
  \select@language{\languagename}%
  \if@filesw
   \protected@write\@auxout{}{\string\select@language{\languagename}}%
   \addtocontents{toc}{\string\select@language{\languagename}}%
   \addtocontents{lof}{\string\select@language{\languagename}}%
   \addtocontents{lot}{\string\select@language{\languagename}}%
  \fi}

```

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

\selectlanguage
\l@duselanguage
\theledlanguageL
\theledlanguageR

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

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

```

Now do the babel fix or polyglossia, if necessary.

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

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

```
    \l@dusedbabelfalse
    \renewcommand*{\selectlanguage}[1]{}}{%
```

Here we deal with the case where babel has been used. \selectlanguage has to be redefined to use our version of \bbl@set@language and to store the left or right language.

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

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

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

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

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

If on Polyglossia

```
  {    \let\old@otherlanguage\otherlanguage%
       \renewcommand{\otherlanguage}[2][]{%
        \selectlanguage[#1]{#2}%
        \ifledRcol  \gdef\theledlanguageR{#2}%
        \else      \gdef\theledlanguageL{#2}%
        \fi}%
      \let\l@duselanguage\select@language%
      \gdef\theledlanguageL{\xpg@main@language}%
      \gdef\theledlanguageR{\xpg@main@language}%
```

That's it.

```
}}
```

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

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

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

```
    \fi
  \fi
}

```

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

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

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

```

\@writelinesinparL
\@writelinesinparR

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

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

```

XVII Parallel columns

\@eledsectionL
\@eledsectionR

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

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

\Columns

The \Columns command results in the previous Left and Right texts being typeset in matching columns. There should be equal numbers of chunks in the left and right texts.

```
\newcommand*{\Columns}{%
  \ifl@dpairing%
    \led@err@Columns@InsideEnv%
  \fi%
  \l@dprintingcolumnstrue%
```

```
 \eledsection@correcting@skip=-\baselineskip% Correction for sections' titles
  \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
   \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
  \fi
```

Start a group and zero counters, etc.

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

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

```
    \check@pstarts
    \loop\if@pstarts
        \global\pstartnumtrue
        \global\pstartnumRtrue
```

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

```
      \global\advance\l@dpscL \@ne
      \global\advance\l@dpscR \@ne
      \restore@pstartL@pc%
      \restore@pstartR@pc%
```

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

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

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

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

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

```
            \l@duselanguage{\theledlanguageL}%
            \do@lineL
            \xifinlist{\the\l@dpscL}{\eled@sections@@}
                {%
                \ifdefstring{\@eledsectmark}{L}%
                  {\csuse{eled@sectmark@\the\l@dpscL}%
                  }{}%
                \global\csundef{eled@sectmark@\the\l@dpscL}%
           \savebox{\@eledsectionL}{\parbox[t][][t]{\Lcolwidth}{\vbox{}\pr
> prevent alignment troubles with RTL language
                }%
                {}%
            \l@duselanguage{\theledlanguageR}%
            \do@lineR
            \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}
                {%
                \ifdefstring{\@eledsectmark}{R}%
                  {\csuse{eled@sectmark@\the\l@dpscR R}%
                  }{}%
              \global\csundef{eled@sectmark@\the\l@dpscR R}%
           \savebox{\@eledsectionR}{\parbox[t][][t]{\Rcolwidth}{\vbox{}\print@eleds
> prevent alignment troubles with RTL language
                {}%
          \hb@xt@ \hsize{%
           \ifdefstring{\columns@position}{L}{}{\hfill }%
            \unhbox\l@dleftbox%
            \ifhbox\@eledsectionL%
               \usebox{\@eledsectionL}%
            \fi%
            \print@columnseparator%
            \unhbox\l@drightbox%
            \ifhbox\@eledsectionR%
              \usebox{\@eledsectionR}%
            \fi%
            \ifdefstring{\columns@position}{R}{}{\hfill}%
          }%
          \checkraw@text
          \checkverseL
          \checkverseR
          \checkpb@columns
        \repeat}
```

Having completed a pair of chunks, write the number of lines in each chunk to the respective section files. Increment pstart counters and reset line numbering if it's by pstart.

```
      \@writelinesinparL
      \@writelinesinparR
      \check@pstarts
        \ifbypstart@%
           \write\linenum@out{\string\@set[1]}
           \resetprevline@
        \fi
       \ifbypstart@R
           \write\linenum@outR{\string\@set[1]}
           \resetprevline@
        \fi
        \Columns@print@after@pend%
    \repeat
```

Having output all chunks, make sure all notes have been output, then zero counts ready for the next set of texts. The boolean tests for stanza are switched to false.

