Parallel typesetting for critical editions: the ledpar (deprecated) package[*]

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

This is documentation of deprecated ledpar package. If your start your project, we suggest that you use eledpar instead. If for old projects you can migrate to eledpar, you can continue to use this documentation and the ledpar package.

Abstract

The ledmac package, which is based on the PLAIN TEX 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.

You can subscribe to the eledmac email list in: `https://lists.berlios.de/pipermail/ledmac-users/`

Contents

---

[*]This file (`ledpar.dtx`) has version number v0.14, last revised 2012/08/14.

[†]`herries dot press at earthlink dot net`

[‡]`maieul at maieul dot net`

List of Figures

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 load the package etex, which needs pdflatex or
xelatex. If you `\maxchunks` is too little you can 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` environ-
ment. 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
`\Rcolwidth` columns, respectively. By default, these are:
```
\setlength{\Lcolwidth}{0.45\textwidth}
\setlength{\Rcolwidth}{0.45\textwidth}
```
They may be adjusted if one text tends to be 'bulkier' than the other.

`\columnrulewidth` The macro `\columnseparator` is called between each left/right pair of lines.
`\columnseparator` 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. When you use `\stanza`, the visible rule may shift when a verse has a hanging indent. To prevent shifting, use `\setstanzaindents` outside the `Leftside` or `Rightside` environment.

4   Facing pages

pages   Numbered text that is to be set on facing pages must be within a `pages` environment. Within the environment the text for the lefthand and righthand pages is placed within the `Leftside` and `Rightside` environments, respectively.

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

```
\begin{pages}
\begin{Leftside} ... \end{Leftside}
\begin{Rightside} ... \end{Rightside}
\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
`\Rcolwidth` 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 like-
Rightside

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

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

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

`\pstart`
`\pend`

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

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

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

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

---

[1] when used with ledpatch v0.2 or greater.

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 fol-
\endnumbering lowed by \endnumbering, like:

\beginnumbering

⟨*text*⟩

\endnumbering

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

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

\memorydump The command \memorydump effectively performs an \endumbering 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 refer-
\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
\numberpstarttrue   the \numberpstarttrue command to have it. You can stop the numerotation
\numberpstartfalse   with \numberpstartfalse. You can redefine the commands \thepstartL and
\thepstartL   \thepstartR to change style. The numbering restarts on each \beginnumbering
\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 \sza@0@' it is because you have forgotten to use \setstanzaindents to set the stanza indents.

\skipnumbering   The command \skipnumbering when inserted in a line of parallel text causes the numbering of that particular line to be skipped. This can useful if you are

putting some kind of marker (even if it is only a blank line) between stanzas. Remember, parallel texts must be numbered and this provides a way to slip in an 'unnumbered' line.

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

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

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

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

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

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

```
                      Third in first stanza &
                      Fourth in first stanza &
    \strut &
    \stanzanum{2}\advanceline{-1} First in second stanza &
                      Second in second stanza &
                      Second in second stanza &
                      Third in second stanza &
                      Fourth in second stanza \& 
\end{astanza}
...
```

\hangingsymbol   Like in ledmac, you could redefine the command \hangingsymbol to insert a character in each hanged line. If you use it, you must run LATEXtwo 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 type-set 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 0.13 (2011/11/08).

```
1 ⟨*code⟩
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{ledpar}[2012/08/14 v0.14 ledmac extension for parallel texts]
4
```

With the option 'shiftedverses' a long verse one 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 infor-
mation 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@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
`\memorydumpR` 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; sometimes you want line numbers that start at 1 at each \pstart; other times you want line numbers that start at 1 at the start of each section and increase regardless of page breaks. ledpar lets you choose different schemes for the left and right texts.

\ifbypstart@R   The \ifbypage@R and \ifbypstart@R flag specifie the current lineation system:
\bypstart@Rtrue
\bypstart@Rfalse
\ifbypage@R   • line-of-page : bypstart@R = false and bypage@R = true.
\bypage@Rtrue
\bypage@Rfalse   • line-of-pstart : bypstart@R = true and bypage@R = false.

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

```
122 \newif\ifbypage@R
123 \newif\ifbypstart@R
124    \bypage@Rfalse
125    \bypstart@Rfalse
```

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

```
126 \newcommand*{\lineationR}[1]{{%
127    \ifnumbering
128       \led@err@LineationInNumbered
129    \else
130       \def\@tempa{#1}\def\@tempb{page}%
131       \ifx\@tempa\@tempb
132          \global\bypage@Rtrue
133          \global\bypstart@Rfalse
134       \else
135          \def\@tempb{pstart}%
```

```
136          \ifx\@tempa\@tempb
137              \global\bypage@Rfalse
138              \global\bypstart@Rtrue
139          \else
140           \def@tempb{section}
141           \ifx\@tempa\@tempb
142              \global\bypage@Rfalse
143              \global\bypstart@Rfalse
144           \else
145             \led@warn@BadLineation
146           \fi
147          \fi
148      \fi
149    \fi}}
```

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

```
150 \newcount\line@marginR
151 \renewcommand*{\linenummargin}[1]{{%
152   \l@dgetline@margin{#1}%
153   \ifnum\@l@dtempcntb>\m@ne
154     \ifledRcol
155       \global\line@marginR=\@l@dtempcntb
156     \else
157       \global\line@margin=\@l@dtempcntb
158     \fi
159   \fi}}
```

By default put right text numbers at the right.

```
160 \line@marginR=\@ne
161
```

\c@firstlinenumR   The following counters tell ledmac which right text lines should be printed with
\c@linenumincrementR   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.

```
162 \newcounter{firstlinenumR}
163   \setcounter{firstlinenumR}{5}
164 \newcounter{linenumincrementR}
165   \setcounter{linenumincrementR}{5}
```

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

```
166 \newcounter{firstsublinenumR}
167   \setcounter{firstsublinenumR}{5}
168 \newcounter{sublinenumincrementR}
169   \setcounter{sublinenumincrementR}{5}
170
```

`\firstlinenum`   These are the user's macros for changing (sub) line numbers. They are defined in
`\linenumincrement`   ledmac v0.7, but just in case I have started by `\provide`ing them.
`\firstsublinenum`
`\sublinenumincrement`
```
171 \providecommand*{\firstlinenum}{}
172 \providecommand*{\linenumincrement}{}
173 \providecommand*{\firstsublinenum}{}
174 \providecommand*{\sublinenumincrement}{}
175 \renewcommand*{\firstlinenum}[1]{%
176   \ifledRcol \setcounter{firstlinenumR}{#1}%
177   \else       \setcounter{firstlinenum}{#1}%
178   \fi}
179 \renewcommand*{\linenumincrement}[1]{%
180   \ifledRcol \setcounter{linenumincrementR}{#1}%
181   \else       \setcounter{linenumincrement}{#1}%
182   \fi}
183 \renewcommand*{\firstsublinenum}[1]{%
184   \ifledRcol \setcounter{firstsublinenumR}{#1}%
185   \else       \setcounter{firstsublinenum}{#1}%
186   \fi}
187 \renewcommand*{\sublinenumincrement}[1]{%
188   \ifledRcol \setcounter{sublinenumincrementR}{#1}%
189   \else       \setcounter{sublinenumincrement}{#1}%
190   \fi}
191
```

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

```
192 \newcommand*{\Rlineflag}{R}
193
```

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

```
194 \newcommand*{\linenumrepR}[1]{\@arabic{#1}}
195 \newcommand*{\sublinenumrepR}[1]{\@arabic{#1}}
196
```

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

```
197 \newcommand*{\leftlinenumR}{%
198   \l@dlinenumR
199   \kern\linenumsep}
```

```
200 \newcommand*{\rightlinenumR}{%
201   \kern\linenumsep
202   \l@dlinenumR}
203 \newcommand*{\l@dlinenumR}{%
204   \numlabfont\linenumrepR{\line@numR}\Rlineflag%
205   \ifsublines@
206     \ifnum\subline@num>\z@
207       \unskip\fullstop\sublinenumrepR{\subline@numR}%
208     \fi
209   \fi}
210
```

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
\subline@numR   marginal line numbering and in notes. The count \subline@numR stores a sub-line
\absline@numR   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.

```
211 \newcount\line@numR
212 \newcount\subline@numR
213 \newcount\absline@numR
214
```

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

```
215 \list@create{\line@listR}
216 \list@create{\insertlines@listR}
217 \list@create{\actionlines@listR}
218 \list@create{\actions@listR}
219
```

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

```
220 \list@create{\linesinpar@listL}
221 \list@create{\linesinpar@listR}
222 \list@create{\maxlinesinpar@list}
223
```

\page@numR   The right text page number.

```
224 \newcount\page@numR
225
```

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.

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

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

```
227   \ifledRcol
228     \list@clear{\line@listR}%
229     \list@clear{\insertlines@listR}%
230     \list@clear{\actionlines@listR}%
231     \list@clear{\actions@listR}%
232     \list@clear{\linesinpar@listR}%
233     \list@clear{\linesonpage@listR}
234   \else
235     \list@clearing@reg
236     \list@clear{\linesinpar@listL}%
237     \list@clear{\linesonpage@listL}%
238   \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).

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

Now get the file and interpret it.

```
240   \get@linelistfile{#1}%
241   \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.

```
242   \ifledRcol
243     \global\page@numR=\m@ne
244     \ifx\actionlines@listR\empty
245       \gdef\next@actionlineR{1000000}%
246     \else
247       \gl@p\actionlines@listR\to\next@actionlineR
248       \gl@p\actions@listR\to\next@actionR
249     \fi
250   \else
251     \global\page@num=\m@ne
252     \ifx\actionlines@list\empty
253       \gdef\next@actionline{1000000}%
254     \else
255       \gl@p\actionlines@list\to\next@actionline
256       \gl@p\actions@list\to\next@action
257     \fi
258   \fi}
259
```

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.

```
260 \newcommand{\@l@regR}{%
261   \ifx\l@dchset@num\relax \else
262     \advance\absline@numR \@ne
263     \set@line@action
264     \let\l@dchset@num\relax
265     \advance\absline@numR \m@ne
266     \advance\line@numR \m@ne%   % do we need this?
267   \fi
268   \advance\absline@numR \@ne
269   \ifx\next@page@numR\relax \else
270     \page@action
271     \let\next@page@numR\relax
272   \fi
273   \ifx\sub@change\relax \else
274     \ifnum\sub@change>\z@
275       \sublines@true
276     \else
277       \sublines@false
278     \fi
279     \sub@action
280     \let\sub@change\relax
281   \fi
282   \ifcase\@lockR
283   \or
284     \@lockR \tw@
285   \or\or
286     \@lockR \z@
287   \fi
288   \ifcase\sub@lockR
289   \or
290     \sub@lockR \tw@
291   \or\or
292     \sub@lockR \z@
293   \fi
294   \ifsublines@
```

```
295     \ifnum\sub@lockR<\tw@
296       \advance\subline@numR \@ne
297     \fi
298   \else
299     \ifnum\@lockR<\tw@
300       \advance\line@numR \@ne \subline@numR \z@
301     \fi
302   \fi}
303
304 \renewcommand*{\@l}[2]{%
305   \fix@page{#1}%
306   \ifledRcol
307     \@l@regR
308   \else
309     \@l@reg
310   \fi}
311
```

\last@page@numR   We have to adjust \fix@page to handle parallel texts.
      \fix@page
```
312 \newcount\last@page@numR
313   \last@page@numR=-10000
314 \renewcommand*{\fix@page}[1]{%
315   \ifledRcol
316     \ifnum #1=\last@page@numR
317     \else
318       \ifbypage@R
319         \line@numR \z@ \subline@numR \z@
320       \fi
321       \page@numR=#1\relax
322       \last@page@numR=#1\relax
323       \def\next@page@numR{#1}%
324     \fi
325   \else
326     \ifnum #1=\last@page@num
327     \else
328       \ifbypage@
329         \line@num \z@ \subline@num \z@
330       \fi
331       \page@num=#1\relax
332       \last@page@num=#1\relax
333       \def\next@page@num{#1}%
334     \fi
335   \fi}
336
```

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

```
337 \renewcommand*{\@adv}[1]{%
338   \ifsublines@
```

```
339    \ifledRcol
340      \advance\subline@numR by #1\relax
341      \ifnum\subline@numR<\z@
342        \led@warn@BadAdvancelineSubline
343        \subline@numR \z@
344      \fi
345    \else
346      \advance\subline@num by #1\relax
347      \ifnum\subline@num<\z@
348        \led@warn@BadAdvancelineSubline
349        \subline@num \z@
350      \fi
351    \fi
352  \else
353    \ifledRcol
354      \advance\line@numR by #1\relax
355      \ifnum\line@numR<\z@
356        \led@warn@BadAdvancelineLine
357        \line@numR \z@
358      \fi
359    \else
360      \advance\line@num by #1\relax
361      \ifnum\line@num<\z@
362        \led@warn@BadAdvancelineLine
363        \line@num \z@
364      \fi
365    \fi
366  \fi
367  \set@line@action}
368
```

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

```
369 \renewcommand*{\@set}[1]{%
370   \ifledRcol
371     \ifsublines@
372       \subline@numR=#1\relax
373     \else
374       \line@numR=#1\relax
375     \fi
376     \set@line@action
377   \else
378     \ifsublines@
379       \subline@num=#1\relax
380     \else
381       \line@num=#1\relax
382     \fi
383     \set@line@action
384   \fi}
385
```

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

```
386 \renewcommand*{\l@d@set}[1]{%
387   \ifledRcol
388     \line@numR=#1\relax
389     \advance\line@numR \@ne
390     \def\l@dchset@num{#1}
391   \else
392     \line@num=#1\relax
393     \advance\line@num \@ne
394     \def\l@dchset@num{#1}
395   \fi}
396 \let\l@dchset@num\relax
397
```

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

```
398 \renewcommand*{\page@action}{%
399   \ifledRcol
400     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
401     \xright@appenditem{\next@page@numR}\to\actions@listR
402   \else
403     \xright@appenditem{\the\absline@num}\to\actionlines@list
404     \xright@appenditem{\next@page@num}\to\actions@list
405   \fi}
```

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

```
406 \renewcommand*{\set@line@action}{%
407   \ifledRcol
408     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
409     \ifsublines@
410       \@l@dtempcnta=-\subline@numR
411     \else
412       \@l@dtempcnta=-\line@numR
413     \fi
414     \advance\@l@dtempcnta by -5000\relax
415     \xright@appenditem{\the\@l@dtempcnta}\to\actions@listR
416   \else
417     \xright@appenditem{\the\absline@num}\to\actionlines@list
418     \ifsublines@
419       \@l@dtempcnta=-\subline@num
420     \else
421       \@l@dtempcnta=-\line@num
422     \fi
423     \advance\@l@dtempcnta by -5000\relax
424     \xright@appenditem{\the\@l@dtempcnta}\to\actions@list
425   \fi}
```

426

\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.

```
427 \renewcommand*{\sub@action}{%
428   \ifledRcol
429     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
430     \ifsublines@
431       \xright@appenditem{-1001}\to\actions@listR
432     \else
433       \xright@appenditem{-1002}\to\actions@listR
434     \fi
435   \else
436     \xright@appenditem{\the\absline@num}\to\actionlines@list
437     \ifsublines@
438       \xright@appenditem{-1001}\to\actions@list
439     \else
440       \xright@appenditem{-1002}\to\actions@list
441     \fi
442   \fi}
443
```

\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.

