Parallel typesetting for critical editions: the ledpar package*

Peter Wilson
Herries Press†
Maïeul Rouquette‡

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.

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

Contents

*This file (ledpar.dtx) has version number v0.6, last revised 2011/08/22.

†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 0.6 (2004/12/10). You do not need to read the source code for this package in order to use it but doing so may help to answer any questions you might have. On a first reading, I suggest that you should skip anything after the general documentation in sections 2 until 8, unless you are particularly interested in the innards of ledpar.

2 The ledpar package

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

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

2.1 General

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

ledpar similarly puts the left and right chunks into boxes but can't immediately output the text after a `\pend` — it has to wait until after both the left and right texts have been collected before it can start processing. This means that several boxes are required and possibly TeX has to store a lot of text in its memory; both the number of potential boxes and memory are limited. If TeX's memory is overfilled the recourse is to reduce the amount of text stored before printing.

`\maxchunks` It is possible to have multiple chunks in the left and right texts before printing them. The macro `\maxchunks{`⟨*num*⟩`}` specifies the maximum number of chunks within the left or right texts. This is initially set as:

```
\maxchunks{10}
```

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

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

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

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

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

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

3 Parallel columns

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

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

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

There is no required pagebreak before or after the columns.

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

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

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

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

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

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

4 Facing pages

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

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

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

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

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

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

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

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

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

number. Using these macros inside the `Leftside` and `Rightside` environments gives you independent control over the left and right numbering schemes. The `\firstsublinenum` and `\sublinenumincrement` macros correspondingly set the numbering scheme for sublines.

`\pstart` `\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 language selection for the side. The right side language is also applied to the right side line numbers.

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

6 Numbering text lines

\beginnumbering
\endnumbering

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

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

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

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

The `\printlines` macro is ordinarily used to print the line number refer-

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

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

7 Verse

If you are typesetting verse with ledmac you can use the `\stanza` construct, and you can also use this in right or left parallel texts. In this case each verse line is a chunk which has two implications. (1) you can unexpectedly exceed the `\maxchunks` limit or the overall limit on the number of boxes, and (2) left and right verse lines are matched, which may not be desirable if one side requires more print lines for verse lines than the other does.

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

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

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

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

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

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

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

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

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

8 Implementation overview

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

ledmac solves the problem of line numbering by putting the paragraph in typeset form into a box, and then extracting the lines one by one from the box for TeX to put them onto the page with the appropriate page breaks. Most of the ledmac code is concerned with handling this box and its contents.

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

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

9 Preliminaries

Announce the name and version of the package, which is targetted for LaTeX2e. The package also requires the ledmac package, preferably at least version 0.6 (2004/12/10).

```
⟨*code⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{ledpar}[2011/08/22 v0.6 ledmac extension for parallel texts]

```

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@dpaging` `\ifledRcol`

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

```
  \l@dpairingfalse
\newif\ifl@dpaging
  \l@dpagingfalse
\newif\ifledRcol
  \ledRcolfalse
```

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

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

```

9.1 Messages

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

\led@err@TooManyPstarts

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

\led@err@BadLeftRightPstarts

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

\led@err@LeftOnRightPage
\led@err@RightOnLeftPage

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

10 Sectioning commands

\section@numR
This is the right side equivalent of \section@num.

Each section will read and write an associated 'line-list file', containing information used to do the numbering. Normally the file will be called ⟨*jobname*⟩.nn, where nn is the section number. However, for right side texts the file is called ⟨*jobname*⟩.nnR. The \extensionchars applies to the right side files just as it does to the normal files.

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

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

```
\newif\ifnumberingR
```

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

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

```

\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.

```
\providecommand*{\beginnumbering}{%
  \ifnumbering
    \led@err@NumberingStarted
    \endnumbering
  \fi
  \global\l@dnumpstartsL \z@
  \global\pst@rtedLfalse
  \global\numberingtrue
  \global\advance\section@num \@ne
  \initnumbering@reg
  \message{Section \the\section@num}%
  \line@list@stuff{\jobname.\extensionchars\the\section@num}%
  \l@dend@stuff}
```

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

```
\newcommand*{\beginnumberingR}{%
  \ifnumberingR
    \led@err@NumberingStarted
    \endnumberingR
  \fi
  \global\l@dnumpstartsR \z@
  \global\pst@rtedRfalse
  \global\numberingRtrue
  \global\advance\section@numR \@ne
  \global\absline@numR \z@
  \global\line@numR \z@
  \global\@lockR \z@
  \global\sub@lockR \z@
  \global\sublines@false
  \global\let\next@page@numR\relax
  \global\let\sub@change\relax
  \message{Section \the\section@numR R }%
  \line@list@stuffR{\jobname.\extensionchars\the\section@numR R}%
  \l@dend@stuff}

```

\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.

```
\def\endnumberingR{%
  \ifnumberingR
    \global\numberingRfalse
    \normal@pars
```

```
    \ifl@dpairing
      \global\pst@rtedRfalse
    \else
      \ifx\insertlines@listR\empty\else
        \global\noteschanged@true
      \fi
      \ifx\line@listR\empty\else
        \global\noteschanged@true
      \fi
    \fi
    \ifnoteschanged@
      \led@mess@NotesChanged
    \fi
  \else
    \led@err@NumberingNotStarted
  \fi}
```

\pausenumberingR
\resumenumberingR

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

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

\memorydumpL
\memorydumpR

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

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

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

11 Line counting

11.1 Choosing the system of lineation

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

`\ifbypage@R` `\bypage@Rtrue` `\bypage@Rfalse`

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

```
\newif\ifbypage@R
  \bypage@Rfalse
```

`\lineationR`

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

```
\newcommand*{\lineationR}[1]{{%
  \ifnumberingR
    \led@err@LineationInNumbered
  \else
    \def\@tempa{#1}\def\@tempb{page}%
    \ifx\@tempa\@tempb
        \global\bypage@Rtrue
    \else
       \def\@tempb{section}%
       \ifx\@tempa\@tempb
           \global\bypage@Rfalse
       \else
         \led@warn@BadLineation
       \fi
    \fi
  \fi}}
```

`\linenummargin` `\line@marginR`

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

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

```
\newcount\line@marginR
\renewcommand*{\linenummargin}[1]{{%
  \l@dgetline@margin{#1}%
  \ifnum\@l@dtempcntb>\m@ne
    \ifledRcol
      \global\line@marginR=\@l@dtempcntb
    \else
      \global\line@margin=\@l@dtempcntb
    \fi
  \fi}}
```

By default put right text numbers at the right.

```
\line@marginR=\@ne
```

`\c@firstlinenumR` `\c@linenumincrementR`

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

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

`\c@firstsublinenumR` `\c@sublinenumincrementR`

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

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

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

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

```
\providecommand*{\firstlinenum}{}
\providecommand*{\linenumincrement}{}
\providecommand*{\firstsublinenum}{}
\providecommand*{\sublinenumincrement}{}
\renewcommand*{\firstlinenum}[1]{%
  \ifledRcol \setcounter{firstlinenumR}{#1}%
  \else      \setcounter{firstlinenum}{#1}%
  \fi}
\renewcommand*{\linenumincrement}[1]{%
  \ifledRcol \setcounter{linenumincrementR}{#1}%
  \else      \setcounter{linenumincrement}{#1}%
  \fi}
```

```
\renewcommand*{\firstsublinenum}[1]{%
  \ifledRcol \setcounter{firstsublinenumR}{#1}%
  \else      \setcounter{firstsublinenum}{#1}%
  \fi}
\renewcommand*{\sublinenumincrement}[1]{%
  \ifledRcol \setcounter{sublinenumincrementR}{#1}%
  \else      \setcounter{sublinenumincrement}{#1}%
  \fi}

```

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

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

```

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

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

```

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

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

```

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

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

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

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

Here are the commands to create these lists:

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

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

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

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

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

```
\newcount\page@numR
```

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.

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

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

```
  \ifledRcol
    \list@clear{\line@listR}%
    \list@clear{\insertlines@listR}%
    \list@clear{\actionlines@listR}%
    \list@clear{\actions@listR}%
    \list@clear{\linesinpar@listR}%
    \list@clear{\linesonpage@listR}
  \else
    \list@clearing@reg
    \list@clear{\linesinpar@listL}%
    \list@clear{\linesonpage@listL}%
  \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).

