

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

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

Abstract

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

Please, for all bug's report, open a ticket on <https://github.com/maieul/ledmac/issues/>

Contents

1	Introduction	3
2	The ledpar package	4
2.1	General	4
3	Parallel columns	5
4	Facing pages	5
5	Left and right texts	6
6	Numbering text lines	8
7	Verse	9
8	Implementation overview	12
9	Preliminaries	12
9.1	Messages	13

*This file (**ledpar.dtx**) has version number v0.8, last revised 2011/09/16.

[†]herries dot press at earthlink dot net

[‡]maieul at maieul dot net

10 Sectioning commands	13
11 Line counting	16
11.1 Choosing the system of lineation	16
11.2 Line-number counters and lists	19
11.3 Reading the line-list file	19
11.4 Commands within the line-list file	20
11.5 Writing to the line-list file	28
12 Marking text for notes	31
13 Parallel environments	32
14 Paragraph decomposition and reassembly	34
14.1 Boxes, counters, <code>\pstart</code> and <code>\pend</code>	34
14.2 Processing one line	37
14.3 Line and page number computation	39
14.4 Line number printing	41
14.5 Add insertions to the vertical list	43
14.6 Penalties	44
14.7 Printing leftover notes	45
15 Footnotes	45
15.1 Outer-level footnote commands	45
15.2 Normal footnote formatting	48
16 Cross referencing	49
17 Side notes	50
18 Familiar footnotes	52
19 Verse	53
20 Naming macros	54
21 Counts and boxes for parallel texts	54
22 Fixing babel	56
23 Parallel columns	58
24 Parallel pages	61
25 The End	68

<i>List of Figures</i>	3
------------------------	---

A Examples	69
A.1 Parallel column example	78
A.2 Example parallel facing pages	80
A.3 Example poetry on parallel facing pages	86
References	91
Index	91
Change History	100

List of Figures

1	Output from <code>villon.tex</code>	71
2	Left page output from <code>djd17nov.tex</code>	72
3	Right page output from <code>djd17nov.tex</code>	73
4	First left page output from <code>djdpoems.tex</code>	74
5	First right page output from <code>djdpoems.tex</code>	75
6	Second left page output from <code>djdpoems.tex</code>	76
7	Second right page output from <code>djdpoems.tex</code>	77

1 Introduction

The `EDMAC` macros [LW90] for typesetting critical editions of texts have been available for use with TeX for some years. Since `EDMAC` became available there had been a small but constant demand for a version of `EDMAC` that could be used with LaTeX. The `ledmac` package was introduced in 2003 in an attempt to satisfy that request.

Some critical editions contain texts in more than one form, such as a set of verses in one language and their translations in another. In such cases there is a desire to be able to typeset the two texts, together with any critical apparatus, in parallel. The `ledpar` package is an extension to `ledmac` that enables two texts and their apparatus to be set in parallel, either in two columns or on pairs of facing pages.

The package has to try and coerce TeX into paths it was not designed for. Use of the package, therefore, may produce some surprising results.

This manual contains a general description of how to use `ledpar` starting in section 2; the complete source code for the package, with extensive documentation (in sections 8 through 25); and an Index to the source code. As `ledpar` is an adjunct to `ledmac` I assume that you have read the `ledmac` manual. Also `ledpar` requires `ledmac` to be used, preferably at least version 0.10 (2011/08/22). You do not need to read the source code for this package in order to use it but doing so may help to answer any questions you might have. On a first reading, I suggest that you should skip anything after the general documentation in sections 2 until 8, unless you are particularly interested in the innards of `ledpar`.

2 The ledpar package

A file may mix *numbered* and *unnumbered* text. Numbered text is printed with marginal line numbers and can include footnotes and endnotes that are referenced to those line numbers: this is how you'll want to print the text that you're editing. Unnumbered text is not printed with line numbers, and you can't use `ledmac`'s note commands with it: this is appropriate for introductions and other material added by the editor around the edited text.

The `ledpar` package lets you typeset two *numbered* texts in parallel. This can be done either as setting the 'Leftside' and 'Rightside' texts in two columns or on facing pages. In the paired pages case footnotes are placed at the bottom of the page on which they are called out — that is, footnotes belonging to the left are set at the foot of a left (even numbered) page, and those for right texts are at the bottom of the relevant right (odd numbered) page. However, in the columnar case, all footnotes are set at the bottom left of the page on which they are called out — they are not set below the relevant column. The line numbering schemes need not be the same for the two texts.

2.1 General

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

`ledpar` 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{10}
```

meaning that there can be up to 10 chunks in the left text and up to 10 chunks in the right text, requiring a total of 20 boxes. If you need more chunks then you can increase `\maxchunks`.

TeX has a limited number of boxes; if you get an error message along the lines of 'no room for a new box', then decrease the number. A chunk also requires a counter so you may get a message along the lines 'no room for a new count', which may be resolved by reducing `\maxchunks`.

On the other hand, if you get a `ledmac` error message along the lines: 'Too

many `\pstart` without printing. Some text will be lost.’ then you will have to either increase `\maxchunks` or use the parallel printing commands (`\Columns` or `\Pages`) more frequently.

When typesetting verse using `\syntax`, each line is treated as a chunk, so be warned that if you are setting parallel verses you might have to increase `\maxchunks` much more than it appears at first sight.

In general, `ledmac` is a TeX resource hog, and `ledpar` only makes things worse in this respect.

3 Parallel columns

`pairs` Numbered text that is to be set in columns must be within a `pairs` environment. Within the environment the text for the lefthand and righthand columns is placed within the `Leftside` and `Rightside` environments, respectively; these are described in more detail below in section 5.

`\Columns` The command `\Columns` typesets the texts in the previous pair of `Leftside` and `Rightside` environments. The general scheme for parallel columns looks like this:

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

There is no required pagebreak before or after the columns.

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

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

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

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

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

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

4 Facing pages

`pages` Numbered text that is to be set on facing pages must be within a `pages` environ-

ment. Within the environment the text for the lefthand and righthand pages is placed within the `Leftside` and `Rightside` environments, respectively.

`\Pages` The command `\Pages` typesets the texts in the previous pair of `Leftside` and `Rightside` environments. The general scheme for parallel pages looks like this:

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

The `Leftside` text is set on lefthand (even numbered) pages and the `Rightside` text is set on righthand (odd numbered) pages. Each `\Pages` command starts a new even numbered page. After parallel typesetting is finished, a new page is started.

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

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

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

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

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

5 Left and right texts

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

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

`Rightside` Within these environments you can designate the line numbering scheme(s) to be used. The `ledmac` package originally used counters for specifying the numbering scheme; now both `ledmac`¹ and the `ledpar` package use macros instead. Following `\firstlinenum{<num>}` the first line number will be `<num>`, and following `\linenumincrement{<num>}` only every `<num>`th line will have a printed

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

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.

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

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

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

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

If you are using the `babel` package with different languages (via, say, `\selectlanguage`) for the left and right texts it is particularly important to select the appropriate language within the `Leftside` and `Rightside` environments. The initial language selected for the right text is the `babel` package's default. Also, it is the *last* `\selectlanguage` in a side that controls the language used in any notes for that side when they get printed. If you are using multilingual notes then it is probably safest to explicitly specify the language(s) for each note rather than relying on the language selection for the side. The right side language is also applied to the right side line numbers.

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

6 Numbering text lines

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

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

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

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

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

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

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

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

This may be useful for parallel columns but for parallel pages it might be more appropriate to redefine it as:

```
\renewcommand*\Rlineflag}{}
```

`\printlinesR` The `\printlines` macro is ordinarily used to print the line number referred to by `\ledsavedprintlines`

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

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

7 Verse

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

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

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

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

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

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

```

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

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

\end{astanza}
...

```

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

```

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

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

\end{astanza}
...

```

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

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

8 Implementation overview

TeX is designed to process a single stream of text, which may include footnotes, tables, and so on. It just keeps converting its input into a stream typeset pages. It was not designed for typesetting two texts in parallel, where it has to alternate from one to the other. Further, TeX essentially processes its input one paragraph at a time — it is very difficult to get at the ‘internals’ of a paragraph such as the individual lines in case you want to number them or put some mark at the start or end of the lines.

`ledmac` 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 `ledmac` code is concerned with handling this box and its contents.

`ledpar`’s solution to the problem of parallel texts is to put the two texts into separate boxes, and then appropriately extract the pairs of lines from the boxes. This involves duplicating much of the original box code for an extra right text box. The other, smaller, part of the code is concerned with coordinating the line extractions from the boxes.

The package code is presented in roughly in the same order as in `ledmac`.

9 Preliminaries

Announce the name and version of the package, which is targetted for LaTeX2e. The package also requires the `ledmac` package, preferably at least version 0.10 (2011/08/22).

```

1 (*code)
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{ledpar}[2011/09/16 v0.8 ledmac extension for parallel texts]
4

```

With the option ‘`shiftedverses`’ a long verse on the left side (or in the right side) don’t make a blank on the corresponding verse, but the blank is put on the bottom of the page. Consequently, the verses on the parallel pages are shifted, but the shifted stop at every end of pages.

```

5 \newif\ifshiftedverses
6 \shiftedversesfalse
7 \DeclareOption{shiftedverses}{\shiftedversestrue}
8 \ProcessOptions

```

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

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

```

9 \l@dpairingfalse
10 \newif\ifl@dpaging
11 \l@dpagingfalse
12 \newif\ifledRcol
13 \ledRcolfalse

```

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

```

\l@Rcolwidth 14 \newdimen\Lcolwidth
15 \Lcolwidth=0.45\textwidth
16 \newdimen\Rcolwidth
17 \Rcolwidth=0.45\textwidth
18

```

9.1 Messages

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

`\led@err@TooManyPstarts`

```

19 \newcommand*{\led@err@TooManyPstarts}{%
20 \ledmac@error{Too many \string\pstart space without printing.
21 \hspace{1em} Some text will be lost}{\@ehc}}

```

`\led@err@BadLeftRightPstarts`

```

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

```

`\led@err@LeftOnRightPage`

`\led@err@RightOnLeftPage`

```

25 \newcommand*{\led@err@LeftOnRightPage}{%
26 \ledmac@error{The left page has ended on a right page}{\@ehc}}
27 \newcommand*{\led@err@RightOnLeftPage}{%
28 \ledmac@error{The right page has ended on a left page}{\@ehc}}

```

10 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.

```

29 \newcount\section@numR
30 \section@numR=\z@

```

`\ifnumberingR` The `\ifnumberingR` flag is set to `true` if we're within a right text numbered section.

```

31 \newif\ifnumberingR

```

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

```

32 \pst@rtedLfalse
33 \newif\ifpst@rtedR
34 \pst@rtedRfalse
35

```

`\beginnumbering` For parallel processing the original `\beginnumbering` is extended to zero `\l@dnumpstartsL` — the number of chunks to be processed. It also sets `\ifpst@rtedL` to `FALSE`.

```

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

```

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

```

49 \newcommand*\beginnumberingR}{%
50   \ifnumberingR
51     \led@err@NumberingStarted
52   \endnumberingR
53   \fi
54   \global\l@dnumpstartsR \z@
55   \global\pst@rtedRfalse
56   \global\numberingRtrue
57   \global\advance\section@numR \@ne
58   \global\absline@numR \z@
59   \global\line@numR \z@
60   \global\@lockR \z@
61   \global\sub@lockR \z@
62   \global\sublines@false
63   \global\let\next@page@numR\relax
64   \global\let\sub@change\relax
65   \message{Section \the\section@numR R }%
66   \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
67   \l@dend@stuff}
68

```

`\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 `ledmac`.

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

```

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

```

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

```

\resumenumberingR 90 \newcommand*\pausenumberingR}{%
91   \endnumberingR\global\numberingRtrue}
92 \newcommand*\resumenumberingR}{%
93   \ifnumberingR
94     \global\pst@rtedRtrue
95     \global\advance\section@numR \@ne
96     \led@mess@SectionContinued{\the\section@numR R}%
97     \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
98     \l@dend@stuff
99   \else
100    \led@err@numberingShouldHaveStarted
101    \endnumberingR
102    \beginnumberingR
103  \fi}
104

```

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

```

105 \newcommand*\memorydumpL}{%

```

```

106 \endnumbering
107 \numberingtrue
108 \global\pst@rtedLtrue
109 \global\advance\section@num \@ne
110 \led@mess@SectionContinued{\the\section@num}%
111 \line@list@stuff{\jobname.\extensionchars\the\section@num}%
112 \l@dend@stuff}
113 \newcommand*\memorydumpR}{%
114 \endnumberingR
115 \numberingRtrue
116 \global\pst@rtedRtrue
117 \global\advance\section@numR \@ne
118 \led@mess@SectionContinued{\the\section@numR R}%
119 \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
120 \l@dend@stuff}
121

```

11 Line counting

11.1 Choosing the system of lineation

Sometimes you want line numbers that start at 1 at the top of each page; other times you want line numbers that start at 1 at the start of each section and increase regardless of page breaks. `ledpar` lets you choose different schemes for the left and right texts.

`\ifbypage@R` The `\ifbypage@R` flag specifies the current lineation system for right texts: `false` for line-of-section, `true` for line-of-page. `ledpar` will use the line-of-section system unless instructed otherwise.

```

122 \newif\ifbypage@R
123 \bypage@Rfalse

```

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

```

124 \newcommand*\lineationR}[1]{%
125 \ifnumberingR
126 \led@err@LineationInNumbered
127 \else
128 \def\@tempa{#1}\def\@tempb{page}%
129 \ifx\@tempa\@tempb
130 \global\bypage@Rtrue
131 \else
132 \def\@tempb{section}%
133 \ifx\@tempa\@tempb
134 \global\bypage@Rfalse
135 \else
136 \led@warn@BadLineation
137 \fi

```



```

138   \fi
139 \fi}}
140

```

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

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

```

141 \newcount\line@marginR
142 \renewcommand*{\linenummargin}[1]{%
143   \l@getline@margin{#1}%
144   \ifnum\@l@tempcntb>\m@ne
145     \ifledRcol
146       \global\line@marginR=\@l@tempcntb
147     \else
148       \global\line@margin=\@l@tempcntb
149     \fi
150 \fi}}

```

By default put right text numbers at the right.

```

151 \line@marginR=\@ne
152

```

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

```

153 \newcounter{firstlinenumR}
154 \setcounter{firstlinenumR}{5}
155 \newcounter{linenumincrementR}
156 \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.

```

157 \newcounter{firstsublinenumR}
158 \setcounter{firstsublinenumR}{5}
159 \newcounter{sublinenumincrementR}
160 \setcounter{sublinenumincrementR}{5}
161

```

`\firstlinenum` `\linenumincrement` `\firstsublinenum` `\sublinenumincrement` These are the user's macros for changing (sub) line numbers. They are defined in `ledmac v0.7`, but just in case I have started by `\provideing` them.

```

162 \providecommand*{\firstlinenum}{-}

```

```

163 \providecommand*\linenumincrement{}
164 \providecommand*\firstsublinenum{}
165 \providecommand*\sublinenumincrement{}
166 \renewcommand*\firstlinenum[1]{%
167   \ifledRcol \setcounter{firstlinenumR}{#1}%
168   \else      \setcounter{firstlinenum}{#1}%
169   \fi}
170 \renewcommand*\linenumincrement[1]{%
171   \ifledRcol \setcounter{linenumincrementR}{#1}%
172   \else      \setcounter{linenumincrement}{#1}%
173   \fi}
174 \renewcommand*\firstsublinenum[1]{%
175   \ifledRcol \setcounter{firstsublinenumR}{#1}%
176   \else      \setcounter{firstsublinenum}{#1}%
177   \fi}
178 \renewcommand*\sublinenumincrement[1]{%
179   \ifledRcol \setcounter{sublinenumincrementR}{#1}%
180   \else      \setcounter{sublinenumincrement}{#1}%
181   \fi}
182

```

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

```

183 \newcommand*\Rlineflag{R}
184

```

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

```

185 \newcommand*\linenumrepR[1]{\@arabic{#1}}
186 \newcommand*\sublinenumrepR[1]{\@arabic{#1}}
187

```

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

```

188 \newcommand*\leftlinenumR{%
189   \l@dlinenumR
190   \kern\linenumsep}
191 \newcommand*\rightlinenumR{%
192   \kern\linenumsep
193   \l@dlinenumR}
194 \newcommand*\l@dlinenumR{%
195   \numlabfont\linenumrepR{\line@numR}\Rlineflag%
196   \ifsublines@
197     \ifnum\subline@num>\z@
198       \unskip\fullstop\sublinenumrepR{\subline@numR}%
199     \fi
200   \fi}
201

```

11.2 Line-number counters and lists

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

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

```
202 \newcount\line@numR
203 \newcount\subline@numR
204 \newcount\absline@numR
205
```

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

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

```
206 \list@create{\line@listR}
207 \list@create{\insertlines@listR}
208 \list@create{\actionlines@listR}
209 \list@create{\actions@listR}
210
```

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

```
211 \list@create{\linesinpar@listL}
212 \list@create{\linesinpar@listR}
213 \list@create{\maxlinesinpar@list}
214
```

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

```
215 \newcount\page@numR
216
```

11.3 Reading the line-list file

`\read@linelist` `\read@linelist{<file>}` is the control sequence that's called by `\beginnumbering` (via `\line@list@stuff`) to open and process a line-list file; its argument is the name of the file.

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

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

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

```

221 \list@clear{\actionlines@listR}%
222 \list@clear{\actions@listR}%
223 \list@clear{\linesinpar@listR}%
224 \list@clear{\linesonpage@listR}
225 \else
226 \list@clearing@reg
227 \list@clear{\linesinpar@listL}%
228 \list@clear{\linesonpage@listL}%
229 \fi

```

Make sure that the `\maxlinesinpar@list` is empty (otherwise things will be thrown out of kilter if there is any old stuff still hanging in there).

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

Now get the file and interpret it.

```

231 \get@linelistfile{#1}%
232 \endgroup

```

When the reading is done, we're all through with the line-list file. All the information we needed from it will now be encoded in our list macros. Finally, we initialize the `\next@actionline` and `\next@action` macros, which specify where and what the next action to be taken is.

```

233 \ifledRcol
234 \global\page@numR=\m@ne
235 \ifx\actionlines@listR\empty
236 \gdef\next@actionlineR{1000000}%
237 \else
238 \gl@p\actionlines@listR\to\next@actionlineR
239 \gl@p\actions@listR\to\next@actionR
240 \fi
241 \else
242 \global\page@num=\m@ne
243 \ifx\actionlines@list\empty
244 \gdef\next@actionline{1000000}%
245 \else
246 \gl@p\actionlines@list\to\next@actionline
247 \gl@p\actions@list\to\next@action
248 \fi
249 \fi}
250

```

This version of `\read@linelist` creates list macros containing data for the entire section, so they could get rather large. The `\memorydump` macro is available if you run into macro memory limitations.

11.4 Commands within the line-list file

This section defines the commands that can appear within a line-list file, except for `\@lab` which is in a later section among the cross-referencing commands it is associated with.