```
444 \newcount\@lockR
445 \newcount\sub@lockR
446
447 \newcommand*{\do@lockonR}{%
448   \xright@appenditem{\the\absline@numR}\to\actionlines@listR
449   \ifsublines@
450     \xright@appenditem{-1005}\to\actions@listR
451     \ifnum\sub@lockR=\z@
452       \sub@lockR \@ne
453     \else
454       \ifnum\sub@lockR=\thr@@
455         \sub@lockR \@ne
456       \fi
457     \fi
458   \else
459     \xright@appenditem{-1003}\to\actions@listR
460     \ifnum\@lockR=\z@
461       \@lockR \@ne
462     \else
463       \ifnum\@lockR=\thr@@
464         \@lockR \@ne
465       \fi
466     \fi
467   \fi}
```

```
468
469 \renewcommand*{\do@lockon}{%
470   \ifx\next\lock@off
471     \global\let\lock@off=\skip@lockoff
472   \else
473     \ifledRcol
474       \do@lockonR
475     \else
476       \do@lockonL
477     \fi
478   \fi}
```

\lock@off   \lock@off adds an entry to the action-code list to turn line number locking off.
\do@lockoff 479
\do@lockoffR 480
\skip@lockoff 481 \newcommand{\do@lockoffR}{%
```
482   \xright@appenditem{\the\absline@numR}\to\actionlines@listR
483   \ifsublines@
484     \xright@appenditem{-1006}\to\actions@listR
485     \ifnum\sub@lockR=\tw@
486       \sub@lockR \thr@@
487     \else
488       \sub@lockR \z@
489     \fi
490   \else
491     \xright@appenditem{-1004}\to\actions@listR
492     \ifnum\@lockR=\tw@
493       \@lockR \thr@@
494     \else
495       \@lockR \z@
496     \fi
497   \fi}
498
499 \renewcommand*{\do@lockoff}{%
500   \ifledRcol
501     \do@lockoffR
502   \else
503     \do@lockoffL
504   \fi}
505 \global\let\lock@off=\do@lockoff
506
```

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

```
507 \providecommand*{\n@num}{}
508 \renewcommand*{\n@num}{%
509   \ifledRcol
510     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
511     \xright@appenditem{-1007}\to\actions@listR
512   \else
```

```
513      \n@num@reg
514    \fi}
515
```

\@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.

```
516       \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, corre-
  sponding 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.

```
517 \renewcommand*{\@ref}[2]{%
518   \ifledRcol
519     \global\insert@countR=#1\relax
520     \loop\ifnum\insert@countR>\z@
521       \xright@appenditem{\the\absline@numR}\to\insertlines@listR
522       \global\advance\insert@countR \m@ne
523     \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.

```
524    \begingroup
525      \let\@ref=\dummy@ref
526      \let\page@action=\relax
527      \let\sub@action=\relax
528      \let\set@line@action=\relax
529      \let\@lab=\relax
530      #2
531      \global\endpage@num=\page@numR
532      \global\endline@num=\line@numR
533      \global\endsubline@num=\subline@numR
534    \endgroup
```

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

```
535      \xright@appenditem%
536        {\the\page@numR|\the\line@numR|%
537         \ifsublines@ \the\subline@numR \else 0\fi|%
538         \the\endpage@num|\the\endline@num|%
539         \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.

```
540    #2
541    \else
```

And when not in right text

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

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

```
544 \providecommand*{\@pend}[1]{}
545 \renewcommand*{\@pend}[1]{%
546    \ifbypstart@\global\line@num=0\fi%
547    \xright@appenditem{#1}\to\linesinpar@listL}
548 \providecommand*{\@pendR}[1]{}
549 \renewcommand*{\@pendR}[1]{%
550    \ifbypstart@R\global\line@numR=0\fi%
551    \xright@appenditem{#1}\to\linesinpar@listR}
552
```

\@lopL   \@lopL{⟨num⟩} adds its argument to the \linesonpage@listL list, and analagously
\@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.

```
553 \providecommand*{\@lopL}[1]{}
554 \renewcommand*{\@lopL}[1]{%
555    \xright@appenditem{#1}\to\linesonpage@listL}
556 \providecommand*{\@lopR}[1]{}
557 \renewcommand*{\@lopR}[1]{%
558    \xright@appenditem{#1}\to\linesonpage@listR}
559
```

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.

```
560 \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

```
561 \newif\iffirst@linenum@out@R
562    \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.

```
563 \newcommand*{\line@list@stuffR}[1]{%
564   \read@linelist{#1}%
565   \iffirst@linenum@out@R
566     \immediate\closeout\linenum@outR
567     \global\first@linenum@out@Rfalse
568     \immediate\openout\linenum@outR=#1
569   \else
570     \closeout\linenum@outR
571     \openout\linenum@outR=#1
572   \fi}
573
```

\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.

```
574 \newcommand*{\new@lineR}{%
575   \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.

```
576 \renewcommand*{\flag@start}{%
577   \ifledRcol
578     \edef\next{\write\linenum@outR{%
579                    \string\@ref[\the\insert@countR][}}%
580     \next
581   \else
582     \edef\next{\write\linenum@out{%
583                    \string\@ref[\the\insert@count][}}%
584     \next
585   \fi}
586 \renewcommand*{\flag@end}{%
587   \ifledRcol
588     \write\linenum@outR{]}%
589   \else
590     \write\linenum@out{]}%
591   \fi}
```

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

```
592 \renewcommand*{\startsub}{\dimen0\lastskip
593   \ifdim\dimen0>0pt \unskip \fi
594   \ifledRcol \write\linenum@outR{\string\sub@on}%
595   \else      \write\linenum@out{\string\sub@on}%
596   \fi
597   \ifdim\dimen0>0pt \hskip\dimen0 \fi}
598 \def\endsub{\dimen0\lastskip
599   \ifdim\dimen0>0pt \unskip \fi
```

```
600    \ifledRcol \write\linenum@outR{\string\sub@off}%
601    \else      \write\linenum@out{\string\sub@off}%
602    \fi
603    \ifdim\dimen0>0pt \hskip\dimen0 \fi}
604
```

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

```
605 \renewcommand*{\advanceline}[1]{%
606    \ifledRcol \write\linenum@outR{\string\@adv[#1]}%
607    \else      \write\linenum@out{\string\@adv[#1]}%
608    \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.

```
609 \renewcommand*{\setline}[1]{%
610    \ifnum#1<\z@
611       \led@warn@BadSetline
612    \else
613       \ifledRcol \write\linenum@outR{\string\@set[#1]}%
614       \else      \write\linenum@out{\string\@set[#1]}%
615       \fi
616    \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.

```
617 \renewcommand*{\setlinenum}[1]{%
618    \ifnum#1<\z@
619       \led@warn@BadSetlinenum
620    \else
621       \ifledRcol \write\linenum@outR{\string\l@d@set[#1]}
622       \else      \write\linenum@out{\string\l@d@set[#1]} \fi
623    \fi}
624
```

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

```
625 \renewcommand*{\startlock}{%
626    \ifledRcol \write\linenum@outR{\string\lock@on}%
627    \else      \write\linenum@out{\string\lock@on}%
628    \fi}
629 \def\endlock{%
630    \ifledRcol \write\linenum@outR{\string\lock@off}%
631    \else      \write\linenum@out{\string\lock@off}%
632    \fi}
633
```

\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.

```
634 \renewcommand*{\skipnumbering}{%
635   \ifledRcol \write\linenum@outR{\string\n@num}%
636             \advanceline{-1}%
637   \else
638     \skipnumbering@reg
639   \fi}
640
```

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.

```
641 \long\def\critext#1#2/{\leavevmode
642   \begingroup
643     \no@expands
644     \xdef\@tag{#1}%
645     \set@line
646     \ifledRcol \global\insert@countR \z@
647     \else      \global\insert@count \z@ \fi
648     \ignorespaces #2\relax
649     \flag@start
650   \endgroup
651   \showlemma{#1}%
652   \ifx\end@lemmas\empty \else
653     \gl@p\end@lemmas\to\x@lemma
654     \x@lemma
655     \global\let\x@lemma=\relax
656   \fi
657   \flag@end}
```

\edtext   And similarly for \edtext.

```
658 \renewcommand{\edtext}[2]{\leavevmode
659   \begingroup
660     \no@expands
661     \xdef\@tag{#1}%
662     \set@line
663     \ifledRcol \global\insert@countR \z@
664     \else      \global\insert@count \z@ \fi
665     \ignorespaces #2\relax
666     \flag@start
667   \endgroup
668   \showlemma{#1}%
669   \ifx\end@lemmas\empty \else
670     \gl@p\end@lemmas\to\x@lemma
671     \x@lemma
672     \global\let\x@lemma=\relax
673   \fi
674   \flag@end}
675
```

\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.

```
676 \renewcommand*{\set@line}{%
677   \ifledRcol
678     \ifx\line@listR\empty
679       \global\noteschanged@true
680       \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
681     \else
682       \gl@p\line@listR\to\@tempb
683       \xdef\l@d@nums{\@tempb|\edfont@info}%
684       \global\let\@tempb=\undefined
685     \fi
686   \else
687     \ifx\line@list\empty
688       \global\noteschanged@true
689       \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
690     \else
691       \gl@p\line@list\to\@tempb
692       \xdef\l@d@nums{\@tempb|\edfont@info}%
693       \global\let\@tempb=\undefined
694     \fi
695   \fi}
696
```

13   Parallel environments

The initial set up for parallel processing is deceptively simple.

<table>
<tr><td>pairs</td><td>The <code>pairs</code> environment is for parallel columns and the <code>pages</code> environment for</td></tr>
<tr><td>pages</td><td>parallel pages.</td></tr>
</table>

pairs
pages

The `pairs` environment is for parallel columns and the `pages` environment for parallel pages.

chapterinpages

```
697 \newenvironment{pairs}{%
698    \l@dpairingtrue
699    \l@dpagingfalse
700 }{%
701    \l@dpairingfalse
702 }
```

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.

```
703 \newenvironment{pages}{%
704    \let\oldchapter\chapter
705    \let\chapter\chapterinpages
706    \l@dpairingtrue
707    \l@dpagingtrue
708    \setlength{\Lcolwidth}{\textwidth}%
709    \setlength{\Rcolwidth}{\textwidth}%
710 }{%
711    \l@dpairingfalse
712    \l@dpagingfalse
713    \let\chapter\oldchapter
714 }
715 \newcommand{\chapterinpages}{\thispagestyle{plain}%
716                  \global\@topnum\z@
717                  \@afterindentfalse
718                  \secdef\@chapter\@schapter}
719
```

ifinstanzaL
ifinstanzaR

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

```
720    \newif\ifinstanzaL
721    \newif\ifinstanzaR
```

Leftside

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

```
722 \newenvironment{Leftside}{%
723    \ledRcolfalse
724    \let\beginnumbering\beginnumbering\setcounter{pstartL}{1}
725    \let\pstart\pstartL
726    \let\thepstart\thepstartL
727    \let\pend\pendL
728    \let\memorydump\memorydumpL
729    \Leftsidehook
730    \let\oldstanza\stanza
```

```
731    \renewcommand{\stanza}{\oldstanza\global\instanzaLtrue}
732 }{
733       \let\stanza\oldstanza
734       \Leftsidehookend}
```

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

```
735 \newcommand*{\Leftsidehook}{}
736 \newcommand*{\Leftsidehookend}{}
737 \newcommand*{\Rightsidehook}{}
738 \newcommand*{\Rightsidehookend}{}
739
```

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.

```
740 \newenvironment{Rightside}{%
741     \ledRcoltrue
742     \let\beginnumbering\beginnumberingR
743     \let\endnumbering\endnumberingR
744     \let\pausenumbering\pausenumberingR
745     \let\resumenumbering\resumenumberingR
746     \let\memorydump\memorydumpR
747     \let\thepstart\thepstartR
748     \let\pstart\pstartR
749     \let\pend\pendR
750     \let\lineation\lineationR
751     \Rightsidehook
752     \let\oldstanza\stanza
753     \renewcommand{\stanza}{\oldstanza\global\instanzaRtrue}
754 }{%
755     \ledRcolfalse
756     \let\stanza\oldstanza
757     \Rightsidehookend
758 }
759
```

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
\one@lineR   decomposition.
\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.

```
760 \newcount\num@linesR
761 \newbox\one@lineR
762 \newcount\par@lineR
```

\pstartL   \pstart starts the paragraph by clearing the \inserts@list list and other rele-
\pstartR   vant 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 num-
            bered.
                 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. The old coun-
            ters are used to have the good reset of the pstart counters at the begining of the
            \Pages command.

```
763
764 \newcounter{pstartL}
765 \newcounter{pstartLold}
766 \renewcommand{\thepstartL}{{\bfseries\@arabic\c@pstartL}. }
767 \newcounter{pstartR}
768 \newcounter{pstartRold}
769 \renewcommand{\thepstartR}{{\bfseries\@arabic\c@pstartR}. }
770
771 \newcommand*{\pstartL}{
772 \if@nobreak
773 \let\@oldnobreak\@nobreaktrue
774 \else
775 \let\@oldnobreak\@nobreakfalse
776 \fi
777 \@nobreaktrue
778   \ifnumbering \else
779     \led@err@PstartNotNumbered
780     \beginnumbering
781   \fi
782   \ifnumberedpar@
783     \led@err@PstartInPstart
784     \pend
```

```
785   \fi
```

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

```
786   \ifpst@rtedL\else
787    \setcounter{pstartLold}{\value{pstartL}}%
788     \list@clear{\inserts@list}%
789     \global\let\next@insert=\empty
790      \global\pst@rtedLtrue
791   \fi
792   \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.

```
793   \global\advance\l@dnumpstartsL \@ne
794   \ifnum\l@dnumpstartsL>\l@dc@maxchunks
795     \led@err@TooManyPstarts
796     \global\l@dnumpstartsL=\l@dc@maxchunks
797   \fi
798   \global\setnamebox{l@dLcolrawbox\the\l@dnumpstartsL}=\vbox\bgroup\ifautopar\else\ifnumbe
799                      \hsize=\Lcolwidth
800   \numberedpar@true}

801 \newcommand*{\pstartR}{
802 \if@nobreak
803 \let\@oldnobreak\@nobreaktrue
804 \else
805 \let\@oldnobreak\@nobreakfalse
806 \fi
807 \@nobreaktrue
808   \ifnumberingR \else
809     \led@err@PstartNotNumbered
810     \beginnumberingR
811   \fi
812   \ifnumberedpar@
813     \led@err@PstartInPstart
814     \pendR
815   \fi
816   \ifpst@rtedR\else
817    \setcounter{pstartRold}{\value{pstartR}}%
818     \list@clear{\inserts@listR}%
819     \global\let\next@insertR=\empty
820     \global\pst@rtedRtrue
821   \fi
822   \begingroup\normal@pars
823   \global\advance\l@dnumpstartsR \@ne
824   \ifnum\l@dnumpstartsR>\l@dc@maxchunks
825     \led@err@TooManyPstarts
826     \global\l@dnumpstartsR=\l@dc@maxchunks
827   \fi
828   \global\setnamebox{l@dRcolrawbox\the\l@dnumpstartsR}=\vbox\bgroup\ifautopar\else\ifnumbe
```

```
829                              \hsize=\Rcolwidth
830      \numberedpar@true}
```

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

```
831 \newcommand*{\pendL}{\ifnumbering \else
832      \led@err@PendNotNumbered
833    \fi
834    \ifnumberedpar@ \else
835      \led@err@PendNoPstart
836    \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.