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

Now get the file and interpret it.

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

```
\ifledRcol
  \global\page@numR=\m@ne
  \ifx\actionlines@listR\empty
    \gdef\next@actionlineR{1000000}%
  \else
    \gl@p\actionlines@listR\to\next@actionlineR
    \gl@p\actions@listR\to\next@actionR
  \fi
\else
  \global\page@num=\m@ne
  \ifx\actionlines@list\empty
    \gdef\next@actionline{1000000}%
  \else
    \gl@p\actionlines@list\to\next@actionline
    \gl@p\actions@list\to\next@action
  \fi
\fi}

```

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

```
\newcommand{\@l@regR}{%
  \ifx\l@dchset@num\relax \else
    \advance\absline@numR \@ne
    \set@line@action
```

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

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

```

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

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

```

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

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

```
344         \line@numR \z@
345       \fi
346     \else
347       \advance\line@num by #1\relax
348       \ifnum\line@num<\z@
349         \led@warn@BadAdvancelineLine
350         \line@num \z@
351       \fi
352     \fi
353   \fi
354   \set@line@action}
355
```

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

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

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

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

```
373 \renewcommand*{\l@d@set}[1]{%
374   \ifledRcol
375     \line@numR=#1\relax
376     \advance\line@numR \@ne
377     \def\l@dchset@num{#1}
378   \else
379     \line@num=#1\relax
380     \advance\line@num \@ne
381     \def\l@dchset@num{#1}
382   \fi}
383 \let\l@dchset@num\relax
384
```

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

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

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

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

```

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

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

```
      \xright@appenditem{-1002}\to\actions@list
    \fi
  \fi}
```

`\do@lockon` `\do@lockonR`

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

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

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

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

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

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

```

\newcommand{\do@lockoffR}{%
  \xright@appenditem{\the\absline@numR}\to\actionlines@listR
```

```
  \ifsublines@
    \xright@appenditem{-1006}\to\actions@listR
    \ifnum\sub@lockR=\tw@
      \sub@lockR \thr@@
    \else
      \sub@lockR \z@
    \fi
  \else
    \xright@appenditem{-1004}\to\actions@listR
    \ifnum\@lockR=\tw@
      \@lockR \thr@@
    \else
      \@lockR \z@
    \fi
  \fi}

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

```

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

```
\providecommand*{\n@num}{}
\renewcommand*{\n@num}{%
  \ifledRcol
    \xright@appenditem{\the\absline@numR}\to\actionlines@listR
    \xright@appenditem{-1007}\to\actions@listR
  \else
    \n@num@reg
  \fi}

```

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

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

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

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

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

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

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

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

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

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

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

```
  #2
  \else
```

And when not in right text

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

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

```
\providecommand*{\@pend}[1]{}
\renewcommand*{\@pend}[1]{%
  \xright@appenditem{#1}\to\linesinpar@listL}
```

```
\providecommand*{\@pendR}[1]{}
\renewcommand*{\@pendR}[1]{%
  \xright@appenditem{#1}\to\linesinpar@listR}
```

\@lopL
\@lopR

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

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

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.

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

\iffirst@linenum@out@R
\first@linenum@out@Rtrue
\first@linenum@out@Rfalse

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

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

\line@list@stuffR

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

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

\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.

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

\flag@start
\flag@end

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

```
\renewcommand*{\flag@start}{%
  \ifledRcol
    \edef\next{\write\linenum@outR{%
                  \string\@ref[\the\insert@countR][}}%
    \next
  \else
    \edef\next{\write\linenum@out{%
                  \string\@ref[\the\insert@count][}}%
    \next
  \fi}
\renewcommand*{\flag@end}{%
  \ifledRcol
    \write\linenum@outR{]}%
  \else
    \write\linenum@out{]}%
  \fi}
```

\startsub
\endsub

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

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

```

\advanceline

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

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

\setline

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

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

```
    \else      \write\linenum@out{\string\@set[#1]}%
    \fi
  \fi}
```

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

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

```

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

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

```

`\skipnumbering` 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.

```
\renewcommand*{\skipnumbering}{%
  \ifledRcol \write\linenum@outR{\string\n@num}%
             \advanceline{-1}%
  \else
    \skipnumbering@reg
  \fi}

```

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.

```
\long\def\critext#1#2/{\leavevmode
  \begingroup
    \no@expands
    \xdef\@tag{#1}%
    \set@line
    \ifledRcol \global\insert@countR \z@
    \else      \global\insert@count \z@ \fi
    \ignorespaces #2\relax
    \flag@start
  \endgroup
  \showlemma{#1}%
  \ifx\end@lemmas\empty \else
    \gl@p\end@lemmas\to\x@lemma
    \x@lemma
    \global\let\x@lemma=\relax
  \fi
  \flag@end}
```

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

```
\renewcommand{\edtext}[2]{\leavevmode
  \begingroup
    \no@expands
    \xdef\@tag{#1}%
    \set@line
    \ifledRcol \global\insert@countR \z@
    \else      \global\insert@count \z@ \fi
    \ignorespaces #2\relax
    \flag@start
  \endgroup
  \showlemma{#1}%
  \ifx\end@lemmas\empty \else
    \gl@p\end@lemmas\to\x@lemma
    \x@lemma
    \global\let\x@lemma=\relax
  \fi
  \flag@end}
```

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

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

```

13 Parallel environments

The initial set up for parallel processing is deceptively simple.

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

```
\newenvironment{pairs}{%
  \l@dpairingtrue
  \l@dpagingfalse
}{%
  \l@dpairingfalse
}
```

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

```
\newenvironment{pages}{%
 \let\oldchapter\chapter
 \let\chapter\chapterinpages
  \l@dpairingtrue
  \l@dpagingtrue
  \setlength{\Lcolwidth}{\textwidth}%
  \setlength{\Rcolwidth}{\textwidth}%
```

```
}{%
  \l@dpairingfalse
  \l@dpagingfalse
  \let\chapter\oldchapter
}
\newcommand{\chapterinpages}{\thispagestyle{plain}%
                    \global\@topnum\z@
                    \@afterindentfalse
                    \secdef\@chapter\@schapter}

```

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

```
\newenvironment{Leftside}{%
  \ledRcolfalse
  \let\pstart\pstartL
  \let\pend\pendL
  \let\memorydump\memorydumpL
  \Leftsidehook
}{\Leftsidehookend}
```

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

`\Rightsidehook` `\Rightsidehookend`

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

```

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

```
\newenvironment{Rightside}{%
  \ledRcoltrue
  \let\beginnumbering\beginnumberingR
  \let\endnumbering\endnumberingR
  \let\pausenumbering\pausenumberingR
  \let\resumenumbering\resumenumberingR
  \let\memorydump\memorydumpR
  \let\pstart\pstartR
  \let\pend\pendR
  \let\lineation\lineationR
  \Rightsidehook
}{%
  \ledRcolfalse
  \Rightsidehookend
}

```

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
\one@lineR
\par@lineR

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

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

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

\pstartL
\pstartR

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

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

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

```
\newcommand*{\pstartL}{
\if@nobreak
\let\@oldnobreak\@nobreaktrue
\else
\let\@oldnobreak\@nobreakfalse
\fi
\@nobreaktrue
  \ifnumbering \else
    \led@err@PstartNotNumbered
    \beginnumbering
  \fi
  \ifnumberedpar@
    \led@err@PstartInPstart
    \pend
```

```
  \fi
```

