

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.

To report bugs, please go to **ledmac**'s GitHub page and click "New Issue": <https://github.com/maieul/ledmac/issues/>. You must open an account with github.com to access my page (maieul/ledmac). GitHub accounts are free for open-source users.

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 and paragraphs	8
7	Verse	9
8	Implementation overview	12

*This file (**ledpar.dtx**) has version number v0.9.1, last revised 2011/10/02.

[†]herries dot press at earthlink dot net

[‡]maieul at maieul dot net

9 Preliminaries	12
9.1 Messages	13
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	44
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	49
16 Cross referencing	49
17 Side notes	51
18 Familiar footnotes	52
19 Verse	53
20 Naming macros	54
21 Counts and boxes for parallel texts	55
22 Fixing babel	56
23 Parallel columns	58
24 Parallel pages	61
25 The End	69

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

A Examples	70
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`. The `\maxchunks` must be called in the preamble.

TeX has a limited number of boxes; if you get an error message along the lines of 'no room for a new box', then 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 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 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.

\firstlinenum 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. **\linenumincrement** Following **\firstlinenum{<num>}** the first line number will be *<num>*, and following **\sublinenumincrement{<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 and paragraphs

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

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

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

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

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

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

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

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

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

```
\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}}
```

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

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

7 Verse

If you are typesetting verse with `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 `\sza000`' 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}[2011/10/02 v0.9.1 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
\ifl@dpaging is also set TRUE if we are doing parallel pages. \ifledRcol is set TRUE if we
\ifledRcol are doing the right hand text. \ifl@dpairing is defined in ledmac.
9 \l@dpairingfalse
10 \newif\ifl@dpaging
11 \l@dpagingfalse
12 \ledRcolfalse

\Lcolwidth The widths of the left and right parallel columns (or pages).
\Rcolwidth 13 \newdimen\Lcolwidth
14 \Lcolwidth=0.45\textwidth
15 \newdimen\Rcolwidth
16 \Rcolwidth=0.45\textwidth
17

```

9.1 Messages

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

```

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

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

\led@err@LeftOnRightPage
\led@err@RightOnLeftPage 24 \newcommand*{\led@err@LeftOnRightPage}{%
25 \ledmac@error{The left page has ended on a right page}{\@ehc}}
26 \newcommand*{\led@err@RightOnLeftPage}{%
27 \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.

```

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

```

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

```
30 \pst@rtedLfalse
31 \newif\ifpst@rtedR
32 \pst@rtedRfalse
33
```

`\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.

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

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

```
47 \newcommand*\beginnumberingR}{%
48   \ifnumberingR
49     \led@err@NumberingStarted
50   \endnumberingR
51 \fi
52 \global\l@dnumpstartsR \z@
53 \global\pst@rtedRfalse
54 \global\numberingRtrue
55 \global\advance\section@numR \@ne
56 \global\absline@numR \z@
57 \global\line@numR \z@
58 \global\@lockR \z@
59 \global\sub@lockR \z@
60 \global\sublines@false
61 \global\let\next@page@numR\relax
62 \global\let\sub@change\relax
63 \message{Section \the\section@numR R }%
64 \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
65 \l@dend@stuff
66 \setcounter{pstartR}{1}
67 }
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` 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` 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` 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` 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.

```
\linenumincrement 162 \providecommand*{\firstlinenum}{}
\firstsublinenum 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@dtempcnta=-\subline@numR
402     \else
403       \l@dtempcnta=-\line@numR
404     \fi
405     \advance\l@dtempcnta by -5000\relax
406     \xright@appenditem{\the\l@dtempcnta}\to\actions@listR
407   \else
408     \xright@appenditem{\the\absline@num}\to\actionlines@list
409     \ifsublines@
410       \l@dtempcnta=-\subline@num
411     \else
412       \l@dtempcnta=-\line@num
413     \fi
414     \advance\l@dtempcnta by -5000\relax
415     \xright@appenditem{\the\l@dtempcnta}\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@lockon}{%
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.

```
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     \gl@p\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). In this environment, there are two text in parallel on 2 pages. To prevent chapters starting on a lefthand page, the `\chapter` command is redefined to not clear pages.


```

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

```

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

```

709 \newenvironment{Leftside}{%
710   \ledRcolfalse
711   \let\beginnumbering\beginnumbering\setcounter{pstartL}{1}
712   \let\pstart\pstartL
713   \let\thepstart\thepstartL
714   \let\pend\pendL
715   \let\memorydump\memorydumpL
716   \Leftsidehook
717 }{\Leftsidehookend}

```

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

```

\Leftsidehookend
\Rightsidehook 718 \newcommand*\Leftsidehook{}
\Rightsidehookend 719 \newcommand*\Leftsidehookend{}
720 \newcommand*\Rightsidehook{}
721 \newcommand*\Rightsidehookend{}
722

```

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.

```

723 \newenvironment{Rightside}{%
724   \ledRcoltrue
725   \let\beginnumbering\beginnumberingR
726   \let\endnumbering\endnumberingR
727   \let\pausenumbering\pausenumberingR
728   \let\resumenumbering\resumenumberingR

```

```

729 \let\memorydump\memorydumpR
730 \let\thepstart\thepstartR
731 \let\pstart\pstartR
732 \let\pend\pendR
733 \let\lineation\lineationR
734 \Rightsidehook
735 }{
736 \ledRcolfalse
737 \Rightsidehookend
738 }
739

```

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
\par@lineR

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

```

740 \newcount\num@linesR
741 \newbox\one@lineR
742 \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.

```

743

```

```

744 \newcounter{pstartL}
745 \renewcommand{\thepstartL}{\bf\@arabic\c@pstartL}. }
746 \newcounter{pstartR}
747 \renewcommand{\thepstartR}{\bf\@arabic\c@pstartR}. }
748
749 \newcommand*{\pstartL}{
750 \if@nobreak
751 \let\@oldnobreak\@nobreaktrue
752 \else
753 \let\@oldnobreak\@nobreakfalse
754 \fi
755 \@nobreaktrue
756 \ifnumbering \else
757 \led@err@PstartNotNumbered
758 \beginnumbering
759 \fi
760 \ifnumberedpar@
761 \led@err@PstartInPstart
762 \pend
763 \fi

```

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

```

764 \ifpst@rtedL\else
765 \list@clear{\inserts@list}%
766 \global\let\next@insert=\empty
767 \global\pst@rtedLtrue
768 \fi
769 \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.

```

770 \global\advance\l@dnumpstartsL \@ne
771 \ifnum\l@dnumpstartsL>\l@dc@maxchunks
772 \led@err@TooManyPstarts
773 \global\l@dnumpstartsL=\l@dc@maxchunks
774 \fi
775 \global\setnamebox{\l@dLcolrawbox\the\l@dnumpstartsL}=\vbox\bgroup\ifautopar\else\ifnumberpstart\the
776 \hspace=\l@colwidth
777 \numberedpar@true}

778 \newcommand*{\pstartR}{
779 \if@nobreak
780 \let\@oldnobreak\@nobreaktrue
781 \else
782 \let\@oldnobreak\@nobreakfalse
783 \fi
784 \@nobreaktrue
785 \ifnumberingR \else
786 \led@err@PstartNotNumbered
787 \beginnumberingR

```

```

788 \fi
789 \ifnumberedpar@
790   \led@err@PstartInPstart
791   \pendR
792 \fi
793 \ifpst@rtedR\else
794   \list@clear{\inserts@listR}%
795   \global\let\next@insertR=\empty
796   \global\pst@rtedRtrue
797 \fi
798 \begingroup\normal@pars
799 \global\advance\l@dnumpstartsR \@ne
800 \ifnum\l@dnumpstartsR>\l@dc@maxchunks
801   \led@err@TooManyPstarts
802   \global\l@dnumpstartsR=\l@dc@maxchunks
803 \fi
804 \global\setnamebox{\l@dc@rawbox\the\l@dnumpstartsR}=\vbox\bgroup\ifautopar\else\ifnumb
805   \hsize=\Rcolwidth
806 \numberedpar@true}

```

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

```

807 \newcommand*{\pendL}{\ifnumbering \else
808   \led@err@PendNotNumbered
809 \fi
810 \ifnumberedpar@ \else
811   \led@err@PendNoPstart
812 \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.

```

813 \l@dzeroopenalties
814 \endgraf\global\num@lines=\prevgraf\egroup
815 \global\par@line=0

```

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

```

816 \endgroup
817 \ignorespaces
818 \@oldnobreak
819 \ifnumberpstart
820 \addtocounter{pstartL}{1}
821 \fi}
822

```

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

```

823 \newcommand*{\pendR}{\ifnumberingR \else
824   \led@err@PendNotNumbered

```

```

825 \fi
826 \ifnumberedpar@ \else
827 \led@err@PendNoPstart
828 \fi
829 \l@dzeropenalties
830 \endgraf\global\num@linesR=\prevgraf\egroup
831 \global\par@lineR=0
832 \endgroup
833 \ignorespaces
834 \@oldnbreak
835 \ifnumberpstart
836 \addtocounter{pstartR}{1}
837 \fi
838 }
839

```

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 analagously for a line
`\l@drightbox` of right text.

```

840 \newbox\l@dleftbox
841 \newbox\l@drightbox
842

```

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

```

\countRline 843 \newcount\countLline
844 \countLline \z@
845 \newcount\countRline
846 \countRline \z@
847

```

`\@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 848 \newcount\@donereallinesL
849 \newcount\@donetotallinesL
850 \newcount\@donereallinesR
851 \newcount\@donetotallinesR
852

```

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

```

853 \newcommand*{\do@lineL}{%
854 \manageparhangingsymbol
855 \advance\countLline \@ne
856 \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscl}%

```

```

857 {\vbadness=10000
858 \splittopskip=\z@
859 \do@lineLhook
860 \l@emptyd@ta
861 \global\setbox\one@line=\vsplit\namebox{\l@dLcolrawbox\the\l@dpscL}
862 to\baselineskip}%
863 \unvbox\one@line \global\setbox\one@line=\lastbox
864 \getline@numL
865 \setbox\l@dleftbox
866 \hb@xt@ \Lcolwidth{%
867 \affixline@num
868 \l@dld@ta
869 \add@inserts
870 \affixside@note
871 \l@dlsn@te
872 {\ledllfill\hb@xt@ \wd\one@line{\new@line\l@dunhbox@line{\one@line}}\ledrlfill\l@drd@
873 \l@drsn@te
874 }}%
875 \add@penaltiesL
876 \global\advance\@donereallinesL\@ne
877 \global\advance\@donetotallinesL\@ne
878 \else
879 \setbox\l@dleftbox \hb@xt@ \Lcolwidth{\hspace*\Lcolwidth}}%
880 \global\advance\@donetotallinesL\@ne
881 \fi}
882
883

```

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

```

\do@lineRhook 884 \newcommand*{\do@lineLhook}{}
885 \newcommand*{\do@lineRhook}{}
886

```

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

```

887 \newcommand*{\do@lineR}{%
888 \manageparhangingsymbol
889 \advance\countRline \@ne
890 \ifvbox\namebox{\l@dRcolrawbox\the\l@dpscR}%
891 {\vbadness=10000
892 \splittopskip=\z@
893 \do@lineRhook
894 \l@emptyd@ta
895 \global\setbox\one@lineR=\vsplit\namebox{\l@dRcolrawbox\the\l@dpscR}
896 to\baselineskip}%
897 \unvbox\one@lineR \global\setbox\one@lineR=\lastbox
898 \getline@numR
899 \setbox\l@drightbox
900 \hb@xt@ \Rcolwidth{%
901 \affixline@numR

```

```

902 \l@dld@ta
903 \add@insertsR
904 \affixside@noteR
905 \l@dlsn@te
906 {\ledllfill\hb@xt@ \wd\one@lineR{\new@lineR\l@dunhbox@line{\one@lineR}}\ledrlfill\l@dtd@ta%
907 \l@drsn@te
908 }}%
909 \add@penaltiesR
910 \global\advance\@donereallinesR\@ne
911 \global\advance\@donetotallinesR\@ne
912 \else
913 \setbox\l@drightbox \hb@xt@ \Rcolwidth{\hspace*\Rcolwidth}}
914 \global\advance\@donetotallinesR\@ne
915 \fi}
916
917

```

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.

```

918 \newcommand*\getline@numR{%
919 \global\advance\absline@numR \@ne
920 \do@actionsR
921 \do@ballastR
922 \ifsublines@
923 \ifnum\sub@lockR<\tw@
924 \global\advance\subline@numR \@ne
925 \fi
926 \else
927 \ifnum\@lockR<\tw@
928 \addtocounter{hbox}{10}%
929 \global\advance\line@numR \@ne
930 \global\subline@numR \z@
931 \fi
932 \fi}
933 \newcommand*\getline@numL{%
934 \global\advance\absline@num \@ne
935 \do@actions
936 \do@ballast
937 \ifsublines@
938 \ifnum\sub@lock<\tw@
939 \global\advance\subline@num \@ne
940 \fi
941 \else
942 \ifnum\@lock<\tw@
943 \global\advance\line@num \@ne
944 \addtocounter{hbox}{10}%
945 \global\subline@num \z@

```

```

946     \fi
947   \fi}
948
949

```

`\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.