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

```

\@l@regR \@l does everything related to the start of a new line of numbered text. Exactly
\@l what it does depends on whether right text is being processed.
251 \newcommand{\@l@regR}{%
252   \ifx\l@dchset@num\relax \else
253     \advance\absline@numR \@ne
254     \set@line@action
255     \let\l@dchset@num\relax
256     \advance\absline@numR \m@ne
257     \advance\line@numR \m@ne%   % do we need this?
258   \fi
259   \advance\absline@numR \@ne
260   \ifx\next@page@numR\relax \else
261     \page@action
262     \let\next@page@numR\relax
263   \fi
264   \ifx\sub@change\relax \else
265     \ifnum\sub@change>\z@
266       \sublines@true
267     \else
268       \sublines@false
269     \fi
270     \sub@action
271     \let\sub@change\relax
272   \fi
273   \ifcase\@lockR
274   \or
275     \@lockR \tw@
276   \or\or
277     \@lockR \z@
278   \fi
279   \ifcase\sub@lockR
280   \or
281     \sub@lockR \tw@
282   \or\or
283     \sub@lockR \z@
284   \fi
285   \ifsublines@
286     \ifnum\sub@lockR<\tw@
287       \advance\subline@numR \@ne
288     \fi
289   \else
290     \ifnum\@lockR<\tw@
291       \advance\line@numR \@ne \subline@numR \z@
292     \fi
293   \fi}
294
295 \renewcommand*{\@l}[2]{%

```

```

296 \fix@page{#1}%
297 \ifledRcol
298 \@l@regR
299 \else
300 \@l@reg
301 \fi}
302

```

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

```

\fix@page
303 \newcount\last@page@numR
304 \last@page@numR=-10000
305 \renewcommand*{\fix@page}[1]{%
306 \ifledRcol
307 \ifnum #1=\last@page@numR
308 \else
309 \ifbypage@R
310 \line@numR \z@ \subline@numR \z@
311 \fi
312 \page@numR=#1\relax
313 \last@page@numR=#1\relax
314 \def\next@page@numR{#1}%
315 \fi
316 \else
317 \ifnum #1=\last@page@num
318 \else
319 \ifbypage@
320 \line@num \z@ \subline@num \z@
321 \fi
322 \page@num=#1\relax
323 \last@page@num=#1\relax
324 \def\next@page@num{#1}%
325 \fi
326 \fi}
327

```

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

```

328 \renewcommand*{\@adv}[1]{%
329 \ifsublines@
330 \ifledRcol
331 \advance\subline@numR by #1\relax
332 \ifnum\subline@numR<\z@
333 \led@warn@BadAdvancelineSubline
334 \subline@numR \z@
335 \fi
336 \else
337 \advance\subline@num by #1\relax
338 \ifnum\subline@num<\z@
339 \led@warn@BadAdvancelineSubline

```

```

340     \subline@num \z@
341     \fi
342 \fi
343 \else
344 \ifledRcol
345     \advance\line@numR by #1\relax
346     \ifnum\line@numR<\z@
347         \led@warn@BadAdvancelineLine
348         \line@numR \z@
349     \fi
350 \else
351     \advance\line@num by #1\relax
352     \ifnum\line@num<\z@
353         \led@warn@BadAdvancelineLine
354         \line@num \z@
355     \fi
356 \fi
357 \fi
358 \set@line@action}
359

```

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

```

360 \renewcommand*{\@set}[1]{%
361 \ifledRcol
362     \ifsublines@
363         \subline@numR=#1\relax
364     \else
365         \line@numR=#1\relax
366     \fi
367     \set@line@action
368 \else
369     \ifsublines@
370         \subline@num=#1\relax
371     \else
372         \line@num=#1\relax
373     \fi
374     \set@line@action
375 \fi}
376

```

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

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

```

377 \renewcommand*{\l@d@set}[1]{%
378 \ifledRcol
379     \line@numR=#1\relax
380     \advance\line@numR \@ne

```

```

381   \def\l@dchset@num{#1}
382   \else
383     \line@num=#1\relax
384     \advance\line@num \@ne
385     \def\l@dchset@num{#1}
386   \fi}
387 \let\l@dchset@num\relax
388

```

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

```

389 \renewcommand*{\page@action}{%
390   \ifledRcol
391     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
392     \xright@appenditem{\next@page@numR}\to\actions@listR
393   \else
394     \xright@appenditem{\the\absline@num}\to\actionlines@list
395     \xright@appenditem{\next@page@num}\to\actions@list
396   \fi}

```

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

```

397 \renewcommand*{\set@line@action}{%
398   \ifledRcol
399     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
400     \ifsublines@
401       \@l@dttempcnta=-\subline@numR
402     \else
403       \@l@dttempcnta=-\line@numR
404     \fi
405     \advance\@l@dttempcnta by -5000\relax
406     \xright@appenditem{\the\@l@dttempcnta}\to\actions@listR
407   \else
408     \xright@appenditem{\the\absline@num}\to\actionlines@list
409     \ifsublines@
410       \@l@dttempcnta=-\subline@num
411     \else
412       \@l@dttempcnta=-\line@num
413     \fi
414     \advance\@l@dttempcnta by -5000\relax
415     \xright@appenditem{\the\@l@dttempcnta}\to\actions@list
416   \fi}
417

```

`\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.

```

418 \renewcommand*{\sub@action}{%
419   \ifledRcol
420     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
421     \ifsublines@

```



```

422     \xright@appenditem{-1001}\to\actions@listR
423     \else
424     \xright@appenditem{-1002}\to\actions@listR
425     \fi
426 \else
427     \xright@appenditem{\the\absline@num}\to\actionlines@list
428     \ifsublines@
429     \xright@appenditem{-1001}\to\actions@list
430     \else
431     \xright@appenditem{-1002}\to\actions@list
432     \fi
433 \fi}
434

```

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

```

435 \newcount\@lockR
436 \newcount\sub@lockR
437
438 \newcommand*{\do@lockonR}{%
439     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
440     \ifsublines@
441     \xright@appenditem{-1005}\to\actions@listR
442     \ifnum\sub@lockR=\z@
443     \sub@lockR \@ne
444     \else
445     \ifnum\sub@lockR=\thr@@
446     \sub@lockR \@ne
447     \fi
448     \fi
449     \else
450     \xright@appenditem{-1003}\to\actions@listR
451     \ifnum\@lockR=\z@
452     \@lockR \@ne
453     \else
454     \ifnum\@lockR=\thr@@
455     \@lockR \@ne
456     \fi
457     \fi
458 \fi}
459
460 \renewcommand*{\do@lockon}{%
461     \ifx\next\lock@off
462     \global\let\lock@off=\skip@lockoff
463     \else
464     \ifledRcol
465     \do@lockonR
466     \else
467     \do@lockonL

```

```

468   \fi
469   \fi}

\lock@off \lock@off adds an entry to the action-code list to turn line number locking off.
\do@lockoff 470
\do@lockoffR 471
\skip@lockoff 472 \newcommand{\do@lockoffR}{%
473   \xright@appenditem{\the\absline@numR}\to\actionlines@listR
474   \ifsublines@
475     \xright@appenditem{-1006}\to\actions@listR
476     \ifnum\sub@lockR=\tw@
477       \sub@lockR \thr@@
478     \else
479       \sub@lockR \z@
480     \fi
481   \else
482     \xright@appenditem{-1004}\to\actions@listR
483     \ifnum\@lockR=\tw@
484       \@lockR \thr@@
485     \else
486       \@lockR \z@
487     \fi
488   \fi}
489
490 \renewcommand*{\do@lockoff}{%
491   \ifledRcol
492     \do@lockoffR
493   \else
494     \do@lockoffL
495   \fi}
496 \global\let\lock@off=\do@lockoff
497

```

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

```

498 \providecommand*\n@num{}
499 \renewcommand*\n@num{%
500   \ifledRcol
501     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
502     \xright@appenditem{-1007}\to\actions@listR
503   \else
504     \n@num@reg
505   \fi}
506

```

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

- #1, the number of entries to add to `\insertlines@list` for this reference. This value for right text, here and within `\edtext`, which computes it and

writes it to the line-list file, will be stored in the count `\insert@countR`.

```
507 \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.

```
508 \renewcommand*\@ref}[2]{%
509 \ifledRcol
510 \global\insert@countR=#1\relax
511 \loop\ifnum\insert@countR>\z@
512 \xright@appenditem{\the\absline@numR}\to\insertlines@listR
513 \global\advance\insert@countR \m@ne
514 \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.

```
515 \begingroup
516 \let\@ref=\dummy@ref
517 \let\page@action=\relax
518 \let\sub@action=\relax
519 \let\set@line@action=\relax
520 \let\@lab=\relax
521 #2
522 \global\endpage@num=\page@numR
523 \global\endline@num=\line@numR
524 \global\endsubline@num=\subline@numR
525 \endgroup
```

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

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

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

```
531 #2
532 \else
```

And when not in right text

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

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

```
535 \providecommand*\@pend}[1]{%
536 \renewcommand*\@pend}[1]{%
537 \xright@appenditem{#1}\to\linesinpar@listL}
538 \providecommand*\@pendR}[1]{%
539 \renewcommand*\@pendR}[1]{%
540 \xright@appenditem{#1}\to\linesinpar@listR}
541
```

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

```
542 \providecommand*\@lopL}[1]{%
543 \renewcommand*\@lopL}[1]{%
544 \xright@appenditem{#1}\to\linesonpage@listL}
545 \providecommand*\@lopR}[1]{%
546 \renewcommand*\@lopR}[1]{%
547 \xright@appenditem{#1}\to\linesonpage@listR}
548
```

11.5 Writing to the line-list file

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

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

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

```
\first@linenum@out@Rfalse 550 \newif\iffirst@linenum@out@R
551 \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.

```
552 \newcommand*\line@list@stuffR}[1]{%
553 \read@linelist{#1}%
554 \iffirst@linenum@out@R
555 \immediate\closeout\linenum@outR
556 \global\first@linenum@out@Rfalse
557 \immediate\openout\linenum@outR=#1
558 \else
559 \closeout\linenum@outR
560 \openout\linenum@outR=#1
```

```
561 \fi}
562
```

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

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

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

```
565 \renewcommand*{\flag@start}{%
566 \ifledRcol
567 \edef\next{\write\linenum@outR{%
568 \string\@ref[\the\insert@countR] []}%
569 \next
570 \else
571 \edef\next{\write\linenum@out{%
572 \string\@ref[\the\insert@count] []}%
573 \next
574 \fi}
575 \renewcommand*{\flag@end}{%
576 \ifledRcol
577 \write\linenum@outR{]}%
578 \else
579 \write\linenum@out{]}%
580 \fi}
```

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

```
581 \renewcommand*{\startsub}{\dimen0\lastskip
582 \ifdim\dimen0>0pt \unskip \fi
583 \ifledRcol \write\linenum@outR{\string\sub@on}%
584 \else \write\linenum@out{\string\sub@on}%
585 \fi
586 \ifdim\dimen0>0pt \hskip\dimen0 \fi}
587 \def\endsub{\dimen0\lastskip
588 \ifdim\dimen0>0pt \unskip \fi
589 \ifledRcol \write\linenum@outR{\string\sub@off}%
590 \else \write\linenum@out{\string\sub@off}%
591 \fi
592 \ifdim\dimen0>0pt \hskip\dimen0 \fi}
593
```

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

```
594 \renewcommand*{\advanceline}[1]{%
595 \ifledRcol \write\linenum@outR{\string\@adv[#1]}%
596 \else \write\linenum@out{\string\@adv[#1]}%
597 \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.

```
598 \renewcommand*{\setline}[1]{%
599   \ifnum#1<\z@
600     \led@warn@BadSetline
601   \else
602     \ifledRcol \write\linenum@outR{\string\@set[#1]}%
603     \else      \write\linenum@out{\string\@set[#1]}%
604     \fi
605   \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.

```
606 \renewcommand*{\setlinenum}[1]{%
607   \ifnum#1<\z@
608     \led@warn@BadSetlinenum
609   \else
610     \ifledRcol \write\linenum@outR{\string\l@d@set[#1]}
611     \else      \write\linenum@out{\string\l@d@set[#1]} \fi
612   \fi}
613
```

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

`\endlock`

```
614 \renewcommand*{\startlock}{%
615   \ifledRcol \write\linenum@outR{\string\lock@on}%
616   \else      \write\linenum@out{\string\lock@on}%
617   \fi}
618 \def\endlock{%
619   \ifledRcol \write\linenum@outR{\string\lock@off}%
620   \else      \write\linenum@out{\string\lock@off}%
621   \fi}
622
```

`\skipnumbering` In numbered text, `\skipnumbering` in a line will suspend the numbering for that particular line. That is, line numbers are unchanged and no line number will be printed.

```
623 \renewcommand*{\skipnumbering}{%
624   \ifledRcol \write\linenum@outR{\string\n@num}%
625             \advanceline{-1}%
626   \else
627     \skipnumbering@reg
628   \fi}
629
```

12 Marking text for notes

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

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

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

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

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

`\critext` Now we begin `\critext` itself.

We slightly modify the original to make accomodation for when right text is being processed.

```
630 \long\def\critext#1#2/{\leavevmode
631   \begingroup
632     \no@expands
633     \xdef\@tag{#1}%
634     \set@line
635     \ifledRcol \global\insert@countR \z@
636     \else      \global\insert@count \z@ \fi
637     \ignorespaces #2\relax
638     \flag@start
639   \endgroup
640   \showlemma{#1}%
641   \ifx@end@lemmas\empty \else
642     \glp@end@lemmas\to\x@lemma
643     \x@lemma
644     \global\let\x@lemma=\relax
645   \fi
646   \flag@end}
```

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

```
647 \renewcommand{\edtext}[2]{\leavevmode
648   \begingroup
649     \no@expands
650     \xdef\@tag{#1}%
651     \set@line
652     \ifledRcol \global\insert@countR \z@
653     \else      \global\insert@count \z@ \fi
654     \ignorespaces #2\relax
655     \flag@start
656   \endgroup
657   \showlemma{#1}%
```

```

658 \ifx\end@lemmas\empty \else
659   \gl@p\end@lemmas\to\x@lemma
660   \x@lemma
661   \global\let\x@lemma=\relax
662 \fi
663 \flag@end}
664

```

`\set@line` The `\set@line` macro is called by `\edtext` to put the line-reference field and font specifier for the current block of text into `\l@d@nums`.

```

665 \renewcommand*{\set@line}{%
666   \ifledRcol
667   \ifx\line@listR\empty
668     \global\noteschanged@true
669     \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
670   \else
671     \gl@p\line@listR\to\@tempb
672     \xdef\l@d@nums{\@tempb|\edfont@info}%
673     \global\let\@tempb=\undefined
674   \fi
675 \else
676   \ifx\line@list\empty
677     \global\noteschanged@true
678     \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
679   \else
680     \gl@p\line@list\to\@tempb
681     \xdef\l@d@nums{\@tempb|\edfont@info}%
682     \global\let\@tempb=\undefined
683   \fi
684 \fi}
685

```

13 Parallel environments

The initial set up for parallel processing is deceptively simple.

```

pairs The pairs environment is for parallel columns and the pages environment for
pages parallel pages.
chapterinpages 686 \newenvironment{pairs}{%}
687   \l@dpairingtrue
688   \l@dpagingfalse
689 }{%
690   \l@dpairingfalse
691 }

```

The `pages` environment additionally sets the ‘column’ widths to the `\textwidth` (as known at the time the package is called). As in this environment there is two text in parallels in 2 pages, chapter have not to start in a left page. So the `\chapter` command is redefined to make no test about clearing page.


```

692 \newenvironment{pages}{%
693   \let\oldchapter\chapter
694   \let\chapter\chapterinpages
695   \l@dpairingtrue
696   \l@dpaddingtrue
697   \setlength{\Lcolwidth}{\textwidth}%
698   \setlength{\Rcolwidth}{\textwidth}%
699 }{%
700   \l@dpairingfalse
701   \l@dpaddingfalse
702   \let\chapter\oldchapter
703 }
704 \newcommand{\chapterinpages}{\thispagestyle{plain}%
705                               \global\@topnum\z@
706                               \@afterindentfalse
707                               \secdef\@chapter\@schapter}
708

```

Leftside Within the `pages` 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.

```

709 \newenvironment{Leftside}{%
710   \ledRcolfalse
711   \let\pstart\pstartL
712   \let\pend\pendL
713   \let\memorydump\memorydumpL
714   \Leftsidehook
715 }{\Leftsidehookend}

```

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

```

\Rightsidehook 716 \newcommand*\Leftsidehook{}
\Rightsidehookend 717 \newcommand*\Leftsidehookend{}
718 \newcommand*\Rightsidehook{}
719 \newcommand*\Rightsidehookend{}
720

```

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.

```

721 \newenvironment{Rightside}{%
722   \ledRcoltrue
723   \let\beginnumbering\beginnumberingR
724   \let\endnumbering\endnumberingR
725   \let\pausenumbering\pausenumberingR
726   \let\resumenumbering\resumenumberingR
727   \let\memorydump\memorydumpR
728   \let\pstart\pstartR

```

```

729 \let\pend\pendR
730 \let\lineation\lineationR
731 \Rightsidehook
732 }{
733 \ledRcolfalse
734 \Rightsidehookend
735 }
736

```

14 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.

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

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

`\one@lineR` 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.

```

737 \newcount\num@linesR
738 \newbox\one@lineR
739 \newcount\par@lineR

```

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

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

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

```

740 \newcommand*{\pstartL}{
741 \if@nobraek
742 \let\@oldnobraek\@nobraektrue
743 \else

```

```

744 \let\@oldnbreak\@nbreakfalse
745 \fi
746 \@nbreaktrue
747 \ifnumbering \else
748   \led@err@PstartNotNumbered
749   \beginnumbering
750 \fi
751 \ifnumberedpar@
752   \led@err@PstartInPstart
753   \pend
754 \fi

```

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

```

755 \ifpstart@rtedL\else
756   \list@clear{\inserts@list}%
757   \global\let\next@insert=\empty
758   \global\pstart@rtedLtrue
759 \fi
760 \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.

```

761 \global\advance\l@dnumpstartsL \@ne
762 \ifnum\l@dnumpstartsL>\l@dc@maxchunks
763   \led@err@TooManyPstarts
764   \global\l@dnumpstartsL=\l@dc@maxchunks
765 \fi
766 \global\setnamebox{\l@dLcolrawbox\the\l@dnumpstartsL}=\vbox\bgroup%
767   \hsize=\Lcolwidth
768 \numberedpar@true}

769 \newcommand*{\pstartR}{
770 \ifnbreak
771 \let\@oldnbreak\@nbreaktrue
772 \else
773 \let\@oldnbreak\@nbreakfalse
774 \fi
775 \@nbreaktrue
776 \ifnumberingR \else
777   \led@err@PstartNotNumbered
778   \beginnumberingR
779 \fi
780 \ifnumberedpar@
781   \led@err@PstartInPstart
782   \pendR
783 \fi
784 \ifpstart@rtedR\else
785   \list@clear{\inserts@listR}%
786   \global\let\next@insertR=\empty
787   \global\pstart@rtedRtrue

```

```

788 \fi
789 \begingroup\normal@pars
790 \global\advance\l@dnumstartsR \@ne
791 \ifnum\l@dnumstartsR>\l@dc@maxchunks
792   \led@err@TooManyPstarts
793   \global\l@dnumstartsR=\l@dc@maxchunks
794 \fi
795 \global\setnamebox{\l@Rcolrawbox\the\l@dnumstartsR}=\vbox\bgroup%
796   \hsize=\Rcolwidth
797 \numberedpar@true}

```

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

```

798 \newcommand*{\pendL}{\ifnumbering \else
799   \led@err@PendNotNumbered
800 \fi
801 \ifnumberedpar@ \else
802   \led@err@PendNoPstart
803 \fi

```

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

```

804 \l@dzeropenalties
805 \endgraf\global\num@lines=\prevgraf\egroup
806 \global\par@line=0

```

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

```

807 \endgroup
808 \ignorespaces
809 \@oldnbreak}
810

```

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

```

811 \newcommand*{\pendR}{\ifnumberingR \else
812   \led@err@PendNotNumbered
813 \fi
814 \ifnumberedpar@ \else
815   \led@err@PendNoPstart
816 \fi
817 \l@dzeropenalties
818 \endgraf\global\num@linesR=\prevgraf\egroup
819 \global\par@lineR=0
820 \endgroup
821 \ignorespaces
822 \@oldnbreak}
823

```

14.2 Processing one line

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

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

```
824 \newbox\l@dleftbox
825 \newbox\l@drightbox
826
```

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

```
\countRline 827 \newcount\countLline
828 \countLline \z@
829 \newcount\countRline
830 \countRline \z@
831
```

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

```
\donetotallinesR 832 \newcount@donereallinesL
833 \newcount@donetotallinesL
834 \newcount@donereallinesR
835 \newcount@donetotallinesR
836
```

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

```
837 \newcommand*{\do@lineL}{%
838 \manageparhangingsymbol
839 \advance\countLline \@ne
840 \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscL}%
841 {\vbadness=10000
842 \splittopskip=\z@
843 \do@lineLhook
844 \l@emptyd@ta
845 \global\setbox\one@line=\vsplit\namebox{\l@dLcolrawbox\the\l@dpscL}
846 to\baselineskip}%
847 \unvbox\one@line \global\setbox\one@line=\lastbox
848 \getline@numL
849 \setbox\l@dleftbox
850 \hb@xt@ \Lcolwidth{%
851 \affixline@num
852 \l@dld@ta
853 \add@inserts
854 \affixside@note
855 \l@dlsn@te
856 {\ledllfill\hb@xt@ \wd\one@line{\new@line\l@dunhbox@line{\one@line}}\ledrfill\l@drd@ta%
```

```

857     \l@drsn@te
858   }}%
859   \add@penaltiesL
860   \global\advance\@donereallinesL\@ne
861   \global\advance\@donetotallinesL\@ne
862 \else
863   \setbox\l@dleftbox \hb@xt@ \Lcolwidth{\hspace*\Lcolwidth}}%
864   \global\advance\@donetotallinesL\@ne
865 \fi}
866
867

```

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

```

\do@lineRhook 868 \newcommand*{\do@lineLhook}{%
869 \newcommand*{\do@lineRhook}{%
870