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

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

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

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

\newcommand*{\pstartR}{
\if@nobreak
\let\@oldnobreak\@nobreaktrue
\else
\let\@oldnobreak\@nobreakfalse
\fi
\@nobreaktrue
  \ifnumberingR \else
    \led@err@PstartNotNumbered
    \beginnumberingR
  \fi
  \ifnumberedpar@
    \led@err@PstartInPstart
    \pendR
  \fi
  \ifpst@rtedR\else
    \list@clear{\inserts@listR}%
    \global\let\next@insertR=\empty
    \global\pst@rtedRtrue
  \fi
  \begingroup\normal@pars
  \global\advance\l@dnumpstartsR \@ne
  \ifnum\l@dnumpstartsR>\l@dc@maxchunks
    \led@err@TooManyPstarts
    \global\l@dnumpstartsR=\l@dc@maxchunks
  \fi
  \global\setnamebox{l@dRcolrawbox\the\l@dnumpstartsR}=\vbox\bgroup%
                      \hsize=\Rcolwidth
  \numberedpar@true}
```

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

```
\newcommand*{\pendL}{\ifnumbering \else
    \led@err@PendNotNumbered
  \fi
  \ifnumberedpar@ \else
    \led@err@PendNoPstart
  \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.

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

End the group that was begun in the \pstart.

```
  \endgroup
  \ignorespaces
  \@oldnobreak}

```

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

```
\newcommand*{\pendR}{\ifnumberingR \else
    \led@err@PendNotNumbered
  \fi
  \ifnumberedpar@ \else
    \led@err@PendNoPstart
  \fi
  \l@dzeropenalties
  \endgraf\global\num@linesR=\prevgraf\egroup
  \global\par@lineR=0
  \endgroup
  \ignorespaces
  \@oldnobreak}

```

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.

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

```

\countLline
\countRline

We need to know the number of lines processed.

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

```

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

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

```
\newcount\@donereallinesL
\newcount\@donetotallinesL
\newcount\@donereallinesR
\newcount\@donetotallinesR

```

\do@lineL

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

```
\newcommand*{\do@lineL}{%
  \advance\countLline \@ne
  \ifvbox\namebox{l@dLcolrawbox\the\l@dpscL}%
  {\vbadness=10000
   \splittopskip=\z@
   \do@lineLhook
   \l@demptyd@ta
   \global\setbox\one@line=\vsplit\namebox{l@dLcolrawbox\the\l@dpscL}
                             to\baselineskip}%
  \unvbox\one@line \global\setbox\one@line=\lastbox
  \getline@num
  \setbox\l@dleftbox
  \hb@xt@ \Lcolwidth{%
    \affixline@num
    \l@dld@ta
    \add@inserts
    \affixside@note
    \l@dlsn@te
    {\ledllfill\hb@xt@ \wd\one@line{\new@line\l@dunhbox@line{\one@line}}\ledrlfill\l@drd@t
     \l@drsn@te
  }}%
  \add@penaltiesL
  \global\advance\@donereallinesL\@ne
  \global\advance\@donetotallinesL\@ne
\else
  \setbox\l@dleftbox \hb@xt@ \Lcolwidth{\hspace*{\Lcolwidth}}%
  \global\advance\@donetotallinesL\@ne
\fi}

```

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

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

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

```
\newcommand*{\do@lineR}{%
  \advance\countRline \@ne
  \ifvbox\namebox{l@dRcolrawbox\the\l@dpscR}%
  {\vbadness=10000
   \splittopskip=\z@
   \do@lineRhook
   \l@demptyd@ta
   \global\setbox\one@lineR=\vsplit\namebox{l@dRcolrawbox\the\l@dpscR}
                             to\baselineskip}%
  \unvbox\one@lineR \global\setbox\one@lineR=\lastbox
  \getline@numR
  \setbox\l@drightbox
  \hb@xt@ \Rcolwidth{%
    \affixline@numR
    \l@dld@ta
    \add@insertsR
    \affixside@noteR
    \l@dlsn@te
    {\ledllfill\hb@xt@ \wd\one@lineR{\new@lineR\l@dunhbox@line{\one@lineR}}\ledrlfill\l@drd@ta%
     \l@drsn@te
  }}%
  \add@penaltiesR
  \global\advance\@donereallinesR\@ne
  \global\advance\@donetotallinesR\@ne
\else
  \setbox\l@drightbox \hb@xt@ \Rcolwidth{\hspace*{\Rcolwidth}}
  \global\advance\@donetotallinesR\@ne
\fi}
```

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.

```
\newcommand*{\getline@numR}{%
  \global\advance\absline@numR \@ne
  \do@actionsR
  \do@ballastR
  \ifsublines@
    \ifnum\sub@lockR<\tw@
      \global\advance\subline@numR \@ne
```

```
    \fi
  \else
    \ifnum\@lockR<\tw@
      \global\advance\line@numR \@ne
      \global\subline@numR \z@
    \fi
  \fi}
```

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

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

`\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.

```
\newcommand*{\do@actions@fixedcodeR}{%
  \ifcase\@l@dtempcnta%
  \or%                      % 1001
    \global\sublines@true
  \or%                      % 1002
    \global\sublines@false
  \or%                      % 1003
    \global\@lockR=\@ne
  \or%                      % 1004
    \ifnum\@lockR=\tw@
      \global\@lockR=\thr@@
    \else
      \global\@lockR=\z@
    \fi
  \or%                      % 1005
     \global\sub@lockR=\@ne
  \or%                      % 1006
    \ifnum\sub@lockR=\tw@
      \global\sub@lockR=\thr@@
    \else
      \global\sub@lockR=\z@
    \fi
```

```
  \or%                      % 1007
    \l@dskipnumbertrue
  \else
    \led@warn@BadAction
  \fi}


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

```

14.4 Line number printing

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

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

```

\providecommand*{\l@dcalcnum}[3]{%
  \ifnum #1 > #2\relax
    \@l@dtempcnta = #1\relax
```

```
    \advance\@l@dtempcnta by -#2\relax
    \divide\@l@dtempcnta by #3\relax
    \multiply\@l@dtempcnta by #3\relax
    \advance\@l@dtempcnta by #2\relax
  \else
    \@l@dtempcnta=#2\relax
  \fi}

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

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

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

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


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

14.5 Add insertions to the vertical list

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

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

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

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

14.6 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.

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

14.7 Printing leftover notes

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

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

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.

```
\renewcommand*{\Afootnote}[1]{%
  \ifnumberedpar@
    \ifledRcol
      \xright@appenditem{\noexpand\vAfootnote{A}%
                    {{\l@d@nums}{\@tag}{#1}}}\to\inserts@listR
      \global\advance\insert@countR \@ne
    \else
      \xright@appenditem{\noexpand\vAfootnote{A}%
                    {{\l@d@nums}{\@tag}{#1}}}\to\inserts@list
      \global\advance\insert@count \@ne
    \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.