```
837    \l@dzeropenalties
838    \endgraf\global\num@lines=\prevgraf\egroup
839    \global\par@line=0
```

End the group that was begun in the \pstart.

```
840    \endgroup
841    \ignorespaces
842    \@oldnobreak
843 \ifnumberpstart
844 \addtocounter{pstartL}{1}
845 \fi}
846
```

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

```
847 \newcommand*{\pendR}{\ifnumberingR \else
848      \led@err@PendNotNumbered
849    \fi
850    \ifnumberedpar@ \else
851      \led@err@PendNoPstart
852    \fi
853    \l@dzeropenalties
854    \endgraf\global\num@linesR=\prevgraf\egroup
855    \global\par@lineR=0
856    \endgroup
857    \ignorespaces
858    \@oldnobreak
859 \ifnumberpstart
860 \addtocounter{pstartR}{1}
861 \fi
862 }
863
```

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.

```
864 \newbox\l@dleftbox
865 \newbox\l@drightbox
866
```

`\countLline` We need to know the number of lines processed.
`\countRline`
```
867 \newcount\countLline
868    \countLline \z@
869 \newcount\countRline
870    \countRline \z@
871
```

`\@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`
```
872 \newcount\@donereallinesL
873 \newcount\@donetotallinesL
874 \newcount\@donereallinesR
875 \newcount\@donetotallinesR
876
```

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

```
877 \newcommand*{\do@lineL}{%
878    \advance\countLline \@ne
879    \ifvbox\namebox{l@dLcolrawbox\the\l@dpscL}%
880    {\vbadness=10000
881     \splittopskip=\z@
882     \do@lineLhook
883     \l@demptyd@ta
884     \global\setbox\one@line=\vsplit\namebox{l@dLcolrawbox\the\l@dpscL}
885                         to\baselineskip}%
886    \unvbox\one@line \global\setbox\one@line=\lastbox
887    \getline@numL
888 \ifnum\@lock>\@ne\inserthangingsymboltrue\else\inserthangingsymbolfalse\fi
889    \setbox\l@dleftbox
890    \hb@xt@ \Lcolwidth{%
891     \affixpstart@numL
892      \affixline@num
893      \l@dld@ta
894      \add@inserts
895      \affixside@note
```

```
896     \l@dlsn@te
897     {\ledllfill\hb@xt@ \wd\one@line{\inserthangingsymbolL\new@line\l@dunhbox@line{\one@line}}\correct
898      \l@drsn@te
899    }}%
900    \add@penaltiesL
901    \global\advance\@donereallinesL\@ne
902    \global\advance\@donetotallinesL\@ne
903 \else
904    \setbox\l@dleftbox \hb@xt@ \Lcolwidth{\hspace*{\Lcolwidth}}%
905    \global\advance\@donetotallinesL\@ne
906 \fi}
907
908
```

\do@lineLhook   Hooks, initially empty, into the respective \do@line(L/R) macros.
\do@lineRhook
```
909 \newcommand*{\do@lineLhook}{}
910 \newcommand*{\do@lineRhook}{}
911
```

\do@lineR   The \do@lineR macro is called to do all the processing for a single line of right
text.
```
912 \newcommand*{\do@lineR}{%
913    \advance\countRline \@ne
914    \ifvbox\namebox{l@dRcolrawbox\the\l@dpscR}%
915    {\vbadness=10000
916     \splittopskip=\z@
917     \do@lineRhook
918     \l@demptyd@ta
919     \global\setbox\one@lineR=\vsplit\namebox{l@dRcolrawbox\the\l@dpscR}
920                            to\baselineskip}%
921    \unvbox\one@lineR \global\setbox\one@lineR=\lastbox
922    \getline@numR
923 \ifnum\@lockR>\@ne\inserthangingsymbolRtrue\else\inserthangingsymbolRfalse\fi
924    \setbox\l@drightbox
925    \hb@xt@ \Rcolwidth{%
926     \affixpstart@numR
927     \affixline@numR
928     \l@dld@ta
929     \add@insertsR
930     \affixside@noteR
931     \l@dlsn@te
932     {\correcthangingR\ledllfill\hb@xt@ \wd\one@lineR{\inserthangingsymbolR\new@lineR\l@dunhbox@line{\
933      \l@drsn@te
934    }}%
935    \add@penaltiesR
936    \global\advance\@donereallinesR\@ne
937    \global\advance\@donetotallinesR\@ne
938 \else
939    \setbox\l@drightbox \hb@xt@ \Rcolwidth{\hspace*{\Rcolwidth}}}
940    \global\advance\@donetotallinesR\@ne
```

```
941 \fi}
942
943
```

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.

```
944 \newcommand*{\getline@numR}{%
945 \ifnumberline
946   \global\advance\absline@numR \@ne
947 \fi
948   \do@actionsR
949   \do@ballastR
950 \ifnumberline
951   \ifsublines@
952     \ifnum\sub@lockR<\tw@
953       \global\advance\subline@numR \@ne
954     \fi
955   \else
956     \ifnum\@lockR<\tw@
957       \global\advance\line@numR \@ne
958       \global\subline@numR \z@
959     \fi
960   \fi
961 \fi
962 }
963 \newcommand*{\getline@numL}{%
964 \ifnumberline
965 \global\advance\absline@num \@ne
966 \fi
967   \do@actions
968   \do@ballast
969 \ifnumberline
970   \ifsublines@
971     \ifnum\sub@lock<\tw@
972       \global\advance\subline@num \@ne
973     \fi
974   \else
975     \ifnum\@lock<\tw@
976       \global\advance\line@num \@ne
977       \global\subline@num \z@
978     \fi
979   \fi
980 \fi
981 }
982
983
```

\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.

```
984 \newcommand*{\do@ballastR}{\global\ballast@count=\z@
985   \begingroup
986     \advance\absline@numR \@ne
987     \ifnum\next@actionlineR=\absline@numR
988       \ifnum\next@actionR>-1001
989         \global\advance\ballast@count by -\c@ballast
990       \fi
991     \fi
992   \endgroup}
```

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

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

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

```
993 \newcommand*{\do@actions@fixedcodeR}{%
994   \ifcase\@l@dtempcnta%
995   \or%                    % 1001
996     \global\sublines@true
997   \or%                    % 1002
998     \global\sublines@false
999   \or%                    % 1003
1000     \global\@lockR=\@ne
1001   \or%                    % 1004
1002     \ifnum\@lockR=\tw@
1003       \global\@lockR=\thr@@
1004     \else
1005       \global\@lockR=\z@
1006     \fi
1007   \or%                    % 1005
1008     \global\sub@lockR=\@ne
1009   \or%                    % 1006
1010     \ifnum\sub@lockR=\tw@
1011       \global\sub@lockR=\thr@@
1012     \else
1013       \global\sub@lockR=\z@
1014     \fi
1015   \or%                    % 1007
1016     \l@dskipnumbertrue
1017   \else
1018     \led@warn@BadAction
1019   \fi}
1020
1021
1022 \newcommand*{\do@actionsR}{%
1023   \global\let\do@actions@nextR=\relax
1024   \@l@dtempcntb=\absline@numR
```

```
1025    \ifnum\@l@dtempcntb<\next@actionlineR\else
1026      \ifnum\next@actionR>-1001\relax
1027        \global\page@numR=\next@actionR
1028        \ifbypage@R
1029          \global\line@numR \z@  \global\subline@numR \z@
1030        \fi
1031      \else
1032        \ifnum\next@actionR<-4999\relax    %  9/05 added relax here
1033          \@l@dtempcnta=-\next@actionR
1034          \advance\@l@dtempcnta by -5001\relax
1035          \ifsublines@
1036            \global\subline@numR=\@l@dtempcnta
1037          \else
1038            \global\line@numR=\@l@dtempcnta
1039          \fi
1040        \else
1041          \@l@dtempcnta=-\next@actionR
1042          \advance\@l@dtempcnta by -1000\relax
1043          \do@actions@fixedcodeR
1044        \fi
1045      \fi
1046      \ifx\actionlines@listR\empty
1047        \gdef\next@actionlineR{1000000}%
1048      \else
1049        \gl@p\actionlines@listR\to\next@actionlineR
1050        \gl@p\actions@listR\to\next@actionR
1051        \global\let\do@actions@nextR=\do@actionsR
1052      \fi
1053    \fi
1054    \do@actions@nextR}
1055
```

14.4   Line number printing

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

\ch@cksub@l@ckR

\ch@ck@l@ckR

\f@x@l@cksR

\affixline@numR

```
1056
1057 \providecommand*{\l@dcalcnum}[3]{%
1058   \ifnum #1 > #2\relax
1059     \@l@dtempcnta = #1\relax
1060     \advance\@l@dtempcnta by -#2\relax
1061     \divide\@l@dtempcnta by #3\relax
1062     \multiply\@l@dtempcnta by #3\relax
1063     \advance\@l@dtempcnta by #2\relax
1064   \else
1065     \@l@dtempcnta=#2\relax
1066   \fi}
1067
1068 \newcommand*{\ch@cksub@l@ckR}{%
1069   \ifcase\sub@lockR
```

```
1070    \or
1071      \ifnum\sublock@disp=\@ne
1072        \@l@dtempcntb \z@  \@l@dtempcnta \@ne
1073      \fi
1074    \or
1075      \ifnum\sublock@disp=\tw@
1076      \else
1077        \@l@dtempcntb \z@  \@l@dtempcnta \@ne
1078      \fi
1079    \or
1080      \ifnum\sublock@disp=\z@
1081        \@l@dtempcntb \z@  \@l@dtempcnta \@ne
1082      \fi
1083    \fi}
1084
1085 \newcommand*{\ch@ck@l@ckR}{%
1086    \ifcase\@lockR
1087    \or
1088      \ifnum\lock@disp=\@ne
1089        \@l@dtempcntb \z@  \@l@dtempcnta \@ne
1090      \fi
1091    \or
1092      \ifnum\lock@disp=\tw@
1093      \else
1094        \@l@dtempcntb \z@  \@l@dtempcnta \@ne
1095      \fi
1096    \or
1097      \ifnum\lock@disp=\z@
1098        \@l@dtempcntb \z@  \@l@dtempcnta \@ne
1099      \fi
1100    \fi}
1101
1102 \newcommand*{\f@x@l@cksR}{%
1103    \ifcase\@lockR
1104    \or
1105      \global\@lockR \tw@
1106    \or \or
1107      \global\@lockR \z@
1108    \fi
1109    \ifcase\sub@lockR
1110    \or
1111      \global\sub@lockR \tw@
1112    \or \or
1113      \global\sub@lockR \z@
1114    \fi}
1115
1116
1117 \newcommand*{\affixline@numR}{%
1118 \ifnumberline
1119 \ifl@dskipnumber
```

```
1120    \global\l@dskipnumberfalse
1121 \else
1122   \ifsublines@
1123     \@l@dtempcntb=\subline@numR
1124     \l@dcalcnum{\subline@numR}{\c@firstsublinenumR}{\c@sublinenumincrementR}%
1125     \ch@cksub@lockR
1126   \else
1127     \@l@dtempcntb=\line@numR
1128     \ifx\linenumberlist\empty
1129       \l@dcalcnum{\line@numR}{\c@firstlinenumR}{\c@linenumincrementR}%
1130     \else
1131       \@l@dtempcnta=\line@numR
1132       \edef\rem@inder{,\linenumberlist,\number\line@numR,}%
1133         \edef\sc@n@list{\def\noexpand\sc@n@list
1134         ####1,\number\@l@dtempcnta,####2|{\def\noexpand\rem@inder{####2}}}%
1135       \sc@n@list\expandafter\sc@n@list\rem@inder|%
1136         \ifx\rem@inder\empty\advance\@l@dtempcnta\@ne\fi
1137       \fi
1138       \ch@ck@l@ckR
1139   \fi
1140   \ifnum\@l@dtempcnta=\@l@dtempcntb
1141     \if@twocolumn
1142       \if@firstcolumn
1143         \gdef\l@dld@ta{\llap{{\leftlinenumR}}}%
1144       \else
1145         \gdef\l@drd@ta{\rlap{{\rightlinenumR}}}%
1146       \fi
1147     \else
1148       \@l@dtempcntb=\line@marginR
1149       \ifnum\@l@dtempcntb>\@ne
1150         \advance\@l@dtempcntb by\page@numR
1151       \fi
1152       \ifodd\@l@dtempcntb
1153         \gdef\l@drd@ta{\rlap{{\rightlinenumR}}}%
1154       \else
1155         \gdef\l@dld@ta{\llap{{\leftlinenumR}}}%
1156       \fi
1157     \fi
1158   \fi
1159   \f@x@l@cksR
1160 \fi
1161 \fi}
```

14.5   Pstart number printing in side

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

- Some commands have versions suffixed by R or L.

- The \affixpstart@num and \affixpstart@numR commands are called in

the `\Pages` command. Consequently, the `pstartL` and `pstartR` counters must be reset at the begining of this command.

```
\affixpstart@numL
\affixpstart@numR 1162
  \leftpstartnumR 1163 \newcommand*{\affixpstart@numL}{%
 \rightpstartnumR 1164 \ifsidepstartnum
  \leftpstartnumL 1165 \if@twocolumn
 \rightpstartnumL 1166     \if@firstcolumn
     \ifpstartnumR 1167       \gdef\l@dld@ta{\llap{{\leftpstartnumL}}}%
                  1168       \else
                  1169       \gdef\l@drd@ta{\rlap{{\rightpstartnumL}}}%
                  1170     \fi
                  1171   \else
                  1172     \@l@dtempcntb=\line@margin
                  1173   \ifnum\@l@dtempcntb>\@ne
                  1174     \advance\@l@dtempcntb \page@num
                  1175   \fi
                  1176   \ifodd\@l@dtempcntb
                  1177     \gdef\l@drd@ta{\rlap{{\rightpstartnumL}}}%
                  1178   \else
                  1179     \gdef\l@dld@ta{\llap{{\leftpstartnumL}}}%
                  1180   \fi
                  1181   \fi
                  1182 \fi
                  1183 }
                  1184 \newcommand*{\affixpstart@numR}{%
                  1185 \ifsidepstartnum
                  1186 \if@twocolumn
                  1187     \if@firstcolumn
                  1188       \gdef\l@dld@ta{\llap{{\leftpstartnumR}}}%
                  1189     \else
                  1190       \gdef\l@drd@ta{\rlap{{\rightpstartnumR}}}%
                  1191     \fi
                  1192   \else
                  1193     \@l@dtempcntb=\line@marginR
                  1194   \ifnum\@l@dtempcntb>\@ne
                  1195     \advance\@l@dtempcntb \page@numR
                  1196   \fi
                  1197   \ifodd\@l@dtempcntb
                  1198     \gdef\l@drd@ta{\rlap{{\rightpstartnumR}}}%
                  1199   \else
                  1200     \gdef\l@dld@ta{\llap{{\leftpstartnumR}}}%
                  1201   \fi
                  1202   \fi
                  1203 \fi
                  1204 }
                  1205
                  1206 \newcommand*{\leftpstartnumL}{
                  1207 \ifpstartnum
```

```
1208 \thepstartL
1209 \kern\linenumsep\global\pstartnumfalse\fi
1210 }
1211 \newcommand*{\rightpstartnumL}{
1212 \ifpstartnum\kern\linenumsep
1213 \thepstartL
1214 \global\pstartnumfalse\fi
1215 }
1216 \newif\ifpstartnumR
1217 \pstartnumRtrue
1218 \newcommand*{\leftpstartnumR}{
1219 \ifpstartnumR
1220 \thepstartR
1221 \kern\linenumsep\global\pstartnumRfalse\fi
1222 }
1223 \newcommand*{\rightpstartnumR}{
1224 \ifpstartnumR\kern\linenumsep
1225 \thepstartR
1226 \global\pstartnumRfalse\fi
1227 }
```

14.6   Add insertions to the vertical list

\inserts@listR   \inserts@listR is the list macro that contains the inserts that we save up for
one right text paragraph.