```

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

```

871 \newcommand*{\do@lineR}{%
872 \manageparhangingsymbol
873   \advance\countRline \@ne
874   \ifvbox\namebox{l@dRcolrawbox\the\l@dpscR}%
875   {\vbadness=10000
876    \splittopskip=\z@
877    \do@lineRhook
878    \l@demptyd@ta
879    \global\setbox\one@lineR=\vsplit\namebox{l@dRcolrawbox\the\l@dpscR}
880     to\baselineskip}%
881   \unvbox\one@lineR \global\setbox\one@lineR=\lastbox
882   \getline@numR
883   \setbox\l@drightbox
884   \hb@xt@ \Rcolwidth{%
885     \affixline@numR
886     \l@dld@ta
887     \add@insertsR
888     \affixside@noter
889     \l@dlsn@te
890     {\ledllfill\hb@xt@ \wd\one@lineR{\new@lineR\l@dunhbox@line{\one@lineR}}\ledrlfill\l@d
891     \l@drsn@te
892   }}%
893   \add@penaltiesR
894   \global\advance\@donereallinesR\@ne
895   \global\advance\@donetotallinesR\@ne
896 \else
897   \setbox\l@drightbox \hb@xt@ \Rcolwidth{\hspace*\Rcolwidth}}
898   \global\advance\@donetotallinesR\@ne
899 \fi}
900
901

```

14.3 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.

```

902 \newcommand*\getline@numR}{%
903   \global\advance\absline@numR \@ne
904   \do@actionsR
905   \do@ballastR
906   \ifsublines@
907     \ifnum\sub@lockR<\tw@
908       \global\advance\subline@numR \@ne
909     \fi
910   \else
911     \ifnum\@lockR<\tw@
912 \addtocounter{hbox}{10}%
913     \global\advance\line@numR \@ne
914     \global\subline@numR \z@
915   \fi
916 \fi}
917 \newcommand*\getline@numL}{%
918   \global\advance\absline@num \@ne
919   \do@actions
920   \do@ballast
921   \ifsublines@
922     \ifnum\sub@lock<\tw@
923       \global\advance\subline@num \@ne
924     \fi
925   \else
926     \ifnum\@lock<\tw@
927       \global\advance\line@num \@ne
928       \addtocounter{hbox}{10}%
929       \global\subline@num \z@
930     \fi
931   \fi}
932
933

```

`\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.

```

934 \newcommand*\do@ballastR}{\global\ballast@count=\z@
935   \beginngroup
936     \advance\absline@numR \@ne
937     \ifnum\next@actionlineR=\absline@numR
938       \ifnum\next@actionR>-1001
939         \global\advance\ballast@count by -\c@ballast
940       \fi
941     \fi
942   \endgroup}

```

`\do@actionsR` The `\do@actionsR` macro looks at the list of actions to take at particular right

```

\do@actions@fixedcodeR
\do@actions@nextR

```

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.

```

943 \newcommand*{\do@actions@fixedcodeR}{%
944   \ifcase\@l@dtmpcnta%
945   \or%           % 1001
946     \global\sublines@true
947   \or%           % 1002
948     \global\sublines@false
949   \or%           % 1003
950     \global\@lockR=\@ne
951   \or%           % 1004
952     \ifnum\@lockR=\tw@
953       \global\@lockR=\thr@@
954     \else
955       \global\@lockR=\z@
956     \fi
957   \or%           % 1005
958     \global\sub@lockR=\@ne
959   \or%           % 1006
960     \ifnum\sub@lockR=\tw@
961       \global\sub@lockR=\thr@@
962     \else
963       \global\sub@lockR=\z@
964     \fi
965   \or%           % 1007
966     \l@dskipnumbertrue
967   \else
968     \led@warn@BadAction
969   \fi}
970
971
972 \newcommand*{\do@actionsR}{%
973   \global\let\do@actions@nextR=\relax
974   \@l@dtmpcntb=\absline@numR
975   \ifnum\@l@dtmpcntb<\next@actionlineR\else
976     \ifnum\next@actionR>-1001\relax
977       \global\page@numR=\next@actionR
978       \ifbypage@R
979         \global\line@numR \z@ \global\subline@numR \z@
980       \fi
981     \else
982       \ifnum\next@actionR<-4999\relax % 9/05 added relax here
983         \@l@dtmpcnta=-\next@actionR
984         \advance\@l@dtmpcnta by -5001\relax
985         \ifsublines@
986           \global\subline@numR=\@l@dtmpcnta
987         \else

```



```

988     \global\line@numR=\@l@tempcnta
989     \fi
990     \else
991     \l@tempcnta=-\next@actionR
992     \advance\l@tempcnta by -1000\relax
993     \do@actions@fixedcodeR
994     \fi
995     \fi
996     \ifx\actionlines@listR\empty
997     \gdef\next@actionlineR{1000000}%
998     \else
999     \glp\actionlines@listR\to\next@actionlineR
1000    \glp\actions@listR\to\next@actionR
1001    \global\let\do@actions@nextR=\do@actionsR
1002    \fi
1003    \fi
1004    \do@actions@nextR}
1005

```

14.4 Line number printing

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

```

\ch@cksub@l@ckR 1006
\ch@ck@l@ckR 1007 \providecommand*\l@dcalcnm}[3]{%
\fx@l@cksR 1008 \ifnum #1 > #2\relax
\affixline@numR 1009 \l@tempcnta = #1\relax
1010 \advance\l@tempcnta by -#2\relax
1011 \divide\l@tempcnta by #3\relax
1012 \multiply\l@tempcnta by #3\relax
1013 \advance\l@tempcnta by #2\relax
1014 \else
1015 \l@tempcnta=#2\relax
1016 \fi}
1017
1018 \newcommand*\ch@cksub@l@ckR}{%
1019 \ifcase\sub@lockR
1020 \or
1021 \ifnum\sublock@disp=\@ne
1022 \l@tempcntb \z@ \l@tempcnta \@ne
1023 \fi
1024 \or
1025 \ifnum\sublock@disp=\tw@
1026 \else
1027 \l@tempcntb \z@ \l@tempcnta \@ne
1028 \fi
1029 \or
1030 \ifnum\sublock@disp=\z@
1031 \l@tempcntb \z@ \l@tempcnta \@ne
1032 \fi

```

```

1033 \fi}
1034
1035 \newcommand*{\ch@ck@l@ckR}{%
1036 \ifcase\@lockR
1037 \or
1038 \ifnum\lock@disp=\@ne
1039 \@l@tempcntb \z@ \@l@tempcnta \@ne
1040 \fi
1041 \or
1042 \ifnum\lock@disp=\tw@
1043 \else
1044 \@l@tempcntb \z@ \@l@tempcnta \@ne
1045 \fi
1046 \or
1047 \ifnum\lock@disp=\z@
1048 \@l@tempcntb \z@ \@l@tempcnta \@ne
1049 \fi
1050 \fi}
1051
1052 \newcommand*{\f@x@l@cksR}{%
1053 \ifcase\@lockR
1054 \or
1055 \global\@lockR \tw@
1056 \or \or
1057 \global\@lockR \z@
1058 \fi
1059 \ifcase\sub@lockR
1060 \or
1061 \global\sub@lockR \tw@
1062 \or \or
1063 \global\sub@lockR \z@
1064 \fi}
1065
1066
1067 \newcommand*{\affixline@numR}{%
1068 \ifl@dskipnumber
1069 \global\l@dskipnumberfalse
1070 \else
1071 \ifsublines@
1072 \@l@tempcntb=\subline@numR
1073 \l@dcalcnm{\subline@numR}{\c@firstsublinenumR}{\c@sublinenumincrementR}%
1074 \ch@cksub@lockR
1075 \else
1076 \@l@tempcntb=\line@numR
1077 \ifx\linenumberlist\empty
1078 \l@dcalcnm{\line@numR}{\c@firstlinenumR}{\c@linenumincrementR}%
1079 \else
1080 \@l@tempcnta=\line@numR
1081 \edef\rem@inder{\linenumberlist,\number\line@numR,}%
1082 \edef\sc@n@list{\def\noexpand\sc@n@list

```

```

1083     ###1,\number\@l@tempcnta,###2|{\def\noexpand\rem@inder{###2}}}%
1084     \sc@n@list\expandafter\sc@n@list\rem@inder|%
1085     \ifx\rem@inder\empty\advance\@l@tempcnta\@ne\fi
1086     \fi
1087     \ch@ck@l@ckR
1088     \fi
1089     \ifnum\@l@tempcnta=\@l@tempcntb
1090     \if@twocolumn
1091     \if@firstcolumn
1092     \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1093     \else
1094     \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1095     \fi
1096     \else
1097     \@l@tempcntb=\line@marginR
1098     \ifnum\@l@tempcntb>\@ne
1099     \advance\@l@tempcntb by\page@numR
1100     \fi
1101     \ifodd\@l@tempcntb
1102     \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1103     \else
1104     \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1105     \fi
1106     \fi
1107     \fi
1108     \f@x@l@cksR
1109 \fi}
1110

```

14.5 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.

```
1111 \list@create{\inserts@listR}
```

`\add@insertsR` The right text version.

```

\add@inserts@nextR 1112 \newcommand*{\add@insertsR}{%
1113     \global\let\add@inserts@nextR=\relax
1114     \ifx\inserts@listR\empty \else
1115     \ifx\next@insertR\empty
1116     \ifx\insertlines@listR\empty
1117     \global\noteschanged@true
1118     \gdef\next@insertR{100000}%
1119     \else
1120     \gl@p\insertlines@listR\to\next@insertR
1121     \fi
1122     \fi
1123     \ifnum\next@insertR=\absline@numR
1124     \gl@p\inserts@listR\to\@insertR

```

```

1125     \@insertR
1126     \global\let\@insertR=\undefined
1127     \global\let\next@insertR=\empty
1128     \global\let\add@inserts@nextR=\add@insertsR
1129     \fi
1130     \fi
1131     \add@inserts@nextR}
1132

```

14.6 Penalties

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

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

```

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

```

This is for a single chunk. However, as we are probably dealing with several chunks

at a time, the above is nor really relevant. I think that it is likely with parallel text that there is no real need to add back any penalties; even if there was, they would have to match across the left and right lines. So, I end up with the following.

```
1133 \newcommand*\add@penaltiesL-{}
1134 \newcommand*\add@penaltiesR-{}
1135
```

14.7 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.

```
1136 \newcommand*\flush@notesR-{}
1137 \xloop
1138 \ifx\inserts@listR\empty \else
1139 \gl@p\inserts@listR\to\insertR
1140 \insertR
1141 \global\let\insertR=\undefined
1142 \repeat}
1143
```

15 Footnotes

15.1 Outer-level footnote commands

`\Afootnote` The outer-level footnote commands will look familiar: they're just called `\Afootnote`, `\Bfootnote`, etc., instead of plain `\footnote`. What they do, however, is quite different, since they have to operate in conjunction with `\edtext` when numbering is in effect.

If we're within a line-numbered paragraph, then, we tack this note onto the `\inserts@list` list, and increment the deferred-page-bottom-note counter.

```
1144 \renewcommand*\Afootnote}[1]{%
1145 \ifnumberedpar@
1146 \ifledRcol
1147 \xright@appenditem{\noexpand\Afootnote{A}%
1148 {{\l@d@nums}{\@tag}{#1}}}\to\inserts@listR
1149 \global\advance\insert@countR \@ne
1150 \else
1151 \xright@appenditem{\noexpand\Afootnote{A}%
1152 {{\l@d@nums}{\@tag}{#1}}}\to\inserts@list
1153 \global\advance\insert@count \@ne
1154 \fi
```

Within free text, there's no need to put off making the insertion for this note.

No line numbers are available, so this isn't generally that useful; but you might want to use it to get around some limitation of `ledmac`.

```
1155 \else
1156 \Afootnote{A}{{0|0|0|0|0|0|0|0}{#1}}%
1157 \fi\ignorespaces}
```

```

\Bfootnote We need similar commands for the other footnote series.
\Cfootnote 1158 \renewcommand*{\Bfootnote}[1]{%
\Dfootnote 1159 \ifnumberedpar@
\Efootnote 1160 \ifledRcol
1161 \xright@appenditem{\noexpand\Bfootnote{B}%
1162 {\l@d@nums}{\@tag}{#1}}\to\inserts@listR
1163 \global\advance\insert@countR \@ne
1164 \else
1165 \xright@appenditem{\noexpand\Bfootnote{B}%
1166 {\l@d@nums}{\@tag}{#1}}\to\inserts@list
1167 \global\advance\insert@count \@ne
1168 \fi
1169 \else
1170 \vBfootnote{B}{0|0|0|0|0|0|0|0}{#1}%
1171 \fi\ignorespaces}

1172 \renewcommand*{\Cfootnote}[1]{%
1173 \ifnumberedpar@
1174 \ifledRcol
1175 \xright@appenditem{\noexpand\Cfootnote{C}%
1176 {\l@d@nums}{\@tag}{#1}}\to\inserts@listR
1177 \global\advance\insert@countR \@ne
1178 \else
1179 \xright@appenditem{\noexpand\Cfootnote{C}%
1180 {\l@d@nums}{\@tag}{#1}}\to\inserts@list
1181 \global\advance\insert@count \@ne
1182 \fi
1183 \else
1184 \vCfootnote{C}{0|0|0|0|0|0|0|0}{#1}%
1185 \fi\ignorespaces}

1186 \renewcommand*{\Dfootnote}[1]{%
1187 \ifnumberedpar@
1188 \ifledRcol
1189 \xright@appenditem{\noexpand\Dfootnote{D}%
1190 {\l@d@nums}{\@tag}{#1}}\to\inserts@listR
1191 \global\advance\insert@countR \@ne
1192 \else
1193 \xright@appenditem{\noexpand\Dfootnote{D}%
1194 {\l@d@nums}{\@tag}{#1}}\to\inserts@list
1195 \global\advance\insert@count \@ne
1196 \fi
1197 \else
1198 \vDfootnote{D}{0|0|0|0|0|0|0|0}{#1}%
1199 \fi\ignorespaces}

1200 \renewcommand*{\Efootnote}[1]{%
1201 \ifnumberedpar@
1202 \ifledRcol
1203 \xright@appenditem{\noexpand\Efootnote{E}%
1204 {\l@d@nums}{\@tag}{#1}}\to\inserts@listR

```

```

1205     \global\advance\insert@countR \@ne
1206     \else
1207     \xright@appenditem{\noexpand\vEfootnote{E}%
1208                       {\l@d@nums}{\@tag}{#1}}\to\inserts@list
1209     \global\advance\insert@count \@ne
1210     \fi
1211 \else
1212     \vEfootnote{E}{0|0|0|0|0|0|0}{#1}%
1213 \fi\ignorespaces}
1214

```

`\mpAfootnote` For footnotes in minipages and the like, we need a similar series of commands.

```

\mpBfootnote 1215 \renewcommand*{\mpAfootnote}[1]{%
\mpCfootnote 1216 \ifnumberedpar@
\mpDfootnote 1217 \ifledRcol
\mpEfootnote 1218 \xright@appenditem{\noexpand\mpvAfootnote{A}%
1219                       {\l@d@nums}{\@tag}{#1}}\to\inserts@listR
1220     \global\advance\insert@countR \@ne
1221     \else
1222     \xright@appenditem{\noexpand\mpvAfootnote{A}%
1223                       {\l@d@nums}{\@tag}{#1}}\to\inserts@list
1224     \global\advance\insert@count \@ne
1225     \fi
1226     \else
1227     \mpvAfootnote{A}{0|0|0|0|0|0|0}{#1}%
1228     \fi\ignorespaces}

1229 \renewcommand*{\mpBfootnote}[1]{%
1230 \ifnumberedpar@
1231 \ifledRcol
1232 \xright@appenditem{\noexpand\mpvBfootnote{B}%
1233                       {\l@d@nums}{\@tag}{#1}}\to\inserts@listR
1234 \global\advance\insert@countR \@ne
1235 \else
1236 \xright@appenditem{\noexpand\mpvBfootnote{B}%
1237                       {\l@d@nums}{\@tag}{#1}}\to\inserts@list
1238 \global\advance\insert@count \@ne
1239 \fi
1240 \else
1241 \mpvBfootnote{B}{0|0|0|0|0|0|0}{#1}%
1242 \fi\ignorespaces}

1243 \renewcommand*{\mpCfootnote}[1]{%
1244 \ifnumberedpar@
1245 \ifledRcol
1246 \xright@appenditem{\noexpand\mpvCfootnote{C}%
1247                       {\l@d@nums}{\@tag}{#1}}\to\inserts@listR
1248 \global\advance\insert@countR \@ne
1249 \else
1250 \xright@appenditem{\noexpand\mpvCfootnote{C}%
1251                       {\l@d@nums}{\@tag}{#1}}\to\inserts@list

```

```

1252 \global\advance\insert@count \@ne
1253 \fi
1254 \else
1255 \mpvCfootnote{C}{\l@d@nums}{\@tag}{#1}}%
1256 \fi\ignorespaces}

1257 \renewcommand*\mpDfootnote}[1]{%
1258 \ifnumberedpar@
1259 \ifledRcol
1260 \xright@appenditem{\noexpand\mpvDfootnote{D}}%
1261 {\l@d@nums}{\@tag}{#1}}\to\inserts@listR
1262 \global\advance\insert@countR \@ne
1263 \else
1264 \xright@appenditem{\noexpand\mpvDfootnote{D}}%
1265 {\l@d@nums}{\@tag}{#1}}\to\inserts@list
1266 \global\advance\insert@count \@ne
1267 \fi
1268 \else
1269 \mpvDfootnote{D}{\l@d@nums}{\@tag}{#1}}%
1270 \fi\ignorespaces}

1271 \renewcommand*\mpEfootnote}[1]{%
1272 \ifnumberedpar@
1273 \ifledRcol
1274 \xright@appenditem{\noexpand\mpvEfootnote{E}}%
1275 {\l@d@nums}{\@tag}{#1}}\to\inserts@listR
1276 \global\advance\insert@countR \@ne
1277 \else
1278 \xright@appenditem{\noexpand\mpvEfootnote{E}}%
1279 {\l@d@nums}{\@tag}{#1}}\to\inserts@list
1280 \global\advance\insert@count \@ne
1281 \fi
1282 \else
1283 \mpvEfootnote{E}{\l@d@nums}{\@tag}{#1}}%
1284 \fi\ignorespaces}

```

15.2 Normal footnote formatting

The `\printlines` macro prints the line numbers for a note—which, in the general case, is a rather complicated task. The seven parameters of the argument are the line numbers as stored in `\l@d@nums`, in the form described on page ??: the starting page, line, and sub-line numbers, followed by the ending page, line, and sub-line numbers, and then the font specifier for the lemma.