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

\Bfootnote We need similar commands for the other footnote series.
\Cfootnote
\Dfootnote
\Efootnote

```
\renewcommand*{\Bfootnote}[1]{%
  \ifnumberedpar@
    \ifledRcol
      \xright@appenditem{\noexpand\vBfootnote{B}%
                    {{\l@d@nums}{\@tag}{#1}}}\to\inserts@listR
      \global\advance\insert@countR \@ne
    \else
      \xright@appenditem{\noexpand\vBfootnote{B}%
                    {{\l@d@nums}{\@tag}{#1}}}\to\inserts@list
      \global\advance\insert@count \@ne
    \fi
  \else
    \vBfootnote{B}{{0|0|0|0|0|0|0}{}{#1}}%
  \fi\ignorespaces}

\renewcommand*{\Cfootnote}[1]{%
  \ifnumberedpar@
    \ifledRcol
      \xright@appenditem{\noexpand\vCfootnote{C}%
                    {{\l@d@nums}{\@tag}{#1}}}\to\inserts@listR
```

```
      \global\advance\insert@countR \@ne
    \else
      \xright@appenditem{\noexpand\vCfootnote{C}%
                    {{\l@d@nums}{\@tag}{#1}}}\to\inserts@list
      \global\advance\insert@count \@ne
    \fi
  \else
    \vCfootnote{C}{{0|0|0|0|0|0|0}{}{#1}}%
  \fi\ignorespaces}
\renewcommand*{\Dfootnote}[1]{%
  \ifnumberedpar@
    \ifledRcol
      \xright@appenditem{\noexpand\vDfootnote{D}%
                    {{\l@d@nums}{\@tag}{#1}}}\to\inserts@listR
      \global\advance\insert@countR \@ne
    \else
      \xright@appenditem{\noexpand\vDfootnote{D}%
                    {{\l@d@nums}{\@tag}{#1}}}\to\inserts@list
      \global\advance\insert@count \@ne
    \fi
  \else
    \vDfootnote{D}{{0|0|0|0|0|0|0}{}{#1}}%
  \fi\ignorespaces}
\renewcommand*{\Efootnote}[1]{%
  \ifnumberedpar@
    \ifledRcol
      \xright@appenditem{\noexpand\vEfootnote{E}%
                    {{\l@d@nums}{\@tag}{#1}}}\to\inserts@listR
      \global\advance\insert@countR \@ne
    \else
      \xright@appenditem{\noexpand\vEfootnote{E}%
                    {{\l@d@nums}{\@tag}{#1}}}\to\inserts@list
      \global\advance\insert@count \@ne
    \fi
  \else
    \vEfootnote{E}{{0|0|0|0|0|0|0}{}{#1}}%
  \fi\ignorespaces}

```

\mpAfootnote
\mpBfootnote
\mpCfootnote
\mpDfootnote
\mpEfootnote

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

```
\renewcommand*{\mpAfootnote}[1]{%
  \ifnumberedpar@
  \ifledRcol
    \xright@appenditem{\noexpand\mpvAfootnote{A}%
                  {{\l@d@nums}{\@tag}{#1}}}\to\inserts@listR
    \global\advance\insert@countR \@ne
  \else
    \xright@appenditem{\noexpand\mpvAfootnote{A}%
                  {{\l@d@nums}{\@tag}{#1}}}\to\inserts@list
```

```
    \global\advance\insert@count \@ne
  \fi
  \else
    \mpvAfootnote{A}{{0|0|0|0|0|0|0}{}{#1}}%
  \fi\ignorespaces}
\renewcommand*{\mpBfootnote}[1]{%
  \ifnumberedpar@
  \ifledRcol
    \xright@appenditem{\noexpand\mpvBfootnote{B}%
                  {{\l@d@nums}{\@tag}{#1}}}\to\inserts@listR
    \global\advance\insert@countR \@ne
  \else
    \xright@appenditem{\noexpand\mpvBfootnote{B}%
                  {{\l@d@nums}{\@tag}{#1}}}\to\inserts@list
    \global\advance\insert@count \@ne
  \fi
  \else
    \mpvBfootnote{B}{{0|0|0|0|0|0|0}{}{#1}}%
  \fi\ignorespaces}
\renewcommand*{\mpCfootnote}[1]{%
  \ifnumberedpar@
  \ifledRcol
    \xright@appenditem{\noexpand\mpvCfootnote{C}%
                  {{\l@d@nums}{\@tag}{#1}}}\to\inserts@listR
    \global\advance\insert@countR \@ne
  \else
    \xright@appenditem{\noexpand\mpvCfootnote{C}%
                  {{\l@d@nums}{\@tag}{#1}}}\to\inserts@list
    \global\advance\insert@count \@ne
  \fi
  \else
    \mpvCfootnote{C}{{0|0|0|0|0|0|0}{}{#1}}%
  \fi\ignorespaces}
\renewcommand*{\mpDfootnote}[1]{%
  \ifnumberedpar@
  \ifledRcol
    \xright@appenditem{\noexpand\mpvDfootnote{D}%
                  {{\l@d@nums}{\@tag}{#1}}}\to\inserts@listR
    \global\advance\insert@countR \@ne
  \else
    \xright@appenditem{\noexpand\mpvDfootnote{D}%
                  {{\l@d@nums}{\@tag}{#1}}}\to\inserts@list
    \global\advance\insert@count \@ne
  \fi
  \else
    \mpvDfootnote{D}{{0|0|0|0|0|0|0}{}{#1}}%
  \fi\ignorespaces}
\renewcommand*{\mpEfootnote}[1]{%
```

```
\ifnumberedpar@
\ifledRcol
  \xright@appenditem{\noexpand\mpvEfootnote{E}%
                {{\l@d@nums}{\@tag}{#1}}}\to\inserts@listR
  \global\advance\insert@countR \@ne
\else
  \xright@appenditem{\noexpand\mpvEfootnote{E}%
                {{\l@d@nums}{\@tag}{#1}}}\to\inserts@list
  \global\advance\insert@count \@ne
\fi
\else
  \mpvEfootnote{E}{{0|0|0|0|0|0|0}{}{#1}}%
\fi\ignorespaces}
```

15.2 Normal footnote formatting

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

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

Just a reminder of the arguments:

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

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

\let\ledsavedprintlines\printlines

```

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.

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

```

\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.

```
\renewcommand*{\edlabel}[1]{\@bsphack
  \ifledRcol
    \write\linenum@outR{\string\@lab}%
    \ifx\labelref@listR\empty
      \xdef\label@refs{\zz@@@}%
    \else
      \gl@p\labelref@listR\to\label@refs
    \fi
    \protected@write\@auxout{}%
      {\string\l@dmake@labelsR\space\thepage|\label@refs|{#1}}%
  \else
    \write\linenum@out{\string\@lab}%
    \ifx\labelref@list\empty
      \xdef\label@refs{\zz@@@}%
    \else
      \gl@p\labelref@list\to\label@refs
    \fi
  \fi
  \protected@write\@auxout{}%
    {\string\l@dmake@labels\space\thepage|\label@refs|{#1}}%
  \@esphack}

```

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

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

```

\@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.

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

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

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
\sidenotemargin

Specifies which margin sidenotes can be in.

```
\newcount\sidenote@marginR
\renewcommand*{\sidenotemargin}[1]{{%
  \l@dgetsidenote@margin{#1}%
  \ifnum\@l@dtempcntb>\m@ne
    \ifledRcol
      \global\sidenote@marginR=\@l@dtempcntb
    \else
      \global\sidenote@margin=\@l@dtempcntb
    \fi
  \fi}}
\sidenotemargin{right}
\global\sidenote@margin=\@ne
```

\l@dlsnote
\l@drsnote
\l@dcsnote

The 'footnotes' for left, right, and moveable sidenotes. The whole scheme is reminiscent of the critical footnotes code.