```

950 \newcommand*{\do@ballastR}{\global\ballast@count=\z@
951   \begingroup
952     \advance\absline@numR \@ne
953     \ifnum\next@actionlineR=\absline@numR
954       \ifnum\next@actionR>-1001
955         \global\advance\ballast@count by -\c@ballast
956       \fi
957     \fi
958   \endgroup}

```

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

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

```

959 \newcommand*{\do@actions@fixedcodeR}{%
960   \ifcase\@l@dttempcnta%
961     \or%                               % 1001
962       \global\sublines@true
963     \or%                               % 1002
964       \global\sublines@false
965     \or%                               % 1003
966       \global\@lockR=\@ne
967     \or%                               % 1004
968       \ifnum\@lockR=\tw@
969         \global\@lockR=\thr@@
970       \else
971         \global\@lockR=\z@
972       \fi
973     \or%                               % 1005
974       \global\sub@lockR=\@ne
975     \or%                               % 1006
976       \ifnum\sub@lockR=\tw@
977         \global\sub@lockR=\thr@@
978       \else
979         \global\sub@lockR=\z@
980       \fi
981     \or%                               % 1007
982       \l@dskipnumbertrue
983     \else
984       \led@warn@BadAction
985   \fi}

```



```

986
987
988 \newcommand*{\do@actionsR}{%
989   \global\let\do@actions@nextR=\relax
990   \@l@dttempcntb=\absline@numR
991   \ifnum\@l@dttempcntb<\next@actionlineR\else
992     \ifnum\next@actionR>-1001\relax
993       \global\page@numR=\next@actionR
994       \ifbypage@R
995         \global\line@numR \z@ \global\subline@numR \z@
996       \fi
997     \else
998       \ifnum\next@actionR<-4999\relax % 9/05 added relax here
999         \@l@dttempcnta=-\next@actionR
1000         \advance\@l@dttempcnta by -5001\relax
1001         \ifsublines@
1002           \global\subline@numR=\@l@dttempcnta
1003         \else
1004           \global\line@numR=\@l@dttempcnta
1005         \fi
1006       \else
1007         \@l@dttempcnta=-\next@actionR
1008         \advance\@l@dttempcnta by -1000\relax
1009         \do@actions@fixedcodeR
1010       \fi
1011     \fi
1012     \ifx\actionlines@listR\empty
1013       \gdef\next@actionlineR{1000000}%
1014     \else
1015       \gl@p\actionlines@listR\to\next@actionlineR
1016       \gl@p\actions@listR\to\next@actionR
1017       \global\let\do@actions@nextR=\do@actionsR
1018     \fi
1019   \fi
1020   \do@actions@nextR}
1021

```

14.4 Line number printing

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

```

\ch@cksub@l@ckR 1022
\ch@ck@l@ckR 1023 \providecommand*{\l@dcalcnun}[3]{%
\fx@l@cksR 1024   \ifnum #1 > #2\relax
\affixline@numR 1025     \@l@dttempcnta = #1\relax
1026     \advance\@l@dttempcnta by -#2\relax
1027     \divide\@l@dttempcnta by #3\relax
1028     \multiply\@l@dttempcnta by #3\relax
1029     \advance\@l@dttempcnta by #2\relax
1030   \else

```

```

1031     \@l@tempcnta=#2\relax
1032 \fi}
1033
1034 \newcommand*{\ch@cksub@l@ckR}{%
1035 \ifcase\sub@lockR
1036 \or
1037     \ifnum\sublock@disp=\@ne
1038         \@l@tempcntb \z@ \@l@tempcnta \@ne
1039     \fi
1040 \or
1041     \ifnum\sublock@disp=\tw@
1042     \else
1043         \@l@tempcntb \z@ \@l@tempcnta \@ne
1044     \fi
1045 \or
1046     \ifnum\sublock@disp=\z@
1047         \@l@tempcntb \z@ \@l@tempcnta \@ne
1048     \fi
1049 \fi}
1050
1051 \newcommand*{\ch@ck@l@ckR}{%
1052 \ifcase\@lockR
1053 \or
1054     \ifnum\lock@disp=\@ne
1055         \@l@tempcntb \z@ \@l@tempcnta \@ne
1056     \fi
1057 \or
1058     \ifnum\lock@disp=\tw@
1059     \else
1060         \@l@tempcntb \z@ \@l@tempcnta \@ne
1061     \fi
1062 \or
1063     \ifnum\lock@disp=\z@
1064         \@l@tempcntb \z@ \@l@tempcnta \@ne
1065     \fi
1066 \fi}
1067
1068 \newcommand*{\f@x@l@ckR}{%
1069 \ifcase\@lockR
1070 \or
1071     \global\@lockR \tw@
1072 \or \or
1073     \global\@lockR \z@
1074 \fi
1075 \ifcase\sub@lockR
1076 \or
1077     \global\sub@lockR \tw@
1078 \or \or
1079     \global\sub@lockR \z@
1080 \fi}

```

```

1081
1082
1083 \newcommand*{\affixline@numR}{%
1084 \ifl@dskipnumber
1085   \global\l@dskipnumberfalse
1086 \else
1087   \ifsublines@
1088     \l@dtempcntb=\subline@numR
1089     \l@dcalcnnum{\subline@numR}{\c@firstsublinenumR}{\c@sublinenumincrementR}%
1090     \ch@cksub@lockR
1091   \else
1092     \l@dtempcntb=\line@numR
1093     \ifx\linenumberlist\empty
1094       \l@dcalcnnum{\line@numR}{\c@firstlinenumR}{\c@linenumincrementR}%
1095     \else
1096       \l@dtempcnta=\line@numR
1097       \edef\rem@inder{\linenumberlist,\number\line@numR,}%
1098       \edef\sc@n@list{\def\noexpand\sc@n@list
1099         ###1,\number\l@dtempcnta,###2|\def\noexpand\rem@inder{###2}}}%
1100       \sc@n@list\expandafter\sc@n@list\rem@inder|
1101       \ifx\rem@inder\empty\advance\l@dtempcnta\@ne\fi
1102     \fi
1103     \ch@ck@l@ckR
1104   \fi
1105   \ifnum\l@dtempcnta=\l@dtempcntb
1106     \if@twocolumn
1107       \if@firstcolumn
1108         \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1109       \else
1110         \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1111       \fi
1112     \else
1113       \l@dtempcntb=\line@marginR
1114       \ifnum\l@dtempcntb>\@ne
1115         \advance\l@dtempcntb by\page@numR
1116       \fi
1117       \ifodd\l@dtempcntb
1118         \gdef\l@drd@ta{\rlap{\rightlinenumR}}}%
1119       \else
1120         \gdef\l@dld@ta{\llap{\leftlinenumR}}}%
1121       \fi
1122     \fi
1123   \fi
1124   \f@x@l@cksR
1125 \fi}
1126

```

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.

```
1127 \list@create{\inserts@listR}
```

`\add@insertsR` The right text version.

```
\add@inserts@nextR 1128 \newcommand*{\add@insertsR}{%
1129   \global\let\add@inserts@nextR=\relax
1130   \ifx\inserts@listR\empty \else
1131     \ifx\next@insertR\empty
1132       \ifx\insertlines@listR\empty
1133         \global\noteschanged@true
1134         \gdef\next@insertR{100000}%
1135       \else
1136         \gl@p\insertlines@listR\to\next@insertR
1137       \fi
1138     \fi
1139     \ifnum\next@insertR=\absline@numR
1140       \gl@p\inserts@listR\to\@insertR
1141       \@insertR
1142       \global\let\@insertR=\undefined
1143       \global\let\next@insertR=\empty
1144       \global\let\add@inserts@nextR=\add@insertsR
1145     \fi
1146   \fi
1147   \add@inserts@nextR}
1148
```

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 not really relevant. I think that it is likely with parallel text that there is no real need to add back any penalties; even if there was, they would have to match across the left and right lines. So, I end up with the following.

```

1149 \newcommand*{\add@penaltiesL}{-}
1150 \newcommand*{\add@penaltiesR}{-}
1151

```

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.

```

1152 \newcommand*{\flush@notesR}{%
1153   \xloop
1154   \ifx\inserts@listR\empty \else
1155     \glp\inserts@listR\to\@insertR
1156     \@insertR
1157     \global\let\@insertR=\undefined
1158   \repeat}
1159

```

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.

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`.

`\Bfootnote` We need similar commands for the other footnote series.

```

1201 \fi\ignorespaces}
1202 \renewcommand*{\Dfootnote}[1]{%
1203 \ifnumberedpar@
1204 \ifledRcol
1205 \xright@appenditem{\noexpand\VDfootnote{D}%
1206 {{\l@d@nums}{\@tag}{#1}}}\to\inserts@listR
1207 \global\advance\insert@countR \@ne
1208 \else
1209 \xright@appenditem{\noexpand\VDfootnote{D}%
1210 {{\l@d@nums}{\@tag}{#1}}}\to\inserts@list
1211 \global\advance\insert@count \@ne
1212 \fi
1213 \else
1214 \VDfootnote{D}{{0|0|0|0|0|0|0|0}{#1}}%
1215 \fi\ignorespaces}
1216 \renewcommand*{\Efootnote}[1]{%
1217 \ifnumberedpar@
1218 \ifledRcol
1219 \xright@appenditem{\noexpand\VEfootnote{E}%
1220 {{\l@d@nums}{\@tag}{#1}}}\to\inserts@listR
1221 \global\advance\insert@countR \@ne
1222 \else
1223 \xright@appenditem{\noexpand\VEfootnote{E}%
1224 {{\l@d@nums}{\@tag}{#1}}}\to\inserts@list
1225 \global\advance\insert@count \@ne
1226 \fi
1227 \else
1228 \VEfootnote{E}{{0|0|0|0|0|0|0|0}{#1}}%
1229 \fi\ignorespaces}
1230

```

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

```

\mpBfootnote 1231 \renewcommand*{\mpAfootnote}[1]{%
\mpCfootnote 1232 \ifnumberedpar@
\mpDfootnote 1233 \ifledRcol
\mpEfootnote 1234 \xright@appenditem{\noexpand\mpvAfootnote{A}%
1235 {{\l@d@nums}{\@tag}{#1}}}\to\inserts@listR
1236 \global\advance\insert@countR \@ne
1237 \else
1238 \xright@appenditem{\noexpand\mpvAfootnote{A}%
1239 {{\l@d@nums}{\@tag}{#1}}}\to\inserts@list
1240 \global\advance\insert@count \@ne
1241 \fi
1242 \else
1243 \mpvAfootnote{A}{{0|0|0|0|0|0|0|0}{#1}}%
1244 \fi\ignorespaces}
1245 \renewcommand*{\mpBfootnote}[1]{%
1246 \ifnumberedpar@

```

```

1247 \ifledRcol
1248   \xright@appenditem{\noexpand\mpvBfootnote{B}%
1249                       {\l@d@nums}{\@tag}{#1}}\to\inserts@listR
1250   \global\advance\insert@countR \@ne
1251 \else
1252   \xright@appenditem{\noexpand\mpvBfootnote{B}%
1253                       {\l@d@nums}{\@tag}{#1}}\to\inserts@list
1254   \global\advance\insert@count \@ne
1255 \fi
1256 \else
1257   \mpvBfootnote{B}{\{0|0|0|0|0|0|0\}}{#1}}%
1258 \fi\ignorespaces}
1259 \renewcommand*{\mpCfootnote}[1]{%
1260   \ifnumberedpar@
1261   \ifledRcol
1262     \xright@appenditem{\noexpand\mpvCfootnote{C}%
1263                         {\l@d@nums}{\@tag}{#1}}\to\inserts@listR
1264     \global\advance\insert@countR \@ne
1265   \else
1266     \xright@appenditem{\noexpand\mpvCfootnote{C}%
1267                         {\l@d@nums}{\@tag}{#1}}\to\inserts@list
1268     \global\advance\insert@count \@ne
1269   \fi
1270   \else
1271     \mpvCfootnote{C}{\{0|0|0|0|0|0|0\}}{#1}}%
1272   \fi\ignorespaces}
1273 \renewcommand*{\mpDfootnote}[1]{%
1274   \ifnumberedpar@
1275   \ifledRcol
1276     \xright@appenditem{\noexpand\mpvDfootnote{D}%
1277                         {\l@d@nums}{\@tag}{#1}}\to\inserts@listR
1278     \global\advance\insert@countR \@ne
1279   \else
1280     \xright@appenditem{\noexpand\mpvDfootnote{D}%
1281                         {\l@d@nums}{\@tag}{#1}}\to\inserts@list
1282     \global\advance\insert@count \@ne
1283   \fi
1284   \else
1285     \mpvDfootnote{D}{\{0|0|0|0|0|0|0\}}{#1}}%
1286   \fi\ignorespaces}
1287 \renewcommand*{\mpEfootnote}[1]{%
1288   \ifnumberedpar@
1289   \ifledRcol
1290     \xright@appenditem{\noexpand\mpvEfootnote{E}%
1291                         {\l@d@nums}{\@tag}{#1}}\to\inserts@listR
1292     \global\advance\insert@countR \@ne
1293   \else
1294     \xright@appenditem{\noexpand\mpvEfootnote{E}%
1295                         {\l@d@nums}{\@tag}{#1}}\to\inserts@list

```



```

1296 \global\advance\insert@count \@ne
1297 \fi
1298 \else
1299 \mpvEfootnote{E}{\{0|0|0|0|0|0|0\}\{#\}}%
1300 \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
1301 \def\printlinesR#1|#2|#3|#4|#5|#6|#7|{\begingroup
1302 \setprintlines{#1}{#2}{#3}{#4}{#5}{#6}%
1303 \ifl@d@pnum #1\fullstop\fi
1304 \ifl@d@plinenum \linenumr@p{#2}\Rlineflag\else \symlinenum\fi
1305 \ifl@d@ssub \fullstop \sublinenumr@p{#3}\fi
1306 \ifl@d@dash \endashchar\fi
1307 \ifl@d@pnum #4\fullstop\fi
1308 \ifl@d@elin \linenumr@p{#5}\Rlineflag\fi
1309 \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumr@p{#6}\fi
1310 \endgroup}
1311
1312 \let\ledsavedprintlines\printlines
1313

```

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.

```

1314 \list@create{\labelref@listR}
1315

```

`\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.

```

1316 \renewcommand*{\edlabel}[1]{\@bspack
1317 \ifl@d@col

```

```

1318 \write\linenum@outR{\string\@lab}%
1319 \ifx\labelref@listR\empty
1320 \xdef\label@refs{\zz@@@}%
1321 \else
1322 \glp\labelref@listR\to\label@refs
1323 \fi
1324 \protected@write\@auxout{%
1325 {\string\l@dmake@labelsR\space\thepage|\label@refs|{#1}}}%
1326 \else
1327 \write\linenum@out{\string\@lab}%
1328 \ifx\labelref@list\empty
1329 \xdef\label@refs{\zz@@@}%
1330 \else
1331 \glp\labelref@list\to\label@refs
1332 \fi
1333 \fi
1334 \protected@write\@auxout{%
1335 {\string\l@dmake@labelsR\space\thepage|\label@refs|{#1}}}%
1336 \@esphack}
1337

```

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

```

1338 \def\l@dmake@labelsR#1|#2|#3|#4{%
1339 \expandafter\ifx\csname the@label#4\endcsname \relax\else
1340 \led@warn@DuplicateLabel{#4}%
1341 \fi
1342 \expandafter\gdef\csname the@label#4\endcsname{#1|#2\Rlineflag|#3}%
1343 \ignorespaces}
1344 \AtBeginDocument{%
1345 \def\l@dmake@labelsR#1|#2|#3|#4{%
1346 }
1347

```

`\@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.