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

Just a reminder of the arguments:

```

\printlinesR #1 | #2 | #3 | #4 | #5 | #6 | #7
\printlinesR start-page | line | subline | end-page | line | subline | font
1285 \def\printlinesR#1|#2|#3|#4|#5|#6|#7{\begingroup

```



```

1286 \setprintlines{#1}{#2}{#3}{#4}{#5}{#6}%
1287 \ifl@d@pnum #1\fullstop\fi
1288 \ifledplinenum \linenumr@p{#2}\Rlineflag\else \symplinenum\fi
1289 \ifl@d@ssub \fullstop \sublinenumr@p{#3}\fi
1290 \ifl@d@dash \endashchar\fi
1291 \ifl@d@pnum #4\fullstop\fi
1292 \ifl@d@elin \linenumr@p{#5}\Rlineflag\fi
1293 \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumr@p{#6}\fi
1294 \endgroup}
1295
1296 \let\ledsavedprintlines\printlines
1297

```

16 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.

```

1298 \list@create{\labelref@listR}
1299

```

`\edlabel` The `\edlabel` command first writes a `\@lab` macro to the `\linenum@out` file. It then checks to see that the `\labelref@list` actually has something in it (if not, it creates a dummy entry), and pops the next value for the current label, storing it in `\label@refs`. Finally it defines the label to be `\empty` so that any future check will turn up the fact that it has been used.

```

1300 \renewcommand*{\edlabel}[1]{\@bsphack
1301 \ifledRcol
1302 \write\linenum@outR{\string\@lab}%
1303 \ifx\labelref@list\empty
1304 \xdef\label@refs{\zz@@}%
1305 \else
1306 \gl@p\labelref@listR\to\label@refs
1307 \fi
1308 \protected@write\@auxout{%
1309 {\string\l@dmake@labelsR\space\thepage|\label@refs|{#1}}%
1310 \else
1311 \write\linenum@out{\string\@lab}%
1312 \ifx\labelref@list\empty
1313 \xdef\label@refs{\zz@@}%
1314 \else
1315 \gl@p\labelref@list\to\label@refs
1316 \fi
1317 \fi
1318 \protected@write\@auxout{%
1319 {\string\l@dmake@labels\space\thepage|\label@refs|{#1}}%
1320 \@esphack}
1321

```

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

```

1322 \def\l@dmake@labelsR#1|#2|#3|#4{%
1323   \expandafter\ifx\csname the@label#4\endcsname \relax\else
1324     \led@warn@DuplicateLabel{#4}%
1325   \fi
1326   \expandafter\gdef\csname the@label#4\endcsname{#1|#2\Rlineflag|#3}%
1327   \ignorespaces}
1328 \AtBeginDocument{%
1329   \def\l@dmake@labelsR#1|#2|#3|#4{%
1330 }
1331
```

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

```

1332 \renewcommand*{\@lab}{%
1333   \ifledRcol
1334     \xright@appenditem{\linenumr@p{\line@numR}}|{%
1335       \ifsublines@ \sublinenumr@p{\subline@numR}\else 0\fi}%
1336     \to\labelref@listR
1337   \else
1338     \xright@appenditem{\linenumr@p{\line@num}}|{%
1339       \ifsublines@ \sublinenumr@p{\subline@num}\else 0\fi}%
1340     \to\labelref@list
1341   \fi}
1342
```

17 Side notes

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

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

```

\sidenotemargin 1343 \newcount\sidenote@marginR
1344 \renewcommand*{\sidenotemargin}[1]{%
1345   \l@dgetsidenote@margin{#1}%
1346   \ifnum\@l@dtempcntb>\m@ne
1347     \ifledRcol
1348       \global\sidenote@marginR=\@l@dtempcntb
1349     \else
1350       \global\sidenote@margin=\@l@dtempcntb
1351     \fi
1352   \fi}}
1353 \sidenotemargin{right}
1354 \global\sidenote@margin=\@ne
1355
```

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

```

\l@dcsnote 1356 \renewcommand*{\l@dlsnote}[1]{%
1357   \ifnumberedpar@
1358     \ifledRcol
1359       \xright@appenditem{\noexpand\l@dlsnote{#1}}%
1360         \to\inserts@listR
1361     \global\advance\insert@countR \@ne
1362   \else
1363     \xright@appenditem{\noexpand\l@dlsnote{#1}}%
1364       \to\inserts@list
1365     \global\advance\insert@count \@ne
1366   \fi
1367 \fi\ignorespaces}
1368 \renewcommand*{\l@drsnote}[1]{%
1369   \ifnumberedpar@
1370     \ifledRcol
1371       \xright@appenditem{\noexpand\l@drsnote{#1}}%
1372         \to\inserts@listR
1373     \global\advance\insert@countR \@ne
1374   \else
1375     \xright@appenditem{\noexpand\l@drsnote{#1}}%
1376       \to\inserts@list
1377     \global\advance\insert@count \@ne
1378   \fi
1379 \fi\ignorespaces}
1380 \renewcommand*{\l@dcsnote}[1]{%
1381   \ifnumberedpar@
1382     \ifledRcol
1383       \xright@appenditem{\noexpand\l@dcsnote{#1}}%
1384         \to\inserts@listR
1385     \global\advance\insert@countR \@ne
1386   \else
1387     \xright@appenditem{\noexpand\l@dcsnote{#1}}%
1388       \to\inserts@list
1389     \global\advance\insert@count \@ne
1390   \fi
1391 \fi\ignorespaces}
1392

```

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

```

1393 \newcommand*{\affixside@noteR}{%
1394   \gdef\@templ@d{ }%
1395   \ifx\@templ@d\l@dcsnotetext \else
1396     \if@twocolumn
1397       \if@firstcolumn
1398         \setl@dlp@rbox{\l@dcsnotetext}%
1399       \else
1400         \setl@drp@rbox{\l@dcsnotetext}%

```

```

1401     \fi
1402   \else
1403     \@l@dttempcntb=\sidenote@marginR
1404     \ifnum\@l@dttempcntb>\@ne
1405       \advance\@l@dttempcntb by\page@num
1406     \fi
1407     \ifodd\@l@dttempcntb
1408       \setl@drp@rbox{\l@dcsnotetext}%
1409     \else
1410       \setl@dlp@rbox{\l@dcsnotetext}%
1411     \fi
1412   \fi
1413 \fi}
1414

```

18 Familiar footnotes

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

```

1415 \renewcommand{\l@dbfnote}[1]{%
1416   \ifnumberedpar@
1417     \ifledRcol
1418       \xright@appenditem{\noexpand\vl@dbfnote{#1}{\@thefnmark}}%
1419         \to\inserts@listR
1420     \global\advance\insert@countR \@ne
1421   \else
1422     \xright@appenditem{\noexpand\vl@dbfnote{#1}{\@thefnmark}}%
1423       \to\inserts@list
1424     \global\advance\insert@count \@ne
1425   \fi
1426 \fi\ignorespaces}
1427

```

`\normalbfnoteX`

```

1428 \renewcommand{\normalbfnoteX}[2]{%
1429   \ifnumberedpar@
1430     \ifledRcol
1431       \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\@nameuse{thefootnote#1}}}%
1432         \to\inserts@listR
1433     \global\advance\insert@countR \@ne
1434   \else
1435     \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\@nameuse{thefootnote#1}}}%
1436       \to\inserts@list
1437     \global\advance\insert@count \@ne
1438   \fi
1439 \fi\ignorespaces}
1440

```

19 Verse

The `\manageparhangingsymbol` command is made to insert the hanging symbol (like in the french typography).

```

1441
1442 \newcommand{\manageparhangingsymbol}{%
1443   \setcounter{hbox}{0}%
1444   \everyhbox{%
1445     \ifnum \value{hbox}=-2%
1446       \hangingsymbol%
1447     \fi%
1448     \addtocounter{hbox}{-1}}
1449 %\end{macrocode}
1450 % Before we can define the main stanza macros we need to be able to save
1451 % and reset
1452 % the category code for \&. To save the current value we use
1453 % \verb+\next+ from the \verb+\loop+ macro.
1454 % \begin{macrocode}
1455 \chardef\next=\catcode'\&
1456 \catcode'\&=\active
1457

```

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

```

1458 \newenvironment{astanza}{%
1459   \startstanzahook
1460   \catcode'\&\active
1461   \global\stanza@count\@ne
1462   \ifnum\usernamecount{sza@00}=\z@
1463     \let\stanza@hang\relax
1464     \let\endlock\relax
1465   \else
1466   %% \interlinepenalty\@M % this screws things up, but I don't know why
1467     \rightskip\z@ plus 1fil\relax
1468     \fi
1469     \ifnum\usernamecount{szp@00}=\z@
1470       \let\sza@penalty\relax
1471     \fi
1472     \def&{%
1473       \endlock\mbox{}%
1474       \sza@penalty
1475       \global\advance\stanza@count\@ne
1476       \@astanza@line}%
1477     \def\&{%
1478       \endlock\mbox{}
1479       \pend
1480       \endstanzaextra}%
1481     \pstart
1482     \@astanza@line

```

```
1483 }{}
1484
```

`\@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.

```
1485 \newcommand*\@astanza@line}{%
1486   \parindent=\csname sza@number\stanza@count @\endcsname\stanzaindentbase
1487   \par
1488   \stanza@hang%\mbox{}%
1489   \ignorespaces}
1490
```

Lastly reset the modified category codes.

```
1491 \catcode'\&=\next
1492
```

20 Naming macros

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

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

```
\unhnamebox 1493 \providecommand*\newnamebox}[1]{%
\unvnamebox 1494   \expandafter\newbox\csname #1\endcsname}
\namebox 1495 \providecommand*\setnamebox}[1]{%
1496   \expandafter\setbox\csname #1\endcsname}
1497 \providecommand*\unhnamebox}[1]{%
1498   \expandafter\unhbox\csname #1\endcsname}
1499 \providecommand*\unvnamebox}[1]{%
1500   \expandafter\unvbox\csname #1\endcsname}
1501 \providecommand*\namebox}[1]{%
1502   \csname #1\endcsname}
1503
```

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

```
\usenamecount 1504 \providecommand*\newnamecount}[1]{%
1505   \expandafter\newcount\csname #1\endcsname}
1506 \providecommand*\usenamecount}[1]{%
1507   \csname #1\endcsname}
1508
```

21 Counts and boxes for parallel texts

In sequential text, each chunk (that enclosed by `\pstart ... \pend`) is put into a box called `\raw@text` and then immediately printed, resulting in the box being

emptied and ready for the next chunk. For parallel processing multiple boxes are needed as printing is delayed. We also need extra counters for various things.

`\maxchunks` The maximum number of chunk pairs before printing has to be called for. The
`\l@dc@maxchunks` default is 10 chunk pairs.

```
1509 \newcount\l@dc@maxchunks
1510 \newcommand{\maxchunks}[1]{\l@dc@maxchunks=#1}
1511 \maxchunks{10}
1512
```

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

`\l@dnumpstartsR` 1513 `\newcount\l@dnumpstartsR`
 1514

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

```
\l@pscl 1515 \newcount\l@dpscl
\l@psclR 1516 \newcount\l@dpsclR
1517
```

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

```
1518 \newcommand*{\l@dsetuprawboxes}{%
1519 \l@dc@tempcntb=\l@dc@maxchunks
1520 \loop\ifnum\l@dc@tempcntb>\z@
1521 \newnamebox{\l@dLcolrawbox\the\l@dc@tempcntb}
1522 \newnamebox{\l@dRcolrawbox\the\l@dc@tempcntb}
1523 \advance\l@dc@tempcntb \m@ne
1524 \repeat}
1525
```

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

```
1526 \newcommand*{\l@dsetupmaxlinecounts}{%
1527 \l@dc@tempcntb=\l@dc@maxchunks
1528 \loop\ifnum\l@dc@tempcntb>\z@
1529 \newnamecount{\l@dmaxlinesinpar\the\l@dc@tempcntb}
1530 \advance\l@dc@tempcntb \m@ne
1531 \repeat}
1532 \newcommand*{\l@dzeromaxlinecounts}{%
1533 \begingroup
1534 \l@dc@tempcntb=\l@dc@maxchunks
1535 \loop\ifnum\l@dc@tempcntb>\z@
1536 \global\usenamecount{\l@dmaxlinesinpar\the\l@dc@tempcntb}=\z@
1537 \advance\l@dc@tempcntb \m@ne
1538 \repeat
1539 \endgroup}
1540
```

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

```

1541 \AtBeginDocument{%
1542   \l@dsetuprawboxes
1543   \l@dsetupmaxlinecounts
1544   \l@dzeromaxlinecounts
1545   \l@dnumpstartsL=\z@
1546   \l@dnumpstartsR=\z@
1547   \l@dpscL=\z@
1548   \l@dpscR=\z@}
1549

```

22 Fixing babel

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

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

```

\ifl@dusedbabel A flag for checking if babel has been used as a package.
\l@dusedbabelfalse 1550 \newif\ifl@dusedbabel
\l@dusedbabeltrue 1551 \l@dusedbabelfalse

\ifl@dsamelang A flag for checking if the same babel language has been used for both the left and
\l@dsamelangfalse right texts.
\l@dsamelangtrue 1552 \newif\ifl@dsamelang
1553 \l@dsamelangtrue

\l@dchecklang I'm going to use \theledlanguageL and \theledlanguageR to hold the names of
the languages used for the left and right texts. This macro sets \ifl@dsamelang
TRUE if they are the same, otherwise it sets it FALSE.
1554 \newcommand*\l@dchecklang}{%
1555   \l@dsamelangfalse
1556   \edef\@tempa{\theledlanguageL}\edef\@tempb{\theledlanguageR}%
1557   \ifx\@tempa\@tempb
1558     \l@dsamelangtrue
1559   \fi}
1560

\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 `\`.

```

1561 \newcommand*\l@dbbl@set@language}[1]{%
1562   \edef\languagename{#1}%
1563   \select@language{\languagename}%
1564   \if@filesw
1565     \protected@write\auxout{}\string\select@language{\languagename}}%
1566     \addtocontents{toc}{\string\select@language{\languagename}}%
1567     \addtocontents{lof}{\string\select@language{\languagename}}%
1568     \addtocontents{lot}{\string\select@language{\languagename}}%
1569   \fi}
1570

```

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

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

```

\theledlanguageR 1571 \providecommand{\selectlanguage}[1]{
1572   \newcommand*\l@duselanguage}[1]{
1573     \gdef\theledlanguageL{
1574       \gdef\theledlanguageR{
1575

```

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

```

1576 \AtBeginDocument{%
1577   \@ifundefined{xpg@main@language}{%
1578     \@ifundefined{bbl@main@language}{%

```

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

```

1579   \l@dusedbabelfalse
1580   \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.

```

1581   \l@dusedbabeltrue
1582   \let\l@doldselectlanguage\selectlanguage
1583   \let\l@doldbbl@set@language\bbl@set@language
1584   \let\bbl@set@language\l@dbbl@set@language
1585   \renewcommand*\selectlanguage}[1]{%
1586     \l@doldselectlanguage{#1}%
1587     \ifledRcol \gdef\theledlanguageR{#1}%
1588     \else      \gdef\theledlanguageL{#1}%
1589   \fi}

```

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

```

1590 \renewcommand*{\l@duselanguage}[1]{%
1591   \l@doldselectlanguage{#1}}

```

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

```

1592 \gdef\theledlanguageL{\bb1@main@language}%
1593 \gdef\theledlanguageR{\bb1@main@language}%
1594 }%
1595 }

```

If on Polyglossia

```

1596 { \apptocmd{\xpg@set@language}{%
1597   \ifledRcol \gdef\theledlanguageR{#1}%
1598   \else      \gdef\theledlanguageL{#1}%
1599   \fi}%
1600 \let\l@duselanguage\xpg@set@language
1601 \gdef\theledlanguageL{\xpg@main@language}%
1602 \gdef\theledlanguageR{\xpg@main@language}%
1603 % \end{macrocode}
1604 % That's it.
1605 % \begin{macrocode}
1606 }}

```

23 Parallel columns

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

```

1607 \newcommand*{\Columns}{%
1608   \ifnum\l@dnumstartsL=\l@dnumstartsR\else
1609     \led@err@BadLeftRightPstarts{\the\l@dnumstartsL}{\the\l@dnumstartsR}%
1610   \fi

```

Start a group and zero counters, etc.

```

1611 \begingroup
1612   \l@dzeropenalties
1613   \endgraf\global\num@lines=\prevgraf
1614     \global\num@linesR=\prevgraf
1615   \global\par@line=\z@
1616   \global\par@lineR=\z@
1617   \global\l@dpscL=\z@
1618   \global\l@dpscR=\z@

```

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

```

1619   \check@pstarts
1620   \loop\if@pstarts

```

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

```

1621     \global\advance\l@dpscL \@ne

```

```

1622     \global\advance\l@dpscR \@ne
        Check if there is text yet to be processed in at least one of the two current chunks,
        and also whether the left and right languages are the same
1623     \checkraw@text
1624     \l@dchecklang
1625 {     \loop\ifaraw@text
        Grab the next pair of left and right text lines and output them, swapping languages
        if they differ
1626         \ifl@dsamelang
1627         \do@lineL
1628         \do@lineR
1629     \else
1630         \l@duselanguage{\theledlanguageL}%
1631         \do@lineL
1632         \l@duselanguage{\theledlanguageR}%
1633         \do@lineR
1634     \fi
1635     \hb@xt@ \hsize{%
1636         \unhbox\l@dleftbox
1637         \hfill \columnseparator \hfill
1638         \unhbox\l@drightbox
1639     }%
1640     \checkraw@text
1641     \repeat}
        Having completed a pair of chunks, write the number of lines in each chunk to the
        respective section files.
1642     \@writelinesinparL
1643     \@writelinesinparR
1644     \check@pstarts
1645     \repeat
        Having output all chunks, make sure all notes have been output, then zero counts
        ready for the next set of texts.
1646     \flush@notes
1647     \flush@notesR
1648     \endgroup
1649     \global\l@dpscL=\z@
1650     \global\l@dpscR=\z@
1651     \global\l@dnumpstartsL=\z@
1652     \global\l@dnumpstartsR=\z@
1653     \ignorespaces}
1654

```

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

```

1655 \newcommand*\columnseparator{%
1656   \smash{\rule[-0.2\baselineskip]{\columnrulewidth}{1.05\baselineskip}}}
1657 \newdimen\columnrulewidth
1658   \columnrulewidth=\z@
1659

```

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

```

\@pstartstrue 1660 \newif\if@pstarts
\@pstartsfalse 1661 \newcommand*\check@pstarts{%
\check@pstarts 1662   \@pstartsfalse
                  1663   \ifnum\l@dnumpstartsL>\l@dpscl
                  1664     \@pstartstrue
                  1665   \else
                  1666     \ifnum\l@dnumpstartsR>\l@dpscl
                  1667       \@pstartstrue
                  1668     \fi
                  1669   \fi}
1670

```

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

```

\checkraw@text 1671 \newif\ifaraw@text
                  1672   \araw@textfalse
                  1673 \newcommand*\checkraw@text{%
                  1674   \araw@textfalse
                  1675   \ifvbox\namebox{l@dLcolrawbox\the\l@dpscl}
                  1676     \araw@texttrue
                  1677   \else
                  1678     \ifvbox\namebox{l@dRcolrawbox\the\l@dpscl}
                  1679       \araw@texttrue
                  1680     \fi
                  1681   \fi}
1682

```

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

```

1683 \newcommand*\@writelinesinparL{%
1684   \edef\next{%
1685     \write\linenum@out{\string\@pend[\the\@donereallinesL]}%
1686   \next
1687   \global\@donereallinesL \z@}
1688 \newcommand*\@writelinesinparR{%
1689   \edef\next{%
1690     \write\linenum@outR{\string\@pendR[\the\@donereallinesR]}%
1691   \next
1692   \global\@donereallinesR \z@}
1693

```

24 Parallel pages

This is considerably more complicated than parallel columns.

```

\l@minpagelines 1694 \newcount\numpagelinesL
\l@minpagelines 1695 \newcount\numpagelinesR
\l@minpagelines 1696 \newcount\l@dminpagelines
\l@minpagelines 1697

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

1698 \newcommand*\Pages{%
1699   \typeout{}
1700   \typeout{***** PAGES *****}
1701   \ifnum\l@dnumstartsL=\l@dnumstartsR\else
1702     \led@err@BadLeftRightPstarts{\the\l@dnumstartsL}{\the\l@dnumstartsR}%
1703   \fi

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

1704   \clearto\devenpage
1705   \begingroup
1706     \l@dzeropenalties
1707     \endgraf\global\num@lines=\prevgraf
1708     \global\num@linesR=\prevgraf
1709     \global\par@line=\z@
1710     \global\par@lineR=\z@
1711     \global\l@dpscL=\z@
1712     \global\l@dpscR=\z@
1713     \writtenlinesLfalse
1714     \writtenlinesRfalse

   Check if there are chunks to be processed.

1715     \check@pstarts
1716     \loop\if@pstarts

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

1717       \global\advance\l@dpscL \@ne
1718       \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.

1719       \getlinesfromparlistL
1720       \getlinesfromparlistR
1721       \l@dcalc@maxoftwo{\cs@linesinparL}{\cs@linesinparR}%
1722       {\usenamecount{l@dmaxlinesinpar\the\l@dpscL}}%
1723       \check@pstarts
1724       \repeat

```

Zero the counts again, ready for the next bit.

```
1725 \global\l@dpscL=\z@
1726 \global\l@dpscR=\z@
```

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

```
1727 \getlinesfrompagelistL
1728 \getlinesfrompagelistR
1729 \l@dcalc@minoftwo{\cs@linesonpageL}{\cs@linesonpageR}%
1730 {\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.

```
1731 \check@pstarts
1732 \if@pstarts
```

Increment the chunk counts to get the first pair.

```
1733 \global\advance\l@dpscL \@ne
1734 \global\advance\l@dpscR \@ne
```

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

```
1735 \global\@donereallinesL=\z@
1736 \global\@donetotallinesL=\z@
1737 \global\@donereallinesR=\z@
1738 \global\@donetotallinesR=\z@
```

Start a loop over the boxes (chunks).

```
1739 \checkraw@text
1740 % \begingroup
1741 { \loop\ifaraw@text
```

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

```
1742 \checkpageL
1743 \l@duselanguage{\theledlanguageL}%
1744 %%% \begingroup
1745 { \loop\ifl@dsamepage
```

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

```
1746 \do@lineL
1747 \advance\l@numpagelinesL \@ne
1748 \ifshiftedverses
1749 \addtocounter{hbox}{-1}
1750 \ifdim\ht\l@dleftbox>0pt\hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}\fi%
1751 \else
1752 \hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
1753 \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.

```

1754         \get@nextboxL
1755         \checkpageL
1756         \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.

```

1757         \ifl@dpagfull
1758         \@writelinesonpageL{\the\numpagelinesL}%
1759         \else
1760         \@writelinesonpageL{1000}%
1761         \fi

```

Zero the left page lines count and clear the page to get onto the facing (odd, right) page.

```

1762         \numpagelinesL \z@
1763         \clearl@dleftpage }%

```

Now do the same for the right text.