```
\renewcommand*{\l@dlsnote}[1]{%
  \ifnumberedpar@
    \ifledRcol
      \xright@appenditem{\noexpand\vl@dlsnote{#1}}%
                        \to\inserts@listR
      \global\advance\insert@countR \@ne
    \else
      \xright@appenditem{\noexpand\vl@dlsnote{#1}}%
                        \to\inserts@list
      \global\advance\insert@count \@ne
    \fi
  \fi\ignorespaces}
\renewcommand*{\l@drsnote}[1]{%
  \ifnumberedpar@
    \ifledRcol
      \xright@appenditem{\noexpand\vl@drsnote{#1}}%
                        \to\inserts@listR
      \global\advance\insert@countR \@ne
    \else
```

```
      \xright@appenditem{\noexpand\vl@drsnote{#1}}%
                         \to\inserts@list
      \global\advance\insert@count \@ne
    \fi
  \fi\ignorespaces}
\renewcommand*{\l@dcsnote}[1]{%
  \ifnumberedpar@
    \ifledRcol
      \xright@appenditem{\noexpand\vl@dcsnote{#1}}%
                         \to\inserts@listR
      \global\advance\insert@countR \@ne
    \else
      \xright@appenditem{\noexpand\vl@dcsnote{#1}}%
                         \to\inserts@list
      \global\advance\insert@count \@ne
    \fi
  \fi\ignorespaces}
```

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

```
\newcommand*{\affixside@noteR}{%
  \gdef\@templ@d{}%
  \ifx\@templ@d\l@dcsnotetext \else
    \if@twocolumn
      \if@firstcolumn
        \setl@dlp@rbox{\l@dcsnotetext}%
      \else
        \setl@drp@rbox{\l@dcsnotetext}%
      \fi
    \else
      \@l@dtempcntb=\sidenote@marginR
      \ifnum\@l@dtempcntb>\@ne
        \advance\@l@dtempcntb by\page@num
      \fi
      \ifodd\@l@dtempcntb
        \setl@drp@rbox{\l@dcsnotetext}%
      \else
        \setl@dlp@rbox{\l@dcsnotetext}%
      \fi
    \fi
  \fi}
```

18 Familiar footnotes

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

```
\renewcommand{\l@dbfnote}[1]{%
```

```
  \ifnumberedpar@
    \ifledRcol
      \xright@appenditem{\noexpand\vl@dbfnote{{#1}}{\@thefnmark}}%
                        \to\inserts@listR
      \global\advance\insert@countR \@ne
    \else
      \xright@appenditem{\noexpand\vl@dbfnote{{#1}}{\@thefnmark}}%
                        \to\inserts@list
      \global\advance\insert@count \@ne
    \fi
  \fi\ignorespaces}

\normalbfnoteX
 \renewcommand{\normalbfnoteX}[2]{%
  \ifnumberedpar@
    \ifledRcol
      \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\@nameuse{thefootnote#1}}}%
                        \to\inserts@listR
      \global\advance\insert@countR \@ne
    \else
      \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\@nameuse{thefootnote#1}}}%
                        \to\inserts@list
      \global\advance\insert@count \@ne
    \fi
  \fi\ignorespaces}
```

19 Verse

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

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

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

```
 \newenvironment{astanza}{%
  \startstanzahook
  \catcode`\&\active
  \global\stanza@count\@ne
  \ifnum\usenamecount{sza@0@}=\z@
    \let\stanza@hang\relax
    \let\endlock\relax
  \else
 %%%    \interlinepenalty\@M % this screws things up, but I don't know why
```

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

```

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

```
\newcommand*{\@astanza@line}{%
  \parindent=\csname sza@\number\stanza@count @\endcsname\stanzaindentbase
  \par
  \stanza@hang%\mbox{}%
  \ignorespaces}

```

Lastly reset the modified category codes.

```
  \catcode`\&=\next

```

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

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

```
\providecommand*{\namebox}[1]{%
                      \csname #1\endcsname}
```

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

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

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.

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

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

```
\newcount\l@dnumpstartsR
```

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

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

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

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

`\l@dsetupmaxlinecounts` `\l@dzeromaxlinecounts` To be able to synchronise left and right texts we need to know the maximum number of text lines there are in each pair of chunks. `\l@dsetupmaxlinecounts` creates

`\maxchunks` new counts called `\l@dmaxlinesinpar1`, etc., and `\l@dzeromaxlinecounts` zeroes all of them.

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

```

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

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

```

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` `\l@dusedbabelfalse` `\l@dusedbabeltrue`

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

```
\newif\ifl@dusedbabel
  \l@dusedbabelfalse
```

`\ifl@dsamelang` `\l@dsamelangfalse` `\l@dsamelangtrue`

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

```
\newif\ifl@dsamelang
  \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.

```
\newcommand*{\l@dchecklang}{%
  \l@dsamelangfalse
  \edef\@tempa{\theledlanguageL}\edef\@temp{\theledlanguageR}%
  \ifx\@tempa\@tempb
    \l@dsamelangtrue
  \fi}

```

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

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

```

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

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

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

```

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

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

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

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

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

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

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

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

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

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

If on Polyglossia

```
  {    \apptocmd{\xpg@set@language}{%
        \ifledRcol \gdef\theledlanguageR{#1}%
        \else      \gdef\theledlanguageL{#1}%
        \fi}%
       \let\l@duselanguage\xpg@set@language
       \gdef\theledlanguageL{\xpg@main@language}%
       \gdef\theledlanguageR{\xpg@main@language}%
% \end{macrocode}
% That's it.
%    \begin{macrocode}
}}
```

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.

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

Start a group and zero counters, etc.

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

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

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

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

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

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

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

```
        \ifl@dsamelang
          \do@lineL
          \do@lineR
        \else
          \l@duselanguage{\theledlanguageL}%
          \do@lineL
          \l@duselanguage{\theledlanguageR}%
          \do@lineR
        \fi
        \hb@xt@ \hsize{%
          \unhbox\l@dleftbox
          \hfill \columnseparator \hfill
          \unhbox\l@drightbox
        }%
        \checkraw@text
      \repeat}
```

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

```
    \@writelinesinparL
    \@writelinesinparR
    \check@pstarts
  \repeat
```

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

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

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

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

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

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

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

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

```
  \fi}
```

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

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

24 Parallel pages

This is considerably more complicated than parallel columns.

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

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

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

```
\newcommand*{\Pages}{%
  \typeout{}
  \typeout{************************** PAGES **************************}
  \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else
    \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}%
  \fi
```

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

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

```
    \writtenlinesLfalse
    \writtenlinesRfalse
```

Check if there are chunks to be processed.

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

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

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

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

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

Zero the counts again, ready for the next bit.

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

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

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

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

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

Increment the chunk counts to get the first pair.

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

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

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

Start a loop over the boxes (chunks).

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

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

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

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

```
          \do@lineL
          \advance\numpagelinesL \@ne
          \hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
```

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.

```
          \get@nextboxL
          \checkpageL
        \repeat
```

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

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

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

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

Now do the same for the right text.

```
\checkpageR
\l@duselanguage{\theledlanguageR}%
{       \loop\ifl@dsamepage
          \do@lineR
          \advance\numpagelinesR \@ne
          \hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}%
          \get@nextboxR
          \checkpageR
        \repeat
        \ifl@dpagefull
          \@writelinesonpageR{\the\numpagelinesR}%
        \else
          \@writelinesonpageR{1000}%
        \fi
        \numpagelinesR=\z@
```

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

```
            \clearl@drightpage}
```

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

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

We have now output the text from all the chunks.

```
    \fi
```

Make sure that there are no inserts hanging around.

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

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

```
  \global\l@dpscL=\z@
  \global\l@dpscR=\z@
  \global\l@dnumpstartsL=\z@
  \global\l@dnumpstartsR=\z@
  \ignorespaces}

```

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

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

```

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

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

```
\newcommand*{\clearl@dleftpage}{%
  \clearpage
  \ifodd\c@page\else
    \led@err@LeftOnRightPage
    \hbox{}%
    \cleardoublepage
  \fi}
\newcommand*{\clearl@drightpage}{%
  \clearpage
  \ifodd\c@page
    \led@err@RightOnLeftPage
    \hbox{}%
    \cleartoevenpage
  \fi}
```

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

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

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

`\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`.