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

```

\print@columnseparator

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

```
\def\print@columnseparator{%
  \ifdim\beforecolumnseparator<0pt%
    \hfill%
  \else%
    \hspace{\beforecolumnseparator}%
  \fi%
  \columnseparator%
  \ifdim\aftercolumnseparator<0pt%
    \hfill%
  \else%
    \hspace{\beforecolumnseparator}%
  \fi%
}%
%\end{macrocode}
% \end{macro}
% \begin{macro}{\checkpb@columns}
% \cs{checkpb@columns} prevent or make pagebreaking in columns, depending of the use of \cs{ledpb} or \cs{lednopb}.
%    \begin{macrocode}

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

\columnseparator
\columnrulewidth

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

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

```

\columnsposition
\columns@position

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

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

eforecolumnseparator
aftercolumnseparator

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

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

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

```

vidthliketwocolumns@L
tionliketwocolumns@L
tionliketwocolumns@L
vidthliketwocolumns@C
tionliketwocolumns@C
tionliketwocolumns@C
vidthliketwocolumns@R
tionliketwocolumns@R
tionliketwocolumns@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.

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

Hsize : Left + Right width

```
  \hsize=\Lcolwidth%
  \advance\hsize\Rcolwidth%
```

Now, calculating the remaining space

```
  \advance\temp-\hsize%
```

And multiply the hsize by 2/3 of this space

```
  \multiply\temp by 2%
  \divide\temp by 3%
  \advance\hsize\temp%
}%

\newcommand{\setpositionliketwocolumns@L}{%
  \renewcommand{\ledrlfill}{\hfill}%
}%

\newcommand{\setnotespositionliketwocolumns@L}{%
}%


\newcommand{\setwidthliketwocolumns@C}{%
% Temporary dimension, initially equal to the standard hsize, i.e. text width
  \newdimen\temp%
  \temp=\hsize%
% Hsize : Left + Right width
  \hsize=\Lcolwidth%
  \advance\hsize\Rcolwidth%
% Now, calculating the remaining space
  \advance\temp-\hsize%
```

And multiply the hsize by 1/2 of this space

```
  \divide\temp by 2%
  \advance\hsize\temp%
}%

\newcommand{\setpositionliketwocolumns@C}{%
  \doinsidelinehook{\hfill}%
  \renewcommand{\ledrlfill}{\hfill}%
}%

\newcommand{\setnotespositionliketwocolumns@C}{%
  \newdimen\temp%
  \newdimen\tempa%
  \temp=\hsize%
  \tempa=\Lcolwidth%
  \advance\tempa\Rcolwidth%
  \advance\temp-\tempa%
  \divide\temp by 2%
  \leftskip=\temp%
  \rightskip=-\temp%
}%
```

```

\newcommand{\setwidthliketwocolumns@R}{%
```

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

```
   \newdimen\temp%
   \temp=\hsize%
```

Hsize : Left + Right width

```
   \hsize=\Lcolwidth%
   \advance\hsize\Rcolwidth%
```

Now, calculating the remaining space

```
   \advance\temp-\hsize%
```

And multiply the hsize by 2/3 of this space

```
   \multiply\temp by 2%
   \divide\temp by 3%
   \advance\hsize\temp%
}%

\newcommand{\setpositionliketwocolumns@R}{%
   \doinsidelinehook{\hfill}%
}%

\newcommand{\setnotespositionliketwocolumns@R}{%
   \newdimen\temp%
   \newdimen\tempa%
   \temp=\hsize%
   \tempa=\Lcolwidth%
   \advance\tempa\Rcolwidth%
   \advance\temp-\tempa%
   \divide\temp by 2%
   \leftskip=\temp%
   \rightskip=-\temp%
}%