```

1764         \checkpageR
1765         \l@duselanguage{\theledlanguageR}%
1766 {
1767         \loop\ifl@dsamepage
1768         \do@lineR
1769         \advance\numpagelinesR \@ne
1770 \addtocounter{hbox}{-1}
1771         \ifdim\ht\l@drightbox>0pt\hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}\fi%
1772         \else
1773         \hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}%
1774         \fi
1775         \get@nextboxR
1776         \checkpageR
1777         \repeat
1778         \ifl@dpagfull
1779         \@writelinesonpageR{\the\numpagelinesR}%
1780         \else
1781         \@writelinesonpageR{1000}%
1782         \fi
1783         \numpagelinesR=\z@

```

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

```

1784         \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.

```

1785         \checkraw@text
1786         \ifaraw@text
1787         \getlinesfrompagelistL
1788         \getlinesfrompagelistR

```

```

1789          \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
1790          {\l@dminpagelines}%
1791          \fi
1792      \repeat}

```

We have now output the text from all the chunks.

```

1793      \fi
      Make sure that there are no inserts hanging around.
1794      \flush@notes
1795      \flush@notesR
1796  \endgroup

```

Zero counts ready for the next set of left/right text chunks.

```

1797  \global\l@dpscL=\z@
1798  \global\l@dpscR=\z@
1799  \global\l@dnumstartsL=\z@
1800  \global\l@dnumstartsR=\z@
1801
1802  \ignorespaces}
1803

```

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

```

\ledstrutR 1804 \newcommand*{\ledstrutL}{\strut}
1805 \newcommand*{\ledstrutR}{\strut}
1806

```

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

```

1807 \providecommand{\cleartoevenpage}[1][\@empty]{%
1808   \clearpage
1809   \ifodd\c@page\hbox{#1}\clearpage\fi}
1810 \newcommand*{\cleartol@devenpage}{%
1811   \ifdim\pagetotal<\topskip% on an empty page
1812   \else
1813     \clearpage
1814   \fi
1815   \ifodd\c@page\hbox{ }\clearpage\fi}
1816 \newcommand*{\clearl@dleftpage}{%
1817   \clearpage
1818   \ifodd\c@page\else
1819     \led@err@LeftOnRightPage
1820     \hbox{ }%
1821     \cleardoublepage
1822   \fi}
1823 \newcommand*{\clearl@drightpage}{%
1824   \clearpage

```



```

1825 \ifodd\c@page
1826   \led@err@RightOnLeftPage
1827   \hbox{}%
1828   \cleartoevenpage
1829 \fi}
1830

```

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

```

\cs@linesinparR 1831 \newcommand*{\getlinesfromparlistL}{%
1832   \ifx\linesinpar@listL\empty
1833     \gdef\cs@linesinparL{0}%
1834   \else
1835     \gl@p\linesinpar@listL\to\cs@linesinparL
1836   \fi}
1837 \newcommand*{\getlinesfromparlistR}{%
1838   \ifx\linesinpar@listR\empty
1839     \gdef\cs@linesinparR{0}%
1840   \else
1841     \gl@p\linesinpar@listR\to\cs@linesinparR
1842   \fi}
1843

```

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

```

\cs@linesonpageR 1844 \newcommand*{\getlinesfrompagelistL}{%
1845   \ifx\linesonpage@listL\empty
1846     \gdef\cs@linesonpageL{1000}%
1847   \else
1848     \gl@p\linesonpage@listL\to\cs@linesonpageL
1849   \fi}
1850 \newcommand*{\getlinesfrompagelistR}{%
1851   \ifx\linesonpage@listR\empty
1852     \gdef\cs@linesonpageR{1000}%
1853   \else
1854     \gl@p\linesonpage@listR\to\cs@linesonpageR
1855   \fi}
1856

```

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

```

1857 \newcommand*{\@writelinesonpageL}[1]{%
1858   \edef\next{\write\linenum@out{\string\@lopL{#1}}}%
1859   \next}
1860 \newcommand*{\@writelinesonpageR}[1]{%
1861   \edef\next{\write\linenum@outR{\string\@lopR{#1}}}%
1862   \next}

```

1863

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

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

```
1864 \newcommand*\l@dcalc@maxoftwo}[3]{%
1865   \ifnum #2>#1\relax
1866     #3=#2\relax
1867   \else
1868     #3=#1\relax
1869   \fi}
1870 \newcommand*\l@dcalc@minoftwo}[3]{%
1871   \ifnum #2<#1\relax
1872     #3=#2\relax
1873   \else
1874     #3=#1\relax
1875   \fi}
1876
```

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

```
\checkpageL 1877 \newif\ifl@dsamepage
\checkpageR 1878   \l@dsamepagetrue
1879 \newif\ifl@dpagfull
1880 \newcommand*\checkpageL}{%
1881   \l@dpagfulltrue
1882   \l@dsamepagetrue
1883   \check@goal
1884   \ifdim\pagetotal<\ledthegoal
1885     \ifnum\numpagelinesL<\l@dminpagelines
1886       \else
1887         \l@dsamepagefalse
1888         \l@dpagfullfalse
1889       \fi
1890     \else
1891       \l@dsamepagefalse
1892       \l@dpagfulltrue
1893     \fi}
1894 \newcommand*\checkpageR}{%
1895   \l@dpagfulltrue
1896   \l@dsamepagetrue
1897   \check@goal
1898   \ifdim\pagetotal<\ledthegoal
1899     \ifnum\numpagelinesR<\l@dminpagelines
```

```

1900     \else
1901         \l@dsamepagefalse
1902         \l@dpagfullfalse
1903     \fi
1904 \else
1905     \l@dsamepagefalse
1906     \l@dpagfulltrue
1907 \fi}
1908

```

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

```

1909 \newdimen\ledthegoal
1910 \ifshiftedverses
1911     \newcommand*\goalfraction{0.95}
1912 \else
1913     \newcommand*\goalfraction{0.9}
1914 \fi
1915
1916 \newcommand*\check@goal{%
1917     \ledthegoal=\goalfraction\pagegoal}
1918

```

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

```

\ifwrittenlinesL 1919 \newif\ifwrittenlinesL
                  1920 \newif\ifwrittenlinesR
                  1921

```

`\get@nextboxL` If the current box is not empty (i.e., still contains some lines) nothing is done.

`\get@nextboxR` Otherwise if and only if a synchronisation point is reached the next box is started.

```

1922 \newcommand*\get@nextboxL{%
1923     \ifvbox\namebox{1@dLcolrawbox\the\l@dpscl}% box is not empty

```

The current box is not empty; do nothing.

```

1924 \else%                                     box is empty

```

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

```

1925     \ifnum\usenamecount{1@dmaxlinesinpar\the\l@dpscl}>\@donetotallinesL
1926     \else

```

Sufficient lines have been output.

```

1927     \ifwrittenlinesL
1928     \else

```

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

```

1929         \@writelinesinparL
1930         \writtenlinesLtrue
1931     \fi
1932     \ifnum\l@dnumstartsL>\l@dpscl

```

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

```

1933     \writtenlinesLfalse
1934     \l@dcalc@maxoftwo{\the\usernamecount{l@dmaxlinesinpar\the\l@dpscL}}%
1935             {\the\@donetotallinesL}%
1936             {\usernamecount{l@dmaxlinesinpar\the\l@dpscL}}%
1937     \global\@donetotallinesL \z@
1938     \global\advance\l@dpscL \@ne
1939     \fi
1940 \fi
1941 \fi}

1942 \newcommand*{\get@nextboxR}{%
1943 \ifvbox\namebox{l@dRcolrawbox\the\l@dpscR}% box is not empty
1944 \else% box is empty
1945 \ifnum\usernamecount{l@dmaxlinesinpar\the\l@dpscR}>\@donetotallinesR
1946 \else
1947 \ifwrittenlinesR
1948 \else
1949 \@writelinesinparR
1950 \writtenlinesRtrue
1951 \fi
1952 \ifnum\l@dnumpstartsR>\l@dpscR
1953 \writtenlinesRfalse
1954 \l@dcalc@maxoftwo{\the\usernamecount{l@dmaxlinesinpar\the\l@dpscR}}%
1955             {\the\@donetotallinesR}%
1956             {\usernamecount{l@dmaxlinesinpar\the\l@dpscR}}%
1957 \global\@donetotallinesR \z@
1958 \global\advance\l@dpscR \@ne
1959 \fi
1960 \fi
1961 \fi}
1962

```

25 The End

`i/codej`

A Examples

This section presents some sample documents.

The figures are from processed versions of the files. Having latexed a file I used DVIPS to get Encapsulated PostScript, then the `epstopdf` script to get a PDF version as well, for example:

```
> latex villon
> latex villon
> latex villon
> dvips -E -o villon.eps villon % produces villon.eps
> epstopdf villon.eps          % produces villon.pdf
```

For a multipage example, DVIPS has an option to output a range of pages (`-p` for the first and `-l` (letter l) for the last). For instance, to output a single page, say page 2:

```
> latex djd17nov
> latex djd17nov
> latex djd17nov
> dvips -E -p2 -l2 -o djd17novL.eps djd17nov % produces djd17novL.eps
> epstopdf djd17novL.eps                    % produces djd17novL.pdf
```

For those who aren't fascinated by LaTeX code, I show the all the typeset results first, then the code that produced them.

I thought that limericks were peculiarly English, but this appears not to be the case. As with most limericks this one is by Anonymous.

Il y avait un jeune homme de Dijon,	There was a young man of Dijon,	1
2 Qui n'avait que peu de religion.	Who had only a little religion,	
Il dit: 'Quant à moi,	He said: 'As for me,	3
4 Je déteste tous les trois,	I detest all the three,	
Le Père, et le Fils, et le Pigeon.'	The Father, the Son, and the Pigeon.'	5

The following is verse LXXIII of François Villon's *Le Testament* (The Testament), composed in 1461.

Dieu mercy et Tacque Thibault,	Thanks to God — and to Tacque Thibaud	
2 Qui tant d'eaue froid m'a fait boire,	Who made me drink so much cold water,	2r
Mis en bas lieu, non pas en hault,	Put me underground instead of higher up	
4 Mengier d'angoisse maints poire, Enferré . . . Quant j'en ay memoire,	And made me eat such bitter fruit, In chains . . . When I think of this,	4r
6 Je Prie pour luy <i>et reliqua</i> ,	I pray for him— <i>et reliqua</i> ;	6r
Que Dieu luy doint, et voire, voire!	May God grant him (yes, by God)	
8 Ce que je pense . . . <i>et cetera</i> .	What I think . . . <i>et cetera</i> .	8r

The translation and notes are by Anthony Bonner, *The Complete Works of François Villon*, published by Bantam Books in 1960.

4 poire d'angoisse] This has a triple meaning: literally it is the fruit of the choke pear, figuratively it means 'bitter fruit', and it also refers to a torture instrument.
6 *et reliqua*] and so on

1r Tacque Thibaud] A favourite of Jean, Duc de Berry and loathed for his exactions and debauchery. Villon uses his name as an insulting nickname for Thibaud d'Auxigny, the Bishop of Orléans.

2r cold water] Can either refer to the normal prison diet of bread and water or to a common medieval torture which involved forced drinking of cold water.

1 De ecclesia S. Stephani Novimagensi

Nobilis itaque comes Otto imperio et dominio Novimagensi sibi, ut praeferretur, impignoratis et commissis proinde praeesse cupiens, anno LIII superius descripto, mense Iunio, una cum iudice, scabinis ceterisque civibus civitatis Novimagensis, pro ipsius et inhabitantium in ea necessitate, commodo et utilitate, ut ecclesia eius parochialis extra civitatem sita destrueretur et infra muros transferretur ac de novo construeretur, a reverendo patre domino Conrado de Hofsteden, archiepiscopo Coloniensi, licentiam, et a venerabilibus dominis decano et capitulo sanctorum Apostolorum Coloniensi, ipsius ecclesiae ab antiquo veris et pacificis patronis, consensus, citra tamen praeiudicium, damnum aut gravamen iurium et bonorum eorundem, impetravit.

Et exinde liberum locum eiusdem civitatis qui dicitur Hundisburg, de praelibati Wilhelmi Romanorum regis, ipsius fundi domini, consensu, ad aedificandum et consecrandum ecclesiam et coemeterium, eisdem decano et capitulo de expresso eiusdem civitatis assensu libera contradiderunt voluntate, obligantes se ipsi comes et civitas dictis decano et capitulo, quod in recompensationem illius areae infra castrum et portam, quae fuit dos ecclesiae, in qua plebanus habitare solebat—quae tunc per novum fossatum civitatis est destructa—aliam aream competentem et ecclesiae novae, ut praefertur, aedificandae satis contiguam, ipsi plebano darent et assignarent. Et desuper apud dictam ecclesiam sanctorum Apostolorum est littera sigillis ipsorum Ottonis comitis et civitatis Novimagensis sigillata.

// One additional line to show synchronization. //

3 p. 227 R 4 p. 97 N 6 p. 129 D 12 f. 72v M 13 p. 228 R 20 p. 130 D

2 proinde] primum D 5 ecclesia eius] ecclesia D: eius eius H extra civitatem *om.* H infra] intra D 6 transferretur] transferreretur NH 7 Hofsteden] Hoffstede D: Hoffsteden H Coloniensi] Colononiensi H dominis] viris H 8 Coloniensi] Coloniae H 10 iurium] virium D 11 liberum] librum H qui] quae D Hundisburg] Hundisburch D: Hundisbrug HMN: Hunsdisbrug R 12 regis] imperatoris D 13 et consecrandum *om.* H eisdem] eiusdem D 15 comes] comites D dictis *om.* H 17 tunc] nunc H 18 ut... aedificandae *om.* H 18–19 contiguam] contiguum M 19 apud *om.* H 20 est] et H littera] litteram H 21 Novimagensis] Novimagii D sigillata] sigillis communita H

6–7 William is confusing two charters that are five years apart. Permission from St. Apostles' Church in Cologne had been obtained as early as 1249. Cf. Sloet, *Oorkondenboek* nr. 707 (14 November 1249): "... nos devotionis tue precibus annuentes, ut ipsam ecclesiam faciens demoliri transferas in locum alium competentem, tibi auctoritate presentium indulgemus..." 11–19 Cf. Sloet, *Oorkondenboek* nr. 762 (June 1254)

1 St. Stephen's Church in Nijmegen

After the noble count Otto had taken in pledge the power over Nijmegen,¹ like I have written above, he wanted to protect the town. So in June 1254 he and the judge, the sheriffs and other citizens of Nijmegen obtained permission to demolish the parish church that lay outside the town walls,² to move it inside the walls and to rebuild it new. This operation was necessary and useful both for Otto himself and for the inhabitants of the town. The reverend father Conrad of Hochstaden, archbishop of Cologne,³ gave his permission. So did the reverend dean and canons of the chapter of St. Apostles' in Cologne, who had long⁴ been the true and benevolent patrons of the church—but they did not allow Otto to do anything without their knowledge, nor to infringe their rights, nor to damage their property.

And so the count and the town voluntarily gave an open space in town called Hundisburg, which was owned by the aforementioned king William, to the dean and chapter of St. Apostles' in order to build and consecrate a church and graveyard. King William approved and the town of Nijmegen explicitly expressed its assent. A new ditch was dug on property of the church near the castle and the harbour,⁵ causing the demolition of the presbytery. In compensation, the count and citizens committed themselves to giving the parish priest another suitable space close enough to the new church that was about to be built. A letter about these transactions, with the seals of count Otto and the town of Nijmegen, is kept at St. Apostles' church.⁶

// One additional line to show synchronization. //

¹In 1247 William II (1227–1256) count of Holland needed money to fight his way to Aachen to be crowned King of the Holy Roman Empire. He gave the town of Nijmegen in pledge to Otto II (1229–1271) count of Guelders.

²Since the early seventh century old St. Stephen's church had been located close to the castle, at today's Kelfkensbos square. Traces of the church and the presbytery were found during excavations in 1998–1999.

³Conrad of Hochstaden († 1261) was archbishop of Cologne in 1238–1261. Nijmegen belonged to the archdiocese of Cologne until 1559.

⁴They probably became the patrons when the chapter was established in the early eleventh century. About the church and the chapter, see Gottfried Stracke, *Köln: St. Aposteln, Stadtspuren – Denkmäler in Köln*, vol. 19, Köln: J. P. Bachem, 1992.

⁵Nowadays, the exact location of the medieval ditch—and of two Roman ones—can be seen in the pavement of Kelfkensbos square.

⁶The original letter is lost. A 15th century transcription of it is kept at the Historisches Archiv der Stadt Köln (HASTK).

1 Arma gravi numero violentaque bella parabam
 2 edere, materiā conveniente modis.
 3 Par erat inferior versus—risisse Cupido
 4 dicitur atque unum surripuisse pedem.

 5 “Quis tibi, saeve puer, dedit hoc in carmina iuris?
 6 Pieridum vates, non tua turba sumus.
 7 Quid si praeripiat flavae Vēnus arma Minervae,
 8 ventilet accensas flava Minerva faces?

 9 Quis probet in silvis Cererem regnare iugosis,
 10 lege pharetratae Virginis arva coli?
 11 Crinibus insignem quis acuta cuspide Phoebum
 12 instruat, Aoniam Marte movente lyram?

6 sumus] note lost 11 acuta] acutā (abl. abs.)

Figure 4: First left page output from `djdpoems.tex`.

1 Arma gravi numero violentaque bella parabam
 2 edere, materiā conveniente modis.
 3 Par erat inferior versus—risisse Cupido
 4 dicitur atque unum surripuisse pedem.

 5 “Quis tibi, saeve puer, dedit hoc in carmina iuris?
 6 Pieridum vates, non tua turba sumus.
 7 Quid si praeripiat flavae Vēnus arma Minervae,
 8 ventilet accensas flava Minerva faces?

 9 Quis probet in silvis Cererem regnare iugosis,
 10 lege pharetratae Virginis arva coli?
 11 Crinibus insignem quis acuta cuspide Phoebum
 12 instruat, Aoniam Marte movente lyram?

6 sumus] note lost 11 acuta] acutā (abl. abs.)

Figure 6: Second left page output from `djdpoems.tex`.