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

\@writelinesonpageL
\@writelinesonpageR

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

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

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

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

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

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

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

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

```
\newif\ifl@dsamepage
  \l@dsamepagetrue
\newif\ifl@dpagefull
\newcommand*{\checkpageL}{%
  \l@dpagefulltrue
  \l@dsamepagetrue
  \check@goal
  \ifdim\pagetotal<\ledthegoal
    \ifnum\numpagelinesL<\l@dminpagelines
    \else
      \l@dsamepagefalse
      \l@dpagefullfalse
    \fi
  \else
```

```
    \l@dsamepagefalse
    \l@dpagefulltrue
  \fi}
\newcommand*{\checkpageR}{%
  \l@dpagefulltrue
  \l@dsamepagetrue
  \check@goal
  \ifdim\pagetotal<\ledthegoal
    \ifnum\numpagelinesR<\l@dminpagelines
    \else
      \l@dsamepagefalse
      \l@dpagefullfalse
    \fi
  \else
    \l@dsamepagefalse
    \l@dpagefulltrue
  \fi}
```

`\ledthegoal` `\goalfraction` `\check@goal`

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

```
\newdimen\ledthegoal
\newcommand*{\goalfraction}{0.9}
\newcommand*{\check@goal}{%
  \ledthegoal=\goalfraction\pagegoal}
```

`\ifwrittenlinesL` `\ifwrittenlinesL`

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

```
\newif\ifwrittenlinesL
\newif\ifwrittenlinesR
```

`\get@nextboxL` `\get@nextboxR`

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

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

The current box is not empty; do nothing.

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

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

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

Sufficient lines have been output.

```
      \ifwrittenlinesL
      \else
```

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

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

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

```
        \writtenlinesLfalse
        \l@dcalc@maxoftwo{\the\usenamecount{l@dmaxlinesinpar\the\l@dpscL}}%
                         {\the\@donetotallinesL}%
                         {\usenamecount{l@dmaxlinesinpar\the\l@dpscL}}%
        \global\@donetotallinesL \z@
        \global\advance\l@dpscL \@ne
      \fi
    \fi
  \fi}
\newcommand*{\get@nextboxR}{%
  \ifvbox\namebox{l@dRcolrawbox\the\l@dpscR}% box is not empty
  \else%                                      box is empty
    \ifnum\usenamecount{l@dmaxlinesinpar\the\l@dpscR}>\@donetotallinesR
    \else
      \ifwrittenlinesR
      \else
        \@writelinesinparR
        \writtenlinesRtrue
      \fi
      \ifnum\l@dnumpstartsR>\l@dpscR
        \writtenlinesRfalse
        \l@dcalc@maxoftwo{\the\usenamecount{l@dmaxlinesinpar\the\l@dpscR}}%
                         {\the\@donetotallinesR}%
                         {\usenamecount{l@dmaxlinesinpar\the\l@dpscR}}%
        \global\@donetotallinesR \z@
        \global\advance\l@dpscR \@ne
      \fi
    \fi
  \fi}

```

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 water,	2r
	Mis en bas lieu, non pas en hault,	Put me underground instead of higher up	
4	Mengier d'angoisse maints poire,	And made me eat such bitter fruit,	4r
	Enferré . . . Quant j'en ay memoire,	In chains . . . When I think of this,	
6	Je Prie pour luy *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 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.

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

// One additional line to show synchronization. //

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

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

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

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 graveyard. King William approved and the town of Nijmegen explicitly expressed its assent. A new ditch was dug on property of the church near the castle and the harbour,[5] causing the demolition of the presbytery. In compensation, the count and citizens committed themselves to giving the parish priest another suitable space close enough to the new church that was about to be built. A letter about these transactions, with the seals of count Otto and the town of Nijmegen, is kept at St. Apostles' church.[6]

// One additional line to show synchronization. //

[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 belonged 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*, Stadtspuren – 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`.

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

"Quis tibi, saeve puer, dedit hoc in carmina iuris?
    Pieridum vates, non tua turba sumus.
Quid si praeripiat flavae Vĕnus arma Minervae,
    ventilet accensas flava Minerva faces?

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

---

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

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.

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

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

"Quis tibi, saeve puer, dedit hoc in carmina iuris?
Pieridum vates, non tua turba sumus.
Quid si praeripiat flavae Vĕnus arma Minervae,
ventilet accensas flava Minerva faces?

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

---

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

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.

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.

```
⟨∗villon⟩
%%% villon.tex Example parallel columns
\documentclass{article}
\addtolength{\textheight}{-10\baselineskip}
\usepackage{ledmac,ledpar}
%% Use r instead of R to flag right text line numbers
\renewcommand{\Rlineflag}{r}
%% Use the flag in the notes
\let\oldBfootfmt\Bfootfmt
\renewcommand{\Bfootfmt}[3]{%
  \let\printlines\printlinesR
  \oldBfootfmt{#1}{#2}{#3}}
\begin{document}

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

\vspace*{\baselineskip}

\begin{pairs}
%% no indentation
\setstanzaindents{0,0,0,0,0,0,0,0,0,0}
%% no number flag
\renewcommand{\Rlineflag}{}
%% draw a rule and widen the columns
\setlength{\columnrulewidth}{0.4pt}
\setlength{\Lcolwidth}{0.46\textwidth}
\setlength{\Rcolwidth}{\Lcolwidth}

\begin{Leftside}
%% set left text line numbering sequence
\firstlinenum{2}
\linenumincrement{2}
\linenummargin{left}
\beginnumbering
\stanza
Il y avait un jeune homme de Dijon, &
Qui n'avait que peu de religion. &
Il dit: `Quant \`{a} moi, &
Je d\'{e}teste tous les trois, &
Le P\`{e}re, et le Fils, et le Pigeon.' \&
\endnumbering
\end{Leftside}
```

```
1954
1955 \begin{Rightside}
1956 %% different right text line numbering sequence
1957 \firstlinenum{1}
1958 \linenumincrement{2}
1959 \linenummargin{right}
1960 \beginnumbering
1961 \stanza
1962 There was a young man of Dijon, &
1963 Who had only a little religion, &
1964 He said: ‘As for me, &
1965 I detest all the three, &
1966 The Father, the Son, and the Pigeon.’ \&
1967 \endnumbering
1968 \end{Rightside}
1969
1970 \Columns
1971 \end{pairs}
1972
1973 \vspace*{\baselineskip}
1974
1975     The following is verse \textsc{lxxiii} of Fran\c{c}ois Villon’s
1976 \textit{Le Testament} (The Testament), composed in 1461.
1977
1978 %% Allow for hanging indentation for long lines
1979 \setstanzaindents{1,0,0,0,0,0,0,0,0}
1980 %% Columns wider than the default
1981 \setlength{\Lcolwidth}{0.46\textwidth}
1982 \setlength{\Rcolwidth}{\Lcolwidth}
1983 \vspace*{\baselineskip}
1984
1985 \begin{pairs}
1986 \begin{Leftside}
1987 \firstlinenum{2}
1988 \linenumincrement{2}
1989 \linenummargin{left}
1990 \beginnumbering
1991 \stanza
1992 Dieu mercy et Tacque Thibault, &
1993 Qui tant d’eaue froid m’a fait boire, &
1994 Mis en bas lieu, non pas en hault, &
1995 Mengier d’angoisse maints \edtext{poire}{\lemma{poire d’angoisse}%
1996   \Afootnote{This has a triple meaning: literally it is the fruit of the
1997   choke pear,
1998   figuratively it means ‘bitter fruit’, and it also refers to a torture
1999   instrument.}}, &
2000 Enferr\’{e} \ldots Quant j’en ay memoire, &
2001 Je Prie pour luy \edtext{\textit{et reliqua}}{\Afootnote{and so on}}, &
2002 Que Dieu luy doint, et voire, voire! &
2003 Ce que je pense \ldots \textit{et cetera}. \&
```

```
\endnumbering
\end{Leftside}

\begin{Rightside}
\firstlinenum{2}
\linenumincrement{2}
\linenummargin{right}
\beginnumbering
\stanza
Thanks to God --- and to \edtext{Tacque Thibaud}{%
  \Bfootnote{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 Orl\'{e}ans.}}  &
Who made me drink so much \edtext{cold water}{%
  \Bfootnote{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.}}, &
Put me underground instead of higher up &
And made me eat such bitter fruit, &
In chains \ldots When I think of this, &
I pray for him---\textit{et reliqua;} &
May God grant him (yes, by God) &
What I think \ldots \textit{et cetera}. \&
\endnumbering
\end{Rightside}

\Columns
\end{pairs}

\vspace*{\baselineskip}

    The translation and notes are by Anthony Bonner,
\textit{The Complete Works of Fran\c{c}ois Villon}, published by
Bantam Books in 1960.

\end{document}

⟨/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.

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