```

s@print@before@pstart
umns@print@after@pend

The \Columns@print@before@pstart and \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.

```
\newcommand{\Columns@print@before@pstart}{%
   \ifboolexpr{%
   test{\ifcsstring{before@pstartL@\the\l@dpscL}{\at@every@pstart}}%
   and test {\ifcsstring{before@pstartR@\the\l@dpscR}{\at@every@pstart}}%
    and test {\ifdefempty{\at@every@pstart}}}%
       {}%
       {%
         \hb@xt@ \hsize{%
           \ifdefstring{\columns@position}{L}{}{\hfill }%
           \par\parbox[t][][t]{\Lcolwidth}{%
             \csuse{before@pstartL@\the\l@dpscL}%
```

```
               }%
               \print@columnseparator%
               \parbox[t][][t]{\Rcolwidth}{%
                  \csuse{before@pstartR@\the\l@dpscR}%
               }%
               \ifdefstring{\columns@position}{R}{}{\hfill}%
            }%
         }%
     \global\csundef{before@pstartL@\the\l@dpscL}%
     \global\csundef{before@pstartR@\the\l@dpscR}%
}%
\newcommand{\Columns@print@after@pend}{%
  \ifboolexpr{%
   test{\ifcsstring{after@pendL@\the\l@dpscL}{\at@every@pend}}%
   and test {\ifcsstring{after@pendR@\the\l@dpscR}{\at@every@pend}}%
     and test {\ifdefempty{\at@every@pend}}}%
         {}%
         {%
            \hb@xt@ \hsize{%
               \ifdefstring{\columns@position}{L}{}{\hfill }%
               \parbox[t][][t]{\Lcolwidth}{%
                  \csuse{after@pendL@\the\l@dpscL}%
               }%
               \print@columnseparator%
               \parbox[t][][t]{\Rcolwidth}{%
                  \csuse{after@pendR@\the\l@dpscR}%
               }%
               \ifdefstring{\columns@position}{R}{}{\hfill}%
            }%
         }%
     \global\csundef{after@pendL@\the\l@dpscL}%
     \global\csundef{after@pendR@\the\l@dpscR}%
}%
```

XVIII Parallel pages

This is considerably more complicated than parallel columns.

XVIII.1 Specific counters

\numpagelinesL
\numpagelinesR
\l@dminpagelines

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

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

```

XVIII.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.

```
\newcommand*{\Pages}{%
  \l@dprintingpagestrue%
  \ifl@dpairing%
    \led@err@Pages@InsideEnv%
  \fi%
  \eledsection@correcting@skip=-2\baselineskip% line correcting for section titles.
  \parledgroup@notespacing@set@correction%
  \typeout{}%
 \typeout{*************************** PAGES ***************************}%
  \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else%
   \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
  \fi%
```

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

```
  \setlength{\Lcolwidth}{\textwidth}%
  \setlength{\Rcolwidth}{\textwidth}%
```

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

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

Sometimes, people what to have the same page number on both left and right sides. To do this, use the \init@sameparallelpage@number command.

```
    \init@sameparallelpage@number
```

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

```
    \let\print@Xnotes\print@Xnotes@forpages%
    \let\print@notesX\print@notesX@forpages%
```

Check if there are chunks to be processed.

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

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

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

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

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

Zero the counts again, ready for the next bit.

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

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

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

Now we start processing the left and right chunks (`\l@dpscL` and `\l@dpscR` count the left and right chunks), starting with the first pair.

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

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

```
      \global\advance\l@dpscL \@ne%
      \global\advance\l@dpscR \@ne%
      \restore@pstartL@pc%
      \restore@pstartR@pc%
```

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

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

Start a loop over the boxes (chunks).