```

1348 \renewcommand*{\@lab}{%
1349 \ifledRcol
1350 \xright@appenditem{\linenumr@p{\line@numR}}|{%
1351 \ifsublines@ \sublinenumr@p{\subline@numR}\else 0\fi}%
1352 \to\labelref@listR
1353 \else
1354 \xright@appenditem{\linenumr@p{\line@num}}|{%
1355 \ifsublines@ \sublinenumr@p{\subline@num}\else 0\fi}%
1356 \to\labelref@list
1357 \fi}
1358

```

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 1359 \newcount\sidenote@marginR
1360 \renewcommand*{\sidenotemargin}[1]{\%
1361   \l@dgetsidenote@margin{#1}%
1362   \ifnum\@l@dttempcntb>\m@ne
1363     \ifledRcol
1364       \global\sidenote@marginR=\@l@dttempcntb
1365     \else
1366       \global\sidenote@margin=\@l@dttempcntb
1367     \fi
1368   \fi}}
1369 \sidenotemargin{right}
1370 \global\sidenote@margin=\@ne
1371
```

`\l@dlsnote` The ‘footnotes’ for left, right, and moveable sidenotes. The whole scheme is rem-

`\l@drsnote` iniscent of the critical footnotes code.

```
\l@dcsnote 1372 \renewcommand*{\l@dlsnote}[1]{\%
1373   \ifnumberedpar@
1374     \ifledRcol
1375       \xright@appenditem{\noexpand\l@dlsnote{#1}}%
1376         \to\inserts@listR
1377       \global\advance\insert@countR \@ne
1378     \else
1379       \xright@appenditem{\noexpand\l@dlsnote{#1}}%
1380         \to\inserts@list
1381       \global\advance\insert@count \@ne
1382     \fi
1383   \fi\ignorespaces}
1384 \renewcommand*{\l@drsnote}[1]{\%
1385   \ifnumberedpar@
1386     \ifledRcol
1387       \xright@appenditem{\noexpand\l@drsnote{#1}}%
1388         \to\inserts@listR
1389       \global\advance\insert@countR \@ne
1390     \else
1391       \xright@appenditem{\noexpand\l@drsnote{#1}}%
1392         \to\inserts@list
1393       \global\advance\insert@count \@ne
1394     \fi
1395   \fi\ignorespaces}
1396 \renewcommand*{\l@dcsnote}[1]{\%
1397   \ifnumberedpar@
1398     \ifledRcol
1399       \xright@appenditem{\noexpand\l@dcsnote{#1}}%
```

```

1400             \to\inserts@listR
1401     \global\advance\insert@countR \@ne
1402     \else
1403       \xright@appenditem{\noexpand\vl@dcnote{#1}}%
1404             \to\inserts@list
1405       \global\advance\insert@count \@ne
1406     \fi
1407   \fi\ignorespaces}
1408

```

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

```

1409 \newcommand*{\affixside@noteR}{%
1410   \gdef\@templ@d{%
1411     \ifx\@templ@d\l@dcnotetext \else
1412       \if@twocolumn
1413         \if@firstcolumn
1414           \setl@dlp@rbox{\l@dcnotetext}%
1415         \else
1416           \setl@drp@rbox{\l@dcnotetext}%
1417         \fi
1418       \else
1419         \@l@tempcntb=\sidenote@marginR
1420         \ifnum\@l@tempcntb>\@ne
1421           \advance\@l@tempcntb by\page@num
1422         \fi
1423         \ifodd\@l@tempcntb
1424           \setl@drp@rbox{\l@dcnotetext}%
1425         \else
1426           \setl@dlp@rbox{\l@dcnotetext}%
1427         \fi
1428       \fi
1429     \fi}
1430

```

18 Familiar footnotes

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

```

1431 \renewcommand{\l@dbfnote}[1]{%
1432   \ifnumberedpar@
1433     \ifledRcol
1434       \xright@appenditem{\noexpand\vl@dbfnote{#1}}{\@thefnmark}}%
1435       \to\inserts@listR
1436     \global\advance\insert@countR \@ne
1437   \else
1438     \xright@appenditem{\noexpand\vl@dbfnote{#1}}{\@thefnmark}}%
1439     \to\inserts@list
1440   \global\advance\insert@count \@ne

```

```

1441 \fi
1442 \fi\ignorespaces}
1443
\normalbfnoteX
1444 \renewcommand{\normalbfnoteX}[2]{%
1445 \ifnumberedpar@
1446 \ifledRcol
1447 \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\@nameuse{thefootnote#1}}}%
1448 \to\inserts@listR
1449 \global\advance\insert@countR \@ne
1450 \else
1451 \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\@nameuse{thefootnote#1}}}%
1452 \to\inserts@list
1453 \global\advance\insert@count \@ne
1454 \fi
1455 \fi\ignorespaces}
1456

```

19 Verse

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

```

1457
1458 \newcommand{\manageparhangingsymbol}{%
1459 \setcounter{hbox}{0}%
1460 \everyhbox{%
1461 \ifnum \value{hbox}=-2%
1462 \hangingsymbol%
1463 \fi%
1464 \addtocounter{hbox}{-1}}
1465 %\end{macrocode}
1466 % Before we can define the main stanza macros we need to be able to save
1467 % and reset
1468 % the category code for \&. To save the current value we use
1469 % \verb+\next+ from the \verb+\loop+ macro.
1470 % \begin{macrocode}
1471 \chardef\next=\catcode'\&
1472 \catcode'\&=\active
1473

```

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

```

1474 \newenvironment{astanza}{%
1475 \startstanzahook
1476 \catcode'\&=\active
1477 \global\stanza@count\@ne
1478 \ifnum\usernamecount{sza@0@}=\z@

```

```

1479 \let\stanza@hang\relax
1480 \let\endlock\relax
1481 \else
1482 %% \interlinepenalty\@M % this screws things up, but I don't know why
1483 \rightskip\z@ plus 1fil\relax
1484 \fi
1485 \ifnum\usernamecount{szp@0@}=\z@
1486 \let\sza@penalty\relax
1487 \fi
1488 \def&{%
1489 \endlock\mbox{}}%
1490 \sza@penalty
1491 \global\advance\stanza@count\@ne
1492 \@astanza@line}%
1493 \def\&{%
1494 \endlock\mbox{}}
1495 \pend
1496 \endstanzaextra}%
1497 \pstart
1498 \@astanza@line
1499 }-{}
1500

```

`\@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.

```

1501 \newcommand*{\@astanza@line}{%
1502 \parindent=\csname szp@number\stanza@count @\endcsname\stanzaindentbase
1503 \par
1504 \stanza@hang%\mbox{}}%
1505 \ignorespaces}
1506

```

Lastly reset the modified category codes.

```

1507 \catcode'\&=\next
1508

```

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 1509 \providecommand*{\newnamebox}[1]{%
\unvnamebox 1510 \expandafter\newbox\csname #1\endcsname}
\namebox 1511 \providecommand*{\setnamebox}[1]{%
1512 \expandafter\setbox\csname #1\endcsname}

```

```

1513 \providecommand*\unhnamebox}[1]{%
1514   \expandafter\unhbox\csname #1\endcsname}
1515 \providecommand*\unvnamebox}[1]{%
1516   \expandafter\unvbox\csname #1\endcsname}
1517 \providecommand*\namebox}[1]{%
1518   \csname #1\endcsname}
1519

```

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

```

\usernamecount 1520 \providecommand*\newnamecount}[1]{%
1521   \expandafter\newcount\csname #1\endcsname}
1522 \providecommand*\usernamecount}[1]{%
1523   \csname #1\endcsname}
1524

```

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.

```

1525 \newcount\l@dc@maxchunks
1526 \newcommand*\maxchunks}[1]{\l@dc@maxchunks=#1}
1527   \maxchunks{10}
1528

```

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

```

\l@dnumpstartsR 1529 \newcount\l@dnumpstartsR
1530

```

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

```

\l@pscR 1531 \newcount\l@dpscl
1532 \newcount\l@dpscR
1533

```

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

```

1534 \newcommand*\l@dsetuprawboxes){%
1535   \@l@dttempcntb=\l@dc@maxchunks
1536   \loop\ifnum\@l@dttempcntb>\z@
1537     \newnamebox{\l@dLcolrawbox\the\@l@dttempcntb}
1538     \newnamebox{\l@dRcolrawbox\the\@l@dttempcntb}
1539     \advance\@l@dttempcntb \m@ne
1540   \repeat}
1541

```

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

```

1542 \newcommand*{\l@dssetupmaxlinecounts}{%
1543   \@l@dttempcntb=\l@dc@maxchunks
1544   \loop\ifnum\@l@dttempcntb>\z@
1545     \newnamecount{l@dmaxlinesinpar\the\@l@dttempcntb}
1546     \advance\@l@dttempcntb \m@ne
1547   \repeat}
1548 \newcommand*{\l@dzeromaxlinecounts}{%
1549   \begingroup
1550   \@l@dttempcntb=\l@dc@maxchunks
1551   \loop\ifnum\@l@dttempcntb>\z@
1552     \global\usenamecount{l@dmaxlinesinpar\the\@l@dttempcntb}=\z@
1553     \advance\@l@dttempcntb \m@ne
1554   \repeat
1555   \endgroup}
1556
```

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

```

1557 \AtBeginDocument{%
1558   \l@dssetuprawboxes
1559   \l@dssetupmaxlinecounts
1560   \l@dzeromaxlinecounts
1561   \l@dnumpsstartsL=\z@
1562   \l@dnumpsstartsR=\z@
1563   \l@dpscl=\z@
1564   \l@dpscr=\z@}
1565
```

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 1566 \newif\ifl@dusedbabel
\l@dusedbabeltrue 1567 \l@dusedbabelfalse
```


`\ifl@dsamelang` A flag for checking if the same `babel` language has been used for both the left and right texts.

```
\l@dsamelangtrue 1568 \newif\ifl@dsamelang
                  1569 \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.

```
1570 \newcommand*{\l@dchecklang}{%
1571   \l@dsamelangfalse
1572   \edef\@tempa{\theledlanguageL}\edef\@tempb{\theledlanguageR}%
1573   \ifx\@tempa\@tempb
1574     \l@dsamelangtrue
1575   \fi}
1576
```

`\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 `\`.

```
1577 \newcommand*{\l@dbbl@set@language}[1]{%
1578   \edef\language#1}%
1579   \select@language{\language}%
1580   \if@filesw
1581     \protected@write\@auxout{}\string\select@language{\language}%
1582     \addtocontents{toc}{\string\select@language{\language}%
1583     \addtocontents{lof}{\string\select@language{\language}%
1584     \addtocontents{lot}{\string\select@language{\language}%
1585   \fi}
1586
```

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` are the names of the languages of the left and right texts. `\l@duselanguage` is similar to `\selectlanguage`.

```
\theledlanguageR 1587 \providecommand{\selectlanguage}[1]{%
                  1588 \newcommand*{\l@duselanguage}[1]{%
                  1589 \gdef\theledlanguageL{}
                  1590 \gdef\theledlanguageR{}
                  1591
```

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

```
1592 \AtBeginDocument{%
1593   \@ifundefined{xpg@main@language}{%
1594     \@ifundefined{bbl@main@language}{%
```

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

```
1595 \l@dusedbabelfalse
1596 \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.

```
1597 \l@dusedbabeltrue
1598 \let\l@doldselectlanguage\selectlanguage
1599 \let\l@doldbbl@set@language\bbl@set@language
1600 \let\bbl@set@language\l@dbbl@set@language
1601 \renewcommand{\selectlanguage}[1]{%
1602   \l@doldselectlanguage{#1}%
1603   \ifledRcol \gdef\theledlanguageR{#1}%
1604   \else      \gdef\theledlanguageL{#1}%
1605   \fi}
```

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

```
1606 \renewcommand*{\l@duselanguage}[1]{%
1607   \l@doldselectlanguage{#1}}
```

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

```
1608 \gdef\theledlanguageL{\bbl@main@language}%
1609 \gdef\theledlanguageR{\bbl@main@language}%
1610 }%
1611 }
```

If on Polyglossia

```
1612 { \apptocmd{\xpg@set@language}{%
1613   \ifledRcol \gdef\theledlanguageR{#1}%
1614   \else      \gdef\theledlanguageL{#1}%
1615   \fi}%
1616   \let\l@duselanguage\xpg@set@language
1617   \gdef\theledlanguageL{\xpg@main@language}%
1618   \gdef\theledlanguageR{\xpg@main@language}%
1619 % \end{macrocode}
1620 % That's it.
1621 % \begin{macrocode}
1622 }}
```

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.

```
1623 \newcommand*{\Columns}{%
1624   \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
1625     \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
1626   \fi}
```

Start a group and zero counters, etc.

```

1627 \begingroup
1628 \l@dzeropenalties
1629 \endgraf\global\num@lines=\prevgraf
1630 \global\num@linesR=\prevgraf
1631 \global\par@line=\z@
1632 \global\par@lineR=\z@
1633 \global\l@dpscL=\z@
1634 \global\l@dpscR=\z@

```

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

```

1635 \check@pstarts
1636 \loop\if@pstarts

```

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

```

1637 \global\advance\l@dpscL \@ne
1638 \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

```

1639 \checkraw@text
1640 \l@dchecklang
1641 { \loop\ifaraw@text

```

Grab the next pair of left and right text lines and output them, swapping languages if they differ

```

1642 \ifl@dsamelang
1643 \do@lineL
1644 \do@lineR
1645 \else
1646 \l@duselanguage{\theledlanguageL}%
1647 \do@lineL
1648 \l@duselanguage{\theledlanguageR}%
1649 \do@lineR
1650 \fi
1651 \hb@xt@ \hsize{%
1652 \unhbox\l@leftbox
1653 \hfill \columnseparator \hfill
1654 \unhbox\l@rightbox
1655 }%
1656 \checkraw@text
1657 \repeat}

```

Having completed a pair of chunks, write the number of lines in each chunk to the respective section files.

```

1658 \@writelinesinparL
1659 \@writelinesinparR
1660 \check@pstarts
1661 \repeat

```

Having output all chunks, make sure all notes have been output, then zero counts ready for the next set of texts.

```

1662     \flush@notes
1663     \flush@notesR
1664 \endgroup

1665 \global\l@dpscL=\z@
1666 \global\l@dpscR=\z@
1667 \global\l@dnumstartsL=\z@
1668 \global\l@dnumstartsR=\z@
1669 \ignorespaces}
1670

```

`\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).

```

1671 \newcommand*{\columnseparator}{%
1672   \smash{\rule[-0.2\baselineskip]{\columnrulewidth}{1.05\baselineskip}}}
1673 \newdimen\columnrulewidth
1674   \columnrulewidth=\z@
1675

```

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

```

\@pstartstrue 1676 \newif\if@pstarts
\@pstartsfalse 1677 \newcommand*{\check@pstarts}{%
\check@pstarts 1678   \@pstartsfalse
                  1679   \ifnum\l@dnumstartsL>\l@dpscL
                  1680     \@pstartstrue
                  1681   \else
                  1682     \ifnum\l@dnumstartsR>\l@dpscR
                  1683       \@pstartstrue
                  1684     \fi
                  1685   \fi}
1686

```

`\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 sets `\araw@textfalse`.

```

\checkraw@text 1687 \newif\ifaraw@text
                  1688   \araw@textfalse
                  1689 \newcommand*{\checkraw@text}{%
                  1690   \araw@textfalse
                  1691   \ifvbox\namebox{\l@dLcolrawbox\the\l@dpscL}
                  1692     \araw@texttrue
                  1693   \else
                  1694     \ifvbox\namebox{\l@dRcolrawbox\the\l@dpscR}
                  1695       \araw@texttrue
                  1696     \fi

```

```

1697 \fi}
1698

```

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

```

1699 \newcommand*{\@writelinesinparL}{%
1700 \edef\next{%
1701 \write\linenum@out{\string\@pend[\the\@donereallinesL]}}%
1702 \next
1703 \global\@donereallinesL \z@}
1704 \newcommand*{\@writelinesinparR}{%
1705 \edef\next{%
1706 \write\linenum@outR{\string\@pendR[\the\@donereallinesR]}}%
1707 \next
1708 \global\@donereallinesR \z@}
1709

```

24 Parallel pages

This is considerably more complicated than parallel columns.

`\numpagelinesL` Counts for the number of lines on a left or right page, and the smaller of the
`\numpagelinesR` number of lines on a pair of facing pages.
`\l@dminpagelines` 1710 \newcount\numpagelinesL
1711 \newcount\numpagelinesR
1712 \newcount\l@dminpagelines
1713

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

```

1714 \newcommand*{\Pages}{%
1715 \typeout{}
1716 \typeout{***** PAGES *****}
1717 \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
1718 \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
1719 \fi

```

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

```

1720 \clearto\l@devenpage
1721 \begingroup
1722 \l@dzeropenalties
1723 \endgraf\global\num@lines=\prevgraf
1724 \global\num@linesR=\prevgraf
1725 \global\par@line=\z@
1726 \global\par@lineR=\z@
1727 \global\l@dpscl=\z@
1728 \global\l@dpscR=\z@

```

```

1729    \writtenlinesLfalse
1730    \writtenlinesRfalse

    Check if there are chunks to be processed.

1731    \check@pstarts
1732    \loop\if@pstarts

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

1733    \global\advance\l@dpscL \@ne
1734    \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.