```
%%% Radboud University, Nijmegen (The Netherlands)
%%% (PRW) Modified slightly by PRW to fit the ledpar manual

\documentclass[10pt, letterpaper, twoside]{article}
\usepackage[latin,english]{babel}
\usepackage{makeidx}
\usepackage{ledmac,ledpar}
\lineation{section}
\linenummargin{inner}
\sidenotemargin{outer}

\makeindex

\renewcommand{\notenumfont}{\footnotesize}
\newcommand{\notetextfont}{\footnotesize}

%\let\Afootnoterule=\relax
\let\Bfootnoterule=\relax
\let\Cfootnoterule=\relax

\addtolength{\skip\Afootins}{1.5mm}
%\addtolength{\skip\Bfootins}{1.5mm}
%\addtolength{\skip\Cfootins}{1.5mm}

\makeatletter

\renewcommand*{\para@vfootnote}[2]{%
  \insert\csname #1footins\endcsname
  \bgroup
    \notefontsetup
    \interlinepenalty=\interfootnotelinepenalty
    \floatingpenalty=\@MM
    \splittopskip=\ht\strutbox \splitmaxdepth=\dp\strutbox
    \leftskip=\z@skip \rightskip=\z@skip
    \l@dparsefootspec #2\ledplinenumtrue%                  new from here
    \ifnum\@nameuse{previous@#1@number}=\l@dparsedstartline\relax
      \ledplinenumfalse
     \fi
     \ifnum\previous@page=\l@dparsedstartpage\relax
     \else \ledplinenumtrue \fi
     \ifnum\l@dparsedstartline=\l@dparsedendline\relax
     \else \ledplinenumtrue \fi
     \expandafter\xdef\csname previous@#1@number\endcsname{\l@dparsedstartline}%
     \xdef\previous@page{\l@dparsedstartpage}%            to here
     \setbox0=\vbox{\hsize=\maxdimen
       \noindent\csname #1footfmt\endcsname#2}%
      \setbox0=\hbox{\unvxh0}%
      \dp0=0pt
      \ht0=\csname #1footfudgefactor\endcsname\wd0
      \box0
```

```
      \penalty0
  \egroup
}

\newcommand*{\previous@A@number}{-1}
\newcommand*{\previous@B@number}{-1}
\newcommand*{\previous@C@number}{-1}
\newcommand*{\previous@page}{-1}

\newcommand{\abb}[1]{#1%
        \let\rbracket\nobrak\relax}
\newcommand{\nobrak}{\textnormal{}}
\newcommand{\morenoexpands}{%
        \let\abb=0%
}

\newcommand{\Aparafootfmt}[3]{%
  \ledsetnormalparstuff
  \scriptsize
  \notenumfont\printlines#1|\enspace
%  \lemmafont#1|#2\enskip
  \notetextfont
  #3\penalty-10\hskip 1em plus 4em minus.4em\relax}

\newcommand{\Bparafootfmt}[3]{%
  \ledsetnormalparstuff
  \scriptsize
  \notenumfont\printlines#1|%
  \ifledplinenum
   \enspace
  \else
   {\hskip 0em plus 0em minus .3em}%
  \fi
  \select@lemmafont#1|#2\rbracket\enskip
  \notetextfont
  #3\penalty-10\hskip 1em plus 4em minus.4em\relax }

\newcommand{\Cparafootfmt}[3]{%
  \ledsetnormalparstuff
  \scriptsize
  \notenumfont\printlines#1|\enspace
%  \lemmafont#1|#2\enskip
  \notetextfont
  #3\penalty-10\hskip 1em plus 4em minus.4em\relax}

\makeatother

\footparagraph{A}
\footparagraph{B}
\footparagraph{C}
```

```
2146
2147 \let\Afootfmt=\Aparafootfmt
2148 \let\Bfootfmt=\Bparafootfmt
2149 \let\Cfootfmt=\Cparafootfmt
2150
2151 \renewcommand*{\Rlineflag}{}
2152
2153 \emergencystretch40pt
2154
2155 \author{Guillelmus de Berchen}
2156 \title{Chronicon Geldriae}
2157 \date{}
2158 \hyphenation{archi-epi-sco-po Huns-dis-brug li-be-ra No-vi-ma-gen-si}
2159 \begin{document}
2160 \begin{pages}
2161 \begin{Leftside}
2162 \beginnumbering\pstart
2163 \selectlanguage{latin}
2164 \section{De ecclesia S. Stephani Novimagensi}
2165
2166 \noindent\setline{1}
2167 Nobilis itaque comes Otto\protect\edindex{Otto II of Guelders}
2168 imperio et dominio Novimagensi sibi, ut praefertur, impignoratis
2169 et commissis
2170 \edtext{proinde}{\Bfootnote{primum D}} praeesse cupiens, anno
2171 \textsc{liiii} superius descripto, mense
2172 Iu\edtext{}{\Afootnote{p.\ 227~R}}nio, una cum iudice, scabinis ceterisque
2173 civibus civitatis Novimagensis, pro ipsius et inhabitantium in ea
2174 necessitate,\edtext{}{\Afootnote{p.\ 97~N}} commodo et utilitate,
2175 ut \edtext{ecclesia eius}{\Bfootnote{ecclesia D: eius eius H}} parochialis
2176 \edtext{\abb{extra civitatem}}{\Bfootnote{\textit{om.}~H}} sita
2177 destrueretur et \edtext{infra}{\Bfootnote{intra D}} muros
2178 \edtext{transfer\edtext{}{\Afootnote{p.\ 129~D}}retur}%
2179 {\Bfootnote{transferreretur NH}}
2180 ac de novo construeretur,
2181 \edtext{a reverendo patre domino
2182 Conrado\protect\edindex{Conrad of Hochstaden} de
2183 \edtext{Hofsteden}{\Bfootnote{Hoffstede D: Hoffsteden H}}, archiepiscopo
2184 \edtext{Coloniensi}{\Bfootnote{Colononiensi H}}, licentiam}%
2185 {\Cfootnote{William is confusing two charters that are five years
2186 apart. Permission from St.\ Apostles' Church in Cologne had been
2187 obtained as early as 1249. Cf.\
2188 Sloet\protect\index{Sloet van de Beele, L.A.J.W.},
2189 \textit{Oorkondenboek} nr.\ 707 (14 November 1249):
2190 ``\ldots{}nos devotionis tue precibus annuentes, ut ipsam ecclesiam
2191 faciens demoliri transferas in locum alium competentem, tibi
2192 auctoritate presentium indulgemus\ldots''}}, et a venerabilibus
2193 \edtext{dominis}{\Bfootnote{viris H}} decano et capitulo sanctorum
2194 Apostolorum\protect\edindex{St. Apostles' (Cologne)}
2195 \edtext{Coloniensi}{\Bfootnote{Coloniae H}}, ipsius ecclesiae ab
```

```
antiquo veris et pacificis patronis, consensum, citra tamen
praeiudicium, damnum aut gravamen \edtext{iurium}{\Bfootnote{virium D}}
et bonorum eorundem, impetravit.
\pend