```
1228 \list@create{\inserts@listR}
```

\add@insertsR   The right text version.

\add@inserts@nextR
```
1229 \newcommand*{\add@insertsR}{%
1230    \global\let\add@inserts@nextR=\relax
1231    \ifx\inserts@listR\empty \else
1232      \ifx\next@insertR\empty
1233        \ifx\insertlines@listR\empty
1234          \global\noteschanged@true
1235          \gdef\next@insertR{100000}%
1236        \else
1237          \gl@p\insertlines@listR\to\next@insertR
1238        \fi
1239      \fi
1240      \ifnum\next@insertR=\absline@numR
1241        \gl@p\inserts@listR\to\@insertR
1242        \@insertR
1243        \global\let\@insertR=\undefined
1244        \global\let\next@insertR=\empty
1245        \global\let\add@inserts@nextR=\add@insertsR
1246      \fi
1247    \fi
1248    \add@inserts@nextR}
1249
```

14.7   Penalties

`\add@penaltiesL`
`\add@penaltiesR`

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

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

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

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

```
1250 \newcommand*{\add@penaltiesL}{}
1251 \newcommand*{\add@penaltiesR}{}
1252
```

14.8   Printing leftover notes

\flush@notesR   The `\flush@notesR` macro is called after the entire right text has been sliced up
and sent on to the vertical list.

```
1253 \newcommand*{\flush@notesR}{%
1254   \@xloop
1255     \ifx\inserts@listR\empty \else
1256       \gl@p\inserts@listR\to\@insertR
1257       \@insertR
1258       \global\let\@insertR=\undefined
1259   \repeat}
1260
```

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.

```
1261 \renewcommand*{\Afootnote}[1]{%
1262   \ifnumberedpar@
1263     \ifledRcol
1264       \xright@appenditem{\noexpand\vAfootnote{A}%
1265                     {{\l@d@nums}{\@tag}{#1}}}\to\inserts@listR
1266       \global\advance\insert@countR \@ne
1267     \else
1268       \xright@appenditem{\noexpand\vAfootnote{A}%
1269                     {{\l@d@nums}{\@tag}{#1}}}\to\inserts@list
1270       \global\advance\insert@count \@ne
1271     \fi
```

   Within free text, there's no need to put off making the insertion for this note.
No line numbers are available, so this isn't generally that useful; but you might
want to use it to get around some limitation of ledmac.

```
1272   \else
1273     \vAfootnote{A}{{0|0|0|0|0|0|0}{}{#1}}%
1274   \fi\ignorespaces}
```

\Bfootnote   We need similar commands for the other footnote series.
\Cfootnote
\Dfootnote
\Efootnote
```
1275 \renewcommand*{\Bfootnote}[1]{%
1276   \ifnumberedpar@
1277     \ifledRcol
1278       \xright@appenditem{\noexpand\vBfootnote{B}%
1279                     {{\l@d@nums}{\@tag}{#1}}}\to\inserts@listR
```

```
1280        \global\advance\insert@countR \@ne
1281      \else
1282        \xright@appenditem{\noexpand\vBfootnote{B}%
1283                          {{\l@d@nums}{\@tag}{#1}}}\to\inserts@list
1284        \global\advance\insert@count \@ne
1285      \fi
1286    \else
1287      \vBfootnote{B}{{0|0|0|0|0|0|0}{}{#1}}%
1288    \fi\ignorespaces}
1289 \renewcommand*{\Cfootnote}[1]{%
1290    \ifnumberedpar@
1291      \ifledRcol
1292        \xright@appenditem{\noexpand\vCfootnote{C}%
1293                          {{\l@d@nums}{\@tag}{#1}}}\to\inserts@listR
1294        \global\advance\insert@countR \@ne
1295      \else
1296        \xright@appenditem{\noexpand\vCfootnote{C}%
1297                          {{\l@d@nums}{\@tag}{#1}}}\to\inserts@list
1298        \global\advance\insert@count \@ne
1299      \fi
1300    \else
1301      \vCfootnote{C}{{0|0|0|0|0|0|0}{}{#1}}%
1302    \fi\ignorespaces}
1303 \renewcommand*{\Dfootnote}[1]{%
1304    \ifnumberedpar@
1305      \ifledRcol
1306        \xright@appenditem{\noexpand\vDfootnote{D}%
1307                          {{\l@d@nums}{\@tag}{#1}}}\to\inserts@listR
1308        \global\advance\insert@countR \@ne
1309      \else
1310        \xright@appenditem{\noexpand\vDfootnote{D}%
1311                          {{\l@d@nums}{\@tag}{#1}}}\to\inserts@list
1312        \global\advance\insert@count \@ne
1313      \fi
1314    \else
1315      \vDfootnote{D}{{0|0|0|0|0|0|0}{}{#1}}%
1316    \fi\ignorespaces}
1317 \renewcommand*{\Efootnote}[1]{%
1318    \ifnumberedpar@
1319      \ifledRcol
1320        \xright@appenditem{\noexpand\vEfootnote{E}%
1321                          {{\l@d@nums}{\@tag}{#1}}}\to\inserts@listR
1322        \global\advance\insert@countR \@ne
1323      \else
1324        \xright@appenditem{\noexpand\vEfootnote{E}%
1325                          {{\l@d@nums}{\@tag}{#1}}}\to\inserts@list
1326        \global\advance\insert@count \@ne
1327      \fi
1328    \else
```

```
1329        \vEfootnote{E}{{0|0|0|0|0|0|0}{}{#1}}%
1330   \fi\ignorespaces}
1331
```

\mpAfootnote   For footnotes in minipages and the like, we need a similar series of commands.
\mpBfootnote
\mpCfootnote
\mpDfootnote
\mpEfootnote

```
1332 \renewcommand*{\mpAfootnote}[1]{%
1333   \ifnumberedpar@
1334   \ifledRcol
1335     \xright@appenditem{\noexpand\mpvAfootnote{A}%
1336                       {{\l@d@nums}{\@tag}{#1}}}\to\inserts@listR
1337     \global\advance\insert@countR \@ne
1338   \else
1339     \xright@appenditem{\noexpand\mpvAfootnote{A}%
1340                       {{\l@d@nums}{\@tag}{#1}}}\to\inserts@list
1341     \global\advance\insert@count \@ne
1342   \fi
1343   \else
1344     \mpvAfootnote{A}{{0|0|0|0|0|0|0}{}{#1}}%
1345   \fi\ignorespaces}
1346 \renewcommand*{\mpBfootnote}[1]{%
1347   \ifnumberedpar@
1348   \ifledRcol
1349     \xright@appenditem{\noexpand\mpvBfootnote{B}%
1350                       {{\l@d@nums}{\@tag}{#1}}}\to\inserts@listR
1351     \global\advance\insert@countR \@ne
1352   \else
1353     \xright@appenditem{\noexpand\mpvBfootnote{B}%
1354                       {{\l@d@nums}{\@tag}{#1}}}\to\inserts@list
1355     \global\advance\insert@count \@ne
1356   \fi
1357   \else
1358     \mpvBfootnote{B}{{0|0|0|0|0|0|0}{}{#1}}%
1359   \fi\ignorespaces}
1360 \renewcommand*{\mpCfootnote}[1]{%
1361   \ifnumberedpar@
1362   \ifledRcol
1363     \xright@appenditem{\noexpand\mpvCfootnote{C}%
1364                       {{\l@d@nums}{\@tag}{#1}}}\to\inserts@listR
1365     \global\advance\insert@countR \@ne
1366   \else
1367     \xright@appenditem{\noexpand\mpvCfootnote{C}%
1368                       {{\l@d@nums}{\@tag}{#1}}}\to\inserts@list
1369     \global\advance\insert@count \@ne
1370   \fi
1371   \else
1372     \mpvCfootnote{C}{{0|0|0|0|0|0|0}{}{#1}}%
1373   \fi\ignorespaces}
1374 \renewcommand*{\mpDfootnote}[1]{%
```

```
1375   \ifnumberedpar@
1376   \ifledRcol
1377     \xright@appenditem{\noexpand\mpvDfootnote{D}%
1378                       {{\l@d@nums}{\@tag}{#1}}}\to\inserts@listR
1379     \global\advance\insert@countR \@ne
1380   \else
1381     \xright@appenditem{\noexpand\mpvDfootnote{D}%
1382                       {{\l@d@nums}{\@tag}{#1}}}\to\inserts@list
1383     \global\advance\insert@count \@ne
1384   \fi
1385   \else
1386     \mpvDfootnote{D}{{0|0|0|0|0|0|0}{}{#1}}%
1387   \fi\ignorespaces}
1388 \renewcommand*{\mpEfootnote}[1]{%
1389   \ifnumberedpar@
1390   \ifledRcol
1391     \xright@appenditem{\noexpand\mpvEfootnote{E}%
1392                       {{\l@d@nums}{\@tag}{#1}}}\to\inserts@listR
1393     \global\advance\insert@countR \@ne
1394   \else
1395     \xright@appenditem{\noexpand\mpvEfootnote{E}%
1396                       {{\l@d@nums}{\@tag}{#1}}}\to\inserts@list
1397     \global\advance\insert@count \@ne
1398   \fi
1399   \else
1400     \mpvEfootnote{E}{{0|0|0|0|0|0|0}{}{#1}}%
1401   \fi\ignorespaces}
```

\l@dedendmini

```
1402 \renewcommand*{\l@dedendmini}{%
1403   \ifl@dpairing
1404     \ifledRcol
1405       \flush@notesR
1406     \else
1407       \flush@notes
1408     \fi
1409   \fi
1410   \ifvoid\mpAfootins\else\mpAfootgroup{A}\fi%
1411   \ifvoid\mpBfootins\else\mpBfootgroup{B}\fi%
1412   \ifvoid\mpCfootins\else\mpCfootgroup{C}\fi%
1413   \ifvoid\mpDfootins\else\mpDfootgroup{D}\fi%
1414   \ifvoid\mpEfootins\else\mpEfootgroup{E}\fi}
```

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
```

```
1415 \def\printlinesR#1|#2|#3|#4|#5|#6|#7|{\begingroup
1416   \setprintlines{#1}{#2}{#3}{#4}{#5}{#6}%
1417   \ifl@d@pnum #1\fullstop\fi
1418   \ifledplinenum \linenumr@p{#2}\Rlineflag\else \symplinenum\fi
1419   \ifl@d@ssub \fullstop \sublinenumr@p{#3}\fi
1420   \ifl@d@dash \endashchar\fi
1421   \ifl@d@pnum #4\fullstop\fi
1422   \ifl@d@elin \linenumr@p{#5}\Rlineflag\fi
1423   \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumr@p{#6}\fi
1424 \endgroup}
1425
1426 \let\ledsavedprintlines\printlines
1427
```

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.

```
1428 \list@create{\labelref@listR}
1429
```

\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.

```
1430 \renewcommand*{\edlabel}[1]{\@bsphack
1431   \ifledRcol
1432     \write\linenum@outR{\string\@lab}%
1433     \ifx\labelref@listR\empty
1434       \xdef\label@refs{\zz@@@}%
1435     \else
1436       \gl@p\labelref@listR\to\label@refs
1437     \fi
1438     \ifvmode
1439      \advancelabel@refs
1440     \fi
1441     \protected@write\@auxout{}%
1442       {\string\l@dmake@labelsR\space\thepage|\label@refs|{#1}}%
```

```
1443    \else
1444      \write\linenum@out{\string\@lab}%
1445      \ifx\labelref@list\empty
1446        \xdef\label@refs{\zz@@@}%
1447      \else
1448        \gl@p\labelref@list\to\label@refs
1449      \fi
1450      \ifvmode
1451       \advancelabel@refs
1452      \fi
1453    \protected@write\@auxout{}%
1454      {\string\l@dmake@labels\space\thepage|\label@refs|{#1}}%
1455    \fi
1456    \@esphack}
1457
```

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

```
1458 \def\l@dmake@labelsR#1|#2|#3|#4{%
1459    \expandafter\ifx\csname the@label#4\endcsname \relax\else
1460      \led@warn@DuplicateLabel{#4}%
1461    \fi
1462    \expandafter\gdef\csname the@label#4\endcsname{#1|#2\Rlineflag|#3}%
1463    \ignorespaces}
1464 \AtBeginDocument{%
1465    \def\l@dmake@labelsR#1|#2|#3|#4{}%
1466 }
1467
```

\@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.

```
1468 \renewcommand*{\@lab}{%
1469    \ifledRcol
1470      \xright@appenditem{\linenumr@p{\line@numR}|%
1471        \ifsublines@ \sublinenumr@p{\subline@numR}\else 0\fi}%
1472      \to\labelref@listR
1473    \else
1474      \xright@appenditem{\linenumr@p{\line@num}|%
1475        \ifsublines@ \sublinenumr@p{\subline@num}\else 0\fi}%
1476      \to\labelref@list
1477    \fi}
1478
```

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 1479 \newcount\sidenote@marginR
1480 \renewcommand*{\sidenotemargin}[1]{{%
1481   \l@dgetsidenote@margin{#1}%
1482   \ifnum\@l@dtempcntb>\m@ne
1483   \ifledRcol
1484   \global\sidenote@marginR=\@l@dtempcntb
1485   \else
1486   \global\sidenote@margin=\@l@dtempcntb
1487   \fi
1488   \fi}}
1489 \sidenotemargin{right}
1490 \global\sidenote@margin=\@ne
1491