1735    \getlinesfromparlistL
1736    \getlinesfromparlistR
1737    \l@dcalc@maxoftwo{\@cs@linesinparL}{\@cs@linesinparR}%
1738    {\@cs@linesinparR}%
1739    \check@pstarts
1740    \repeat

    Zero the counts again, ready for the next bit.

1741    \global\l@dpscL=\z@
1742    \global\l@dpscR=\z@

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

1743    \getlinesfrompagelistL
1744    \getlinesfrompagelistR
1745    \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
1746    {\@cs@linesonpageR}%

    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.

1747    \check@pstarts
1748    \if@pstarts

    Increment the chunk counts to get the first pair.

1749    \global\advance\l@dpscL \@ne
1750    \global\advance\l@dpscR \@ne

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

1751    \global\@donereallinesL=\z@
1752    \global\@donetotallinesL=\z@
1753    \global\@donereallinesR=\z@
1754    \global\@donetotallinesR=\z@

    Start a loop over the boxes (chunks).

1755    \checkraw@text
1756 %    \begingroup
1757 {    \loop\ifaraw@text

```

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

```

1758      \checkpageL
1759      \l@duselanguage{\theledlanguageL}%
1760      %%
1760      \begingroup
1761      {
1761      \loop\ifl@dsamepage
1761      Process the next (left) text line, adding it to the page.
1762      \do@lineL
1763      \advance\numpagelinesL \@ne
1764      \ifshiftedverses
1765      \addtocounter{hbox}{-1}
1766      \ifdim\ht\l@dleftbox>0pt\hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}\fi%
1767      \else
1768      \hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
1769      \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.

```

1770      \get@nextboxL
1771      \checkpageL
1772      \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.

```

1773      \ifl@dpagfull
1774      \@writelinesonpageL{\the\numpagelinesL}%
1775      \else
1776      \@writelinesonpageL{1000}%
1777      \fi

```

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

```

1778      \numpagelinesL \z@
1779      \clearl@dleftpage }%

```

Now do the same for the right text.

```

1780      \checkpageR
1781      \l@duselanguage{\theledlanguageR}%
1782      {
1782      \loop\ifl@dsamepage
1783      \do@lineR
1784      \advance\numpagelinesR \@ne
1785      \ifshiftedverses
1786      \addtocounter{hbox}{-1}
1787      \ifdim\ht\l@drightbox>0pt\hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}\fi%
1788      \else
1789      \hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}%
1790      \fi

```

```

1791         \get@nextboxR
1792         \checkpageR
1793     \repeat
1794     \ifl@dpagfull
1795         \@writelinesonpageR{\the\numpagelinesR}%
1796     \else
1797         \@writelinesonpageR{1000}%
1798     \fi
1799     \numpagelinesR=\z@

```

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

```
1800     \clearl@drighthousepage}
```

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.

```

1801         \checkraw@text
1802         \ifaraw@text
1803             \getlinesfrompagelistL
1804             \getlinesfrompagelistR
1805             \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
1806                 {\l@dminpagelines}%
1807         \fi
1808     \repeat}

```

We have now output the text from all the chunks.

```
1809     \fi
```

Make sure that there are no inserts hanging around.

```

1810     \flush@notes
1811     \flush@notesR
1812 \endgroup

```

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

```

1813 \global\l@dpscL=\z@
1814 \global\l@dpscR=\z@
1815 \global\l@dnumpestartsL=\z@
1816 \global\l@dnumpestartsR=\z@
1817
1818 \ignorespaces}
1819

```

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

```

\ledstrutR 1820 \newcommand*{\ledstrutL}{\strut}
1821 \newcommand*{\ledstrutR}{\strut}
1822

```

`\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` `\clearl@drighthousepage`

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

```

1823 \providecommand{\cleartoevenpage}[1][\@empty]{%
1824   \clearpage
1825   \ifodd\c@page\hbox{ }#1\clearpage\fi}
1826 \newcommand*{\clearl@devenpage}{%
1827   \ifdim\pagetotal<\topskip% on an empty page
1828   \else
1829     \clearpage
1830   \fi
1831   \ifodd\c@page\hbox{ }\clearpage\fi}
1832 \newcommand*{\clearl@dleftpage}{%
1833   \clearpage
1834   \ifodd\c@page\else
1835     \led@err@LeftOnRightPage
1836     \hbox{ }%
1837   \cleardoublepage
1838 \fi}
1839 \newcommand*{\clearl@drighthouse}{%
1840   \clearpage
1841   \ifodd\c@page
1842     \led@err@RightOnLeftPage
1843     \hbox{ }%
1844   \cleartoevenpage
1845 \fi}
1846

```

`\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@linesinparL 1847 \newcommand*{\getlinesfromparlistL}{%
1848   \ifx\linesinpar@listL\empty
1849   \gdef\cs@linesinparL{0}%
1850   \else
1851     \gl@p\linesinpar@listL\to\cs@linesinparL
1852   \fi}
1853 \newcommand*{\getlinesfromparlistR}{%
1854   \ifx\linesinpar@listR\empty
1855   \gdef\cs@linesinparR{0}%
1856   \else
1857     \gl@p\linesinpar@listR\to\cs@linesinparR
1858   \fi}
1859

```

`\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@linesonpageL 1860 \newcommand*{\getlinesfrompagelistL}{%
1861   \ifx\linesonpage@listL\empty

```

```

1862 \gdef\@cs@linesonpageL{1000}%
1863 \else
1864 \gl@p\linesonpage@listL\to\@cs@linesonpageL
1865 \fi}
1866 \newcommand*\getlinesfrompagelistR{%
1867 \ifx\linesonpage@listR\empty
1868 \gdef\@cs@linesonpageR{1000}%
1869 \else
1870 \gl@p\linesonpage@listR\to\@cs@linesonpageR
1871 \fi}
1872

```

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

```

1873 \newcommand*\@writelinesonpageL}[1]{%
1874 \edef\next{\write\linenum@out{\string\@lopL{#1}}}%
1875 \next}
1876 \newcommand*\@writelinesonpageR}[1]{%
1877 \edef\next{\write\linenum@outR{\string\@lopR{#1}}}%
1878 \next}
1879

```

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

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

```

1880 \newcommand*\l@dcalc@maxoftwo}[3]{%
1881 \ifnum #2>#1\relax
1882 #3=#2\relax
1883 \else
1884 #3=#1\relax
1885 \fi}
1886 \newcommand*\l@dcalc@minoftwo}[3]{%
1887 \ifnum #2<#1\relax
1888 #3=#2\relax
1889 \else
1890 #3=#1\relax
1891 \fi}
1892

```

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

```

\checkpageL 1893 \newif\ifl@dsamepage
\checkpageR 1894 \l@dsamepagetrue

```

```

1895 \newif\ifl@dpagfull
1896 \newcommand*{\checkpageL}{%
1897   \l@dpagfulltrue
1898   \l@dsamepagetrue
1899   \check@goal
1900   \ifdim\pagetotal<\ledthegoal
1901     \ifnum\numpagelinesL<\l@dminpagelines
1902       \else
1903         \l@dsamepagefalse
1904         \l@dpagfullfalse
1905       \fi
1906     \else
1907       \l@dsamepagefalse
1908       \l@dpagfulltrue
1909     \fi}
1910 \newcommand*{\checkpageR}{%
1911   \l@dpagfulltrue
1912   \l@dsamepagetrue
1913   \check@goal
1914   \ifdim\pagetotal<\ledthegoal
1915     \ifnum\numpagelinesR<\l@dminpagelines
1916       \else
1917         \l@dsamepagefalse
1918         \l@dpagfullfalse
1919       \fi
1920     \else
1921       \l@dsamepagefalse
1922       \l@dpagfulltrue
1923     \fi}
1924

```

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

```

1925 \newdimen\ledthegoal
1926 \ifshiftedverses
1927   \newcommand*{\goalfraction}{0.95}
1928 \else
1929   \newcommand*{\goalfraction}{0.9}
1930 \fi
1931
1932 \newcommand*{\check@goal}{%
1933   \ledthegoal=\goalfraction\pagegoal}
1934

```

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

```

\ifwrittenlinesL 1935 \newif\ifwrittenlinesL
1936 \newif\ifwrittenlinesR
1937

```

`\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.

```

1938 \newcommand*{\get@nextboxL}{%
1939   \ifvbox\namebox{1@dLcolrawbox\the\1@dpscL}% box is not empty

   The current box is not empty; do nothing.
1940   \else%                                     box is empty

   The box is empty; check if enough lines (real and blank) have been output.
1941     \ifnum\usernamecount{1@dmaxlinesinpar\the\1@dpscL}>\@donetotallinesL
1942     \else

   Sufficient lines have been output.
1943       \ifwrittenlinesL
1944       \else

   Write out the number of lines done, and set the boolean so this is only done once.
1945         \@writelinesinparL
1946         \writtenlinesLtrue
1947       \fi
1948       \ifnum\1@dnumstartsL>\1@dpscL

   There are still unprocessed boxes. Recalculate the maximum number of lines
   needed, and move onto the next box (by incrementing \1@dpscL).
1949         \writtenlinesLfalse
1950         \1@dcalc@maxoftwo{\the\usernamecount{1@dmaxlinesinpar\the\1@dpscL}}%
1951                               {\the\@donetotallinesL}%
1952                               {\usernamecount{1@dmaxlinesinpar\the\1@dpscL}}%
1953         \global\@donetotallinesL \z@
1954         \global\advance\1@dpscL \@ne
1955       \fi
1956     \fi
1957   \fi}

1958 \newcommand*{\get@nextboxR}{%
1959   \ifvbox\namebox{1@dRcolrawbox\the\1@dpscR}% box is not empty
1960   \else%                                     box is empty
1961     \ifnum\usernamecount{1@dmaxlinesinpar\the\1@dpscR}>\@donetotallinesR
1962     \else
1963       \ifwrittenlinesR
1964       \else
1965         \@writelinesinparR
1966         \writtenlinesRtrue
1967       \fi
1968       \ifnum\1@dnumstartsR>\1@dpscR
1969       \writtenlinesRfalse
1970       \1@dcalc@maxoftwo{\the\usernamecount{1@dmaxlinesinpar\the\1@dpscR}}%
1971                               {\the\@donetotallinesR}%
1972                               {\usernamecount{1@dmaxlinesinpar\the\1@dpscR}}%
1973       \global\@donetotallinesR \z@
1974       \global\advance\1@dpscR \@ne

```

```
1975      \fi
1976      \fi
1977      \fi}
1978
```

25 The End

i/code_i

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,	And made me eat such bitter fruit,	4r
	Enferré ... Quant j'en ay memoire,	In chains ... When I think of this,	
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.

Figure 1: Output from `villon.tex`.

1 De ecclesia S. Stephani Novimagensi

Nobilis itaque comes Otto imperio et dominio Novimagensi sibi, ut praefer-
tur, 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 utili-
tate, 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 de-
cano et capitulo sanctorum Apostolorum Coloniensi, ipsius ecclesiae ab antiquo
veris et pacificis patronis, consensum, citra tamen praeiudicium, damnum aut
gravamen iurium et bonorum eorundem, impetravit.

Et exinde liberum locum eiusdem civitatis qui dicitur Hundisburg, de praeli-
bati Wilhelmi Romanorum regis, ipsius fundi domini, consensu, ad aedifican-
dum 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 con-
tiguam, 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 1254
 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
 5 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
 10 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 grave-
 15 yard. 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
 20 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—rissime 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 praecripiat 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`.

¹I.e., the even lines, which were hexameters (with six feet) became pentameters (with five feet).

²Muses

³Ceres was the Roman goddess of the harvest.

⁴By '*Virgo*' ('Virgin') Ovid means Diana, the Roman goddess of the hunt.

⁵Lines 7R-12R show some paradoxical situations that would occur if the gods didn't stay with their own business.

12R Aonian] Mount Parnassus, where the Muses live, is located in Aonia.

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 praecripiat 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 `djdpoe.ms.tex`.

⁶I.e., the even lines, which were hexameters (with six feet) became pentameters (with five feet).

⁷Muses

⁸Ceres was the Roman goddess of the harvest.

⁹By '*Virgo*' ('Virgin') Ovid means Diana, the Roman goddess of the hunt.

¹⁰Lines 7R–12R show some paradoxical situations that would occur if the gods didn't stay with their own business.

12R Aonian] Mount Parnassus, where the Muses live, is located in Aonia.

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.

```

1979 (*villon)
1980 %% villon.tex Example parallel columns
1981 \documentclass{article}
1982 \addtolength{\textheight}{-10\baselineskip}
1983 \usepackage{ledmac,ledpar}
1984 %% Use r instead of R to flag right text line numbers
1985 \renewcommand{\Rlineflag}{r}
1986 %% Use the flag in the notes
1987 \let\oldBfootfmt\Bfootfmt
1988 \renewcommand{\Bfootfmt}[3]{%
1989   \let\printlines\printlinesR
1990   \oldBfootfmt{#1}{#2}{#3}}
1991 \begin{document}
1992
1993 I thought that limericks were peculiarly English, but this appears not
1994 to be the case. As with most limericks this one is by Anonymous.
1995
1996 \vspace*{\baselineskip}
1997
1998 \begin{pairs}
1999 %% no indentation
2000 \setstanzaindents{0,0,0,0,0,0,0,0,0}
2001 %% no number flag
2002 \renewcommand{\Rlineflag}{}
2003 %% draw a rule and widen the columns
2004 \setlength{\columnrulewidth}{0.4pt}
2005 \setlength{\Lcolwidth}{0.46\textwidth}
2006 \setlength{\Rcolwidth}{\Lcolwidth}
2007
2008 \begin{Leftside}
2009 %% set left text line numbering sequence
2010 \firstlinenum{2}
2011 \linenumincrement{2}
2012 \linenummargin{left}
2013 \beginnumbering
2014 \stanza
2015 Il y avait un jeune homme de Dijon, &
2016 Qui n'avait que peu de religion. &
2017 Il dit: 'Quant \'{a} moi, &
2018 Je d'\{e\}teste tous les trois, &
2019 Le P\{e\}re, et le Fils, et le Pigeon.' \&
2020 \endnumbering
2021 \end{Leftside}

```

```

2022
2023 \begin{Rightside}
2024 %% different right text line numbering sequence
2025 \firstlinenum{1}
2026 \linenumincrement{2}
2027 \linenummargin{right}
2028 \beginnumbering
2029 \stanza
2030 There was a young man of Dijon, &
2031 Who had only a little religion, &
2032 He said: 'As for me, &
2033 I detest all the three, &
2034 The Father, the Son, and the Pigeon.' \&
2035 \endnumbering
2036 \end{Rightside}
2037
2038 \Columns
2039 \end{pairs}
2040
2041 \vspace*{\baselineskip}
2042
2043     The following is verse \textsc{lxixiii} of Fran\c{c}ois Villon's
2044 \textit{Le Testament} (The Testament), composed in 1461.
2045
2046 %% Allow for hanging indentation for long lines
2047 \setstanzaindents{1,0,0,0,0,0,0,0}
2048 %% Columns wider than the default
2049 \setlength{\Lcolwidth}{0.46\textwidth}
2050 \setlength{\Rcolwidth}{\Lcolwidth}
2051 \vspace*{\baselineskip}
2052
2053 \begin{pairs}
2054 \begin{Leftside}
2055 \firstlinenum{2}
2056 \linenumincrement{2}
2057 \linenummargin{left}
2058 \beginnumbering
2059 \stanza
2060 Dieu mercy et Tacque Thibault, &
2061 Qui tant d'eaue froid m'a fait boire, &
2062 Mis en bas lieu, non pas en hault, &
2063 Mengier d'angoisse maints \edtext{poire}{\lemma{poire d'angoisse}}%
2064 \Afootnote{This has a triple meaning: literally it is the fruit of the
2065  choke pear,
2066  figuratively it means 'bitter fruit', and it also refers to a torture
2067  instrument.}}, &
2068 Enferr'\{e} \ldots Quant j'en ay memoire, &
2069 Je Prie pour luy \edtext{\textit{et reliqua}}{\Afootnote{and so on}}, &
2070 Que Dieu luy doint, et voire, voire! &
2071 Ce que je pense \ldots \textit{et cetera}. \&

```