```
      \checkraw@text%
```

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

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

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

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

```
    \ifdefstring{\@eledsectnotoc}{L}{\ledsectnotoc}{}%
       \csuse{before@pstartL@\the\l@dpscL}%
       \global\csundef{before@pstartL@\the\l@dpscL}%
       \do@lineL%
       \xifinlist{\the\l@dpscL}{\eled@sections@@}
         {\print@eledsectionL}%
         {}%
       \advance\numpagelinesL \@ne%
```

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

```
       \ifshiftedpstarts%
             \ifdim\ht\l@dleftbox>0pt\hb@xt@%
              \hsize{\ledstrutL\unhbox\l@dleftbox}%
             \else%
               \dimen0=\pagetotal%
               \advance\dimen0 by \baselineskip%
               \global\pagetotal=\dimen0%
             \fi%
       \else%
             \parledgroup@correction@notespacing{L}
        \hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
       \fi%
```

Perhaps we have to move to the next (left) box. Check if we have got all we can onto the page. If not, repeat for the next line. Check if we have to print the optional argument of the last pend. Check if the page is full. Check if the verse is split in two subsequent pages. Check there is any forced page breaks. Reset the verse skipnumber boolean

```
       \get@nextboxL%
\global\l@dskipversenumberfalse%
          \ifprint@last@after@pendL%
            \csuse{after@pendL@\the\l@dpscL}%
          \global\csundef{after@pendL@\the\l@dpscL}%
           \fi%
       \checkpageL%
       \checkverseL%
       \checkpbL%
      \repeat%
```

That (left) page has been filled. Output the number of real lines on the page — if the page break is because the page has been filled with lines, use the actual number, otherwise the page has been ended early in order to synchronise with the facing page so use an impossibly large number.

```
            \ifl@dpagefull%
              \@writelinesonpageL{\the\numpagelinesL}%
            \else%
              \@writelinesonpageL{1000}%
            \fi%
```

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

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

Now do the same for the right text.

```
          \checkpageR%
          \l@duselanguage{\theledlanguageR}%
{             \loop\ifl@dsamepage%
              \initnumbering@sectcountR%
          \ifdefstring{\@eledsectnotoc}{R}{\ledsectnotoc}{}%
              \csuse{before@pstartR@\the\l@dpscR}%
              \global\csundef{before@pstartR@\the\l@dpscR}%
              \do@lineR%
              \xifinlist{\the\l@dpscR}{\eled@sectionsR@@}%
                {\print@eledsectionR}%
                {}%
              \advance\numpagelinesR \@ne%
              \ifshiftedpstarts%
                      \ifdim\ht\l@drightbox>0pt\hb@xt@%
                      \hsize{\ledstrutR\unhbox\l@drightbox}%
                      \else%
                        \dimen0=\pagetotal%
                        \advance\dimen0 by \baselineskip%
                        \global\pagetotal=\dimen0%
                      \fi%
              \else%
                     \parledgroup@correction@notespacing{R}%
               \hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}%
              \fi%
              \get@nextboxR%
     \global\l@dskipversenumberRfalse%
                  \ifprint@last@after@pendR%
                     \csuse{after@pendR@\the\l@dpscR}%
                  \global\csundef{after@pendR@\the\l@dpscR}%
                   \fi%
              \checkpageR%
              \checkverseR%
              \checkpbR%
            \repeat%
            \ifl@dpagefull%
              \@writelinesonpageR{\the\numpagelinesR}%
            \else%
              \@writelinesonpageR{1000}%
            \fi%
            \numpagelinesR=\z@%
            \parledgroup@correction@notespacing@init%
```

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

```
            \clearl@drightpage}%
```

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

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

We have now output the text from all the chunks.

```
    \fi%
```

Make sure that there are no inserts hanging around.

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

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

```
  \global\l@dpscL=\z@%
  \global\l@dpscR=\z@%
  \global\l@dnumpstartsL=\z@%
  \global\l@dnumpstartsR=\z@%
    \global\instanzaLfalse%
    \global\instanzaRfalse%
 \l@dprintingpagesfalse%
 \finish@sameparallelpage@number%In order to have continuous page number
  \finish@Pages@notes%Needed to prevent final notes overlap line number
  \ignorespaces}


```

XVIII.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 typessetting, and that there is no more long notes left for the next pages.

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

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

```
    \ifnocritical@%
      \global\newnamebox{##1footins}
    \fi
    \ifnofamiliar@%
      \global\newnamebox{footins##1}
    \fi
```

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

```
    \ifvoid\csuse{##1footins}%
      \ifvoid\csuse{footins##1}\else%
        \newpage\null%
        \listbreak%
      \fi%
    \else%
      \newpage\null%
      \listbreak%
    \fi%
  }%
  \dolistloop{\@series}%
}%
```

XVIII.4 Struts

`\ledstrutL` `\ledstrutR`

Struts inserted into leftand right text lines.

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