\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 1492 \renewcommand*{\l@dlsnote}[1]{%
1493   \ifnumberedpar@
1494   \ifledRcol
1495   \xright@appenditem{\noexpand\vl@dlsnote{#1}}%
1496   \to\inserts@listR
1497   \global\advance\insert@countR \@ne
1498   \else
1499   \xright@appenditem{\noexpand\vl@dlsnote{#1}}%
1500   \to\inserts@list
1501   \global\advance\insert@count \@ne
1502   \fi
1503   \fi\ignorespaces}
1504 \renewcommand*{\l@drsnote}[1]{%
1505   \ifnumberedpar@
1506   \ifledRcol
1507   \xright@appenditem{\noexpand\vl@drsnote{#1}}%
1508   \to\inserts@listR
1509   \global\advance\insert@countR \@ne
1510   \else
1511   \xright@appenditem{\noexpand\vl@drsnote{#1}}%
1512   \to\inserts@list
1513   \global\advance\insert@count \@ne
1514   \fi
1515   \fi\ignorespaces}
1516 \renewcommand*{\l@dcsnote}[1]{%
1517   \ifnumberedpar@
1518   \ifledRcol
1519   \xright@appenditem{\noexpand\vl@dcsnote{#1}}%
1520   \to\inserts@listR
1521   \global\advance\insert@countR \@ne
1522   \else
1523   \xright@appenditem{\noexpand\vl@dcsnote{#1}}%

```
1524                              \to\inserts@list
1525      \global\advance\insert@count \@ne
1526    \fi
1527  \fi\ignorespaces}
1528
```

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

```
1529 \newcommand*{\affixside@noteR}{%
1530  \gdef\@templ@d{}%
1531  \ifx\@templ@d\l@dcsnotetext \else
1532    \if@twocolumn
1533      \if@firstcolumn
1534        \setl@dlp@rbox{\l@dcsnotetext}%
1535      \else
1536        \setl@drp@rbox{\l@dcsnotetext}%
1537      \fi
1538    \else
1539      \@l@dtempcntb=\sidenote@marginR
1540      \ifnum\@l@dtempcntb>\@ne
1541        \advance\@l@dtempcntb by\page@num
1542      \fi
1543      \ifodd\@l@dtempcntb
1544        \setl@drp@rbox{\l@dcsnotetext}%
1545      \else
1546        \setl@dlp@rbox{\l@dcsnotetext}%
1547      \fi
1548    \fi
1549  \fi}
1550
```

18   Familiar footnotes

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

```
1551 \renewcommand{\l@dbfnote}[1]{%
1552  \ifnumberedpar@
1553    \ifledRcol
1554      \xright@appenditem{\noexpand\vl@dbfnote{{#1}}{\@thefnmark}}%
1555                          \to\inserts@listR
1556      \global\advance\insert@countR \@ne
1557    \else
1558      \xright@appenditem{\noexpand\vl@dbfnote{{#1}}{\@thefnmark}}%
1559                          \to\inserts@list
1560      \global\advance\insert@count \@ne
1561    \fi
1562  \fi\ignorespaces}
1563
```

\normalbfnoteX

```
1564 \renewcommand{\normalbfnoteX}[2]{%
1565   \ifnumberedpar@
1566     \ifledRcol
1567       \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\@nameuse{thefootnote#1}}}%
1568                         \to\inserts@listR
1569       \global\advance\insert@countR \@ne
1570     \else
1571       \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\@nameuse{thefootnote#1}}}%
1572                         \to\inserts@list
1573       \global\advance\insert@count \@ne
1574     \fi
1575   \fi\ignorespaces}
1576
```

19   Verse

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

\inserthangingsymbolL
\inserthangingsymbolR

```
1577 \newif\ifinserthangingsymbolR
1578 \newcommand{\inserthangingsymbolL}{%
1579 \ifinserthangingsymbol%
1580 \ifinstanzaL%
1581 \hfill\hangingsymbol%
1582 \fi%
1583 \fi}
1584 \newcommand{\inserthangingsymbolR}{%
1585 \ifinserthangingsymbolR%
1586 \ifinstanzaR%
1587 \hfill\hangingsymbol%
1588 \fi%
1589 \fi}
```

When a verse is hanged, the column separator is shifted. To prevent it, the
\do@lineL and \do@lineR commands call \correcthangingL and \correcthangingR
commands. These commands insert horizontal skip which length is equal to the
hang indent.

\correcthangingL
\correcthangingR

```
1590 \newcommand{\correcthangingL}{%
1591 \ifl@dpaging\else%
1592 \ifinstanzaL%
1593 \ifinserthangingsymbol%
1594 \hskip \@ifundefined{sza@0@}{0}{\expandafter%
1595             \noexpand\csname sza@0@\endcsname}\stanzaindentbase%
1596 \fi%
1597 \fi%
```

```
1598 \fi}
1599
1600 \newcommand{\correcthangingR}{%
1601 \ifl@dpaging\else%
1602 \ifinstanzaR%
1603 \ifinserthangingsymbolR%
1604 \hskip \@ifundefined{sza@0@}{0}{\expandafter%
1605          \noexpand\csname sza@0@\endcsname}\stanzaindentbase%
1606 \fi%
1607 \fi%
1608 \fi}
```

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

```
1609   \chardef\next=\catcode`\&
1610   \catcode`\&=\active
1611
```

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

```
1612 \newenvironment{astanza}{%
1613   \startstanzahook
1614   \catcode`\&\active
1615   \global\stanza@count\@ne
1616   \ifnum\usenamecount{sza@0@}=\z@
1617     \let\stanza@hang\relax
1618     \let\endlock\relax
1619   \else
1620 %%%    \interlinepenalty\@M % this screws things up, but I don't know why
1621     \rightskip\z@ plus 1fil\relax
1622   \fi
1623   \ifnum\usenamecount{szp@0@}=\z@
1624     \let\sza@penalty\relax
1625   \fi
1626   \def&{%
1627     \endlock\mbox{}%
1628     \sza@penalty
1629     \global\advance\stanza@count\@ne
1630     \@astanza@line}%
1631   \def\&{%
1632     \endlock\mbox{}
1633     \pend
1634     \endstanzaextra}%
1635   \pstart
1636   \@astanza@line
1637 }{}
1638
```

\@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.

```
1639 \newcommand*{\@astanza@line}{%
1640   \parindent=\csname sza@\number\stanza@count @\endcsname\stanzaindentbase
1641   \par
1642   \stanza@hang%\mbox{}%
1643   \ignorespaces}
1644
```

Lastly reset the modified category codes.

```
1645   \catcode'\&=\next
1646
```

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
\unvnamebox
\namebox

```
1647 \providecommand*{\newnamebox}[1]{%
1648   \expandafter\newbox\csname #1\endcsname}
1649 \providecommand*{\setnamebox}[1]{%
1650   \expandafter\setbox\csname #1\endcsname}
1651 \providecommand*{\unhnamebox}[1]{%
1652   \expandafter\unhbox\csname #1\endcsname}
1653 \providecommand*{\unvnamebox}[1]{%
1654   \expandafter\unvbox\csname #1\endcsname}
1655 \providecommand*{\namebox}[1]{%
1656                   \csname #1\endcsname}
1657
```

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

```
1658 \providecommand*{\newnamecount}[1]{%
1659   \expandafter\newcount\csname #1\endcsname}
1660 \providecommand*{\usenamecount}[1]{%
1661                   \csname #1\endcsname}
1662
```

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

```
1663 \newcount\l@dc@maxchunks
1664 \newcommand{\maxchunks}[1]{\l@dc@maxchunks=#1}
1665     \maxchunks{10}
1666
```

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

```
1667 \newcount\l@dnumpstartsR
1668
```

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

```
1669 \newcount\l@dpscL
1670 \newcount\l@dpscR
1671
```

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

```
1672 \newcommand*{\l@dsetuprawboxes}{%
1673     \@l@dtempcntb=\l@dc@maxchunks
1674     \loop\ifnum\@l@dtempcntb>\z@
1675         \newnamebox{l@dLcolrawbox\the\@l@dtempcntb}
1676         \newnamebox{l@dRcolrawbox\the\@l@dtempcntb}
1677         \advance\@l@dtempcntb \m@ne
1678     \repeat}
1679
```

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

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

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

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

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
\l@dusedbabeltrue
```
1704 \newif\ifl@dusedbabel
1705   \l@dusedbabelfalse
```

\ifl@dsamelang   A flag for checking if the same `babel` language has been used for both the left and
\l@dsamelangfalse   right texts.
\l@dsamelangtrue
```
1706 \newif\ifl@dsamelang
1707   \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.
```
1708 \newcommand*{\l@dchecklang}{%
1709   \l@dsamelangfalse
1710   \edef\@tempa{\theledlanguageL}\edef\@temp{\theledlanguageR}%
1711   \ifx\@tempa\@tempb
1712     \l@dsamelangtrue
1713   \fi}
1714
```

\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 \.

```
1715 \newcommand*{\l@dbbl@set@language}[1]{%
1716   \edef\languagename{#1}%
1717   \select@language{\languagename}%
1718   \if@filesw
1719     \protected@write\@auxout{}{\string\select@language{\languagename}}%
1720     \addtocontents{toc}{\string\select@language{\languagename}}%
1721     \addtocontents{lof}{\string\select@language{\languagename}}%
1722     \addtocontents{lot}{\string\select@language{\languagename}}%
1723   \fi}
1724
```

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

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

```
1725 \providecommand{\selectlanguage}[1]{}
1726 \newcommand*{\l@duselanguage}[1]{}
1727 \gdef\theledlanguageL{}
1728 \gdef\theledlanguageR{}
1729
```

Now do the babel fix or polyglossia, if necessary.

```
1730 \AtBeginDocument{%
1731   \@ifundefined{xpg@main@language}{%
1732     \@ifundefined{bbl@main@language}{%
```

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

```
1733       \l@dusedbabelfalse
1734       \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.

```
1735       \l@dusedbabeltrue
1736       \let\l@doldselectlanguage\selectlanguage
1737       \let\l@doldbbl@set@language\bbl@set@language
1738       \let\bbl@set@language\l@dbbl@set@language
1739       \renewcommand{\selectlanguage}[1]{%
1740         \l@doldselectlanguage{#1}%
1741         \ifledRcol \gdef\theledlanguageR{#1}%
1742         \else      \gdef\theledlanguageL{#1}%
1743         \fi}
```

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

```
1744       \renewcommand*{\l@duselanguage}[1]{%
1745         \l@doldselectlanguage{#1}}
```

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

```
1746        \gdef\theledlanguageL{\bbl@main@language}%
1747        \gdef\theledlanguageR{\bbl@main@language}%
1748        }%
1749    }
```

If on Polyglossia

```
1750    {    \apptocmd{\xpg@set@language}{%
1751            \ifledRcol \gdef\theledlanguageR{#1}%
1752            \else        \gdef\theledlanguageL{#1}%
1753            \fi}%
1754        \let\l@duselanguage\xpg@set@language
1755        \gdef\theledlanguageL{\xpg@main@language}%
1756        \gdef\theledlanguageR{\xpg@main@language}%
1757 % \end{macrocode}
1758 % That's it.
1759 %    \begin{macrocode}
1760 }}
```

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.

```
1761 \newcommand*{\Columns}{%
1762    \setcounter{pstartL}{\value{pstartLold}}
1763    \setcounter{pstartR}{\value{pstartRold}}
1764    \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
1765        \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
1766    \fi
```

Start a group and zero counters, etc.

```
1767    \begingroup
1768        \l@dzeropenalties
1769        \endgraf\global\num@lines=\prevgraf
1770            \global\num@linesR=\prevgraf
1771        \global\par@line=\z@
1772        \global\par@lineR=\z@
1773        \global\l@dpscL=\z@
1774        \global\l@dpscR=\z@
```

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

```
1775        \check@pstarts
1776        \loop\if@pstarts
1777            \global\pstartnumtrue
1778            \global\pstartnumRtrue
```

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

```
1779        \global\advance\l@dpscL \@ne
1780        \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

```
1781        \checkraw@text
1782        \l@dchecklang
1783 {       \loop\ifaraw@text
```

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

```
1784          \ifl@dsamelang
1785            \do@lineL
1786            \do@lineR
1787          \else
1788            \l@duselanguage{\theledlanguageL}%
1789            \do@lineL
1790            \l@duselanguage{\theledlanguageR}%
1791            \do@lineR
1792          \fi
1793          \hb@xt@ \hsize{%
1794           \hfill \unhbox\l@dleftbox
1795             \hfill \columnseparator  \hfill
1796             \unhbox\l@drightbox
1797          }%
1798          \checkraw@text
1799        \repeat}
```

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

```
1800        \@writelinesinparL
1801        \@writelinesinparR
1802      \check@pstarts
1803          \ifbypstart@
1804          \write\linenum@out{\string\@set[1]}
1805        \fi
1806       \ifbypstart@R
1807          \write\linenum@outR{\string\@set[1]}
1808        \fi
1809      \addtocounter{pstartL}{1}
1810      \addtocounter{pstartR}{1}
1811    \repeat
```

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

```
1812    \flush@notes
1813    \flush@notesR
1814  \endgroup
```

```
1815    \global\l@dpscL=\z@
1816    \global\l@dpscR=\z@
1817    \global\l@dnumpstartsL=\z@
1818    \global\l@dnumpstartsR=\z@
1819    \ignorespaces
1820      \global\instanzaLfalse
1821      \global\instanzaRfalse}
1822
```

\columnseparator The separator between line pairs in parallel columns is in the form of a vertical
\columnrulewidth 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).

```
1823 \newcommand*{\columnseparator}{%
1824    \smash{\rule[-0.2\baselineskip]{\columnrulewidth}{1.05\baselineskip}}}
1825 \newdimen\columnrulewidth
1826    \columnrulewidth=\z@
1827
```

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

```
1828 \newif\if@pstarts
1829 \newcommand*{\check@pstarts}{%
1830    \@pstartsfalse
1831    \ifnum\l@dnumpstartsL>\l@dpscL
1832      \@pstartstrue
1833    \else
1834      \ifnum\l@dnumpstartsR>\l@dpscR
1835        \@pstartstrue
1836      \fi
1837    \fi
1838 }
1839
```

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

```
1840 \newif\ifaraw@text
1841    \araw@textfalse
1842 \newcommand*{\checkraw@text}{%
1843    \araw@textfalse
1844    \ifvbox\namebox{l@dLcolrawbox\the\l@dpscL}
1845      \araw@texttrue
1846    \else
1847      \ifvbox\namebox{l@dRcolrawbox\the\l@dpscR}
1848        \araw@texttrue
1849      \fi
1850    \fi
1851 }
1852
```

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

```
1853 \newcommand*{\@writelinesinparL}{%
1854   \edef\next{%
1855     \write\linenum@out{\string\@pend[\the\@donereallinesL]}}%
1856   \next
1857   \global\@donereallinesL \z@}
1858 \newcommand*{\@writelinesinparR}{%
1859   \edef\next{%
1860     \write\linenum@outR{\string\@pendR[\the\@donereallinesR]}}%
1861   \next
1862   \global\@donereallinesR \z@}
1863
```

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`
```
1864 \newcount\numpagelinesL
1865 \newcount\numpagelinesR
1866 \newcount\l@dminpagelines
1867
```

`\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.

```
1868 \newcommand*{\Pages}{%
1869   \setcounter{pstartL}{\value{pstartLold}}
1870   \setcounter{pstartR}{\value{pstartRold}}
1871   \typeout{}
1872   \typeout{*************************** PAGES ***************************}
1873   \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
1874     \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
1875   \fi
```

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

```
1876   \cleartol@devenpage
1877   \begingroup
1878     \l@dzeropenalties
1879     \endgraf\global\num@lines=\prevgraf
1880            \global\num@linesR=\prevgraf
1881     \global\par@line=\z@
1882     \global\par@lineR=\z@
1883     \global\l@dpscL=\z@
1884     \global\l@dpscR=\z@
1885     \writtenlinesLfalse
1886     \writtenlinesRfalse
```

Check if there are chunks to be processed.