```

2072 \endnumbering
2073 \end{Leftside}
2074
2075 \begin{Rightside}
2076 \firstlinenum{2}
2077 \linenumincrement{2}
2078 \linenummargin{right}
2079 \beginnumbering
2080 \stanza
2081 Thanks to God --- and to \edtext{Tacque Thibaud}{%
2082   \Bfootnote{A favourite of Jean, Duc de Berry and loathed for his exactions
2083     and debauchery. Villon uses his name as an insulting nickname for
2084     Thibaud d'Auxigny, the Bishop of Orl\{'e}ans.}} &
2085   Who made me drink so much \edtext{cold water}{%
2086     \Bfootnote{Can either refer to the normal prison diet of bread and
2087       water or to a common medieval torture which involved forced drinking
2088       of cold water.}}, &
2089   Put me underground instead of higher up &
2090   And made me eat such bitter fruit, &
2091   In chains \ldots When I think of this, &
2092   I pray for him---\textit{et reliqua;} &
2093   May God grant him (yes, by God) &
2094   What I think \ldots \textit{et cetera}. \&
2095 \endnumbering
2096 \end{Rightside}
2097
2098 \Columns
2099 \end{pairs}
2100
2101 \vspace*{\baselineskip}
2102
2103   The translation and notes are by Anthony Bonner,
2104 \textit{The Complete Works of Fran\c{c}ois Villon}, published by
2105 Bantam Books in 1960.
2106
2107 \end{document}
2108
2109 </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.

```

2110 <*djd17nov>
2111 %%% This is djdl7nov.tex, a sample critical text edition
2112 %%% written in LaTeX2e with the ledmac and ledpar packages.
2113 %%% (c) 2003--2004 by Dr. Dirk-Jan Dekker,

```



```

2114 %% Radboud University, Nijmegen (The Netherlands)
2115 %% (PRW) Modified slightly by PRW to fit the ledpar manual
2116
2117 \documentclass[10pt, letterpaper, twoside]{article}
2118 \usepackage[latin,english]{babel}
2119 \usepackage{makeidx}
2120 \usepackage{ledmac,ledpar}
2121 \lineation{section}
2122 \linenummargin{inner}
2123 \sidenotemargin{outer}
2124
2125 \makeindex
2126
2127 \renewcommand{\notenumfont}{\footnotesize}
2128 \newcommand{\notetextfont}{\footnotesize}
2129
2130 \let\Afootnoterule=\relax
2131 \let\Bfootnoterule=\relax
2132 \let\Cfootnoterule=\relax
2133
2134 \addtolength{\skip\Afootins}{1.5mm}
2135 %\addtolength{\skip\Bfootins}{1.5mm}
2136 %\addtolength{\skip\Cfootins}{1.5mm}
2137
2138 \makeatletter
2139
2140 \renewcommand*{\para@vfootnote}[2]{%
2141   \insert\csname #1footins\endcsname
2142   \bgroup
2143     \notefontsetup
2144     \interlinepenalty=\interfootnotelinepenalty
2145     \floatingpenalty=\@MM
2146     \splittopskip=\ht\strutbox \splitmaxdepth=\dp\strutbox
2147     \leftskip=\z@skip \rightskip=\z@skip
2148     \l@dparsfootspec #2\ledplinenumtrue%           new from here
2149     \ifnum\@nameuse{previous@#1@number}=\l@dparsestartline\relax
2150       \ledplinenumfalse
2151     \fi
2152     \ifnum\previous@page=\l@dparsestartpage\relax
2153     \else \ledplinenumtrue \fi
2154     \ifnum\l@dparsestartline=\l@dparseendline\relax
2155     \else \ledplinenumtrue \fi
2156     \expandafter\xdef\csname previous@#1@number\endcsname{\l@dparsestartline}%
2157     \xdef\previous@page{\l@dparsestartpage}%           to here
2158     \setbox0=\vbox{\hsize=\maxdimen
2159       \noindent\csname #1footfmt\endcsname#2}%
2160     \setbox0=\hbox{\unvbox0}%
2161     \dp0=0pt
2162     \ht0=\csname #1footfudgefactor\endcsname\wd0
2163     \box0

```

```

2164     \penalty0
2165     \egroup
2166 }
2167
2168 \newcommand*{\previous@A@number}{-1}
2169 \newcommand*{\previous@B@number}{-1}
2170 \newcommand*{\previous@C@number}{-1}
2171 \newcommand*{\previous@page}{-1}
2172
2173 \newcommand{\abb}[1]{#1%
2174     \let\rbracket\nobrak\relax}
2175 \newcommand{\nobrak}{\textnormal{}}
2176 \newcommand{\morenoexpands}{%
2177     \let\abb=0%
2178 }
2179
2180 \newcommand{\Aparafootfmt}[3]{%
2181     \ledsetnormalparstuff
2182     \scriptsize
2183     \notenumfont\printlines#1\enspace
2184 %   \lemmafont#1/#2\enskip
2185     \notetextfont
2186     #3\penalty-10\hskip 1em plus 4em minus.4em\relax}
2187
2188 \newcommand{\Bparafootfmt}[3]{%
2189     \ledsetnormalparstuff
2190     \scriptsize
2191     \notenumfont\printlines#1/%
2192     \ifledplinenum
2193     \enspace
2194     \else
2195     {\hskip 0em plus 0em minus .3em}%
2196     \fi
2197     \select@lemmafont#1/#2\rbracket\enskip
2198     \notetextfont
2199     #3\penalty-10\hskip 1em plus 4em minus.4em\relax }
2200
2201 \newcommand{\Cparafootfmt}[3]{%
2202     \ledsetnormalparstuff
2203     \scriptsize
2204     \notenumfont\printlines#1\enspace
2205 %   \lemmafont#1/#2\enskip
2206     \notetextfont
2207     #3\penalty-10\hskip 1em plus 4em minus.4em\relax}
2208
2209 \makeatother
2210
2211 \footparagraph{A}
2212 \footparagraph{B}
2213 \footparagraph{C}

```

```

2214
2215 \let\Afootfmt=\Aparafootfmt
2216 \let\Bfootfmt=\Bparafootfmt
2217 \let\Cfootfmt=\Cparafootfmt
2218
2219 \renewcommand*{\Rlineflag}{}
2220
2221 \emergencystretch40pt
2222
2223 \author{Guillelmus de Berchen}
2224 \title{Chronicon Geldriae}
2225 \date{}
2226 \hyphenation{archi-epi-sco-po Huns-dis-brug li-be-ra No-vi-ma-gen-si}
2227 \begin{document}
2228 \begin{pages}
2229 \begin{Leftside}
2230 \beginnumbering\pstart
2231 \selectlanguage{latin}
2232 \section{De ecclesia S. Stephani Novimagensi}
2233
2234 \noindent\setline{1}
2235 Nobilis itaque comes Otto\protect\edindex{Otto II of Guelders}
2236 imperio et dominio Novimagensi sibi, ut praeferetur, impignoratis
2237 et commissis
2238 \edtext{proinde}{\Bfootnote{primum D}} praeesse cupiens, anno
2239 \textsc{liiii} superius descripto, mense
2240 Iu\edtext{}{\Afootnote{p.\ 227~R}}nio, una cum iudice, scabinis ceterisque
2241 civibus civitatis Novimagensis, pro ipsius et inhabitantium in ea
2242 necessitate,\edtext{}{\Afootnote{p.\ 97~N}} commodo et utilitate,
2243 ut \edtext{ecclesia eius}{\Bfootnote{ecclesia D: eius eius H}} parochialis
2244 \edtext{\abb{extra civitatem}}{\Bfootnote{\textit{om.}~H}} sita
2245 destrueretur et \edtext{infra}{\Bfootnote{intra D}} muros
2246 \edtext{transfer}\edtext{}{\Afootnote{p.\ 129~D}}retur}%
2247 {\Bfootnote{transferreretur NH}}
2248 ac de novo construeretur,
2249 \edtext{a reverendo patre domino
2250 Conrado\protect\edindex{Conrad of Hochstaden} de
2251 \edtext{Hofsteden}{\Bfootnote{Hoffstede D: Hoffsteden H}}, archiepiscopo
2252 \edtext{Coloniensi}{\Bfootnote{Colononiensi H}}, licentiam}%
2253 {\Cfootnote{William is confusing two charters that are five years
2254 apart. Permission from St.\ Apostles' Church in Cologne had been
2255 obtained as early as 1249. Cf.\
2256 Sloet\protect\index{Sloet van de Beele, L.A.J.W.},
2257 \textit{Oorkondenboek} nr.\ 707 (14 November 1249):
2258 ‘‘\ldots}nos devotionis tue precibus annuentes, ut ipsam ecclesiam
2259 faciens demoliri transferas in locum alium competentem, tibi
2260 auctoritate presentium indulgemus\ldots’’}}, et a venerabilibus
2261 \edtext{dominis}{\Bfootnote{viris H}} decano et capitulo sanctorum
2262 Apostolorum\protect\edindex{St. Apostles' (Cologne)}
2263 \edtext{Coloniensi}{\Bfootnote{Coloniae H}}, ipsius ecclesiae ab

```

```

2264 antiquo veris et pacificis patronis, consensum, citra tamen
2265 praeiudicium, damnum aut gravamen \edtext{iurium}{\Bfootnote{virium D}}
2266 et bonorum eorundem, impetravit.
2267 \pend
2268
2269 \pstart
2270 \edtext{Et exinde \edtext{liberum}{\Bfootnote{librum H}}}
2271 locum eiusdem civitatis
2272 \edtext{qui}{\Bfootnote{quae D}} dicitur
2273 \edtext{Hundisburg}{\Bfootnote{Hundisburch D: Hundisbrug HMN:
2274 Hunsdisbrug R}}\protect\edindex{Hundisburg},
2275 de praelibati Wilhelmi\protect\edindex{William II of Holland} Romanorum
2276 \edtext{regis}{\Bfootnote{imperatoris D}}, ipsius fundi
2277 do\edtext{}{\Afootnote{f.\ 72v~M}}mini, consensu, ad aedificandum
2278 \edtext{\abb{et consecrandum}}{\Bfootnote{\textit{om.}\ H}}
2279 ecclesi\edtext{}{\Afootnote{p.\ 228~R}}am et coemeterium,
2280 \edtext{eisdem}{\Bfootnote{eiusdem D}} decano et capitulo de expresso
2281 eiusdem civitatis assensu libera contradiderunt voluntate, obligantes
2282 se ipsi \edtext{comes}{\Bfootnote{comites D}} et civitas
2283 \edtext{\abb{dictis}}{\Bfootnote{\textit{om.}\ H}} decano et capitulo,
2284 quod in recompensationem illius areae infra castrum et portam, quae
2285 fuit dos ecclesiae, in qua plebanus habitare solebat---quae
2286 \edtext{tunc}{\Bfootnote{nunc H}} per novum fossatum civitatis est
2287 destructa---aliam aream competentem et ecclesiae novae,
2288 \edtext{ut praefertur, aedificandae}{%
2289 \lemma{\abb{ut\ldots aedificandae}}{\Bfootnote{\textit{om.}\ H}} satis
2290 \edtext{contiguam}{\Bfootnote{contiguum M}}, ipsi plebano darent et
2291 assignarent.}{\Cfootnote{Cf.\ Sloet, \textit{Oorkondenboek} nr.\ 762
2292 (June 1254)}} Et desuper
2293 \edtext{\abb{apud}}{\Bfootnote{\textit{om.}\ H}} dictam ecclesiam
2294 sanctorum Apostolorum \edtext{est}{\Bfootnote{et H}}
2295 \edtext{littera}{\Bfootnote{litteram H}} sigillis ipsorum
2296 Ottonis\edtext{}{\Afootnote{p.\ 130~D}} comitis et civitatis
2297 \edtext{Novimagensis}{\Bfootnote{Novimagii D}}
2298 \edtext{sigillata}{\Bfootnote{sigillis communita H}}.
2299 \pend
2300
2301 \pstart
2302 // One additional line to show synchronization. //
2303 \pend
2304 \endnumbering
2305 \end{Leftside}
2306
2307 \begin{Rightside}
2308 \sidenotemargin{right}\selectlanguage{english}
2309 \beginnumbering
2310 \pstart
2311 \addtocounter{section}{-1}%
2312 \leavevmode\section{St.\ Stephen's Church in Nijmegen}
2313

```

2314 \noindent\setline{1}%
 2315 After the noble count Otto had taken in pledge the power over
 2316 Nijmegen,\footnote{In 1247 William II\protect\index{William II of Holland}
 2317 (1227--1256) count of Holland needed money to fight his way to
 2318 Aachen\protect\index{Aachen} to be crowned King of the Holy Roman
 2319 Empire. He gave the town of Nijmegen in pledge to Otto
 2320 II\protect\index{Otto II of Guelders} (1229--1271) count of Guelders.}
 2321 like I have written above, he wanted to protect the town. So in June
 2322 1254\ledsidenote{1254} he and the judge, the sheriffs and other
 2323 citizens of Nijmegen obtained permission to demolish the parish
 2324 church that lay outside the town walls,\footnote{Since the early
 2325 seventh century old St.\ Stephen's church had been located close
 2326 to the castle, at today's
 2327 Kelfkensbos\protect\index{Kelfkensbos (Nijmegen)}} square.
 2328 Traces of the church and the presbytery were found during excavations
 2329 in 1998--1999.} to move it inside the walls and to rebuild it new.
 2330 This operation was necessary and useful both for Otto himself and
 2331 for the inhabitants of the town. The reverend father Conrad of
 2332 Hochstaden, archbishop of
 2333 Cologne,\footnote{Conrad of Hochstaden ({\textdag} 1261) was
 2334 archbishop of Cologne in 1238--1261. Nijmegen belonged to the
 2335 archdiocese of Cologne until 1559.} gave his permission. So did the
 2336 reverend dean and canons of the chapter of St.\
 2337 Apostles'\protect\index{St. Apostles' (Cologne)} in Cologne, who had
 2338 long\footnote{They probably became the patrons when the chapter was
 2339 established in the early eleventh century. About the church and the
 2340 chapter, see Gottfried Stracke\protect\index{Stracke, G.},
 2341 \textit{K}\{"o}ln: \ St.\ Aposteln}, Stadtspuren -- Denkm\{"a}ler in
 2342 K\{"o}ln, vol.\ 19, K\{"o}ln: J.\,P.\ Bachem, 1992.} been the true
 2343 and benevolent patrons of the church---but they did not allow Otto
 2344 to do anything without their knowledge, nor to infringe their rights,
 2345 nor to damage their property.
 2346 \pend
 2347
 2348 \pstart
 2349 And so the count and the town voluntarily gave an open space in town
 2350 called Hundisburg, which was owned by the aforementioned king William,
 2351 to the dean and chapter of St.\ Apostles' in order to build and
 2352 consecrate a church and graveyard. King William approved and the
 2353 town of Nijmegen explicitly expressed its assent. A new ditch was dug
 2354 on property of the church near the castle and the
 2355 harbour,\footnote{Nowadays, the exact location of the medieval
 2356 ditch---and of two Roman ones---can be seen in the pavement of
 2357 Kelfkensbos\protect\index{Kelfkensbos (Nijmegen)}} square.} causing
 2358 the demolition of the presbytery. In compensation, the count and
 2359 citizens committed themselves to giving the parish priest another
 2360 suitable space close enough to the new church that was about to be
 2361 built. A letter about these transactions, with the seals of count
 2362 Otto and the town of Nijmegen, is kept at St.\ Apostles'
 2363 church.\footnote{The original letter is lost. A 15th century

```

2364 transcription of it is kept at the Historisches Archiv der
2365 Stadt K\{"o}ln (HASTK).}
2366 \pend
2367
2368 \pstart
2369 // One additional line to show synchronization. //
2370 \pend
2371 \endnumbering
2372 \end{Rightside}
2373 \Pages
2374 \end{pages}
2375
2376 %%%%%%%%%%%
2377 \printindex
2378 \end{document}
2379 %%%%%%%%%%%
2380
2381 </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.