```

XVIII.5 Page clearing

`\cleartoevenpage` `\cleartol@devenpage`

`\cleartoevenpage`, which is defined in the memoir class, is like `\clear(double)page` except that we end up on an even page. `\cleartol@devenpage` is similar except that it first checks to see if it is already on an empty page.

```
\providecommand{\cleartoevenpage}[1][\@empty]{%
  \clearpage
  \ifodd\c@page\hbox{}#1\clearpage\fi}
\newcommand*{\cleartol@devenpage}{%
  \ifdim\pagetotal<\topskip%  on an empty page
  \else
    \clearpage
  \fi
  \ifodd\c@page\hbox{}\clearpage\fi}
```

`\clearl@dleftpage` `\clearl@drightpage`

`\clearl@dleftpage` and `\clearl@drightpage` get us onto an odd and even 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

```
\newcommand*{\clearl@dleftpage}{%
  \ifdim\pagetotal=0pt\hbox{}\fi%
  \newpage%
  \ifodd\c@page\else
    \led@err@LeftOnRightPage
    \hbox{}%
    \cleardoublepage
  \fi}

\newcommand*{\clearl@drightpage}{%
  \ifdim\pagetotal=0pt\hbox{}\fi%
  \newpage%
  \stepcounter{sameparallelpage@number}%
  \ifodd\c@page
    \led@err@RightOnLeftPage
    \hbox{}%
    \cleartoevenpage
  \fi}

```

XVIII.6 Lines managing

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

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

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

```

etlinesfrompagelistL
\@cs@linesonpageL
etlinesfrompagelistR
\@cs@linesonpageR

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

```
\newcommand*{\getlinesfrompagelistL}{%
  \ifx\linesonpage@listL\empty
    \gdef\@cs@linesonpageL{1000}%
  \else
```

```
    \gl@p\linesonpage@listL\to\@cs@linesonpageL
  \fi}
\newcommand*{\getlinesfrompagelistR}{%
  \ifx\linesonpage@listR\empty
    \gdef\@cs@linesonpageR{1000}%
  \else
    \gl@p\linesonpage@listR\to\@cs@linesonpageR
  \fi}

```

\@writelinesonpageL
\@writelinesonpageR

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

```
\newcommand*{\@writelinesonpageL}[1]{%
  \edef\next{\write\linenum@out{\string\@lopL{#1}}}%
  \next}
\newcommand*{\@writelinesonpageR}[1]{%
  \edef\next{\write\linenum@outR{\string\@lopR{#1}}}%
  \next}

```

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

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

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

```
\newcommand*{\l@dcalc@maxoftwo}[3]{%
  \ifnum #2>#1\relax
    #3=#2\relax
  \else
    #3=#1\relax
  \fi}
\newcommand*{\l@dcalc@minoftwo}[3]{%
  \ifnum #2<#1\relax
    #3=#2\relax
  \else
    #3=#1\relax
  \fi}

```

XVIII.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.

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

```
\newif\ifl@dpagefull

\newcommand*{\checkpageL}{%
  \l@dpagefulltrue
  \l@dsamepagetrue
  \check@goal
  \ifdim\pagetotal<\ledthegoal
    \ifnum\numpagelinesL<\l@dminpagelines
    \else
      \l@dsamepagefalse
      \l@dpagefullfalse
    \fi
  \else
    \l@dsamepagefalse
    \l@dpagefulltrue
  \fi%
  \ifprint@last@after@pendL%
     \l@dpagefullfalse%
     \l@dsamepagefalse%
     \print@last@after@pendLfalse%
  \fi%
  }%

\newcommand*{\checkpageR}{%
  \l@dpagefulltrue
  \l@dsamepagetrue
  \check@goal
  \ifdim\pagetotal<\ledthegoal
    \ifnum\numpagelinesR<\l@dminpagelines
    \else
      \l@dsamepagefalse
      \l@dpagefullfalse
    \fi
  \else
    \l@dsamepagefalse
    \l@dpagefulltrue
  \fi%
  \ifprint@last@after@pendR%
     \l@dpagefullfalse%
     \l@dsamepagefalse%
     \print@last@after@pendRfalse%
  \fi%
  }%

```

\checkpbL
\checkpbR

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