```
1887        \check@pstarts
1888        \loop\if@pstarts
```

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

```
1889          \global\advance\l@dpscL \@ne
1890          \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.

```
1891          \getlinesfromparlistL
1892          \getlinesfromparlistR
1893          \l@dcalc@maxoftwo{\@cs@linesinparL}{\@cs@linesinparR}%
1894              {\usenamecount{l@dmaxlinesinpar\the\l@dpscL}}%
1895          \check@pstarts
1896        \repeat
```

Zero the counts again, ready for the next bit.

```
1897        \global\l@dpscL=\z@
1898        \global\l@dpscR=\z@
```

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

```
1899        \getlinesfrompagelistL
1900        \getlinesfrompagelistR
1901        \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
1902            {\l@dminpagelines}%
```

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

```
1903        \check@pstarts
1904        \if@pstarts
```

Increment the chunk counts to get the first pair.

```
1905          \global\advance\l@dpscL \@ne
1906          \global\advance\l@dpscR \@ne
```

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

```
1907          \global\@donereallinesL=\z@
1908          \global\@donetotallinesL=\z@
1909          \global\@donereallinesR=\z@
1910          \global\@donetotallinesR=\z@
```

Start a loop over the boxes (chunks).

```
1911          \checkraw@text
```

```
1912 %        \begingroup
1913 {          \loop\ifaraw@text
```

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

```
1914              \checkpageL
1915              \l@duselanguage{\theledlanguageL}%
1916 %%%          \begingroup
1917 {            \loop\ifl@dsamepage
1918
```

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

```
1919                \do@lineL
1920                \advance\numpagelinesL \@ne
1921                \ifshiftedverses
1922                \ifdim\ht\l@dleftbox>0pt\hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}\fi%
1923                \else
1924                \hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
1925                \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.

```
1926
1927                \get@nextboxL
1928                \checkpageL
1929              \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.

```
1930              \ifl@dpagefull
1931                \@writelinesonpageL{\the\numpagelinesL}%
1932              \else
1933                \@writelinesonpageL{1000}%
1934              \fi
```

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

```
1935              \numpagelinesL \z@
1936              \clearl@dleftpage }%
```

Now do the same for the right text.

```
1937              \checkpageR
1938              \l@duselanguage{\theledlanguageR}%
1939 {            \loop\ifl@dsamepage
1940                \do@lineR
1941                \advance\numpagelinesR \@ne
1942                \ifshiftedverses
1943                \ifdim\ht\l@drightbox>0pt\hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}\fi%
1944                \else
1945                \hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}%
1946                \fi
1947                \get@nextboxR
```

```
1948                \checkpageR
1949              \repeat
1950              \ifl@dpagefull
1951                \@writelinesonpageR{\the\numpagelinesR}%
1952              \else
1953                \@writelinesonpageR{1000}%
1954              \fi
1955              \numpagelinesR=\z@
```

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

```
1956              \clearl@drightpage}
```

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

```
1957              \checkraw@text
1958              \ifaraw@text
1959                \getlinesfrompagelistL
1960                \getlinesfrompagelistR
1961                \l@dcalc@minoftwo{\@cs@linesonpageL}{\@cs@linesonpageR}%
1962                              {\l@dminpagelines}%
1963              \fi
1964          \repeat}
```

We have now output the text from all the chunks.

```
1965      \fi
```

Make sure that there are no inserts hanging around.

```
1966      \flush@notes
1967      \flush@notesR
1968    \endgroup
```

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

```
1969    \global\l@dpscL=\z@
1970    \global\l@dpscR=\z@
1971    \global\l@dnumpstartsL=\z@
1972    \global\l@dnumpstartsR=\z@
1973      \global\instanzaLfalse
1974      \global\instanzaRfalse
1975    \ignorespaces}
1976
```

\ledstrutL   Struts inserted into leftand right text lines.
\ledstrutR
```
1977 \newcommand*{\ledstrutL}{\strut}
1978 \newcommand*{\ledstrutR}{\strut}
1979
```

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

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

```
1980 \providecommand{\cleartoevenpage}[1][\@empty]{%
1981   \clearpage
1982   \ifodd\c@page\hbox{}#1\clearpage\fi}
1983 \newcommand*{\cleartol@devenpage}{%
1984   \ifdim\pagetotal<\topskip%  on an empty page
1985   \else
1986     \clearpage
1987   \fi
1988   \ifodd\c@page\hbox{}\clearpage\fi}
1989 \newcommand*{\clearl@dleftpage}{%
1990   \clearpage
1991   \ifodd\c@page\else
1992     \led@err@LeftOnRightPage
1993     \hbox{}%
1994     \cleardoublepage
1995   \fi}
1996 \newcommand*{\clearl@drightpage}{%
1997   \clearpage
1998   \ifodd\c@page
1999     \led@err@RightOnLeftPage
2000     \hbox{}%
2001     \cleartoevenpage
2002   \fi}
2003
```

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

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

```
2004 \newcommand*{\getlinesfromparlistL}{%
2005   \ifx\linesinpar@listL\empty
2006     \gdef\@cs@linesinparL{0}%
2007   \else
2008     \gl@p\linesinpar@listL\to\@cs@linesinparL
2009   \fi}
2010 \newcommand*{\getlinesfromparlistR}{%
2011   \ifx\linesinpar@listR\empty
2012     \gdef\@cs@linesinparR{0}%
2013   \else
2014     \gl@p\linesinpar@listR\to\@cs@linesinparR
2015   \fi}
2016
```

`\getlinesfrompagelistL`
`\@cs@linesonpageL`
`\getlinesfrompagelistR`
`\@cs@linesonpageR`

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

```
2017 \newcommand*{\getlinesfrompagelistL}{%
2018   \ifx\linesonpage@listL\empty
```

```
2019        \gdef\@cs@linesonpageL{1000}%
2020     \else
2021        \gl@p\linesonpage@listL\to\@cs@linesonpageL
2022     \fi}
2023 \newcommand*{\getlinesfrompagelistR}{%
2024     \ifx\linesonpage@listR\empty
2025        \gdef\@cs@linesonpageR{1000}%
2026     \else
2027        \gl@p\linesonpage@listL\to\@cs@linesonpageR
2028     \fi}
2029
```

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

```
2030 \newcommand*{\@writelinesonpageL}[1]{%
2031     \edef\next{\write\linenum@out{\string\@lopL{#1}}}%
2032     \next}
2033 \newcommand*{\@writelinesonpageR}[1]{%
2034     \edef\next{\write\linenum@outR{\string\@lopR{#1}}}%
2035     \next}
2036
```

\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⟩.

```
2037 \newcommand*{\l@dcalc@maxoftwo}[3]{%
2038     \ifnum #2>#1\relax
2039        #3=#2\relax
2040     \else
2041        #3=#1\relax
2042     \fi}
2043 \newcommand*{\l@dcalc@minoftwo}[3]{%
2044     \ifnum #2<#1\relax
2045        #3=#2\relax
2046     \else
2047        #3=#1\relax
2048     \fi}
2049
```

\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@dpagefull is set FALSE and
\l@dsamepagefalse \ifl@dsamepage is set TRUE. If the page is spatially full then \ifl@dpagefull
\ifl@dpagefull is set TRUE and \ifl@dsamepage is set FALSE. If it is not spatially full but
\l@dpagefulltrue the maximum number of lines have been output then both \ifl@dpagefull and
\l@dpagefullfalse \ifl@dsamepage are set FALSE.
\checkpageL
\checkpageR

```
2050 \newif\ifl@dsamepage
2051     \l@dsamepagetrue
```

```
2052 \newif\ifl@dpagefull
2053 \newcommand*{\checkpageL}{%
2054   \l@dpagefulltrue
2055   \l@dsamepagetrue
2056   \check@goal
2057   \ifdim\pagetotal<\ledthegoal
2058     \ifnum\numpagelinesL<\l@dminpagelines
2059     \else
2060       \l@dsamepagefalse
2061       \l@dpagefullfalse
2062     \fi
2063   \else
2064     \l@dsamepagefalse
2065     \l@dpagefulltrue
2066   \fi}
2067 \newcommand*{\checkpageR}{%
2068   \l@dpagefulltrue
2069   \l@dsamepagetrue
2070   \check@goal
2071   \ifdim\pagetotal<\ledthegoal
2072     \ifnum\numpagelinesR<\l@dminpagelines
2073     \else
2074       \l@dsamepagefalse
2075       \l@dpagefullfalse
2076     \fi
2077   \else
2078     \l@dsamepagefalse
2079     \l@dpagefulltrue
2080   \fi}
2081
```

\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.

```
2082 \newdimen\ledthegoal
2083 \ifshiftedverses
2084     \newcommand*{\goalfraction}{0.95}
2085 \else
2086     \newcommand*{\goalfraction}{0.9}
2087 \fi
2088
2089 \newcommand*{\check@goal}{%
2090   \ledthegoal=\goalfraction\pagegoal}
2091
```

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

\ifwrittenlinesL
```
2092 \newif\ifwrittenlinesL
2093 \newif\ifwrittenlinesR
2094
```

\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.

```
2095 \newcommand*{\get@nextboxL}{%
2096   \ifvbox\namebox{l@dLcolrawbox\the\l@dpscL}% box is not empty
```

The current box is not empty; do nothing.

```
2097   \else%                                              box is empty
```

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

```
2098     \ifnum\usenamecount{l@dmaxlinesinpar\the\l@dpscL}>\@donetotallinesL
2099     \else
```

Sufficient lines have been output.

```
2100       \ifwrittenlinesL
2101       \else
```

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

```
2102         \@writelinesinparL
2103         \writtenlinesLtrue
2104       \fi
2105       \ifnum\l@dnumpstartsL>\l@dpscL
```

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

```
2106         \writtenlinesLfalse
2107         \ifbypstart@
2108          \ifnum\value{pstartL}<\value{pstartLold}
2109          \else
2110            \global\line@num=0
2111          \fi
2112         \fi
2113        \addtocounter{pstartL}{1}
2114        \global\pstartnumtrue
2115        \l@dcalc@maxoftwo{\the\usenamecount{l@dmaxlinesinpar\the\l@dpscL}}%
2116                    {\the\@donetotallinesL}%
2117                    {\usenamecount{l@dmaxlinesinpar\the\l@dpscL}}%
2118        \global\@donetotallinesL \z@
2119        \global\advance\l@dpscL \@ne
2120      \fi
2121    \fi
2122  \fi}
2123 \newcommand*{\get@nextboxR}{%
2124   \ifvbox\namebox{l@dRcolrawbox\the\l@dpscR}% box is not empty
2125   \else%                                              box is empty
2126     \ifnum\usenamecount{l@dmaxlinesinpar\the\l@dpscR}>\@donetotallinesR
2127     \else
2128       \ifwrittenlinesR
2129       \else
2130         \@writelinesinparR
```

```
2131            \writtenlinesRtrue
2132          \fi
2133          \ifnum\l@dnumpstartsR>\l@dpscR
2134            \writtenlinesRfalse
2135            \ifbypstart@R
2136                \ifnum\value{pstartR}<\value{pstartRold}
2137                \else
2138                    \global\line@numR=0
2139                \fi
2140            \fi
2141            \addtocounter{pstartR}{1}
2142            \global\pstartnumRtrue
2143            \l@dcalc@maxoftwo{\the\usenamecount{l@dmaxlinesinpar\the\l@dpscR}}%
2144                            {\the\@donetotallinesR}%
2145                            {\usenamecount{l@dmaxlinesinpar\the\l@dpscR}}%
2146            \global\@donetotallinesR \z@
2147            \global\advance\l@dpscR \@ne
2148          \fi
2149      \fi
2150  \fi}
2151
```

25   The End

¡/code¿

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 2r water,
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 *et reliqua*,	I pray for him—*et reliqua;* 6r
Que Dieu luy doint, et voire, voire!	May God grant him (yes, by God)
8 Ce que je pense . . . *et cetera*.	What I think . . . *et cetera*. 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 praefertur, impignoratis et commissis proinde praeesse cupiens, anno LIIII superius descripto, mense Iunio, una cum iudice, scabinis ceterisque civibus civitatis Novimagensis, pro ipsius et inhabitantium in ea necessitate, commodo et utilitate, ut ecclesia eius parochialis extra civitatem sita destrueretur et infra muros ₅ transferretur ac de novo construeretur, a reverendo patre domino Conrado de Hofsteden, archiepiscopo Coloniensi, licentiam, et a venerabilibus dominis decano et capitulo sanctorum Apostolorum Coloniensi, ipsius ecclesiae ab antiquo veris et pacificis patronis, consensum, citra tamen praeiudicium, damnum aut gravamen iurium et bonorum eorundem, impetravit.   ₁₀

Et exinde liberum locum eiusdem civitatis qui dicitur Hundisburg, de praelibati Wilhelmi Romanorum regis, ipsius fundi domini, consensu, ad aedificandum et consecrandum ecclesiam et coemeterium, eisdem decano et capitulo de expresso eiusdem civitatis assensu libera contradiderunt voluntate, obligantes se ipsi comes et civitas dictis decano et capitulo, quod in recompensationem ₁₅ illius areae infra castrum et portam, quae fuit dos ecclesiae, in qua plebanus habitare solebat—quae tunc per novum fossatum civitatis est destructa—aliam aream competentem et ecclesiae novae, ut praefertur, aedificandae satis contiguam, ipsi plebano darent et assignarent. Et desuper apud dictam ecclesiam sanctorum Apostolorum est littera sigillis ipsorum Ottonis comitis et civitatis ₂₀ 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)

Figure 2: Left page output from `djd17nov.tex`.

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
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 long[4] been
the true and benevolent patrons of the church—but they did not allow Otto to
do anything without their knowledge, nor to infringe their rights, nor to damage
their property.

And so the count and the town voluntarily gave an open space in town called
Hundisburg, which was owned by the aforementioned king William, to the dean
and chapter of St. Apostles' in order to build and consecrate a church and grave-
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
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. //

---

[1]In 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.

[2]Since 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.

[3]Conrad of Hochstaden († 1261) was archbishop of Cologne in 1238–1261. Nijmegen be-
longed to the archdiocese of Cologne until 1559.

[4]They 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*, Stadt-
spuren – Denkmäler in Köln, vol. 19, Köln: J. P. Bachem, 1992.

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

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

Figure 3: Right page output from `djd17nov.tex`.

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

                            ———

5   "Quis tibi, saeve puer, dedit hoc in carmina iuris?
6   Pieridum vates, non tua turba sumus.
7   Quid si praeripiat flavae Vĕnus 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.)

2

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

1R   I was preparing to sing of weapons and violent wars,

2R   in heavy numbers, with the subject matter suited to the verse measure.

3R   The even lines were as long as the odd ones, but Cupid laughed,

4R   they said, and he stole away one foot.[1]

———

5R   "O cruel boy, who gave you the right over poetry?

6R   We poets belong to the Pierides,[2] we are not your folk.