```

2382 (*djdpoems)
2383 %% djdpoems.tex  example parallel verses on facing pages
2384 \documentclass{article}
2385 \usepackage{ledmac,ledpar}
2386 \addtolength{\textheight}{-15\baselineskip}
2387
2388 \maxchunks{24} % default value = 10
2389 \setstanzaindents{6,0,1,0,1}
2390
2391 \newcommand{\longdash}{-----}
2392
2393 \footparagraph{A} % for left pages
2394 \footparagraph{B} % for right pages
2395 \firstlinenum{1}
2396 \linenumincrement{1}
2397
2398 \let\oldBfootfmt\Bfootfmt
2399 \renewcommand{\Bfootfmt}[3]{%
2400   \let\printlines\printlinesR
2401   \oldBfootfmt{#1}{#2}{#3}}

```

```

2402
2403 \begin{document}
2404
2405 \newcommand{\interstanza}{\pstart\centering\longdash\skipnumbering\pend}
2406
2407 \begin{pages}
2408 \begin{Leftside}
2409 \def\endstanzaextra{\interstanza}
2410 \beginnumbering
2411
2412 \stanza
2413 Arma gravi numero violentaque bella parabam &
2414 edere, materi\={a} conveniente modis. &
2415 Par erat inferior versus---risisse Cupido &
2416 dicitur atque unum surripuisse pedem. \&
2417
2418 \stanza
2419 ‘Quis tibi, saeve puer, dedit hoc in carmina iuris? &
2420 Pieridum vates, non tua turba \edtext{sumus}{\Afootnote{note lost}}. &
2421 Quid si praeripiat flavae V\{e}nus arma Minervae, &
2422 ventilet accensas flava Minerva faces? \&
2423
2424 \stanza
2425 Quis probet in silvis Cererem regnare iugosis, &
2426 lege pharetratae Virginis arva coli? &
2427 Crinibus insignem quis \edtext{acuta}{\Afootnote{acut\={a} (abl.\ abs.)}}
2428 cuspidе Phoebum &
2429 instruat, Aoniam Marte movente lyram? \&
2430 \endnumbering
2431 \end{Leftside}
2432
2433 \begin{Rightside}
2434 \def\endstanzaextra{\interstanza}
2435 \beginnumbering
2436 \firstlinenum{1}
2437 \linenumincrement{1}
2438 \setstanzaindents{6,0,1,0,1,0}
2439
2440 \stanza
2441 I was preparing to sing of weapons and violent wars, &
2442 in heavy numbers, with the subject matter suited to the verse measure. &
2443 The even lines were as long as the odd ones, but Cupid laughed, &
2444 they said, and he stole away one foot.\footnote{I.e., the even lines,
2445 which were hexameters (with six feet) became pentameters
2446 (with five feet).} \&
2447
2448 \stanza
2449 ‘O cruel boy, who gave you the right over poetry? &
2450 We poets belong to the Pierides,\footnote{Muses} we are not your folk. &
2451 \edlabel{beginparadox}What if Venus should seize away the arms of

```

```

2452 Minerva with the golden hair, &
2453 if Minerva with the golden hair should fan alight the kindled torch
2454 of love? \&
2455
2456 \stanza
2457 Who would approve of Ceres\footnote{Ceres was the Roman goddess of
2458 the harvest.} reigning on the woodland ridges, &
2459 and of land tilled under the law of the Maid with the
2460 quiver\footnote{By '\textit{Virgo}' ('Virgin') Ovid means Diana, the
2461 Roman goddess of the hunt.}? &
2462 Who would provide Phoebus with his beautiful hair with a sharp-pointed
2463 spear, &
2464 while Mars stirs the \edtext{Aonian}{\Bfootnote{Mount Parnassus,
2465 where the Muses live, is located in Aonia.}}
2466 lyre?\edlabel{endparadox}\footnote{Lines
2467 \xlineref{beginparadox}--\xlineref{endparadox} show some paradoxical
2468 situations that would occur if the gods didn't stay with their own
2469 business.} \&
2470 \endnumbering
2471 \end{Rightside}
2472
2473 \Pages
2474 \end{pages}
2475
2476 \begin{pages}
2477 \begin{Leftside}
2478 \def\endstanzaextra{\interstanza}
2479 \beginnumbering
2480
2481 \begin{astanza}
2482 Arma gravi numero violentaque bella parabam &
2483 edere, materi\={a} conveniente modis. &
2484 Par erat inferior versus---risisse Cupido &
2485 dicitur atque unum surripuisse pedem. \&
2486 \end{astanza}
2487
2488 \begin{astanza}
2489 'Quis tibi, saeve puer, dedit hoc in carmina iuris? &
2490 Pieridum vates, non tua turba \edtext{sumus}{\Afootnote{note lost}}. &
2491 Quid si praeripiat flavae V\ufe}nus arma Minervae, &
2492 ventilet accensas flava Minerva faces? \&
2493 \end{astanza}
2494
2495 \begin{astanza}
2496 Quis probet in silvis Cererem regnare iugosis, &
2497 lege pharetratae Virginis arva coli? &
2498 Crinibus insignem quis \edtext{acuta}{\Afootnote{acut\={a} (abl.\ abs.)}}
2499 cuspidem Phoebum &
2500 instruat, Aoniam Marte movente lyram? \&
2501 \end{astanza}

```



```

2502
2503 \endnumbering
2504 \end{Leftside}
2505
2506 \begin{Rightside}
2507 \def\endstanzaextra{\interstanza}
2508 \beginnumbering
2509 \firstlinenum{1}
2510 \linenumincrement{1}
2511 \setstanzaindents{6,0,1,0,1,0}
2512
2513 \begin{astanza}
2514 I was preparing to sing of weapons and violent wars, &
2515 in heavy numbers, with the subject matter suited to the verse measure. &
2516 The even lines were as long as the odd ones, but Cupid laughed, &
2517 they said, and he stole away one foot.\footnote{I.e., the even lines,
2518 which were hexameters (with six feet) became pentameters
2519 (with five feet).} \&
2520 \end{astanza}
2521
2522 \begin{astanza}
2523 ‘‘O cruel boy, who gave you the right over poetry? &
2524 We poets belong to the Pierides,\footnote{Muses} we are not your folk. &
2525 \edlabel{beginparadox}What if Venus should seize away the arms of
2526 Minerva with the golden hair, &
2527 if Minerva with the golden hair should fan alight the kindled torch
2528 of love? \&
2529 \end{astanza}
2530
2531 \begin{astanza}
2532 Who would approve of Ceres\footnote{Ceres was the Roman goddess of the
2533 harvest.} reigning on the woodland ridges, &
2534 and of land tilled under the law of the Maid with the
2535 quiver\footnote{By ‘\textit{Virgo}’ (‘Virgin’) Ovid means Diana,
2536 the Roman goddess of the hunt.}? &
2537 Who would provide Phoebus with his beautiful hair with a sharp-pointed
2538 spear, &
2539 while Mars stirs the \edtext{Aonian}{\Bfootnote{Mount Parnassus, where
2540 the Muses live, is located in Aonia.}}
2541 lyre?\edlabel{endparadox}\footnote{Lines
2542 \xlineref{beginparadox}--\xlineref{endparadox} show some paradoxical
2543 situations that would occur if the gods didn’t stay with their
2544 own business.} \&
2545 \end{astanza}
2546
2547 \endnumbering
2548 \end{Rightside}
2549
2550 \Pages
2551 \end{pages}

```