A.1 Parallel column example

This made-up example, `villon.tex`, is included to show parallel columns and how they can be interspersed in regular text. The verses are set using the `\stanza` construct, where each verse line is a chunk. The code is given below and the result is shown in Figure 1.

```

1963 (*villon)
1964 %% villon.tex Example parallel columns
1965 \documentclass{article}
1966 \addtolength{\textheight}{-10\baselineskip}
1967 \usepackage{ledmac,ledpar}
1968 %% Use r instead of R to flag right text line numbers
1969 \renewcommand{\Rlineflag}{r}
1970 %% Use the flag in the notes
1971 \let\oldBfootfmt\Bfootfmt
1972 \renewcommand{\Bfootfmt}[3]{%
1973   \let\printlines\printlinesR
1974   \oldBfootfmt{#1}{#2}{#3}}
1975 \begin{document}
1976
1977 I thought that limericks were peculiarly English, but this appears not
1978 to be the case. As with most limericks this one is by Anonymous.
1979
1980 \vspace*{\baselineskip}
1981
1982 \begin{pairs}
1983 %% no indentation
1984 \setstanzaindents{0,0,0,0,0,0,0,0,0}
1985 %% no number flag
1986 \renewcommand{\Rlineflag}{}
1987 %% draw a rule and widen the columns
1988 \setlength{\columnrulewidth}{0.4pt}
1989 \setlength{\Lcolwidth}{0.46\textwidth}
1990 \setlength{\Rcolwidth}{\Lcolwidth}
1991
1992 \begin{Leftside}
1993 %% set left text line numbering sequence
1994 \firstlinenum{2}
1995 \linenumincrement{2}
1996 \linenummargin{left}
1997 \beginnumbering
1998 \stanza
1999 Il y avait un jeune homme de Dijon, &
2000 Qui n'avait que peu de religion. &
2001 Il dit: 'Quant \'{a} moi, &
2002 Je d'\{e}teste tous les trois, &
2003 Le P\{e}re, et le Fils, et le Pigeon.' \&
2004 \endnumbering
2005 \end{Leftside}

```

```

2006
2007 \begin{Rightside}
2008 %% different right text line numbering sequence
2009 \firstlinenum{1}
2010 \linenumincrement{2}
2011 \linenummargin{right}
2012 \beginnumbering
2013 \stanza
2014 There was a young man of Dijon, &
2015 Who had only a little religion, &
2016 He said: 'As for me, &
2017 I detest all the three, &
2018 The Father, the Son, and the Pigeon.' \&
2019 \endnumbering
2020 \end{Rightside}
2021
2022 \Columns
2023 \end{pairs}
2024
2025 \vspace*{\baselineskip}
2026
2027     The following is verse \textsc{lxiii} of Fran\c{c}ois Villon's
2028 \textit{Le Testament} (The Testament), composed in 1461.
2029
2030 %% Allow for hanging indentation for long lines
2031 \setstanzaindents{1,0,0,0,0,0,0,0}
2032 %% Columns wider than the default
2033 \setlength{\Lcolwidth}{0.46\textwidth}
2034 \setlength{\Rcolwidth}{\Lcolwidth}
2035 \vspace*{\baselineskip}
2036
2037 \begin{pairs}
2038 \begin{Leftside}
2039 \firstlinenum{2}
2040 \linenumincrement{2}
2041 \linenummargin{left}
2042 \beginnumbering
2043 \stanza
2044 Dieu mercy et Tacque Thibault, &
2045 Qui tant d'eaue froid m'a fait boire, &
2046 Mis en bas lieu, non pas en hault, &
2047 Mengier d'angoisse maints \edtext{poire}{\lemma{poire d'angoisse}}%
2048 \Afootnote{This has a triple meaning: literally it is the fruit of the
2049 choke pear,
2050 figuratively it means 'bitter fruit', and it also refers to a torture
2051 instrument.}}, &
2052 Enferr'\{e} \ldots Quant j'en ay memoire, &
2053 Je Prie pour luy \edtext{\textit{et reliqua}}{\Afootnote{and so on}}, &
2054 Que Dieu luy doint, et voire, voire! &
2055 Ce que je pense \ldots \textit{et cetera}. \&

```

```

2056 \endnumbering
2057 \end{Leftside}
2058
2059 \begin{Rightside}
2060 \firstlinenum{2}
2061 \linenumincrement{2}
2062 \linenummargin{right}
2063 \beginnumbering
2064 \stanza
2065 Thanks to God --- and to \edtext{Tacque Thibaud}{%
2066   \Bfootnote{A favourite of Jean, Duc de Berry and loathed for his exactions
2067   and debauchery. Villon uses his name as an insulting nickname for
2068   Thibaud d'Auxigny, the Bishop of Orl\{e}ans.}} &
2069 Who made me drink so much \edtext{cold water}{%
2070   \Bfootnote{Can either refer to the normal prison diet of bread and
2071   water or to a common medieval torture which involved forced drinking
2072   of cold water.}}, &
2073 Put me underground instead of higher up &
2074 And made me eat such bitter fruit, &
2075 In chains \ldots When I think of this, &
2076 I pray for him---\textit{et reliqua;} &
2077 May God grant him (yes, by God) &
2078 What I think \ldots \textit{et cetera}. \&
2079 \endnumbering
2080 \end{Rightside}
2081
2082 \Columns
2083 \end{pairs}
2084
2085 \vspace*{\baselineskip}
2086
2087   The translation and notes are by Anthony Bonner,
2088 \textit{The Complete Works of Fran\c{c}ois Villon}, published by
2089 Bantam Books in 1960.
2090
2091 \end{document}
2092
2093 </villon>

```

A.2 Example parallel facing pages

This example, illustrated in Figures 2 and 3, was provided in November 2004 by Dirk-Jan Dekker of the Department of Medieval History at Radboud University, Nijmegen.

```

2094 (*djd17nov)
2095 %%% This is djd17nov.tex, a sample critical text edition
2096 %%% written in LaTeX2e with the ledmac and ledpar packages.
2097 %%% (c) 2003--2004 by Dr. Dirk-Jan Dekker,

```



```

2098 %% Radboud University, Nijmegen (The Netherlands)
2099 %% (PRW) Modified slightly by PRW to fit the ledpar manual
2100
2101 \documentclass[10pt, letterpaper, twoside]{article}
2102 \usepackage[latin,english]{babel}
2103 \usepackage{makeidx}
2104 \usepackage{ledmac,ledpar}
2105 \lineation{section}
2106 \linenummargin{inner}
2107 \sidenotemargin{outer}
2108
2109 \makeindex
2110
2111 \renewcommand{\notenumfont}{\footnotesize}
2112 \newcommand{\notetextfont}{\footnotesize}
2113
2114 %\let\Afootnoterule=\relax
2115 \let\Bfootnoterule=\relax
2116 \let\Cfootnoterule=\relax
2117
2118 \addtolength{\skip\Afootins}{1.5mm}
2119 %\addtolength{\skip\Bfootins}{1.5mm}
2120 %\addtolength{\skip\Cfootins}{1.5mm}
2121
2122 \makeatletter
2123
2124 \renewcommand*{\para@vfootnote}[2]{%
2125   \insert\csname #1footins\endcsname
2126   \bgroup
2127     \notefontsetup
2128     \interlinepenalty=\interfootnotelinepenalty
2129     \floatingpenalty=@MM
2130     \splittopskip=\ht\strutbox \splitmaxdepth=\dp\strutbox
2131     \leftskip=\z@skip \rightskip=\z@skip
2132     \l@dparsedfootspec #2\ledplinenumtrue%           new from here
2133     \ifnum\@nameuse{previous@#1@number}=\l@dparsedstartline\relax
2134       \ledplinenumfalse
2135       \fi
2136     \ifnum\previous@page=\l@dparsedstartpage\relax
2137     \else \ledplinenumtrue \fi
2138     \ifnum\l@dparsedstartline=\l@dparsedendline\relax
2139     \else \ledplinenumtrue \fi
2140     \expandafter\xdef\csname previous@#1@number\endcsname{\l@dparsedstartline}%
2141     \xdef\previous@page{\l@dparsedstartpage}%           to here
2142     \setbox0=\vbox{\hsize=\maxdimen
2143       \noindent\csname #1footfmt\endcsname#2}%
2144     \setbox0=\hbox{\unvvh0}%
2145     \dp0=0pt
2146     \ht0=\csname #1footfudgefactor\endcsname\wd0
2147     \box0

```

```

2148     \penalty0
2149   \egroup
2150 }
2151
2152 \newcommand*{\previous@A@number}{-1}
2153 \newcommand*{\previous@B@number}{-1}
2154 \newcommand*{\previous@C@number}{-1}
2155 \newcommand*{\previous@page}{-1}
2156
2157 \newcommand{\abb}[1]{#1%
2158     \let\rbracket\nobrak\relax}
2159 \newcommand{\nobrak}{\textnormal{}}
2160 \newcommand{\morenoexpands}{%
2161     \let\abb=0%
2162 }
2163
2164 \newcommand{\Aparafootfmt}[3]{%
2165     \ledsetnormalparstuff
2166     \scriptsize
2167     \notenumfont\printlines#1\enspace
2168 %   \lemmafонт#1/#2\enskip
2169     \notetextfont
2170 #3\penalty-10\hskip 1em plus 4em minus.4em\relax}
2171
2172 \newcommand{\Bparafootfmt}[3]{%
2173     \ledsetnormalparstuff
2174     \scriptsize
2175     \notenumfont\printlines#1/%
2176     \ifledplinenum
2177     \enspace
2178     \else
2179     {\hskip 0em plus 0em minus .3em}%
2180     \fi
2181     \select@lemmafонт#1/#2\rbracket\enskip
2182     \notetextfont
2183 #3\penalty-10\hskip 1em plus 4em minus.4em\relax }
2184
2185 \newcommand{\Cparafootfmt}[3]{%
2186     \ledsetnormalparstuff
2187     \scriptsize
2188     \notenumfont\printlines#1\enspace
2189 %   \lemmafонт#1/#2\enskip
2190     \notetextfont
2191 #3\penalty-10\hskip 1em plus 4em minus.4em\relax}
2192
2193 \makeatother
2194
2195 \footparagraph{A}
2196 \footparagraph{B}
2197 \footparagraph{C}

```

```

2198
2199 \let\Afootfmt=\Aparafootfmt
2200 \let\Bfootfmt=\Bparafootfmt
2201 \let\Cfootfmt=\Cparafootfmt
2202
2203 \renewcommand*{\Rlineflag}{}
2204
2205 \emergencystretch40pt
2206
2207 \author{Guillelmus de Berchen}
2208 \title{Chronicon Geldriae}
2209 \date{}
2210 \hyphenation{archi-epi-sco-po Huns-dis-brug li-be-ra No-vi-ma-gen-si}
2211 \begin{document}
2212 \begin{pages}
2213 \begin{Leftside}
2214 \beginnumbering\pstart
2215 \selectlanguage{latin}
2216 \section{De ecclesia S. Stephani Novimagensi}
2217
2218 \noindent\setline{1}
2219 Nobilis itaque comes Otto\protect\edindex{Otto II of Guelders}
2220 imperio et dominio Novimagensi sibi, ut praefertur, impignoratis
2221 et commissis
2222 \edtext{proinde}{\Bfootnote{primum D}} praeesse cupiens, anno
2223 textsc{liiii} superius descripto, mense
2224 Iu\edtext{}{\Afootnote{p.\ 227~R}}nio, una cum iudice, scabinis ceterisque
2225 civibus civitatis Novimagensis, pro ipsius et inhabitantium in ea
2226 necessitate,\edtext{}{\Afootnote{p.\ 97~N}} commodo et utilitate,
2227 ut \edtext{ecclesia eius}{\Bfootnote{ecclesia D: eius eius H}} parochialis
2228 \edtext{\abb{extra civitatem}}{\Bfootnote{\textit{om.}~H}} sita
2229 destrueretur et \edtext{infra}{\Bfootnote{intra D}} muros
2230 \edtext{transfer\edtext{}{\Afootnote{p.\ 129~D}}retur}%
2231 {\Bfootnote{transferreretur NH}}
2232 ac de novo construeretur,
2233 \edtext{a reverendo patre domino
2234 Conrado\protect\edindex{Conrad of Hochstaden} de
2235 \edtext{Hofsteden}{\Bfootnote{Hoffstede D: Hoffsteden H}}, archiepiscopo
2236 \edtext{Coloniensi}{\Bfootnote{Colononiensi H}}, licentiam%
2237 {\Cfootnote{William is confusing two charters that are five years
2238 apart. Permission from St.\ Apostles' Church in Cologne had been
2239 obtained as early as 1249. Cf.\
2240 Sloet\protect\index{Sloet van de Beele, L.A.J.W.},
2241 \textit{Oorkondenboek} nr.\ 707 (14 November 1249):
2242 ‘‘\ldots}nos devotionis tue precibus annuentes, ut ipsam ecclesiam
2243 faciens demoliri transferas in locum alium competentem, tibi
2244 auctoritate presentium indulgemus\ldots’’}}, et a venerabilibus
2245 \edtext{dominis}{\Bfootnote{viris H}} decano et capitulo sanctorum
2246 Apostolorum\protect\edindex{St. Apostles' (Cologne)}
2247 \edtext{Coloniensi}{\Bfootnote{Coloniae H}}, ipsius ecclesiae ab

```

```

2248 antiquo veris et pacificis patronis, consensum, citra tamen
2249 praeiudicium, damnum aut gravamen \edtext{iurium}{\Bfootnote{virium D}}
2250 et bonorum eorundem, impetravit.
2251 \pend
2252
2253 \pstart
2254 \edtext{Et exinde \edtext{liberum}{\Bfootnote{librum H}}
2255 locum eiusdem civitatis
2256 \edtext{qui}{\Bfootnote{quae D}} dicitur
2257 \edtext{Hundisburg}{\Bfootnote{Hundisburch D: Hundisbrug HMN:
2258 Hunsdisbrug R}}\protect\edindex{Hundisburg},
2259 de praelibati Wilhelmi\protect\edindex{William II of Holland} Romanorum
2260 \edtext{regis}{\Bfootnote{imperatoris D}}, ipsius fundi
2261 do\edtext{}{\Afootnote{f.\ 72v~M}}mini, consensu, ad aedificandum
2262 \edtext{\abb{et consecrandum}}{\Bfootnote{\textit{om.}\ H}}
2263 ecclesi\edtext{}{\Afootnote{p.\ 228~R}}am et coemeterium,
2264 \edtext{eisdem}{\Bfootnote{eiusdem D}} decano et capitulo de expresso
2265 eiusdem civitatis assensu libera contradiderunt voluntate, obligantes
2266 se ipsi \edtext{comes}{\Bfootnote{comites D}} et civitas
2267 \edtext{\abb{dictis}}{\Bfootnote{\textit{om.}\ H}} decano et capitulo,
2268 quod in recompensationem illius areae infra castrum et portam, quae
2269 fuit dos ecclesiae, in qua plebanus habitare solebat---quae
2270 \edtext{tunc}{\Bfootnote{nunc H}} per novum fossatum civitatis est
2271 destructa---aliam aream competentem et ecclesiae novae,
2272 \edtext{ut praefertur, aedificandae}{%
2273 \lemma{\abb{ut\ldots aedificandae}}\Bfootnote{\textit{om.}\ H}} satis
2274 \edtext{contiguam}{\Bfootnote{contiguum M}}, ipsi plebano darent et
2275 assignarent.}{\Cfootnote{Cf.\ Sloet, \textit{Oorkondenboek} nr.\ 762
2276 (June 1254)}} Et desuper
2277 \edtext{\abb{apud}}{\Bfootnote{\textit{om.}\ H}} dictam ecclesiam
2278 sanctorum Apostolorum \edtext{est}{\Bfootnote{et H}}
2279 \edtext{littera}{\Bfootnote{litteram H}} sigillis ipsorum
2280 Ottonis\edtext{}{\Afootnote{p.\ 130~D}} comitis et civitatis
2281 \edtext{Novimagensis}{\Bfootnote{Novimagii D}}
2282 \edtext{sigillata}{\Bfootnote{sigillis communita H}}.
2283 \pend
2284
2285 \pstart
2286 // One additional line to show synchronization. //
2287 \pend
2288 \endnumbering
2289 \end{Leftside}
2290
2291 \begin{Rightside}
2292 \sidenotemargin{right}\selectlanguage{english}
2293 \beginnumbering
2294 \pstart
2295 \addtocounter{section}{-1}%
2296 \leavevmode\section{St.\ Stephen's Church in Nijmegen}
2297

```

2298 \noindent\setline{1}%
 2299 After the noble count Otto had taken in pledge the power over
 2300 Nijmegen,\footnote{In 1247 William II\protect\index{William II of Holland}
 2301 (1227--1256) count of Holland needed money to fight his way to
 2302 Aachen\protect\index{Aachen} to be crowned King of the Holy Roman
 2303 Empire. He gave the town of Nijmegen in pledge to Otto
 2304 II\protect\index{Otto II of Guelders} (1229--1271) count of Guelders.}
 2305 like I have written above, he wanted to protect the town. So in June
 2306 1254\ledsidenote{1254} he and the judge, the sheriffs and other
 2307 citizens of Nijmegen obtained permission to demolish the parish
 2308 church that lay outside the town walls,\footnote{Since the early
 2309 seventh century old St.\ Stephen's church had been located close
 2310 to the castle, at today's
 2311 Kelfkensbos\protect\index{Kelfkensbos (Nijmegen)} square.
 2312 Traces of the church and the presbytery were found during excavations
 2313 in 1998--1999.} to move it inside the walls and to rebuild it new.
 2314 This operation was necessary and useful both for Otto himself and
 2315 for the inhabitants of the town. The reverend father Conrad of
 2316 Hochstaden, archbishop of
 2317 Cologne,\footnote{Conrad of Hochstaden ({\textdagger} 1261) was
 2318 archbishop of Cologne in 1238--1261. Nijmegen belonged to the
 2319 archdiocese of Cologne until 1559.} gave his permission. So did the
 2320 reverend dean and canons of the chapter of St.\
 2321 Apostles'\protect\index{St. Apostles' (Cologne)} in Cologne, who had
 2322 long\footnote{They probably became the patrons when the chapter was
 2323 established in the early eleventh century. About the church and the
 2324 chapter, see Gottfried Stracke\protect\index{Stracke, G.},
 2325 \textit{K}\{o}ln:\ St.\ Aposteln}, Stadtpuren -- Denkm\{a}ler in
 2326 K\{o}ln, vol.\ 19, K\{o}ln: J.\,P.\ Bachem, 1992.} been the true
 2327 and benevolent patrons of the church---but they did not allow Otto
 2328 to do anything without their knowledge, nor to infringe their rights,
 2329 nor to damage their property.
 2330 \pend
 2331
 2332 \pstart
 2333 And so the count and the town voluntarily gave an open space in town
 2334 called Hundisburg, which was owned by the aforementioned king William,
 2335 to the dean and chapter of St.\ Apostles' in order to build and
 2336 consecrate a church and graveyard. King William approved and the
 2337 town of Nijmegen explicitly expressed its assent. A new ditch was dug
 2338 on property of the church near the castle and the
 2339 harbour,\footnote{Nowadays, the exact location of the medieval
 2340 ditch---and of two Roman ones---can be seen in the pavement of
 2341 Kelfkensbos\protect\index{Kelfkensbos (Nijmegen)} square.} causing
 2342 the demolition of the presbytery. In compensation, the count and
 2343 citizens committed themselves to giving the parish priest another
 2344 suitable space close enough to the new church that was about to be
 2345 built. A letter about these transactions, with the seals of count
 2346 Otto and the town of Nijmegen, is kept at St.\ Apostles'
 2347 church.\footnote{The original letter is lost. A 15th century

```

2348 transcription of it is kept at the Historisches Archiv der
2349 Stadt K\{o}ln (HASTK).}
2350 \pend
2351
2352 \pstart
2353 // One additional line to show synchronization. //
2354 \pend
2355 \endnumbering
2356 \end{Rightside}
2357 \Pages
2358 \end{pages}
2359
2360 %%%%%%%%%%%
2361 \printindex
2362 \end{document}
2363 %%%%%%%%%%%
2364
2365 </djd17nov>

```

A.3 Example poetry on parallel facing pages

This example, illustrated in Figures 4 to 7, was originally provided in November 2004 by Dirk-Jan Dekker for an earlier version of `ledpar`. I have updated it, and also extended it to show the difference between the `\stanza` command and the `astanza` environment. `\stanza` is used for the first pair of pages and `astanza` for the second pair. Note the definition of `\endstanzaextra` to give a short line after each stanza.

```

2366 (*djdpoems)
2367 %% djdpoms.tex  example parallel verses on facing pages
2368 \documentclass{article}
2369 \usepackage{ledmac,ledpar}
2370 \addtolength{\textheight}{-15\baselineskip}
2371
2372 \maxchunks{24} % default value = 10
2373 \setstanzaindents{6,0,1,0,1}
2374
2375 \newcommand{\longdash}{-----}
2376
2377 \footparagraph{A} % for left pages
2378 \footparagraph{B} % for right pages
2379 \firstlinenum{1}
2380 \linenumincrement{1}
2381
2382 \let\oldBfootfmt\Bfootfmt
2383 \renewcommand{\Bfootfmt}[3]{%
2384   \let\printlines\printlinesR
2385   \oldBfootfmt{#1}{#2}{#3}}

```

```

2386
2387 \begin{document}
2388
2389 \newcommand{\interstanza}{\pstart\centering\longdash\skipnumbering\pend}
2390
2391 \begin{pages}
2392 \begin{Leftside}
2393 \def\endstanzaextra{\interstanza}
2394 \beginnumbering
2395
2396 \stanza
2397 Arma gravi numero violentaque bella parabam &
2398 edere, materi\={a} conveniente modis. &
2399 Par erat inferior versus---rissime Cupido &
2400 dicitur atque unum surripuisse pedem. \&
2401
2402 \stanza
2403 ‘‘Quis tibi, saeve puer, dedit hoc in carmina iuris? &
2404 Pieridum vates, non tua turba \edtext{sumus}{\Afootnote{note lost}}. &
2405 Quid si praeripiat flavae V\{e}nus arma Minervae, &
2406 ventilet accensas flava Minerva faces? \&
2407
2408 \stanza
2409 Quis probet in silvis Cererem regnare iugosis, &
2410 lege pharetratae Virginis arva coli? &
2411 Crinibus insignem quis \edtext{acuta}{\Afootnote{acut\={a} (abl.\ abs.)}}
2412 cuspide Phoebum &
2413 instruat, Aoniam Marte movente lyram? \&
2414 \endnumbering
2415 \end{Leftside}
2416
2417 \begin{Rightside}
2418 \def\endstanzaextra{\interstanza}
2419 \beginnumbering
2420 \firstlinenum{1}
2421 \linenumincrement{1}
2422 \setstanzaindents{6,0,1,0,1,0}
2423
2424 \stanza
2425 I was preparing to sing of weapons and violent wars, &
2426 in heavy numbers, with the subject matter suited to the verse measure. &
2427 The even lines were as long as the odd ones, but Cupid laughed, &
2428 they said, and he stole away one foot.\footnote{I.e., the even lines,
2429 which were hexameters (with six feet) became pentameters
2430 (with five feet).} \&
2431
2432 \stanza
2433 ‘‘O cruel boy, who gave you the right over poetry? &
2434 We poets belong to the Pierides,\footnote{Muses} we are not your folk. &
2435 \edlabel{beginparadox}What if Venus should seize away the arms of