```
\newcommand{\checkpbL}{
   \IfStrEq{\led@pb@setting}{after}{
```

```
   \xifinlistcs{\the\absline@num}{l@prev@pb}{\l@dpagefulltrue\l@dsamepage
   \xifinlistcs{\the\absline@num}{l@prev@nopb}{\l@dpagefullfalse\l@dsamep
   }{}
   \IfStrEq{\led@pb@setting}{before}{
     \numdef{\next@absline}{\the\absline@num+1}
   \xifinlistcs{\next@absline}{l@prev@pb}{\l@dpagefulltrue\l@dsamepagefal
   \xifinlistcs{\next@absline}{l@prev@nopb}{\l@dpagefullfalse\l@dsamepage
   }{}
}

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

\checkverseL
\checkverseR

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

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

\setgoalfraction
\ledthegoal
\goalfraction
\check@goal

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

`\ledthegoal` is calculated via `\check@goal`.

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

\newcommand*{\check@goal}{%
  \ledthegoal=\@goalfraction\pagegoal}
\newcommand{\setgoalfraction}[1]{%
  \xdef\@goalfraction{#1}%
}
```

\ifwrittenlinesL
\ifwrittenlinesL

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

```
\newif\ifwrittenlinesL
\newif\ifwrittenlinesR

```

XVIII.8 Getting boxes content

\get@nextboxL
\get@nextboxR

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

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

The current box is not empty; do nothing.

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

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

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

Sufficient lines have been output.

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

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

```
        \@writelinesinparL
        \writtenlinesLtrue
      \fi
      \ifnum\l@dnumpstartsL>\l@dpscL
```

There are still unprocessed boxes. Recalculate the maximum number of lines needed, and move onto the next box (by incrementing `\l@dpscL`). If needed, restart the line numbering.

```
        \writtenlinesLfalse
```

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

Check the number of lines

```
     \l@dcalc@maxoftwo{\the\usenamecount{l@dmaxlinesinpar\the\l@dpscL}}%
                         {\the\@donetotallinesL}%
                                                  {\usenamecount{l@dmaxlinesinpar\the\l@dpscL}}%
        \global\@donetotallinesL \z@
```

Go to the next pstart

```
        \global\advance\l@dpscL \@ne
        \global\pstartnumtrue%
        \restore@pstartL@pc%
```

Add notes of parallel ledgroup.

```
        \parledgroup@notes@endL
        \parledgroup@correction@notespacing@final{L}
      \else

      \fi
    \fi
  \fi}
\newcommand*{\get@nextboxR}{%
 \ifvbox\namebox{l@dRcolrawbox\the\l@dpscR}% box is not empty
  \else%                                      box is empty
   \ifnum\usenamecount{l@dmaxlinesinpar\the\l@dpscR}>\@donetotallinesR
      \parledgroup@notes@endR
    \else
    \ifnum\usenamecount{l@dmaxlinesinpar\the\l@dpscR}=\@donetotallinesR
        \parledgroup@notes@endR
      \fi
      \ifwrittenlinesR\else
        \@writelinesinparR
        \writtenlinesRtrue
      \fi
      \ifnum\l@dnumpstartsR>\l@dpscR
        \writtenlinesRfalse
        \ifbypstart@R
             \global\line@numR=0%
             \resetprevline@%
        \fi
        \csuse{after@pendR@\the\l@dpscR}%
        \global\csundef{after@pendR@\the\l@dpscR}%
```

```
     \l@dcalc@maxoftwo{\the\usenamecount{l@dmaxlinesinpar\the\l@dpscR}}%
                         {\the\@donetotallinesR}%
                                                   {\usename-
count{l@dmaxlinesinpar\the\l@dpscR}}%
        \global\@donetotallinesR \z@
        \global\advance\l@dpscR \@ne
        \global\pstartnumRtrue%
        \restore@pstartR@pc%
        \parledgroup@notes@endR
        \parledgroup@correction@notespacing@final{R}
      \else
          \print@last@after@pendRtrue%
      \fi
    \fi
  \fi}

```

XVIII.9 Same page number in both side

The sameparallelpagenumber allow to have the same page number for the left and the right side We can not do it by changing the value of the page counter, since its value is used to determine whether a page is left or right. Consequently, we have to do it by patching \thepage inside a \Pages macro.

eparallelpage@number

This macro is called at the beginning of \Pages. It patches the \thepage macro in order to and to use the value of sameparallelpage@number LaTeXcounter instead of those of page LaTeXcounter. As we are inside a group, the patch is local, and, consequently, the page printed after the \Pages will use the normal page number scheme.

The value of sameparallelpage@number is increase by 1 when we change from right page to left page.