```
2552
2553 \end{document}
2554
2555 \end{djdpoms}
```

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> 1468, 1471, 1472, 1476, 1493, 1507, 2019, 2034, 2071, 2094, 2416, 2422, 2429, 2446, 2454, 2469, 2485, 2492, 2500, 2519, 2528, 2544	<code>\@donetotallinesR</code> 848, 911, 914, 1754, 1961, 1971, 1973
<code>\@M</code> 1482	<code>\@insertR</code> 1140–1142, 1155–1157
<code>\@MM</code> 2145	<code>\@l</code> 251, 564
<code>\@adv</code> 328, 595, 596	<code>\@l@dttempcnta</code> 401, 403, 405, 406, 410, 412, 414, 415, 960, 999, 1000, 1002, 1004, 1007, 1008, 1025–1029, 1031, 1038, 1043, 1047, 1055, 1060, 1064, 1096, 1099, 1101, 1105
<code>\@afterindentfalse</code> 706	<code>\@l@dttempcntb</code> 144, 146, 148, 990, 991, 1038, 1043, 1047, 1055, 1060, 1064, 1088, 1092, 1105, 1113–1115, 1117, 1362, 1364, 1366, 1419–1421, 1423, 1535–1539, 1543–1546, 1550–1553
<code>\@arabic</code> 185, 186, 745, 747	<code>\@l@reg</code> 300
<code>\@astanza@line</code> 1492, 1498, 1501	<code>\@l@regR</code> 251
<code>\@auxout</code> 1324, 1334, 1581	<code>\@lab</code> 520, 1318, 1327, 1348
<code>\@chapter</code> 707	<code>\@lock</code> 942
<code>\@cs@linesinparL</code> 1737, 1847	<code>\@lockR</code> 58, 273, 275, 277, 290, 435, 451, 452, 454, 455, 483, 484, 486, 927, 966, 968, 969, 971, 1052, 1069, 1071, 1073
<code>\@cs@linesinparR</code> 1737, 1847	
<code>\@cs@linesonpageL</code> .. 1745, 1805, 1860	
<code>\@cs@linesonpageR</code> .. 1745, 1805, 1860	
<code>\@donereallinesL</code> 848, 876, 1701, 1703, 1751	
<code>\@donereallinesR</code> 848, 910, 1706, 1708, 1753	
<code>\@donetotallinesL</code> 848, 877, 880, 1752, 1941, 1951, 1953	

`\@lopL` 542, 1874
`\@lopR` 542, 1877
`\@nameuse` 1447, 1451, 2149
`\@nbreakfalse` 753, 782
`\@nbreaktrue` 751, 755, 780, 784
`\@oldnbreak` 751, 753, 780, 782, 818, 834
`\@pend` 535, 1701
`\@pendR` 535, 1706
`\@pstartfalse` 1676
`\@pstartstrue` 1676
`\@ref` 507, 568, 572
`\@ref@reg` 533
`\@schapter` 707
`\@set` 360, 602, 603
`\@tag` 633, 650, 1164, 1168, 1178, 1182,
1192, 1196, 1206, 1210, 1220,
1224, 1235, 1239, 1249, 1253,
1263, 1267, 1277, 1281, 1291, 1295
`\@temp` 1572
`\@templd` 1410, 1411
`\@writelinesinparL` .. 1658, 1699, 1945
`\@writelinesinparR` .. 1659, 1699, 1965
`\@writelinesonpageL` . 1774, 1776, 1873
`\@writelinesonpageR` . 1795, 1797, 1873
`\@xloop` 1153
`\add@insertsR` 903, 1128
`\add@penaltiesL` 875, 1149
`\add@penaltiesR` 909, 1149
`\addtocontents` 1582–1584
`\addtocounter` 820, 836,
928, 944, 1464, 1765, 1786, 2311
`\addtolength` ... 1982, 2134–2136, 2386
`\advanceline` 594, 625
`\affixline@num` 867
`\affixline@numR` 901, 1022
`\affixside@note` 870
`\affixside@noteR` 904, 1409
`\Afootfmt` 2215
`\Afootins` 2134
`\Afootnote` 1160, 2064,
2069, 2240, 2242, 2246, 2277,
2279, 2296, 2420, 2427, 2490, 2498
`\Afootnoterule` 2130
`\Aparafootfmt` 2180, 2215
`\apptocmd` 1612
`\araw@textfalse` 1687
`\araw@texttrue` 1687
`astanza` (environment) 7, 1474
`\AtBeginDocument` ... 1344, 1557, 1592
`\author` 2223

B

`_` 2240, 2242, 2246, 2254, 2255, 2257,
2277–2279, 2283, 2289, 2291,
2293, 2296, 2312, 2325, 2336,
2341, 2342, 2351, 2362, 2427, 2498
A
`\abb` 2173,
2177, 2244, 2278, 2283, 2289, 2293
`\absline@num` 394, 408, 427, 934
`\absline@numR` ... 56, 202, 253, 256,
259, 391, 399, 420, 439, 473,
501, 512, 919, 952, 953, 990, 1139
`\actionlines@list`
..... 243, 246, 394, 408, 427
`\actionlines@listR`
..... 206, 221, 235, 238, 391,
399, 420, 439, 473, 501, 1012, 1015
`\actions@list` . 247, 395, 415, 429, 431
`\actions@listR`
. 206, 222, 239, 392, 406, 422,
424, 441, 450, 475, 482, 502, 1016
`\add@inserts` 869
`\add@inserts@nextR` 1128
`\ballast@count` 950, 955
`\bbl@main@language` 1608, 1609
`\bbl@set@language` 1599, 1600
`\beginnumbering` .. 6, 34, 711, 725,
758, 2013, 2028, 2058, 2079,
2230, 2309, 2410, 2435, 2479, 2508
`\beginnumberingR` ... 47, 102, 725, 787
`\Bfootfmt` 1987, 1988, 2216, 2398, 2399
`\Bfootins` 2135
`\Bfootnote` 1174, 2082, 2086,
2238, 2243–2245, 2247, 2251,
2252, 2261, 2263, 2265, 2270,
2272, 2273, 2276, 2278, 2280,
2282, 2283, 2286, 2289, 2290,
2293–2295, 2297, 2298, 2464, 2539
`\Bfootnoterule` 2131
`\box` 2163
`\Bparafootfmt` 2188, 2216
`\bypage@Rfalse` 122, 134
`\bypage@Rtrue` 122, 130
C
`\c@ballast` 955

- `\c@firstlinenumR` 153, 1094
`\c@firstsublinenumR` 157, 1089
`\c@linenumincrementR` 153, 1094
`\c@page` ... 564, 1825, 1831, 1834, 1841
`\c@pstartL` 745
`\c@pstartR` 747
`\c@sublinenumincrementR` .. 157, 1089
`\centering` 2405
`\Cfootfmt` 2217
`\Cfootins` 2136
`\Cfootnote` 1174, 2253, 2291
`\Cfootnoterule` 2132
`\ch@ck@l@ckR` 1022
`\ch@cksub@l@ckR` 1022
`\ch@cksub@lockR` 1090
`\chapter` 693, 694, 702
`\chapterinpages` 686, 694, 704
`\chardef` 1471
`\check@goal` 1899, 1913, 1925
`\check@pstarts`
 1635, 1660, 1676, 1731, 1739, 1747
`\checkpageL` 1758, 1771, 1893
`\checkpageR` 1780, 1792, 1893
`\checkraw@text`
 1639, 1656, 1687, 1755, 1801
`\cleardoublepage` 1837
`\clearl@ddleftpage` 1779, 1823
`\clearl@drighthpage` 1800, 1823
`\cleartoevenpage` 1823
`\cleartol@devenpage` 1720, 1823
`\closeout` 555, 559
`\columnrulewidth` 4, 1671, 2004
`\Columns` 3, 1623, 2038, 2098
`\columnseparator` 4, 1653, 1671
`\countLline` 843, 855
`\countRline` 843, 889
`\Cparafootfmt` 2201, 2217
`\critext` 630
- D**
- `\date` 2225
`\DeclareOption` 7
Dekker, Dirk-Jan 78, 84
`\Dfootnote` 1174
`\dimen` 581, 582, 586–588, 592
`\divide` 1027
`\do@actions` 935
`\do@actions@fixedcodeR` 959
`\do@actions@nextR` 959
`\do@actionsR` 920, 959
`\do@ballast` 936
`\do@ballastR` 921, 950
`\do@lineL` 853, 1643, 1647, 1762
`\do@lineLhook` 859, 884
`\do@lineR` 887, 1644, 1649, 1783
`\do@lineRhook` 884, 893
`\do@lockoff` 470
`\do@lockoffL` 494
`\do@lockoffR` 470
`\do@lockon` 435
`\do@lockonL` 467
`\do@lockonR` 435
`\documentclass` 1981, 2117, 2384
`\dp` 2146, 2161
`\dummy@ref` 516
- E**
- `\edfont@info` 669, 672, 678, 681
`\edindex` . 2235, 2250, 2262, 2274, 2275
`\edlabel` . 1316, 2451, 2466, 2525, 2541
`\edtext` 647, 2063,
 2069, 2081, 2085, 2238, 2240,
 2242–2246, 2249, 2251, 2252,
 2261, 2263, 2265, 2270, 2272,
 2273, 2276–2280, 2282, 2283,
 2286, 2288, 2290, 2293–2298,
 2420, 2427, 2464, 2490, 2498, 2539
`\Efootnote` 1174
`\emergencystretch` 2221
`\empty` .. 76, 79, 235, 243, 641, 658,
 667, 676, 766, 795, 1012, 1093,
 1101, 1130–1132, 1143, 1154,
 1319, 1328, 1848, 1854, 1861, 1867
`\end@lemmas` 641, 642, 658, 659
`\endashchar` 1306
`\endgraf` 814, 830, 1629, 1723
`\endline@num` 523, 529
`\endlock` 614, 1480, 1489, 1494
`\endnumbering` 6, 37, 69, 106,
 726, 2020, 2035, 2072, 2095,
 2304, 2371, 2430, 2470, 2503, 2547
`\endnumberingR` 50, 69, 91, 101, 114, 726
`\endpage@num` 522, 529
`\endstanzaextra`
 1496, 2409, 2434, 2478, 2507
`\endsub` 581
`\endsubline@num` 524, 530
`\enskip` 2184, 2197, 2205
`\enspace` 2183, 2193, 2204

environments:

astanza 7, 1474
 Leftside 5, 709
 pages 4, 686
 pairs 3, 686
 Rightside 5, 723
 \everyhbox 1460
 \extensionchars .. 45, 64, 97, 111, 119

F

\f@x@l@cksR 1022
 \first@linenum@out@Rfalse .. 550, 556
 \first@linenum@out@Rtrue 550
 \firstlinenum 5, 162, 2010,
 2025, 2055, 2076, 2395, 2436, 2509
 \firstsublinenum 5, 162
 \fix@page 296, 303
 \flag@end 565, 646, 663
 \flag@start 565, 638, 655
 \floatingpenalty 2145
 \flush@notes 1662, 1810
 \flush@notesR 1152, 1663, 1811
 \footnote
 . 2316, 2324, 2333, 2338, 2355,
 2363, 2444, 2450, 2457, 2460,
 2466, 2517, 2524, 2532, 2535, 2541
 \footnotesize 2127, 2128
 \footparagraph . 2211–2213, 2393, 2394
 \fullstop . 198, 1303, 1305, 1307, 1309

G

\get@linelistfile 231
 \get@nextboxL 1770, 1938
 \get@nextboxR 1791, 1938
 \getline@numL 864, 933
 \getline@numR 898, 918
 \getlinesfrompagelistL
 1743, 1803, 1860
 \getlinesfrompagelistR
 1744, 1804, 1860
 \getlinesfromparlistL ... 1735, 1847
 \getlinesfromparlistR ... 1736, 1847
 \gl@p 238, 239,
 246, 247, 642, 659, 671, 680,
 1015, 1016, 1136, 1140, 1155,
 1322, 1331, 1851, 1857, 1864, 1870
 \goalfraction 4, 1925

H

\hangingsymbol 9, 1462

\hb@xt@ ... 866, 872, 879, 900, 906,
 913, 1651, 1766, 1768, 1787, 1789
 \hsize 776, 805,
 1651, 1766, 1768, 1787, 1789, 2158
 \hyphenation 2226

I

\if@filesw 1580
 \if@firstcolumn 1107, 1413
 \if@nobreak 750, 779
 \if@pstarts 1636, 1676, 1732, 1748
 \ifaraw@text ... 1641, 1687, 1757, 1802
 \ifautopar 775, 804
 \ifbypage@ 319
 \ifbypage@R 122, 309, 994
 \ifdim 582, 586, 588,
 592, 1766, 1787, 1827, 1900, 1914
 \iffirst@linenum@out@R 550, 554
 \ifl@d@dash 1306
 \ifl@d@elin 1308, 1309
 \ifl@d@esl 1309
 \ifl@d@pnum 1303, 1307
 \ifl@d@ssub 1305
 \ifl@dpagefull 1773, 1794, 1893
 \ifl@dpaging 9
 \ifl@dpairing 9, 73
 \ifl@dsamelang 1568, 1642
 \ifl@dsamepage 1761, 1782, 1893
 \ifl@dskipnumber 1084
 \ifl@dusedbabel 1566
 \ifledplinenum 1304, 2192
 \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, 1162, 1176, 1190, 1204,
 1218, 1233, 1247, 1261, 1275,
 1289, 1317, 1349, 1363, 1374,
 1386, 1398, 1433, 1446, 1603, 1613
 \ifnoteschanged@ 83
 \ifnumberedpar@ ... 760, 789, 810,
 826, 1161, 1175, 1189, 1203,
 1217, 1232, 1246, 1260, 1274,
 1288, 1373, 1385, 1397, 1432, 1445
 \ifnumbering 35, 756, 807
 \ifnumberingR 48, 70, 93, 125, 785, 823
 \ifnumberpstart ... 775, 804, 819, 835
 \ifodd 1117, 1423, 1825, 1831, 1834, 1841

- \ifpst@rtedL 30, 764
 - \ifpst@rtedR 30, 793
 - \ifshiftedverses . 5, 1764, 1785, 1926
 - \ifsublines@ 196,
 - 285, 329, 362, 369, 400, 409,
 - 421, 428, 440, 474, 528, 530,
 - 922, 937, 1001, 1087, 1351, 1355
 - \ifvbox 856, 890, 1691, 1694, 1939, 1959
 - \ifwrittenlinesL 1935, 1943
 - \ifwrittenlinesR 1936, 1963
 - \initnumbering@reg 43
 - \insert 2141
 - \insert@count 506, 572, 636,
 - 653, 1169, 1183, 1197, 1211,
 - 1225, 1240, 1254, 1268, 1282,
 - 1296, 1381, 1393, 1405, 1440, 1453
 - \insert@countR 507, 568, 635,
 - 652, 1165, 1179, 1193, 1207,
 - 1221, 1236, 1250, 1264, 1278,
 - 1292, 1377, 1389, 1401, 1436, 1449
 - \insertlines@listR
 - 76, 206, 220, 512, 1132, 1136
 - \inserts@list
 - .. 765, 1168, 1182, 1196, 1210,
 - 1224, 1239, 1253, 1267, 1281,
 - 1295, 1380, 1392, 1404, 1439, 1452
 - \inserts@listR
 - .. 794, 1127, 1130, 1140, 1154,
 - 1155, 1164, 1178, 1192, 1206,
 - 1220, 1235, 1249, 1263, 1277,
 - 1291, 1376, 1388, 1400, 1435, 1448
 - \interfootnotelinepenalty 2144
 - \interlinepenalty 1482, 2144
 - \interstanza
 - 2405, 2409, 2434, 2478, 2507
- L**
- \l@d@nums 669, 672, 678,
 - 681, 1164, 1168, 1178, 1182,
 - 1192, 1196, 1206, 1210, 1220,
 - 1224, 1235, 1239, 1249, 1253,
 - 1263, 1267, 1277, 1281, 1291, 1295
 - \l@d@set 377, 610, 611
 - \l@dbbl@set@language 1577, 1600
 - \l@dbfnote 1431
 - \l@dc@maxchunks 771, 773,
 - 800, 802, 1525, 1535, 1543, 1550
 - \l@dcalc@maxoftwo
 - 1737, 1880, 1950, 1970
 - \l@dcalc@minoftwo .. 1745, 1805, 1880
 - \l@dcalcnun 1022
 - \l@dchecklang 1570, 1640
 - \l@dchset@num 252, 255, 377
 - \l@dcsnote 1372
 - \l@dcsnotetext
 - 1411, 1414, 1416, 1424, 1426
 - \l@demptyd@ta 860, 894
 - \l@dend@stuff 46, 65, 98, 112, 120
 - \l@dgetline@margin 143
 - \l@dgetsidenote@margin 1361
 - \l@dld@ta 868, 902, 1108, 1120
 - \l@dleftbox
 - .. 840, 865, 879, 1652, 1766, 1768
 - \l@dlinenumR 188
 - \l@dlsn@te 871, 905
 - \l@dlsnote 1372
 - \l@dmake@labels 1335
 - \l@dmake@labelsR 1325, 1338
 - \l@dminpagelines
 - 1710, 1746, 1806, 1901, 1915
 - \l@dnumpsstartsL . 39, 770, 771, 773,
 - 775, 1529, 1561, 1624, 1625,
 - 1667, 1679, 1717, 1718, 1815, 1948
 - \l@dnumpsstartsR . 52, 799, 800, 802,
 - 804, 1529, 1562, 1624, 1625,
 - 1668, 1682, 1717, 1718, 1816, 1968
 - \l@doldbbl@set@language 1599
 - \l@doldselectlanguage 1598, 1602, 1607
 - \l@dpagefullfalse 1893
 - \l@dpagefulltrue 1893
 - \l@dpagingfalse 11, 688, 701
 - \l@dpagingtrue 696
 - \l@dpairingfalse 9, 690, 700
 - \l@dpairingtrue 687, 695
 - \l@dparsedendline 2154
 - \l@dparsedstartline . 2149, 2154, 2156
 - \l@dparsedstartpage 2152, 2157
 - \l@dparsefootspec 2148
 - \l@dpscL 856, 861, 1531, 1563, 1633,
 - 1637, 1665, 1679, 1691, 1727,
 - 1733, 1738, 1741, 1749, 1813,
 - 1939, 1941, 1948, 1950, 1952, 1954
 - \l@dpscR 890, 895, 1532, 1564,
 - 1634, 1638, 1666, 1682, 1694,
 - 1728, 1734, 1742, 1750, 1814,
 - 1959, 1961, 1968, 1970, 1972, 1974
 - \l@drrd@ta 872, 906, 1110, 1118
 - \l@drightbox
 - .. 840, 899, 913, 1654, 1787, 1789
 - \l@drsn@te 873, 907

- \l@drsnote 1372
- \l@dsamelangfalse 1568, 1571
- \l@dsamelangtrue 1568, 1574
- \l@dsamepagefalse 1893
- \l@dsamepagetrue 1893
- \l@dssetupmaxlinecounts .. 1542, 1559
- \l@dssetuprawboxes 1534, 1558
- \l@dskipnumberfalse 1085
- \l@dskipnumbertrue 982
- \l@dunhbox@line 872, 906
- \l@dusedbabelfalse 1566, 1595
- \l@dusedbabeltrue 1566, 1597
- \l@duselanguage
 - 1587, 1646, 1648, 1759, 1781
- \l@dzromaxlinecounts ... 1542, 1560
- \l@dzroopenalties 813, 829, 1628, 1722
- \l@pscl 1531
- \l@pscr 1531
- \label@refs
 - 1320, 1322, 1325, 1329, 1331, 1335
- \labelref@list 1328, 1331, 1356
- \labelref@listR 1314, 1319, 1322, 1352
- \language name .. 1578, 1579, 1581–1584
- \last@page@num 317, 323
- \last@page@numR 303