```

```

2436 Minerva with the golden hair, &
2437 if Minerva with the golden hair should fan alight the kindled torch
2438 of love? \&
2439
2440 \stanza
2441 Who would approve of Ceres\footnote{Ceres was the Roman goddess of
2442 the harvest.} reigning on the woodland ridges, &
2443 and of land tilled under the law of the Maid with the
2444 quiver\footnote{By '\textit{Virgo}' ('Virgin') Ovid means Diana, the
2445 Roman goddess of the hunt.}? &
2446 Who would provide Phoebus with his beautiful hair with a sharp-pointed
2447 spear, &
2448 while Mars stirs the \edtext{Aonian}{\Bfootnote{Mount Parnassus,
2449 where the Muses live, is located in Aonia.}}
2450 lyre?\edlabel{endparadox}\footnote{Lines
2451 \xlineref{beginparadox}--\xlineref{endparadox} show some paradoxical
2452 situations that would occur if the gods didn't stay with their own
2453 business.} \&
2454 \endnumbering
2455 \end{Rightside}
2456
2457 \Pages
2458 \end{pages}
2459
2460 \begin{pages}
2461 \begin{Leftside}
2462 \def\endstanzaextra{\interstanza}
2463 \beginnumbering
2464
2465 \begin{astanza}
2466 Arma gravi numero violentaque bella parabam &
2467 edere, materi\={a} conveniente modis. &
2468 Par erat inferior versus---risisse Cupido &
2469 dicitur atque unum surripuisse pedem. \&
2470 \end{astanza}
2471
2472 \begin{astanza}
2473 'Quis tibi, saeve puer, dedit hoc in carmina iuris? &
2474 Pieridum vates, non tua turba \edtext{sumus}{\Afootnote{note lost}}. &
2475 Quid si praeripiat flavae V\ue}nus arma Minervae, &
2476 ventilet accensas flava Minerva faces? \&
2477 \end{astanza}
2478
2479 \begin{astanza}
2480 Quis probet in silvis Cererem regnare iugosis, &
2481 lege pharetratae Virginis arva coli? &
2482 Crinibus insignem quis \edtext{acuta}{\Afootnote{acut\={a} (abl.\ abs.)}}
2483 cuspide Phoebum &
2484 instruat, Aoniam Marte movente lyram? \&
2485 \end{astanza}

```



```

2486
2487 \endnumbering
2488 \end{Leftside}
2489
2490 \begin{Rightside}
2491 \def\endstanzaextra{\interstanza}
2492 \beginnumbering
2493 \firstlinenum{1}
2494 \linenumincrement{1}
2495 \setstanzaindents{6,0,1,0,1,0}
2496
2497 \begin{astanza}
2498 I was preparing to sing of weapons and violent wars, &
2499 in heavy numbers, with the subject matter suited to the verse measure. &
2500 The even lines were as long as the odd ones, but Cupid laughed, &
2501 they said, and he stole away one foot.\footnote{I.e., the even lines,
2502 which were hexameters (with six feet) became pentameters
2503 (with five feet).} \&
2504 \end{astanza}
2505
2506 \begin{astanza}
2507 ‘O cruel boy, who gave you the right over poetry? &
2508 We poets belong to the Pierides,\footnote{Muses} we are not your folk. &
2509 \edlabel{beginparadox}What if Venus should seize away the arms of
2510 Minerva with the golden hair, &
2511 if Minerva with the golden hair should fan alight the kindled torch
2512 of love? \&
2513 \end{astanza}
2514
2515 \begin{astanza}
2516 Who would approve of Ceres\footnote{Ceres was the Roman goddess of the
2517 harvest.} reigning on the woodland ridges, &
2518 and of land tilled under the law of the Maid with the
2519 quiver\footnote{By ‘\textit{Virgo}’ (‘Virgin’) Ovid means Diana,
2520 the Roman goddess of the hunt.}? &
2521 Who would provide Phoebus with his beautiful hair with a sharp-pointed
2522 spear, &
2523 while Mars stirs the \edtext{Aonian}{\Bfootnote{Mount Parnassus, where
2524 the Muses live, is located in Aonia.}}
2525 lyre?\edlabel{endparadox}\footnote{Lines
2526 \xlineref{beginparadox}--\xlineref{endparadox} show some paradoxical
2527 situations that would occur if the gods didn’t stay with their
2528 own business.} \&
2529 \end{astanza}
2530
2531 \endnumbering
2532 \end{Rightside}
2533
2534 \Pages
2535 \end{pages}

```

```
2536
2537 \end{document}
2538
2539 </djdpoems>
```

References

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

Index

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

Symbols	
<code>\&</code> 1452, 1455, 1456, 1460, 1477, 1491, 2003, 2018, 2055, 2078, 2400, 2406, 2413, 2430, 2438, 2453, 2469, 2476, 2484, 2503, 2512, 2528	<code>\@donetotallinesR</code> 832, 895, 898, 1738, 1945, 1955, 1957
<code>\@M</code> 1466	<code>\@insertR</code> 1124–1126, 1139–1141
<code>\@MM</code> 2129	<code>\@l</code> 251, 564
<code>\@adv</code> 328, 595, 596	<code>\@l@dttempcnta</code> 401, 403, 405, 406, 410, 412, 414, 415, 944, 983, 984, 986, 988, 991, 992, 1009–1013, 1015, 1022, 1027, 1031, 1039, 1044, 1048, 1080, 1083, 1085, 1089
<code>\@afterindentfalse</code> 706	<code>\@l@dttempcntb</code> 144, 146, 148, 974, 975, 1022, 1027, 1031, 1039, 1044, 1048, 1072, 1076, 1089, 1097–1099, 1101, 1346, 1348, 1350, 1403–1405, 1407, 1519–1523, 1527–1530, 1534–1537
<code>\@arabic</code> 185, 186	<code>\@l@reg</code> 300
<code>\@astanza@line</code> 1476, 1482, 1485	<code>\@l@regR</code> 251
<code>\@auxout</code> 1308, 1318, 1565	<code>\@lab</code> 520, 1302, 1311, 1332
<code>\@chapter</code> 707	<code>\@lock</code> 926
<code>\@cs@linesinparL</code> 1721, 1831	<code>\@lockR</code> 60, 273, 275, 277, 290, 435, 451, 452, 454, 455, 483, 484, 486, 911, 950, 952, 953, 955, 1036, 1053, 1055, 1057
<code>\@cs@linesinparR</code> 1721, 1831	<code>\@lopL</code> 542, 1858
<code>\@cs@linesonpageL</code> 1729, 1789, 1844	
<code>\@cs@linesonpageR</code> 1729, 1789, 1844	
<code>\@donereallinesL</code> 832, 860, 1685, 1687, 1735	
<code>\@donereallinesR</code> 832, 894, 1690, 1692, 1737	
<code>\@donetotallinesL</code> 832, 861, 864, 1736, 1925, 1935, 1937	

- `\@lopR` 542, 1861
`\@nameuse` 1431, 1435, 2133
`\@nobeakfalse` 744, 773
`\@nobeaktrue` 742, 746, 771, 775
`\@oldnobeak` 742, 744, 771, 773, 809, 822
`\@pend` 535, 1685
`\@pendR` 535, 1690
`\@pstartfalse` 1660
`\@pstartstrue` 1660
`\@ref` 507, 568, 572
`\@ref@reg` 533
`\@schapter` 707
`\@set` 360, 602, 603
`\@tag` 633, 650, 1148, 1152, 1162, 1166,
1176, 1180, 1190, 1194, 1204,
1208, 1219, 1223, 1233, 1237,
1247, 1251, 1261, 1265, 1275, 1279
`\@temp` 1556
`\@templ@d` 1394, 1395
`\@writelinesinparL` .. 1642, 1683, 1929
`\@writelinesinparR` .. 1643, 1683, 1949
`\@writelinesonpageL` . 1758, 1760, 1857
`\@writelinesonpageR` . 1779, 1781, 1857
`\@xloop` 1137
- `_` 2224, 2226, 2230, 2238, 2239, 2241,
2261–2263, 2267, 2273, 2275,
2277, 2280, 2296, 2309, 2320,
2325, 2326, 2335, 2346, 2411, 2482
- A**
- `\abb` 2157,
2161, 2228, 2262, 2267, 2273, 2277
`\absline@num` 394, 408, 427, 918
`\absline@numR` ... 58, 202, 253, 256,
259, 391, 399, 420, 439, 473,
501, 512, 903, 936, 937, 974, 1123
`\actionlines@list`
..... 243, 246, 394, 408, 427
`\actionlines@listR`
..... 206, 221, 235, 238, 391,
399, 420, 439, 473, 501, 996, 999
`\actions@list` . 247, 395, 415, 429, 431
`\actions@listR`
. 206, 222, 239, 392, 406, 422,
424, 441, 450, 475, 482, 502, 1000
`\add@inserts` 853
`\add@inserts@nextR` 1112
`\add@insertsR` 887, 1112
- `\add@penaltiesL` 859, 1133
`\add@penaltiesR` 893, 1133
`\addtocontents` 1566–1568
`\addtocounter`
. 912, 928, 1448, 1749, 1770, 2295
`\addtolength` ... 1966, 2118–2120, 2370
`\advanceline` 594, 625
`\affixline@num` 851
`\affixline@numR` 885, 1006
`\affixside@note` 854
`\affixside@noteR` 888, 1393
`\Afootfmt` 2199
`\Afootins` 2118
`\Afootnote` 1144, 2048,
2053, 2224, 2226, 2230, 2261,
2263, 2280, 2404, 2411, 2474, 2482
`\Afootnoterule` 2114
`\Aparafootfmt` 2164, 2199
`\apptocmd` 1596
`\araw@textfalse` 1671
`\araw@texttrue` 1671
astanza (environment) 7, 1458
`\AtBeginDocument` ... 1328, 1541, 1576
`\author` 2207
- B**
- `\ballast@count` 934, 939
`\bbl@main@language` 1592, 1593
`\bbl@set@language` 1583, 1584
`\beginnumbering` 6, 36, 723,
749, 1997, 2012, 2042, 2063,
2214, 2293, 2394, 2419, 2463, 2492
`\beginnumberingR` ... 49, 102, 723, 778
`\Bfootfmt` 1971, 1972, 2200, 2382, 2383
`\Bfootins` 2119
`\Bfootnote` 1158, 2066, 2070,
2222, 2227–2229, 2231, 2235,
2236, 2245, 2247, 2249, 2254,
2256, 2257, 2260, 2262, 2264,
2266, 2267, 2270, 2273, 2274,
2277–2279, 2281, 2282, 2448, 2523
`\Bfootnoterule` 2115
`\box` 2147
`\Bparafootfmt` 2172, 2200
`\bypage@Rfalse` 122, 134
`\bypage@Rtrue` 122, 130
- C**
- `\c@ballast` 939
`\c@firstlinenumR` 153, 1078

- `\c@firstsublinenumR` [157](#), 1073
`\c@linenumincrementR` [153](#), 1078
`\c@page` 564, 1809, 1815, 1818, 1825
`\c@sublinenumincrementR` [157](#), 1073
`\centering` 2389
`\Cfootfmt` 2201
`\Cfootins` 2120
`\Cfootnote` [1158](#), 2237, 2275
`\Cfootnoterule` 2116
`\ch@ck@l@ckR` [1006](#)
`\ch@cksub@l@ckR` [1006](#)
`\ch@cksub@lockR` 1074
`\chapter` 693, 694, 702
`\chapterinpages` [686](#), 694, 704
`\chardef` 1455
`\check@goal` 1883, 1897, [1909](#)
`\check@pstarts`
 1619, 1644, [1660](#), 1715, 1723, 1731
`\checkpageL` 1742, 1755, [1877](#)
`\checkpageR` 1764, 1776, [1877](#)
`\checkraw@text`
 1623, 1640, [1671](#), 1739, 1785
`\clear@doublepage` 1821
`\clearl@dleftpage` 1763, [1807](#)
`\clearl@drightpage` 1784, [1807](#)
`\clear@toevenpage` [1807](#)
`\clear@to@devenpage` 1704, [1807](#)
`\closeout` 555, 559
`\columnrulewidth` 4, [1655](#), 1988
`\Columns` 3, [1607](#), 2022, 2082
`\columnseparator` 4, 1637, [1655](#)
`\countLline` [827](#), 839
`\countRline` [827](#), 873
`\Cparafootfmt` 2185, 2201
`\critext` [630](#)
- D**
- `\date` 2209
`\DeclareOption` 7
Dekker, Dirk-Jan 78, 84
`\Dfootnote` [1158](#)
`\dimen` 581, 582, 586–588, 592
`\divide` 1011
`\do@actions` 919
`\do@actions@fixedcodeR` 943
`\do@actions@nextR` 943
`\do@actionsR` 904, [943](#)
`\do@ballast` 920
`\do@ballastR` 905, [934](#)
`\do@lineL` [837](#), 1627, 1631, 1746
`\do@lineLhook` [843](#), [868](#)
`\do@lineR` [871](#), 1628, 1633, 1767
`\do@lineRhook` [868](#), 877
`\do@lockoff` [470](#)
`\do@lockoffL` 494
`\do@lockoffR` [470](#)
`\do@lockon` [435](#)
`\do@lockonL` 467
`\do@lockonR` [435](#)
`\documentclass` 1965, 2101, 2368
`\dp` 2130, 2145
`\dummy@ref` 516
- E**
- `\edfont@info` 669, 672, 678, 681
`\edindex` 2219, 2234, 2246, 2258, 2259
`\edlabel` [1300](#), 2435, 2450, 2509, 2525
`\edtext` [647](#), 2047,
 2053, 2065, 2069, 2222, 2224,
 2226–2230, 2233, 2235, 2236,
 2245, 2247, 2249, 2254, 2256,
 2257, 2260–2264, 2266, 2267,
 2270, 2272, 2274, 2277–2282,
 2404, 2411, 2448, 2474, 2482, 2523
`\Efootnote` [1158](#)
`\emergencystretch` 2205
`\empty` 76, 79, 235, 243, 641, 658,
 667, 676, 757, 786, 996, 1077,
 1085, 1114–1116, 1127, 1138,
 1303, 1312, 1832, 1838, 1845, 1851
`\end@lemmas` 641, 642, 658, 659
`\endashchar` 1290
`\endgraf` 805, 818, 1613, 1707
`\endline@num` 523, 529
`\endlock` [614](#), 1464, 1473, 1478
`\endnumbering` 6, 39, [69](#), 106,
 724, 2004, 2019, 2056, 2079,
 2288, 2355, 2414, 2454, 2487, 2531
`\endnumberingR` 52, [69](#), 91, 101, 114, 724
`\endpage@num` 522, 529
`\endstanzaextra`
 1480, 2393, 2418, 2462, 2491
`\endsub` [581](#)
`\endsubline@num` 524, 530
`\enskip` 2168, 2181, 2189
`\enspace` 2167, 2177, 2188
environments:
 `astanza` 7, [1458](#)
 `Leftside` 5, [709](#)
 `pages` 4, [686](#)

- pairs 3, 686
 Rightside 5, 721
 \everyhbox 1444
 \extensionchars .. 47, 66, 97, 111, 119
- F**
- \f@x@l@cksR 1006
 \first@linenum@out@Rfalse .. 550, 556
 \first@linenum@out@Rtrue 550
 \firstlinenum 5, 162, 1994,
 2009, 2039, 2060, 2379, 2420, 2493
 \firstsublinenum 5, 162
 \fix@page 296, 303
 \flag@end 565, 646, 663
 \flag@start 565, 638, 655
 \floatingpenalty 2129
 \flush@notes 1646, 1794
 \flush@notesR 1136, 1647, 1795
 \footnote
 . 2300, 2308, 2317, 2322, 2339,
 2347, 2428, 2434, 2441, 2444,
 2450, 2501, 2508, 2516, 2519, 2525
 \footnotesize 2111, 2112
 \footparagraph . 2195–2197, 2377, 2378
 \fullstop . 198, 1287, 1289, 1291, 1293
- G**
- \get@linelistfile 231
 \get@nextboxL 1754, 1922
 \get@nextboxR 1775, 1922
 \getline@numL 848, 917
 \getline@numR 882, 902
 \getlinesfrompagelistL
 1727, 1787, 1844
 \getlinesfrompagelistR
 1728, 1788, 1844
 \getlinesfromparlistL ... 1719, 1831
 \getlinesfromparlistR ... 1720, 1831
 \gl@p 238, 239,
 246, 247, 642, 659, 671, 680,
 999, 1000, 1120, 1124, 1139,
 1306, 1315, 1835, 1841, 1848, 1854
 \goalfraction 4, 1909
- H**
- \hangingsymbol 9, 1446
 \hb@xt@ ... 850, 856, 863, 884, 890,
 897, 1635, 1750, 1752, 1771, 1773
 \hsize 767, 796,
 1635, 1750, 1752, 1771, 1773, 2142
- \hyphenation 2210
- I**
- \if@filesw 1564
 \if@firstcolumn 1091, 1397
 \if@nobreak 741, 770
 \if@pstarts ... 1620, 1660, 1716, 1732
 \ifaraw@text ... 1625, 1671, 1741, 1786
 \ifbypage@ 319
 \ifbypage@R 122, 309, 978
 \ifdim 582, 586, 588,
 592, 1750, 1771, 1811, 1884, 1898
 \iffirst@linenum@out@R ... 550, 554
 \ifl@d@dash 1290
 \ifl@d@elin 1292, 1293
 \ifl@d@esl 1293
 \ifl@d@pnum 1287, 1291
 \ifl@d@ssub 1289
 \ifl@d@pagefull 1757, 1778, 1877
 \ifl@d@paging 9
 \ifl@d@pairing 9, 73
 \ifl@d@samelang 1552, 1626
 \ifl@d@samepage 1745, 1766, 1877
 \ifl@d@skipnumber 1068
 \ifl@d@usedbabel 1550
 \ifledplinenum 1288, 2176
 \ifledRcol 9, 145, 167,
 171, 175, 179, 218, 233, 297,
 306, 330, 344, 361, 378, 390,
 398, 419, 464, 491, 500, 509,
 566, 576, 583, 589, 595, 602,
 610, 615, 619, 624, 635, 652,
 666, 1146, 1160, 1174, 1188,
 1202, 1217, 1231, 1245, 1259,
 1273, 1301, 1333, 1347, 1358,
 1370, 1382, 1417, 1430, 1587, 1597
 \ifnoteschanged@ 83
 \ifnumberedpar@ ... 751, 780, 801,
 814, 1145, 1159, 1173, 1187,
 1201, 1216, 1230, 1244, 1258,
 1272, 1357, 1369, 1381, 1416, 1429
 \ifnumbering 37, 747, 798
 \ifnumberingR
 31, 50, 70, 93, 125, 776, 811
 \ifodd 1101, 1407, 1809, 1815, 1818, 1825
 \ifpst@rtedL 32, 755
 \ifpst@rtedR 32, 784
 \ifshiftedverses . 5, 1748, 1769, 1910
 \ifsublines@
 . 196, 285, 329, 362, 369, 400,