```
\newcounter{sameparallelpage@number}
\newcommand{\init@sameparallelpage@number}{%
  \setcounter{sameparallelpage@number}{\c@page}%
  \ifsameparallelpagenumber%
    \patchcmd{\thepage}{page}{sameparallelpage@number}{}{}%
  \fi%
}%
```

eparallelpage@number

This macro is called at the end of \Pages. If the sameparallelpage@number is enabled, it set the page number to the last value of sameparallelpage@number counter, in order to have a continuity of page numbering between pages printed with \Pages and normal pages.

```
\newcommand{\finish@sameparallelpage@number}{%
  \ifsameparallelpagenumber%
    \setcounter{page}{\c@sameparallelpage@number}%
  \fi%
}%
```

```
%   \end{macrocode}
% \end{macro}
% \section{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 \verb+\vbox{}+ is inserted, and consequently parallel chapters are mis-aligned.
%
% So we patch the \cs{chapter} command in order to prevent this problem.
% \begin{macro}{\chapter}
%    \begin{macrocode}
\pretocmd{\chapter}{%
  \ifl@dprintingpages%
    \vbox{}%
  \fi%
  }%
  {}%
  {}%
```

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

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

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

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

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

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

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

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

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

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

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

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

\ledpbR
\ledpbnumR
\lednopbnum
\lednopbnumR

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

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

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

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

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

XX Parallel ledgroup

\parledgroup@
\parledgroupseries@
\parledgrouptype@

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

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

edgroup@notes@startL
edgroup@notes@startR

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

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

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

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

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

\insert@noterule@ledgroup

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

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

\parledgroupnotespacing

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

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

edgroup@notespacing@correction
oup@notespacing@set@correction

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

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

ion@notespacing@init

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

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

on@notespacing@final

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

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

rrection@notespacing

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

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

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

```
{}
```

```
\newcommand{\parledgroup@correction@notespacing}[1]{%
    \csuse{ifledgroupnotes#1@}%
      \vspace{-\parledgroup@notespacing@correction}%
    \dimdef{\parledgroup@notespacing@correction@accumulated}{\parledgroup@
    \dimdef{\parledgroup@notespacing@correction@modulo}{\parledgroup@notes
    \ifdimless{\parledgroup@notespacing@correction@modulo}{\baselineskip}
  \@ne%
    \dimdef{\parledgroup@notespacing@correction@modulo}{\parledgroup@notes
  \baselineskip}%
     }% mean greater than equal
    \fi%
}
```

\parledgroup@beforenotesL
\parledgroup@beforenotesR

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

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

\parledgroup@beforenotes@save

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

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

XXI Compatibility with eledmac

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

```
\ifeledmaccompat@%


  \unless\ifnocritical@
  \let\onlyXside\Xonlyside
  \fi
\fi
```

XXII The End

</code>

Appendix A Some things to do when changing version

Appendix A.1 Migration to eledpar 1.4.3

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

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

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

See the two following examples:

With standard `\hangingsymbol`:

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

With the modification of the hangingsymbol:

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

Appendix A.2 Migration from eledpar to reledpar

As for migration from eledmac to reledmac:

- Some commands and options have been deleted because they are deprecated.
- Some customization by `\renewcommand` have been replaced by commands.
- Some commands name have been changed in order to have a more logical and uniform.

Appendix A.2.1 Deprecated commands and options

Here, you will find a tabular of deprecated commands and their alternative. Notes that the use of these command can have been changed. Please read the handbook.

Deprecated command	*Replaced by*

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

Appendix A.2.2 `\renewcommand` replaced by command

Many use of `\renewcommand` has been replaced by use of specific commands. Please read handbook on specific command.