- \lastbox 863, 897
- \lastskip 581, 587
- \Lcolwidth 3, 4, 13, 697, 776,
 - 866, 879, 2005, 2006, 2049, 2050
- \ldots 2068,
 - 2071, 2091, 2094, 2258, 2260, 2289
- \led@err@BadLeftRightPstarts ...
 - 21, 1625, 1718
- \led@err@LeftOnRightPage ... 24, 1835
- \led@err@LineationInNumbered ... 126
- \led@err@NumberingNotStarted ... 87
- \led@err@NumberingShouldHaveStarted
 - 100
- \led@err@NumberingStarted ... 36, 49
- \led@err@PendNoPstart 811, 827
- \led@err@PendNotNumbered ... 808, 824
- \led@err@PstartInPstart ... 761, 790
- \led@err@PstartNotNumbered . 757, 786
- \led@err@RightOnLeftPage ... 24, 1842
- \led@err@TooManyPstarts 18, 772, 801
- \led@mess@NotesChanged 84
- \led@mess@SectionContinued
 - 96, 110, 118
- \led@warn@BadAction 984
- \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 1340
- \ledllfill 872, 906
- \ledmac@error 19, 22, 25, 27
- \ledplinenumfalse 2150
- \ledplinenumtrue ... 2148, 2153, 2155
- \ledRcolfalse 12, 710, 736
- \ledRcoltrue 724
- \ledrlfill 872, 906
- \ledsavedprintlines 7, 1301
- \ledsetnormalparstuff 2181, 2189, 2202
- \ledsidenote 2322
- \ledstrutL 1766, 1768, 1820
- \ledstrutR 1787, 1789, 1820
- \ledthegoal 1900, 1914, 1925
- \leftlinenumR 188, 1108, 1120
- Leftside (environment) 5, 709
- \Leftsidehook 716, 718
- \Leftsidehookend 717, 718
- \lemma 2063, 2289
- \lemmafont 2184, 2205
- \line@list 676, 680
- \line@list@stuff 45, 111
- \line@list@stuffR ... 64, 97, 119, 552
- \line@listR . 79, 206, 219, 530, 667, 671
- \line@margin 148
- \line@marginR 141, 1113
- \line@num 320, 351, 352,
 - 354, 372, 383, 384, 412, 943, 1354
- \line@numR . 57, 195, 202, 257, 291,
 - 310, 345, 346, 348, 365, 379,
 - 380, 403, 523, 527, 929, 995,
 - 1004, 1092, 1094, 1096, 1097, 1350
- \lineation 733, 2121
- \lineationR 124, 733
- \linenum@out 571, 579, 584, 590, 596,
 - 603, 611, 616, 620, 1327, 1701, 1874
- \linenum@outR
 - . 549, 555, 557, 559, 560, 564,
 - 567, 577, 583, 589, 595, 602,
 - 610, 615, 619, 624, 1318, 1706, 1877
- \linenumlist 1093, 1097
- \linenumincrement .. 5, 162, 2011,
 - 2026, 2056, 2077, 2396, 2437, 2510
- \linenummargin
 - 141, 2012, 2027, 2057, 2078, 2122

`\linenumr@p` 1304, 1308, 1350, 1354
`\linenumrepR` 185, 195
`\linenumsep` 190, 192
`\linesinpar@listL`
 211, 227, 537, 1848, 1851
`\linesinpar@listR`
 211, 223, 540, 1854, 1857
`\linesonpage@listL` 228, 544, 1861, 1864
`\linesonpage@listR` 224, 547, 1867, 1870
`\list@clear`
 219–224, 227, 228, 230, 765, 794
`\list@clearing@reg` 226
`\list@create`
 206–209, 211–213, 1127, 1314
`\lock@disp` 1054, 1058, 1063
`\lock@off` 461, 462, 470, 619, 620
`\lock@on` 615, 616
`\longdash` 2391, 2405

M

`\manageparhangingsymbol` 854, 888, 1458
`\maxchunks` 3, 1525, 2388
`\maxdimen` 2158
`\maxlinesinpar@list` 211, 230
`\memorydump` 6, 715, 729
`\memorydumpL` 105, 715
`\memorydumpR` 105, 729
`\message` 44, 63
`\morenoexpands` 2176
`\mpAfootnote` 1231
`\mpBfootnote` 1231
`\mpCfootnote` 1231
`\mpDfootnote` 1231
`\mpEfootnote` 1231
`\mpvAfootnote` 1234, 1238, 1243
`\mpvBfootnote` 1248, 1252, 1257
`\mpvCfootnote` 1262, 1266, 1271
`\mpvDfootnote` 1276, 1280, 1285
`\mpvEfootnote` 1290, 1294, 1299
`\multiply` 1028

N

`\n@num` 498, 624
`\n@num@reg` 504
`\namebox` 856, 861, 890,
 895, 1509, 1691, 1694, 1939, 1959
`\NeedsTeXFormat` 2
`\new@line` 872
`\new@lineR` 563, 906
`\newbox` 741, 840, 841, 1510

`\newcounter` 153, 155, 157, 159, 744, 746
`\newif` 5, 10, 31, 122, 550, 1566, 1568,
 1676, 1687, 1893, 1895, 1935, 1936
`\newnamebox` 1509, 1537, 1538
`\newnamecount` 1520, 1545
`\newwrite` 549
`\next@action` 247
`\next@actionline` 244, 246
`\next@actionlineR`
 236, 238, 953, 991, 1013, 1015
`\next@actionR` 239,
 954, 992, 993, 998, 999, 1007, 1016
`\next@insert` 766
`\next@insertR`
 795, 1131, 1134, 1136, 1139, 1143
`\next@page@num` 324, 395
`\next@page@numR` 61, 260, 262, 314, 392
`\no@expands` 632, 649
`\nobrak` 2174, 2175
`\noindent` 2159, 2234, 2314
`\normal@pars` 72, 769, 798
`\normalbfnoteX` 1444
`\notefontsetup` 2143
`\notenumfont` 2127, 2183, 2191, 2204
`\noteschanged@true`
 77, 80, 668, 677, 1133
`\notetextfont` 2128, 2185, 2198, 2206
`\num@lines` 814, 1629, 1723
`\num@linesR` 740, 830, 1630, 1724
`\numberedpar@true` 777, 806
`\numberingRfalse` 71
`\numberingRtrue` 54, 91, 115
`\numberingtrue` 41, 107
`\numberpstartfalse` 7
`\numberpstarttrue` 7
`\numlabfont` 195
`\numpagelinesL`
 1710, 1763, 1774, 1778, 1901
`\numpagelinesR`
 1710, 1784, 1795, 1799, 1915

O

`\oldBfootfmt` 1987, 1990, 2398, 2401
`\oldchapter` 693, 702
`\one@line` 861, 863, 872
`\one@lineR` 740, 895, 897, 906
`\openout` 557, 560

P

`\page@action` 261, 389, 517

- \page@num 242, 322, 1421
 - \page@numR
 215, 234, 312, 522, 527, 993, 1115
 - \pagegoal 1933
 - \Pages 4, 1714, 2373, 2473, 2550
 - pages (environment) 4, 686
 - \pagetotal 1827, 1900, 1914
 - pairs (environment) 3, 686
 - \par@line 815, 1631, 1725
 - \par@lineR 740, 831, 1632, 1726
 - \para@vfootnote 2140
 - \pausenumbering 727
 - \pausenumberingR 90, 727
 - \pend .. 5, 714, 732, 762, 1495, 2267,
 2299, 2303, 2346, 2366, 2370, 2405
 - \pendL 714, 807
 - \pendR 732, 791, 823
 - \prevgraf
 . 814, 830, 1629, 1630, 1723, 1724
 - \previous@A@number 2168
 - \previous@B@number 2169
 - \previous@C@number 2170
 - \previous@page 2152, 2157, 2171
 - \printindex 2377
 - \printlines
 1312, 1989, 2183, 2191, 2204, 2400
 - \printlinesR 7, 1301, 1989, 2400
 - \ProcessOptions 8
 - \protected@write ... 1324, 1334, 1581
 - \ProvidesPackage 3
 - \pst@rtedLfalse 30, 40
 - \pst@rtedLtrue 108, 767
 - \pst@rtedRfalse 32, 53, 74
 - \pst@rtedRtrue 94, 116, 796
 - \pstart 5, 19, 23, 712, 731, 1497, 2230,
 2269, 2301, 2310, 2348, 2368, 2405
 - \pstartL 712, 743
 - \pstartR 731, 743
- R**
- \rbracket 2174, 2197
 - \Rcolwidth 3, 4,
 13, 698, 805, 900, 913, 2006, 2050
 - \read@linelist 217, 553
 - \rem@inder 1097, 1099–1101
 - \resumenumbering 728
 - \resumenumberingR 90, 728
 - \rightlinenumR 188, 1110, 1118
 - Rightside (environment) 5, 723
 - \Rightsidehook 718, 734
 - \Rightsidehookend 718, 737
 - \rlap 1110, 1118
 - \Rlineflag 7, 183, 195,
 1304, 1308, 1342, 1985, 2002, 2219
 - \rule 1672
- S**
- \sc@n@list 1098, 1100
 - \secdef 707
 - \section@num 42, 44, 45, 109–111
 - \section@numR
 ... 28, 55, 63, 64, 95–97, 117–119
 - \select@language ... 1579, 1581–1584
 - \select@lemmfont 2197
 - \selectlanguage ... 1587, 2231, 2308
 - \set@line 634, 651, 665
 - \set@line@action
 254, 358, 367, 374, 397, 519
 - \setl@dlp@rbox 1414, 1426
 - \setl@drp@rbox 1416, 1424
 - \setline 598, 2234, 2314
 - \setlinenum 606
 - \setnamebox 775, 804, 1509
 - \setprintlines 1302
 - \setstanzaindents
 2000, 2047, 2389, 2438, 2511
 - \shiftedversesfalse 6
 - \shiftedversestrue 7
 - \showlemma 640, 657
 - \sidenote@margin 1366, 1370
 - \sidenote@marginR 1359, 1419
 - \sidenotemargin ... 1359, 2123, 2308
 - \skip 2134–2136
 - \skip@lockoff 462, 470
 - \skipnumbering 8, 623, 2405
 - \skipnumbering@reg 627
 - \smash 1672
 - \splitmaxdepth 2146
 - \splittopskip 858, 892, 2146
 - \stanza ... 2014, 2029, 2059, 2080,
 2412, 2418, 2424, 2440, 2448, 2456
 - \stanza@count 1477, 1491, 1502
 - \stanza@chang 1479, 1504
 - \stanzaindentbase 1502
 - \startlock 614
 - \startstanzahook 1475
 - \startsub 581
 - \sub@action 270, 418, 518
 - \sub@change 62, 264, 265, 271
 - \sub@lock 938

`\sub@lockR` 59, 279, 281, 283,
 286, 436, 442, 443, 445, 446,
 476, 477, 479, 923, 974, 976,
 977, 979, 1035, 1075, 1077, 1079
`\sub@off` 589, 590
`\sub@on` 583, 584
`\subline@num` 197, 320, 337,
 338, 340, 370, 410, 939, 945, 1355
`\subline@numR` 198,
 202, 287, 291, 310, 331, 332,
 334, 363, 401, 524, 528, 924,
 930, 995, 1002, 1088, 1089, 1351
`\sublinenumincrement` 5, 162
`\sublinenumr@p` . 1305, 1309, 1351, 1355
`\sublinenumrepR` 185, 198
`\sublines@false` 60, 268, 964
`\sublines@true` 266, 962
`\sublock@disp` 1037, 1041, 1046
`\symplinenum` 1304
`\sza@penalty` 1486, 1490

T

`\textdagger` 2333
`\textheight` 1982, 2386
`\textit` 2044, 2069, 2071, 2092, 2094,
 2104, 2244, 2257, 2278, 2283,
 2289, 2291, 2293, 2341, 2460, 2535
`\textnormal` 2175
`\textsc` 2043, 2239
`\textwidth` 14, 16, 697, 698, 2005, 2049
`\theledlanguageL` 1572, 1587, 1646, 1759
`\theledlanguageR` 1572, 1587, 1648, 1781
`\thepage` 564, 1325, 1335
`\thepstart` 713, 730
`\thepstartL` 7, 713, 745, 775
`\thepstartR` 7, 730, 747, 804
`\thr@@` 445, 454, 477, 484, 969, 977
`\title` 2224
`\topskip` 1827

U

`\unhbox` 1514,
 1652, 1654, 1766, 1768, 1787, 1789
`\unhnamebox` 1509
`\unvbox` 863, 897, 1516
`\unvnamebox` 1509
`\unvxh` 2160

`\usernamecount`
 . 1478, 1485, 1520, 1552, 1738,
 1941, 1950, 1952, 1961, 1970, 1972
`\usepackage` 1983, 2118–2120, 2385

V

`\vAfootnote` 1163, 1167, 1172
`\value` 1461
`\vbadness` 857, 891
`\vbfnoteX` 1447, 1451
`\vBfootnote` 1177, 1181, 1186
`\vbox` 775, 804, 2158
`\vCfootnote` 1191, 1195, 1200
`\vDfootnote` 1205, 1209, 1214
`\vEfootnote` 1219, 1223, 1228
`\vl@dbfnote` 1434, 1438
`\vl@dcnote` 1399, 1403
`\vl@dlsnote` 1375, 1379
`\vl@drsnote` 1387, 1391
`\vsplit` 861, 895

W

`\wd` 872, 906, 2162
`\writtenlinesLfalse` 1729, 1949
`\writtenlinesLtrue` 1946
`\writtenlinesRfalse` 1730, 1969
`\writtenlinesRtrue` 1966

X

`\x@lemma` 642–644, 659–661
`\xlineref` 2467, 2542
`\xpg@main@language` 1617, 1618
`\xpg@set@language` 1612, 1616
`\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, 1163,
 1167, 1177, 1181, 1191, 1195,
 1205, 1209, 1219, 1223, 1234,
 1238, 1248, 1252, 1262, 1266,
 1276, 1280, 1290, 1294, 1350,
 1354, 1375, 1379, 1387, 1391,
 1399, 1403, 1434, 1438, 1447, 1451

Z

`\z@skip` 2147
`\zz@@@` 1320, 1329

Change History

v0.9	General: Possibility to number <code>\pstart</code> 7	<code>\ledsavedprintlines</code> : Simplified <code>\printlinesR</code> by using <code>\setprintlines</code> 47
v0.1	General: First public release 1	<code>\ledstrutR</code> : Added <code>\ledstrutL</code> and <code>\ledstrutR</code> 62
v0.2	General: Added section of babel re- lated code 54 Fix babel problems 1 <code>\Columns</code> : Added <code>\l@dchecklang</code> and <code>\l@duselanguage</code> to <code>\Columns</code> 57 <code>\Pages</code> : Added <code>\l@duselanguage</code> to <code>\Pages</code> 61	<code>\normalbfnoteX</code> : Removed extraneous spaces from <code>\normalbfnoteX</code> 51 <code>\Pages</code> : Added <code>\ledstrutL</code> to <code>\Pages</code> 61 Added <code>\ledstrutR</code> to <code>\Pages</code> . 61 <code>\Rightsidehookend</code> : Added <code>\Leftsidehook</code> , <code>\Leftsidehookend</code> , <code>\Rightsidehook</code> and <code>\Rightsidehookend</code> 31 <code>\sublinenumrepR</code> : Added <code>\linenumrepR</code> and <code>\sublinenumrepR</code> 16
v0.3	General: Reorganize for ledarab . . 1 <code>\affixline@numR</code> : Changed <code>\affixline@numR</code> to match new ledmac 39 <code>\do@actions@nextR</code> : Used <code>\do@actions@fixedcode</code> in <code>\do@actionsR</code> 38 <code>\do@lineL</code> : Added <code>\do@lineLhook</code> to <code>\do@lineL</code> 35 Simplified <code>\do@lineL</code> by using macros for some common code 35 <code>\do@lineR</code> : Changed <code>\do@lineR</code> similarly to <code>\do@lineL</code> 36 <code>\do@lineRhook</code> : Added <code>\do@lineLhook</code> and <code>\do@lineRhook</code> 36 Leftside: Added hooks into Left- side environment 31 <code>\flag@end</code> : Removed extraneous spaces from <code>\flag@end</code> 27 <code>\ifledRcol</code> : Moved <code>\ifl@dpairing</code> to ledmac 11 <code>\ifpst@rtedR</code> : Moved <code>\ifpst@rtedL</code> to ledmac 12 <code>\l@dlinenumR</code> : Simplified <code>\leftlinenumR</code> and <code>\rightlinenumR</code> by introducing <code>\l@dlinenumR</code> 16 <code>\l@dnumpstartsR</code> : Moved <code>\l@dnumpstartsL</code> to ledmac . 53	v0.3a General: Minor <code>\linenummargin</code> fix 1 <code>\line@marginR</code> : Don't just set <code>\line@marginR</code> in <code>\linenummargin</code> 15 v0.3b General: Improved parallel page balancing 1 <code>\Pages</code> : Added <code>\l@dminpagelines</code> calculation for succeeding page pairs 62 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 <code>\section</code> and other titles in numbered sections 1

v0.6		<code>\hangingsymbol</code> to define the character.	1
	General: Be able to use <code>\chapter</code> in parallel pages.		1
v0.7			
	General: Option ‘ <code>shiftedverses</code> ’ which make there is no blank between two parallel verses with unequal length.		1
v0.8			
	General: Possibility to have a symbol on each hanging of verses, like in the french typography. Redefine the commande		
		<code>\hangingsymbol</code> to define the character.	1
		General: Possibility to number the pstart with the commands <code>\numberpstarttrue</code>	1
		<code>\ifledRcol</code> : Moved <code>\iflledRcol</code> and <code>\ifnumberingR</code> to ledmac	11
		General: The numbering of the pstarts restarts on each <code>\beginnumbering</code>	1