7R   What if Venus should seize away the arms of Minerva with the golden hair,

8R   if Minerva with the golden hair should fan alight the kindled torch of love?

———

9R   Who would approve of Ceres[3] reigning on the woodland ridges,

10R   and of land tilled under the law of the Maid with the quiver[4]?

11R   Who would provide Phoebus with his beautiful hair with a sharp-pointed spear,

12R   while Mars stirs the Aonian lyre?[5]

———

---

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

[2] Muses

[3] Ceres was the Roman goddess of the harvest.

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

[5] Lines 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.

3

Figure 5: First right page output from `djdpoems.tex`.

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

                                                              ———

5   "Quis tibi, saeve puer, dedit hoc in carmina iuris?
6   Pieridum vates, non tua turba sumus.
7   Quid si praeripiat flavae Věnus 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.)

4

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

1R   I was preparing to sing of weapons and violent wars,

2R   in heavy numbers, with the subject matter suited to the verse measure.

3R   The even lines were as long as the odd ones, but Cupid laughed,

4R   they said, and he stole away one foot.[6]

———

5R   "O cruel boy, who gave you the right over poetry?

6R   We poets belong to the Pierides,[7] we are not your folk.

7R   What if Venus should seize away the arms of Minerva with the golden hair,

8R   if Minerva with the golden hair should fan alight the kindled torch of love?

———

9R   Who would approve of Ceres[8] reigning on the woodland ridges,

10R   and of land tilled under the law of the Maid with the quiver[9]?

11R   Who would provide Phoebus with his beautiful hair with a sharp-pointed spear,

12R   while Mars stirs the Aonian lyre?[10]

———

---

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

[7] Muses

[8] Ceres was the Roman goddess of the harvest.

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

[10] Lines 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.

5

Figure 7: Second right page output from djdpoems.tex.

A.1   Parallel column example

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

```
2152 ⟨*villon⟩
2153 %%% villon.tex Example parallel columns
2154 \documentclass{article}
2155 \addtolength{\textheight}{-10\baselineskip}
2156 \usepackage{ledmac,ledpar}
2157 %% Use r instead of R to flag right text line numbers
2158 \renewcommand{\Rlineflag}{r}
2159 %% Use the flag in the notes
2160 \let\oldBfootfmt\Bfootfmt
2161 \renewcommand{\Bfootfmt}[3]{%
2162   \let\printlines\printlinesR
2163   \oldBfootfmt{#1}{#2}{#3}}
2164 \begin{document}
2165
2166 I thought that limericks were peculiarly English, but this appears not
2167 to be the case. As with most limericks this one is by Anonymous.
2168
2169 \vspace*{\baselineskip}
2170
2171 \begin{pairs}
2172 %% no indentation
2173 \setstanzaindents{0,0,0,0,0,0,0,0,0}
2174 %% no number flag
2175 \renewcommand{\Rlineflag}{}
2176 %% draw a rule and widen the columns
2177 \setlength{\columnrulewidth}{0.4pt}
2178 \setlength{\Lcolwidth}{0.46\textwidth}
2179 \setlength{\Rcolwidth}{\Lcolwidth}
2180
2181 \begin{Leftside}
2182 %% set left text line numbering sequence
2183 \firstlinenum{2}
2184 \linenumincrement{2}
2185 \linenummargin{left}
2186 \beginnumbering
2187 \stanza
2188 Il y avait un jeune homme de Dijon, &
2189 Qui n'avait que peu de religion. &
2190 Il dit: 'Quant \'{a} moi, &
2191 Je d\'{e}teste tous les trois, &
2192 Le P\'{e}re, et le Fils, et le Pigeon.' \&
2193 \endnumbering
2194 \end{Leftside}
```

```
2195
2196 \begin{Rightside}
2197 %% different right text line numbering sequence
2198 \firstlinenum{1}
2199 \linenumincrement{2}
2200 \linenummargin{right}
2201 \beginnumbering
2202 \stanza
2203 There was a young man of Dijon, &
2204 Who had only a little religion, &
2205 He said: 'As for me, &
2206 I detest all the three, &
2207 The Father, the Son, and the Pigeon.' \&
2208 \endnumbering
2209 \end{Rightside}
2210
2211 \Columns
2212 \end{pairs}
2213
2214 \vspace*{\baselineskip}
2215
2216     The following is verse \textsc{lxxiii} of Fran\c{c}ois Villon's
2217 \textit{Le Testament} (The Testament), composed in 1461.
2218
2219 %% Allow for hanging indentation for long lines
2220 \setstanzaindents{1,0,0,0,0,0,0,0,0}
2221 %% Columns wider than the default
2222 \setlength{\Lcolwidth}{0.46\textwidth}
2223 \setlength{\Rcolwidth}{\Lcolwidth}
2224 \vspace*{\baselineskip}
2225
2226 \begin{pairs}
2227 \begin{Leftside}
2228 \firstlinenum{2}
2229 \linenumincrement{2}
2230 \linenummargin{left}
2231 \beginnumbering
2232 \stanza
2233 Dieu mercy et Tacque Thibault, &
2234 Qui tant d'eaue froid m'a fait boire, &
2235 Mis en bas lieu, non pas en hault, &
2236 Mengier d'angoisse maints \edtext{poire}{\lemma{poire d'angoisse}%
2237   \Afootnote{This has a triple meaning: literally it is the fruit of the
2238   choke pear,
2239   figuratively it means 'bitter fruit', and it also refers to a torture
2240   instrument.}}, &
2241 Enferr\'{e} \ldots Quant j'en ay memoire, &
2242 Je Prie pour luy \edtext{\textit{et reliqua}}{\Afootnote{and so on}}, &
2243 Que Dieu luy doint, et voire, voire! &
2244 Ce que je pense \ldots \textit{et cetera}. \&
```

```
2245 \endnumbering
2246 \end{Leftside}
2247
2248 \begin{Rightside}
2249 \firstlinenum{2}
2250 \linenumincrement{2}
2251 \linenummargin{right}
2252 \beginnumbering
2253 \stanza
2254 Thanks to God --- and to \edtext{Tacque Thibaud}{%
2255   \Bfootnote{A favourite of Jean, Duc de Berry and loathed for his exactions
2256   and debauchery. Villon uses his name as an insulting nickname for
2257   Thibaud d'Auxigny, the Bishop of Orl\'{e}ans.}}  &
2258 Who made me drink so much \edtext{cold water}{%
2259   \Bfootnote{Can either refer to the normal prison diet of bread and
2260    water or to a common medieval torture which involved forced drinking
2261    of cold water.}}, &
2262 Put me underground instead of higher up &
2263 And made me eat such bitter fruit, &
2264 In chains \ldots When I think of this, &
2265 I pray for him---\textit{et reliqua;} &
2266 May God grant him (yes, by God) &
2267 What I think \ldots \textit{et cetera}. \&
2268 \endnumbering
2269 \end{Rightside}
2270
2271 \Columns
2272 \end{pairs}
2273
2274 \vspace*{\baselineskip}
2275
2276     The translation and notes are by Anthony Bonner,
2277 \textit{The Complete Works of Fran\c{c}ois Villon}, published by
2278 Bantam Books in 1960.
2279
2280 \end{document}
2281
2282 ⟨/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.

```
2283 ⟨∗djd17nov⟩
2284 %%% This is djd17nov.tex, a sample critical text edition
2285 %%% written in LaTeX2e with the ledmac and ledpar packages.
2286 %%% (c) 2003--2004 by Dr. Dirk-Jan Dekker,
```

```
2287 %%% Radboud University, Nijmegen (The Netherlands)
2288 %%% (PRW) Modified slightly by PRW to fit the ledpar manual
2289
2290 \documentclass[10pt, letterpaper, twoside]{article}
2291 \usepackage[latin,english]{babel}
2292 \usepackage{makeidx}
2293 \usepackage{ledmac,ledpar}
2294 \lineation{section}
2295 \linenummargin{inner}
2296 \sidenotemargin{outer}
2297
2298 \makeindex
2299
2300 \renewcommand{\notenumfont}{\footnotesize}
2301 \newcommand{\notetextfont}{\footnotesize}
2302
2303 %\let\Afootnoterule=\relax
2304 \let\Bfootnoterule=\relax
2305 \let\Cfootnoterule=\relax
2306
2307 \addtolength{\skip\Afootins}{1.5mm}
2308 %\addtolength{\skip\Bfootins}{1.5mm}
2309 %\addtolength{\skip\Cfootins}{1.5mm}
2310
2311 \makeatletter
2312
2313 \renewcommand*{\para@vfootnote}[2]{%
2314   \insert\csname #1footins\endcsname
2315   \bgroup
2316     \notefontsetup
2317     \interlinepenalty=\interfootnotelinepenalty
2318     \floatingpenalty=\@MM
2319     \splittopskip=\ht\strutbox \splitmaxdepth=\dp\strutbox
2320     \leftskip=\z@skip \rightskip=\z@skip
2321     \l@dparsefootspec #2\ledplinenumtrue%                   new from here
2322     \ifnum\@nameuse{previous@#1@number}=\l@dparsedstartline\relax
2323      \ledplinenumfalse
2324      \fi
2325     \ifnum\previous@page=\l@dparsedstartpage\relax
2326     \else \ledplinenumtrue \fi
2327     \ifnum\l@dparsedstartline=\l@dparsedendline\relax
2328     \else \ledplinenumtrue \fi
2329     \expandafter\xdef\csname previous@#1@number\endcsname{\l@dparsedstartline}%
2330     \xdef\previous@page{\l@dparsedstartpage}%              to here
2331     \setbox0=\vbox{\hsize=\maxdimen
2332      \noindent\csname #1footfmt\endcsname#2}%
2333     \setbox0=\hbox{\unvxh0}%
2334     \dp0=0pt
2335     \ht0=\csname #1footfudgefactor\endcsname\wd0
2336      \box0
```

```
2337        \penalty0
2338    \egroup
2339 }
2340
2341 \newcommand*{\previous@A@number}{-1}
2342 \newcommand*{\previous@B@number}{-1}
2343 \newcommand*{\previous@C@number}{-1}
2344 \newcommand*{\previous@page}{-1}
2345
2346 \newcommand{\abb}[1]{#1%
2347          \let\rbracket\nobrak\relax}
2348 \newcommand{\nobrak}{\textnormal{}}
2349 \newcommand{\morenoexpands}{%
2350          \let\abb=0%
2351 }
2352
2353 \newcommand{\Aparafootfmt}[3]{%
2354    \ledsetnormalparstuff
2355    \scriptsize
2356    \notenumfont\printlines#1|\enspace
2357 %  \lemmafont#1|#2\enskip
2358    \notetextfont
2359    #3\penalty-10\hskip 1em plus 4em minus.4em\relax}
2360
2361 \newcommand{\Bparafootfmt}[3]{%
2362    \ledsetnormalparstuff
2363    \scriptsize
2364    \notenumfont\printlines#1|%
2365    \ifledplinenum
2366      \enspace
2367    \else
2368      {\hskip 0em plus 0em minus .3em}%
2369    \fi
2370    \select@lemmafont#1|#2\rbracket\enskip
2371    \notetextfont
2372    #3\penalty-10\hskip 1em plus 4em minus.4em\relax }
2373
2374 \newcommand{\Cparafootfmt}[3]{%
2375    \ledsetnormalparstuff
2376    \scriptsize
2377    \notenumfont\printlines#1|\enspace
2378 %  \lemmafont#1|#2\enskip
2379    \notetextfont
2380    #3\penalty-10\hskip 1em plus 4em minus.4em\relax}
2381
2382 \makeatother
2383
2384 \footparagraph{A}
2385 \footparagraph{B}
2386 \footparagraph{C}
```

```
2387
2388 \let\Afootfmt=\Aparafootfmt
2389 \let\Bfootfmt=\Bparafootfmt
2390 \let\Cfootfmt=\Cparafootfmt
2391
2392 \renewcommand*{\Rlineflag}{}
2393
2394 \emergencystretch40pt
2395
2396 \author{Guillelmus de Berchen}
2397 \title{Chronicon Geldriae}
2398 \date{}
2399 \hyphenation{archi-epi-sco-po Huns-dis-brug li-be-ra No-vi-ma-gen-si}
2400 \begin{document}
2401 \begin{pages}
2402 \begin{Leftside}
2403 \beginnumbering\pstart
2404 \selectlanguage{latin}
2405 \section{De ecclesia S. Stephani Novimagensi}
2406
2407 \noindent\setline{1}
2408 Nobilis itaque comes Otto\protect\edindex{Otto II of Guelders}
2409 imperio et dominio Novimagensi sibi, ut praefertur, impignoratis
2410 et commissis
2411 \edtext{proinde}{\Bfootnote{primum D}} praeesse cupiens, anno
2412 \textsc{liiii} superius descripto, mense
2413 Iu\edtext{}{\Afootnote{p.\ 227~R}}nio, una cum iudice, scabinis ceterisque
2414 civibus civitatis Novimagensis, pro ipsius et inhabitantium in ea
2415 necessitate,\edtext{}{\Afootnote{p.\ 97~N}} commodo et utilitate,
2416 ut \edtext{ecclesia eius}{\Bfootnote{ecclesia D: eius eius H}} parochialis
2417 \edtext{\abb{extra civitatem}}{\Bfootnote{\textit{om.}~H}} sita
2418 destrueretur et \edtext{infra}{\Bfootnote{intra D}} muros
2419 \edtext{transfer\edtext{}{\Afootnote{p.\ 129~D}}retur}%
2420 {\Bfootnote{transferreretur NH}}
2421 ac de novo construeretur,
2422 \edtext{a reverendo patre domino
2423 Conrado\protect\edindex{Conrad of Hochstaden} de
2424 \edtext{Hofsteden}{\Bfootnote{Hoffstede D: Hoffsteden H}}, archiepiscopo
2425 \edtext{Coloniensi}{\Bfootnote{Colononiensi H}}, licentiam}%
2426 {\Cfootnote{William is confusing two charters that are five years
2427 apart. Permission from St.\ Apostles' Church in Cologne had been
2428 obtained as early as 1249. Cf.\
2429 Sloet\protect\index{Sloet van de Beele, L.A.J.W.},
2430 \textit{Oorkondenboek} nr.\ 707 (14 November 1249):
2431 ''\ldots{}nos devotionis tue precibus annuentes, ut ipsam ecclesiam
2432 faciens demoliri transferas in locum alium competentem, tibi
2433 auctoritate presentium indulgemus\ldots''}}, et a venerabilibus
2434 \edtext{dominis}{\Bfootnote{viris H}} decano et capitulo sanctorum
2435 Apostolorum\protect\edindex{St. Apostles' (Cologne)}
2436 \edtext{Coloniensi}{\Bfootnote{Coloniae H}}, ipsius ecclesiae ab
```