- 409, 421, 428, 440, 474, 528,
530, 906, 921, 985, 1071, 1335, 1339
- `\ifvbox` 840, 874, 1675, 1678, 1923, 1943
- `\ifwrittenlinesL` 1919, 1927
- `\ifwrittenlinesR` 1920, 1947
- `\initnumbering@reg` 45
- `\insert` 2125
- `\insert@count` 506, 572, 636,
653, 1153, 1167, 1181, 1195,
1209, 1224, 1238, 1252, 1266,
1280, 1365, 1377, 1389, 1424, 1437
- `\insert@countR` 507, 568, 635,
652, 1149, 1163, 1177, 1191,
1205, 1220, 1234, 1248, 1262,
1276, 1361, 1373, 1385, 1420, 1433
- `\insertlines@listR`
.... 76, 206, 220, 512, 1116, 1120
- `\inserts@list`
.. 756, 1152, 1166, 1180, 1194,
1208, 1223, 1237, 1251, 1265,
1279, 1364, 1376, 1388, 1423, 1436
- `\inserts@listR`
.. 785, 1111, 1114, 1124, 1138,
1139, 1148, 1162, 1176, 1190,
1204, 1219, 1233, 1247, 1261,
1275, 1360, 1372, 1384, 1419, 1432
- `\interfootnotelinepenalty` 2128
- `\interlinepenalty` 1466, 2128
- `\interstanza`
.... 2389, 2393, 2418, 2462, 2491
- L**
- `\l@d@nums` 669, 672, 678,
681, 1148, 1152, 1162, 1166,
1176, 1180, 1190, 1194, 1204,
1208, 1219, 1223, 1233, 1237,
1247, 1251, 1261, 1265, 1275, 1279
- `\l@d@set` 377, 610, 611
- `\l@dbbl@set@language` 1561, 1584
- `\l@dbfnote` 1415
- `\l@dc@maxchunks` 762, 764,
791, 793, 1509, 1519, 1527, 1534
- `\l@dcalc@maxoftwo`
.... 1721, 1864, 1934, 1954
- `\l@dcalc@minoftwo` .. 1729, 1789, 1864
- `\l@dcalcnun` 1006
- `\l@dchecklang` 1554, 1624
- `\l@dchset@num` 252, 255, 377
- `\l@dcsnote` 1356
- `\l@dcsnotetext`
.... 1395, 1398, 1400, 1408, 1410
- `\l@demptyd@ta` 844, 878
- `\l@dend@stuff` 48, 67, 98, 112, 120
- `\l@dgetline@margin` 143
- `\l@dgetsidenote@margin` 1345
- `\l@dld@ta` 852, 886, 1092, 1104
- `\l@dleftbox`
.. 824, 849, 863, 1636, 1750, 1752
- `\l@dlinenumR` 188
- `\l@dlsn@te` 855, 889
- `\l@dlsnote` 1356
- `\l@dmake@labels` 1319
- `\l@dmake@labelsR` 1309, 1322
- `\l@dminpagelines`
.... 1694, 1730, 1790, 1885, 1899
- `\l@dnumpstartsL` . 41, 761, 762, 764,
766, 1513, 1545, 1608, 1609,
1651, 1663, 1701, 1702, 1799, 1932
- `\l@dnumpstartsR` . 54, 790, 791, 793,
795, 1513, 1546, 1608, 1609,
1652, 1666, 1701, 1702, 1800, 1952
- `\l@doldbbl@set@language` 1583
- `\l@doldselectlanguage` 1582, 1586, 1591
- `\l@dpagefullfalse` 1877
- `\l@dpagefulltrue` 1877
- `\l@dpaddingfalse` 11, 688, 701
- `\l@dpaddingtrue` 696
- `\l@dpairingfalse` 9, 690, 700
- `\l@dpairingtrue` 687, 695
- `\l@dparsedendline` 2138
- `\l@dparsedstartline` . 2133, 2138, 2140
- `\l@dparsedstartpage` 2136, 2141
- `\l@dparsefootspec` 2132
- `\l@dpscL` 840, 845, 1515, 1547, 1617,
1621, 1649, 1663, 1675, 1711,
1717, 1722, 1725, 1733, 1797,
1923, 1925, 1932, 1934, 1936, 1938
- `\l@dpscR` 874, 879, 1516, 1548,
1618, 1622, 1650, 1666, 1678,
1712, 1718, 1726, 1734, 1798,
1943, 1945, 1952, 1954, 1956, 1958
- `\l@drd@ta` 856, 890, 1094, 1102
- `\l@drightbox`
.. 824, 883, 897, 1638, 1771, 1773
- `\l@drsn@te` 857, 891
- `\l@drsnote` 1356
- `\l@dsamelangfalse` 1552, 1555
- `\l@dsamelangtrue` 1552, 1558
- `\l@dsamepagefalse` 1877

- \l@dsamepagetrue 1877
- \l@dsetupmaxlinecounts .. 1526, 1543
- \l@dsetuprawboxes 1518, 1542
- \l@dskipnumberfalse 1069
- \l@dskipnumbertrue 966
- \l@dunhbox@line 856, 890
- \l@dusedbabelfalse 1550, 1579
- \l@dusedbabeltrue 1550, 1581
- \l@duselanguage
 - 1571, 1630, 1632, 1743, 1765
- \l@dzeromaxlinecounts ... 1526, 1544
- \l@dzeropenalties 804, 817, 1612, 1706
- \l@pscl 1515
- \l@pscr 1515
- \label@refs
 - 1304, 1306, 1309, 1313, 1315, 1319
- \labelref@list 1312, 1315, 1340
- \labelref@listR 1298, 1303, 1306, 1336
- \languagename .. 1562, 1563, 1565–1568
- \last@page@num 317, 323
- \last@page@numR 303
- \lastbox 847, 881
- \lastskip 581, 587
- \Lcolwidth 3, 4, 14, 697, 767,
 - 850, 863, 1989, 1990, 2033, 2034
- \ldots 2052,
 - 2055, 2075, 2078, 2242, 2244, 2273
- \led@err@BadLeftRightPstarts ...
 - 22, 1609, 1702
- \led@err@LeftOnRightPage ... 25, 1819
- \led@err@LineationInNumbered ... 126
- \led@err@NumberingNotStarted ... 87
- \led@err@NumberingShouldHaveStarted
 - 100
- \led@err@NumberingStarted 38, 51
- \led@err@PendNoPstart 802, 815
- \led@err@PendNotNumbered ... 799, 812
- \led@err@PstartInPstart ... 752, 781
- \led@err@PstartNotNumbered . 748, 777
- \led@err@RightOnLeftPage ... 25, 1826
- \led@err@TooManyPstarts 19, 763, 792
- \led@mess@NotesChanged 84
- \led@mess@SectionContinued
 - 96, 110, 118
- \led@warn@BadAction 968
- \led@warn@BadAdvancelineLine 347, 353
- \led@warn@BadAdvancelineSubline .
 - 333, 339
- \led@warn@BadLineation 136
- \led@warn@BadSetline 600
- \led@warn@BadSetlinenum 608
- \led@warn@DuplicateLabel 1324
- \ledllfill 856, 890
- \ledmac@error 20, 23, 26, 28
- \ledplinenumfalse 2134
- \ledplinenumtrue ... 2132, 2137, 2139
- \ledRcolfalse 13, 710, 733
- \ledRcoltrue 722
- \ledrlfill 856, 890
- \ledsavedprintlines 7, 1285
- \ledsetnormalparstuff 2165, 2173, 2186
- \ledsidenote 2306
- \ledstrutL 1750, 1752, 1804
- \ledstrutR 1771, 1773, 1804
- \ledthegoal 1884, 1898, 1909
- \leftlinenumR 188, 1092, 1104
- Leftside (environment) 5, 709
- \Leftsidehook 714, 716
- \Leftsidehookend 715, 716
- \lemma 2047, 2273
- \lemmafnt 2168, 2189
- \line@list 676, 680
- \line@list@stuff 47, 111
- \line@list@stuffR ... 66, 97, 119, 552
- \line@listR . 79, 206, 219, 530, 667, 671
- \line@margin 148
- \line@marginR 141, 1097
- \line@num 320, 351, 352,
 - 354, 372, 383, 384, 412, 927, 1338
- \line@numR . 59, 195, 202, 257, 291,
 - 310, 345, 346, 348, 365, 379,
 - 380, 403, 523, 527, 913, 979,
 - 988, 1076, 1078, 1080, 1081, 1334
- \lineation 730, 2105
- \lineationR 124, 730
- \linenum@out 571, 579, 584, 590, 596,
 - 603, 611, 616, 620, 1311, 1685, 1858
- \linenum@outR
 - . 549, 555, 557, 559, 560, 564,
 - 567, 577, 583, 589, 595, 602,
 - 610, 615, 619, 624, 1302, 1690, 1861
- \linenumberlist 1077, 1081
- \linenumincrement .. 5, 162, 1995,
 - 2010, 2040, 2061, 2380, 2421, 2494
- \linenummargin
 - 141, 1996, 2011, 2041, 2062, 2106
- \linenumr@p 1288, 1292, 1334, 1338
- \linenumrepR 185, 195
- \linenumsep 190, 192

`\linesinpar@listL`
 [211](#), 227, 537, 1832, 1835
`\linesinpar@listR`
 [211](#), 223, 540, 1838, 1841
`\linesonpage@listL` 228, 544, 1845, 1848
`\linesonpage@listR` 224, 547, 1851, 1854
`\list@clear`
 . 219–224, 227, 228, 230, 756, 785
`\list@clearing@reg` 226
`\list@create`
 ... 206–209, 211–213, 1111, 1298
`\lock@disp` 1038, 1042, 1047
`\lock@off` 461, 462, [470](#), 619, 620
`\lock@on` 615, 616
`\longdash` 2375, 2389

M

`\manageparhangingsymbol` 838, 872, 1442
`\maxchunks` [3](#), [1509](#), 2372
`\maxdimen` 2142
`\maxlinesinpar@list` [211](#), 230
`\memorydump` [6](#), 713, 727
`\memorydumpL` [105](#), 713
`\memorydumpR` [105](#), 727
`\message` 46, 65
`\morenoexpands` 2160
`\mpAfootnote` [1215](#)
`\mpBfootnote` [1215](#)
`\mpCfootnote` [1215](#)
`\mpDfootnote` [1215](#)
`\mpEfootnote` [1215](#)
`\mpvAfootnote` 1218, 1222, 1227
`\mpvBfootnote` 1232, 1236, 1241
`\mpvCfootnote` 1246, 1250, 1255
`\mpvDfootnote` 1260, 1264, 1269
`\mpvEfootnote` 1274, 1278, 1283
`\multiply` 1012

N

`\n@num` [498](#), 624
`\n@num@reg` 504
`\namebox` 840, 845, 874,
 879, [1493](#), 1675, 1678, 1923, 1943
`\NeedsTeXFormat` 2
`\new@line` 856
`\new@lineR` [563](#), 890
`\newbox` 738, 824, 825, 1494
`\newcounter` 153, 155, 157, 159

`\newif` 5, 10,
 12, 31, 33, 122, 550, 1550, 1552,
 1660, 1671, 1877, 1879, 1919, 1920
`\newnamebox` [1493](#), 1521, 1522
`\newnamecount` [1504](#), 1529
`\newwrite` 549
`\next@action` 247
`\next@actionline` 244, 246
`\next@actionlineR`
 236, 238, 937, 975, 997, 999
`\next@actionR` 239,
 938, 976, 977, 982, 983, 991, 1000
`\next@insert` 757
`\next@insertR`
 786, 1115, 1118, 1120, 1123, 1127
`\next@page@num` 324, 395
`\next@page@numR` 63, 260, 262, 314, 392
`\no@expands` 632, 649
`\nobrak` 2158, 2159
`\noindent` 2143, 2218, 2298
`\normal@pars` 72, 760, 789
`\normalbfnoteX` [1428](#)
`\notefontsetup` 2127
`\notenumfont` ... 2111, 2167, 2175, 2188
`\noteschanged@true`
 77, 80, 668, 677, 1117
`\notetextfont` .. 2112, 2169, 2182, 2190
`\num@lines` 805, 1613, 1707
`\num@linesR` [737](#), 818, 1614, 1708
`\numberedpar@true` 768, 797
`\numberingRfalse` 71
`\numberingRtrue` 56, 91, 115
`\numberingtrue` 43, 107
`\numlabfont` 195
`\numpagelinesL`
 [1694](#), 1747, 1758, 1762, 1885
`\numpagelinesR`
 [1694](#), 1768, 1779, 1783, 1899

O

`\oldBfootfmt` ... 1971, 1974, 2382, 2385
`\oldchapter` 693, 702
`\one@line` 845, 847, 856
`\one@lineR` [737](#), 879, 881, 890
`\openout` 557, 560

P

`\page@action` 261, [389](#), 517
`\page@num` 242, 322, 1405

- `\page@numR` 215, 234, 312, 522, 527, 977, 1099
`\pagegoal` 1917
`\Pages` 4, 1698, 2357, 2457, 2534
`pages` (environment) 4, 686
`\pagetotal` 1811, 1884, 1898
`pairs` (environment) 3, 686
`\par@line` 806, 1615, 1709
`\par@lineR` 737, 819, 1616, 1710
`\para@vfootnote` 2124
`\pausenumbering` 725
`\pausenumberingR` 90, 725
`\pend` .. 5, 712, 729, 753, 1479, 2251,
2283, 2287, 2330, 2350, 2354, 2389
`\pendL` 712, 798
`\pendR` 729, 782, 811
`\prevgraf`
. 805, 818, 1613, 1614, 1707, 1708
`\previous@A@number` 2152
`\previous@B@number` 2153
`\previous@C@number` 2154
`\previous@page` 2136, 2141, 2155
`\printindex` 2361
`\printlines`
1296, 1973, 2167, 2175, 2188, 2384
`\printlinesR` 7, 1285, 1973, 2384
`\ProcessOptions` 8
`\protected@write` ... 1308, 1318, 1565
`\ProvidesPackage` 3
`\pst@rtedLfalse` 32, 42
`\pst@rtedLtrue` 108, 758
`\pst@rtedRfalse` 34, 55, 74
`\pst@rtedRtrue` 94, 116, 787
`\pstart` 5, 20, 24, 711, 728, 1481, 2214,
2253, 2285, 2294, 2332, 2352, 2389
`\pstartL` 711, 740
`\pstartR` 728, 740
- R**
- `\rbracket` 2158, 2181
`\Rcolwidth` 3, 4,
14, 698, 796, 884, 897, 1990, 2034
`\read@linelist` 217, 553
`\rem@inder` 1081, 1083–1085
`\resumenummering` 726
`\resumenummeringR` 90, 726
`\rightlinenumR` 188, 1094, 1102
`Rightside` (environment) 5, 721
`\Rightsidehook` 716, 731
`\Rightsidehookend` 716, 734
- `\rlap` 1094, 1102
`\Rlineflag` 7, 183, 195,
1288, 1292, 1326, 1969, 1986, 2203
`\rule` 1656
- S**
- `\sc@n@list` 1082, 1084
`\secdef` 707
`\section@num` 44, 46, 47, 109–111
`\section@numR`
... 29, 57, 65, 66, 95–97, 117–119
`\select@language` ... 1563, 1565–1568
`\select@lemmfont` 2181
`\selectlanguage` ... 1571, 2215, 2292
`\set@line` 634, 651, 665
`\set@line@action`
... 254, 358, 367, 374, 397, 519
`\setl@dlp@rbox` 1398, 1410
`\setl@drp@rbox` 1400, 1408
`\setline` 598, 2218, 2298
`\setlinenum` 606
`\setnamebox` 766, 795, 1493
`\setprintlines` 1286
`\setstanzaindents`
... 1984, 2031, 2373, 2422, 2495
`\shiftedversesfalse` 6
`\shiftedversestrue` 7
`\showlemma` 640, 657
`\sidenote@margin` 1350, 1354
`\sidenote@marginR` 1343, 1403
`\sidenotemargin` ... 1343, 2107, 2292
`\skip` 2118–2120
`\skip@lockoff` 462, 470
`\skipnumbering` 7, 623, 2389
`\skipnumbering@reg` 627
`\smash` 1656
`\splitmaxdepth` 2130
`\splittopskip` 842, 876, 2130
`\stanza` ... 1998, 2013, 2043, 2064,
2396, 2402, 2408, 2424, 2432, 2440
`\stanza@count` 1461, 1475, 1486
`\stanza@hang` 1463, 1488
`\stanzaindentbase` 1486
`\startlock` 614
`\startstanzahook` 1459
`\startsub` 581
`\sub@action` 270, 418, 518
`\sub@change` 64, 264, 265, 271
`\sub@lock` 922

- `\sub@lockR` 61, 279, 281, 283,
 286, 436, 442, 443, 445, 446,
 476, 477, 479, 907, 958, 960,
 961, 963, 1019, 1059, 1061, 1063
`\sub@off` 589, 590
`\sub@on` 583, 584
`\subline@num` 197, 320, 337,
 338, 340, 370, 410, 923, 929, 1339
`\subline@numR`
 . 198, 202, 287, 291, 310, 331,
 332, 334, 363, 401, 524, 528,
 908, 914, 979, 986, 1072, 1073, 1335
`\sublinenumincrement` 5, 162
`\sublinenumr@p` . 1289, 1293, 1335, 1339
`\sublinenumrepR` 185, 198
`\sublines@false` 62, 268, 948
`\sublines@true` 266, 946
`\sublock@disp` 1021, 1025, 1030
`\symplinenum` 1288
`\sza@penalty` 1470, 1474
- T**
- `\textdagger` 2317
`\textheight` 1966, 2370
`\textit` 2028, 2053, 2055, 2076, 2078,
 2088, 2228, 2241, 2262, 2267,
 2273, 2275, 2277, 2325, 2444, 2519
`\textnormal` 2159
`\textsc` 2027, 2223
`\textwidth` 15, 17, 697, 698, 1989, 2033
`\theledlanguageL` 1556, 1571, 1630, 1743
`\theledlanguageR` 1556, 1571, 1632, 1765
`\thepage` 564, 1309, 1319
`\thr@@` 445, 454, 477, 484, 953, 961
`\title` 2208
`\topskip` 1811
- U**
- `\unhbox` 1498,
 1636, 1638, 1750, 1752, 1771, 1773
`\unhnamebox` 1493
`\unvbox` 847, 881, 1500
`\unvnamebox` 1493
`\unvxh` 2144
`\usernamecount`
 . 1462, 1469, 1504, 1536, 1722,
 1925, 1934, 1936, 1945, 1954, 1956
- `\usepackage` 1967, 2102–2104, 2369
- V**
- `\vAfootnote` 1147, 1151, 1156
`\value` 1445
`\vbadness` 841, 875
`\vbfnoteX` 1431, 1435
`\vBfootnote` 1161, 1165, 1170
`\vbox` 766, 795, 2142
`\vCfootnote` 1175, 1179, 1184
`\vDfootnote` 1189, 1193, 1198
`\vEfootnote` 1203, 1207, 1212
`\vl@dbfnote` 1418, 1422
`\vl@dcsnote` 1383, 1387
`\vl@dlsnote` 1359, 1363
`\vl@drsnote` 1371, 1375
`\vsplit` 845, 879
- W**
- `\wd` 856, 890, 2146
`\writtenlinesLfalse` 1713, 1933
`\writtenlinesLtrue` 1930
`\writtenlinesRfalse` 1714, 1953
`\writtenlinesRtrue` 1950
- X**
- `\x@lemma` 642–644, 659–661
`\xlineref` 2451, 2526
`\xpg@main@language` 1601, 1602
`\xpg@set@language` 1596, 1600
`\xright@appenditem`
 391, 392, 394, 395, 399,
 406, 408, 415, 420, 422, 424,
 427, 429, 431, 439, 441, 450,
 473, 475, 482, 501, 502, 512,
 526, 537, 540, 544, 547, 1147,
 1151, 1161, 1165, 1175, 1179,
 1189, 1193, 1203, 1207, 1218,
 1222, 1232, 1236, 1246, 1250,
 1260, 1264, 1274, 1278, 1334,
 1338, 1359, 1363, 1371, 1375,
 1383, 1387, 1418, 1422, 1431, 1435
- Z**
- `\z@skip` 2131
`\zz@@@` 1304, 1313

Change History

v0.1			
General: First public release	1	
v0.2			
General: Added section of babel re-	lated code	54
Fix babel problems	1	
\Columns: Added \l@dchecklang	and \l@duselanguage	to	
\Columns	57	
\Pages: Added \l@duselanguage	to \Pages	60
v0.3			
General: Reorganize for ledarab	..	1	
\affixline@numR: Changed	\affixline@numR to match new		
ledmac	39	
\do@actions@nextR: Used	\do@actions@fixedcode	in	
\do@actionsR	38	
\do@lineL: Added \do@lineLhook	to \do@lineL	35
Simplified \do@lineL by using	macros for some common code	35	
\do@lineR: Changed \do@lineR	similarly to \do@lineL	36
\do@lineRhook: Added \do@lineLhook	and \do@lineRhook	36
Leftside: Added hooks into Left-	side environment	31
\flag@end: Removed extraneous	spaces from \flag@end	27
\ifledRcol: Moved \ifl@dpairing	to ledmac	11
\ifpst@rtedR: Moved \ifpst@rtedL	to ledmac	12
\l@dlinenumR: Simplified	\leftlinenumR and \rightlinenumR	by introducing \l@dlinenumR	16
\l@dnumpstartsR: Moved	\l@dnumpstartsL to ledmac	..	53
\ledsavedprintlines: Simplified	\printlinesR by using		
\setprintlines	46	
	\ledstrutR: Added \ledtrutL and		
	\ledstrutR	62
	\normalbfnoteX: Removed		
	extraneous spaces from		
	\normalbfnoteX	50
	\Pages: Added \ledstrutL to		
	\Pages	60
	Added \ledstrutR to \Pages	..	61
	\Rightsidehookend: Added		
	\Leftsidehook, \Leftsidehookend,		
	\Rightsidehook and \Rightsidehookend	31
	\sublinenumrepR: Added		
	\linenumrepR and \sublinenumrepR	16
v0.3a			
General: Minor \linenummargin	fix	1
\line@marginR: Don't just	set \line@marginR	in	
\linenummargin	15	
v0.3b			
General: Improved parallel page	balancing	1
\Pages: Added \l@dminpagelines	calculation for succeeding page	pairs
			61
v0.3c			
General: Compatibilty with Poly-	glossia	1
v0.4			
General: No more ledparpatch. All	patches are now in the main	file.
			1
v0.5			
General: Corrections about	\section and other titles in	numbered sections
			1
v0.6			
General: Be able to us \chapter	in	parallel pages.
			1
v0.7			
General: Option 'shiftedverses'			

which make there is no blank
between two parallel verses with
inequal length. 1
v0.8
General: Possibility to have a sym-

bol on each hanging of verses,
like in the french typogra-
phy. Redefine the commande
`\hangingsymbol` to define the
character. 1