```
2437 antiquo veris et pacificis patronis, consensum, citra tamen
2438 praeiudicium, damnum aut gravamen \edtext{iurium}{\Bfootnote{virium D}}
2439 et bonorum eorundem, impetravit.
2440 \pend
2441
2442 \pstart
2443 \edtext{Et exinde \edtext{liberum}{\Bfootnote{librum H}}
2444 locum eiusdem civitatis
2445 \edtext{qui}{\Bfootnote{quae D}} dicitur
2446 \edtext{Hundisburg}{\Bfootnote{Hundisburch D: Hundisbrug HMN:
2447 Hunsdisbrug R}}\protect\edindex{Hundisburg},
2448 de praelibati Wilhelmi\protect\edindex{William II of Holland} Romanorum
2449 \edtext{regis}{\Bfootnote{imperatoris D}}, ipsius fundi
2450 do\edtext{}{\Afootnote{f.\ 72v~M}}mini, consensu, ad aedificandum
2451 \edtext{\abb{et consecrandum}}{\Bfootnote{\textit{om.}\ H}}
2452 ecclesi\edtext{}{\Afootnote{p.\ 228~R}}am et coemeterium,
2453 \edtext{eisdem}{\Bfootnote{eiusdem D}} decano et capitulo de expresso
2454 eiusdem civitatis assensu libera contradiderunt voluntate, obligantes
2455 se ipsi \edtext{comes}{\Bfootnote{comites D}} et civitas
2456 \edtext{\abb{dictis}}{\Bfootnote{\textit{om.}\ H}} decano et capitulo,
2457 quod in recompensationem illius areae infra castrum et portam, quae
2458 fuit dos ecclesiae, in qua plebanus habitare solebat---quae
2459 \edtext{tunc}{\Bfootnote{nunc H}} per novum fossatum civitatis est
2460 destructa---aliam aream competentem et ecclesiae novae,
2461 \edtext{ut praefertur, aedificandae}{%
2462 \lemma{\abb{ut\ldots aedificandae}}\Bfootnote{\textit{om.}\ H}} satis
2463 \edtext{contiguam}{\Bfootnote{contiguum M}}, ipsi plebano darent et
2464 assignarent.}{\Cfootnote{Cf.\ Sloet, \textit{Oorkondenboek} nr.\ 762
2465 (June 1254)}} Et desuper
2466 \edtext{\abb{apud}}{\Bfootnote{\textit{om.}\ H}} dictam ecclesiam
2467 sanctorum Apostolorum \edtext{est}{\Bfootnote{et H}}
2468 \edtext{littera}{\Bfootnote{litteram H}} sigillis ipsorum
2469 Ottonis\edtext{}{\Afootnote{p.\ 130~D}} comitis et civitatis
2470 \edtext{Novimagensis}{\Bfootnote{Novimagii D}}
2471 \edtext{sigillata}{\Bfootnote{sigillis communita H}}.
2472 \pend
2473
2474 \pstart
2475  // One additional line to show synchronization. //
2476 \pend
2477 \endnumbering
2478 \end{Leftside}
2479
2480 \begin{Rightside}
2481 \sidenotemargin{right}\selectlanguage{english}
2482 \beginnumbering
2483 \pstart
2484 \addtocounter{section}{-1}%
2485 \leavevmode\section{St.\ Stephen's Church in Nijmegen}
2486
```

2487 `\noindent\setline{1}%`
2488 *After the noble count Otto had taken in pledge the power over*
2489 *Nijmegen,\footnote{In 1247 William II\protect\index{William II of Holland}*
2490 *(1227--1256) count of Holland needed money to fight his way to*
2491 *Aachen\protect\index{Aachen} to be crowned King of the Holy Roman*
2492 *Empire. He gave the town of Nijmegen in pledge to Otto*
2493 *II\protect\index{Otto II of Guelders} (1229--1271) count of Guelders.}*
2494 *like I have written above, he wanted to protect the town. So in June*
2495 *1254\ledsidenote{1254} he and the judge, the sheriffs and other*
2496 *citizens of Nijmegen obtained permission to demolish the parish*
2497 *church that lay outside the town walls,\footnote{Since the early*
2498 *seventh century old St.\ Stephen's church had been located close*
2499 *to the castle, at today's*
2500 *Kelfkensbos\protect\index{Kelfkensbos (Nijmegen)} square.*
2501 *Traces of the church and the presbytery were found during excavations*
2502 *in 1998--1999.} to move it inside the walls and to rebuild it new.*
2503 *This operation was necessary and useful both for Otto himself and*
2504 *for the inhabitants of the town. The reverend father Conrad of*
2505 *Hochstaden, archbishop of*
2506 *Cologne,\footnote{Conrad of Hochstaden ({\textdagger} 1261) was*
2507 *archbishop of Cologne in 1238--1261. Nijmegen belonged to the*
2508 *archdiocese of Cologne until 1559.} gave his permission. So did the*
2509 *reverend dean and canons of the chapter of St.*
2510 *Apostles'\protect\index{St. Apostles' (Cologne)} in Cologne, who had*
2511 *long\footnote{They probably became the patrons when the chapter was*
2512 *established in the early eleventh century. About the church and the*
2513 *chapter, see Gottfried Stracke\protect\index{Stracke, G.},*
2514 *\textit{K\"{o}ln:\ St.\ Aposteln}, Stadtspuren -- Denkm\"{a}ler in*
2515 *K\"{o}ln, vol.\ 19, K\"{o}ln: J.\,P.\ Bachem, 1992.} been the true*
2516 *and benevolent patrons of the church---but they did not allow Otto*
2517 *to do anything without their knowledge, nor to infringe their rights,*
2518 *nor to damage their property.*
2519 `\pend`
2520
2521 `\pstart`
2522 *And so the count and the town voluntarily gave an open space in town*
2523 *called Hundisburg, which was owned by the aforementioned king William,*
2524 *to the dean and chapter of St.\ Apostles' in order to build and*
2525 *consecrate a church and graveyard. King William approved and the*
2526 *town of Nijmegen explicitly expressed its assent. A new ditch was dug*
2527 *on property of the church near the castle and the*
2528 *harbour,\footnote{Nowadays, the exact location of the medieval*
2529 *ditch---and of two Roman ones---can be seen in the pavement of*
2530 *Kelfkensbos\protect\index{Kelfkensbos (Nijmegen)} square.} causing*
2531 *the demolition of the presbytery. In compensation, the count and*
2532 *citizens committed themselves to giving the parish priest another*
2533 *suitable space close enough to the new church that was about to be*
2534 *built. A letter about these transactions, with the seals of count*
2535 *Otto and the town of Nijmegen, is kept at St.\ Apostles'*
2536 *church.\footnote{The original letter is lost. A 15th century*

```
2537 transcription of it is kept at the Historisches Archiv der
2538 Stadt K\"{o}ln (HAStK).}
2539 \pend
2540
2541 \pstart
2542 // One additional line to show synchronization. //
2543 \pend
2544 \endnumbering
2545 \end{Rightside}
2546 \Pages
2547 \end{pages}
2548
2549 %%%%%%%%%%%%%%%%%%%%%%%%
2550 \printindex
2551 \end{document}
2552 %%%%%%%%%%%%%%%%%%
2553
2554 ⟨/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.

```
2555 ⟨*djdpoems⟩
2556 %%% djdpoems.tex  example parallel verses on facing pages
2557 \documentclass{article}
2558 \usepackage{ledmac,ledpar}
2559 \addtolength{\textheight}{-15\baselineskip}
2560
2561 \maxchunks{24} % default value = 10
2562 \setstanzaindents{6,0,1,0,1}
2563
2564 \newcommand{\longdash}{---------}
2565
2566 \footparagraph{A} % for left pages
2567 \footparagraph{B} % for right pages
2568 \firstlinenum{1}
2569 \linenumincrement{1}
2570
2571 \let\oldBfootfmt\Bfootfmt
2572 \renewcommand{\Bfootfmt}[3]{%
2573    \let\printlines\printlinesR
2574    \oldBfootfmt{#1}{#2}{#3}}
```

```
2575
2576 \begin{document}
2577
2578 \newcommand{\interstanza}{\pstart\centering\longdash\skipnumbering\pend}
2579
2580 \begin{pages}
2581 \begin{Leftside}
2582 \def\endstanzaextra{\interstanza}
2583 \beginnumbering
2584
2585 \stanza
2586 Arma gravi numero violentaque bella parabam &
2587  edere, materi\={a} conveniente modis. &
2588 Par erat inferior versus---risisse Cupido &
2589  dicitur atque unum surripuisse pedem. \&
2590
2591 \stanza
2592 ``Quis tibi, saeve puer, dedit hoc in carmina iuris? &
2593  Pieridum vates, non tua turba \edtext{sumus}{\Afootnote{note lost}}. &
2594 Quid si praeripiat flavae V\u{e}nus arma Minervae, &
2595  ventilet accensas flava Minerva faces? \&
2596
2597 \stanza
2598 Quis probet in silvis Cererem regnare iugosis, &
2599  lege pharetratae Virginis arva coli? &
2600 Crinibus insignem quis \edtext{acuta}{\Afootnote{acut\={a} (abl.\ abs.)}}
2601 cuspide Phoebum &
2602  instruat, Aoniam Marte movente lyram? \&
2603 \endnumbering
2604 \end{Leftside}
2605
2606 \begin{Rightside}
2607 \def\endstanzaextra{\interstanza}
2608 \beginnumbering
2609 \firstlinenum{1}
2610 \linenumincrement{1}
2611 \setstanzaindents{6,0,1,0,1,0}
2612
2613 \stanza
2614 I was preparing to sing of weapons and violent wars, &
2615 in heavy numbers, with the subject matter suited to the verse measure. &
2616 The even lines were as long as the odd ones, but Cupid laughed, &
2617 they said, and he stole away one foot.\footnote{I.e., the even lines,
2618 which were hexameters (with six feet) became pentameters
2619 (with five feet).} \&
2620
2621 \stanza
2622 ``O cruel boy, who gave you the right over poetry? &
2623 We poets belong to the Pierides,\footnote{Muses} we are not your folk. &
2624 \edlabel{beginparadox}What if Venus should seize away the arms of
```

```
2625 Minerva with the golden hair, &
2626  if Minerva with the golden hair should fan alight the kindled torch
2627 of love? \&
2628
2629 \stanza
2630 Who would approve of Ceres\footnote{Ceres was the Roman goddess of
2631 the harvest.} reigning on the woodland ridges, &
2632  and of land tilled under the law of the Maid with the
2633 quiver\footnote{By '\textit{Virgo}' ('Virgin') Ovid means Diana, the
2634 Roman goddess of the hunt.}? &
2635 Who would provide Phoebus with his beautiful hair with a sharp-pointed
2636 spear, &
2637  while Mars stirs the \edtext{Aonian}{\Bfootnote{Mount Parnassus,
2638 where the Muses live, is located in Aonia.}}
2639 lyre?\edlabel{endparadox}\footnote{Lines
2640 \xlineref{beginparadox}--\xlineref{endparadox} show some paradoxical
2641 situations that would occur if the gods didn't stay with their own
2642 business.} \&
2643 \endnumbering
2644 \end{Rightside}
2645
2646 \Pages
2647 \end{pages}
2648
2649 \begin{pages}
2650 \begin{Leftside}
2651 \def\endstanzaextra{\interstanza}
2652 \beginnumbering
2653
2654 \begin{astanza}
2655 Arma gravi numero violentaque bella parabam &
2656  edere, materi\={a} conveniente modis. &
2657 Par erat inferior versus---risisse Cupido &
2658  dicitur atque unum surripuisse pedem. \&
2659 \end{astanza}
2660
2661 \begin{astanza}
2662 ''Quis tibi, saeve puer, dedit hoc in carmina iuris? &
2663  Pieridum vates, non tua turba \edtext{sumus}{\Afootnote{note lost}}. &
2664 Quid si praeripiat flavae V\u{e}nus arma Minervae, &
2665  ventilet accensas flava Minerva faces? \&
2666 \end{astanza}
2667
2668 \begin{astanza}
2669 Quis probet in silvis Cererem regnare iugosis, &
2670  lege pharetratae Virginis arva coli? &
2671 Crinibus insignem quis \edtext{acuta}{\Afootnote{acut\={a} (abl.\ abs.)}}
2672 cuspide Phoebum &
2673  instruat, Aoniam Marte movente lyram? \&
2674 \end{astanza}
```

2675
2676 \endnumbering
2677 \end{Leftside}
2678
2679 \begin{Rightside}
2680 \def\endstanzaextra{\interstanza}
2681 \beginnumbering
2682 \firstlinenum{1}
2683 \linenumincrement{1}
2684 \setstanzaindents{6,0,1,0,1,0}
2685
2686 \begin{astanza}
2687 *I was preparing to sing of weapons and violent wars, &*
2688 *in heavy numbers, with the subject matter suited to the verse measure. &*
2689 *The even lines were as long as the odd ones, but Cupid laughed, &*
2690 *they said, and he stole away one foot.\footnote{I.e., the even lines,*
2691 *which were hexameters (with six feet) became pentameters*
2692 *(with five feet).} \&*
2693 \end{astanza}
2694
2695 \begin{astanza}
2696 *''O cruel boy, who gave you the right over poetry? &*
2697 *We poets belong to the Pierides,\footnote{Muses} we are not your folk. &*
2698 \edlabel{beginparadox}*What if Venus should seize away the arms of*
2699 *Minerva with the golden hair, &*
2700 * if Minerva with the golden hair should fan alight the kindled torch*
2701 *of love? \&*
2702 \end{astanza}
2703
2704 \begin{astanza}
2705 *Who would approve of Ceres\footnote{Ceres was the Roman goddess of the*
2706 *harvest.} reigning on the woodland ridges, &*
2707 * and of land tilled under the law of the Maid with the*
2708 *quiver\footnote{By '\textit{Virgo}' ('Virgin') Ovid means Diana,*
2709 *the Roman goddess of the hunt.}? &*
2710 *Who would provide Phoebus with his beautiful hair with a sharp-pointed*
2711 *spear, &*
2712 * while Mars stirs the \edtext{Aonian}{\Bfootnote{Mount Parnassus, where*
2713 *the Muses live, is located in Aonia.}}*
2714 *lyre?\edlabel{endparadox}\footnote{Lines*
2715 *\xlineref{beginparadox}--\xlineref{endparadox} show some paradoxical*
2716 *situations that would occur if the gods didn't stay with their*
2717 *own business.} \&*
2718 \end{astanza}
2719
2720 \endnumbering
2721 \end{Rightside}
2722
2723 \Pages
2724 \end{pages}

```
2725
2726 \end{document}
2727
2728 ⟨/djdpoems⟩
```

References

[LW90]   John Lavagnino and Dominik Wujastyk. 'An overview of EDMAC: a PLAIN TeX format for critical editions'. *TUGboat*, **11**, 4, pp. 623–643, November 1990. (Code available from CTAN in `macros/plain/contrib/edmac`)

[Wil02]   Peter Wilson. *The memoir class for configurable typesetting.* November 2002. (Available from CTAN in `macros/latex/contrib/memoir`)

[Wil04]   Peter Wilson. *ledmac A presumptuous attempt to port ED-MAC,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.

Change History

v0.1
    General: First public release 1
v0.10
    General: \edlabel commands on
        the right side are now correctly
        indicated. 1
        \edlabel commands which start
        a paragraph are now put in the
        right place. 1
v0.11
    General: Change \do@lineL and

\do@lineR to allow line num-
bering by pstart(like in ledmac
0.15). 36
Lineation can be by pstart (like
in ledmac 0.15). 14
New management of hang-
ingsymbol insertion, preventing
undesirable insertions. 54
Prevent shift of column separator
when a verse is hanged 54
\affixline@numR: Changed