\pstart
\edtext{Et exinde \edtext{liberum}{\Bfootnote{librum H}}
locum eiusdem civitatis
\edtext{qui}{\Bfootnote{quae D}} dicitur
\edtext{Hundisburg}{\Bfootnote{Hundisburch D: Hundisbrug HMN:
Hunsdisbrug R}}\protect\edindex{Hundisburg},
de praelibati Wilhelmi\protect\edindex{William II of Holland} Romanorum
\edtext{regis}{\Bfootnote{imperatoris D}}, ipsius fundi
do\edtext{}{\Afootnote{f.\ 72v~M}}mini, consensu, ad aedificandum
\edtext{\abb{et consecrandum}}{\Bfootnote{\textit{om.}\ H}}
ecclesi\edtext{}{\Afootnote{p.\ 228~R}}am et coemeterium,
\edtext{eisdem}{\Bfootnote{eiusdem D}} decano et capitulo de expresso
eiusdem civitatis assensu libera contradiderunt voluntate, obligantes
se ipsi \edtext{comes}{\Bfootnote{comites D}} et civitas
\edtext{\abb{dictis}}{\Bfootnote{\textit{om.}\ H}} decano et capitulo,
quod in recompensationem illius areae infra castrum et portam, quae
fuit dos ecclesiae, in qua plebanus habitare solebat---quae
\edtext{tunc}{\Bfootnote{nunc H}} per novum fossatum civitatis est
destructa---aliam aream competentem et ecclesiae novae,
\edtext{ut praefertur, aedificandae}{%
\lemma{\abb{ut\ldots aedificandae}}\Bfootnote{\textit{om.}\ H}} satis
\edtext{contiguam}{\Bfootnote{contiguum M}}, ipsi plebano darent et
assignarent.}{\Cfootnote{Cf.\ Sloet, \textit{Oorkondenboek} nr.\ 762
(June 1254)}} Et desuper
\edtext{\abb{apud}}{\Bfootnote{\textit{om.}\ H}} dictam ecclesiam
sanctorum Apostolorum \edtext{est}{\Bfootnote{et H}}
\edtext{littera}{\Bfootnote{litteram H}} sigillis ipsorum
Ottonis\edtext{}{\Afootnote{p.\ 130~D}} comitis et civitatis
\edtext{Novimagensis}{\Bfootnote{Novimagii D}}
\edtext{sigillata}{\Bfootnote{sigillis communita H}}.
\pend

\pstart
 // One additional line to show synchronization. //
\pend
\endnumbering
\end{Leftside}

\begin{Rightside}
\sidenotemargin{right}\selectlanguage{english}
\beginnumbering
\pstart
\addtocounter{section}{-1}%
\leavevmode\section{St.\ Stephen's Church in Nijmegen}

```

```
\noindent\setline{1}%
After the noble count Otto had taken in pledge the power over
Nijmegen,\footnote{In 1247 William II\protect\index{William II of Holland}
(1227--1256) count of Holland needed money to fight his way to
Aachen\protect\index{Aachen} to be crowned King of the Holy Roman
Empire. He gave the town of Nijmegen in pledge to Otto
II\protect\index{Otto II of Guelders} (1229--1271) count of Guelders.}
like I have written above, he wanted to protect the town. So in June
1254\ledsidenote{1254} he and the judge, the sheriffs and other
citizens of Nijmegen obtained permission to demolish the parish
church that lay outside the town walls,\footnote{Since the early
seventh century old St.\ Stephen's church had been located close
to the castle, at today's
Kelfkensbos\protect\index{Kelfkensbos (Nijmegen)} square.
Traces of the church and the presbytery were found during excavations
in 1998--1999.} 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,\footnote{Conrad of Hochstaden ({\textdagger} 1261) was
archbishop of Cologne in 1238--1261. Nijmegen belonged to the
archdiocese of Cologne until 1559.} gave his permission. So did the
reverend dean and canons of the chapter of St.\
Apostles'\protect\index{St. Apostles' (Cologne)} in Cologne, who had
long\footnote{They probably became the patrons when the chapter was
established in the early eleventh century. About the church and the
chapter, see Gottfried Stracke\protect\index{Stracke, G.},
\textit{K\"{o}ln:\ St.\ Aposteln}, Stadtspuren -- Denkm\"{a}ler in
K\"{o}ln, vol.\ 19, K\"{o}ln: J.\,P.\ Bachem, 1992.} 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.
\pend

\pstart
And so the count and the town voluntarily gave an open space in town
called Hundisburg, which was owned by the aforementioned king William,
to the dean and chapter of St.\ Apostles' in order to build and
consecrate a church and graveyard. King William approved and the
town of Nijmegen explicitly expressed its assent. A new ditch was dug
on property of the church near the castle and the
harbour,\footnote{Nowadays, the exact location of the medieval
ditch---and of two Roman ones---can be seen in the pavement of
Kelfkensbos\protect\index{Kelfkensbos (Nijmegen)} square.} 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.\footnote{The original letter is lost. A 15th century
```

```
transcription of it is kept at the Historisches Archiv der
Stadt K\"{o}ln (HAStK).}
\pend

\pstart
// One additional line to show synchronization. //
\pend
\endnumbering
\end{Rightside}
\Pages
\end{pages}

%%%%%%%%%%%%%%%%%%%%%%%%%%
\printindex
\end{document}
%%%%%%%%%%%%%%%%%%
```

⟨/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.

⟨∗djdpoems⟩

```
%%% djdpoems.tex  example parallel verses on facing pages
\documentclass{article}
\usepackage{ledmac,ledpar}
\addtolength{\textheight}{-15\baselineskip}

\maxchunks{24} % default value = 10
\setstanzaindents{6,0,1,0,1}

\newcommand{\longdash}{---------}

\footparagraph{A} % for left pages
\footparagraph{B} % for right pages
\firstlinenum{1}
\linenumincrement{1}

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

```
2334
2335 \begin{document}
2336
2337 \newcommand{\interstanza}{\pstart\centering\longdash\skipnumbering\pend}
2338
2339 \begin{pages}
2340 \begin{Leftside}
2341 \def\endstanzaextra{\interstanza}
2342 \beginnumbering
2343
2344 \stanza
2345 Arma gravi numero violentaque bella parabam &
2346  edere, materi\={a} conveniente modis. &
2347 Par erat inferior versus---risisse Cupido &
2348  dicitur atque unum surripuisse pedem. \&
2349
2350 \stanza
2351 ``Quis tibi, saeve puer, dedit hoc in carmina iuris? &
2352  Pieridum vates, non tua turba \edtext{sumus}{\Afootnote{note lost}}. &
2353 Quid si praeripiat flavae V\u{e}nus arma Minervae, &
2354  ventilet accensas flava Minerva faces? \&
2355
2356 \stanza
2357 Quis probet in silvis Cererem regnare iugosis, &
2358  lege pharetratae Virginis arva coli? &
2359 Crinibus insignem quis \edtext{acuta}{\Afootnote{acut\={a} (abl.\ abs.)}}
2360 cuspide Phoebum &
2361  instruat, Aoniam Marte movente lyram? \&
2362 \endnumbering
2363 \end{Leftside}
2364
2365 \begin{Rightside}
2366 \def\endstanzaextra{\interstanza}
2367 \beginnumbering
2368 \firstlinenum{1}
2369 \linenumincrement{1}
2370 \setstanzaindents{6,0,1,0,1,0}
2371
2372 \stanza
2373 I was preparing to sing of weapons and violent wars, &
2374 in heavy numbers, with the subject matter suited to the verse measure. &
2375 The even lines were as long as the odd ones, but Cupid laughed, &
2376 they said, and he stole away one foot.\footnote{I.e., the even lines,
2377 which were hexameters (with six feet) became pentameters
2378 (with five feet).} \&
2379
2380 \stanza
2381 ``O cruel boy, who gave you the right over poetry? &
2382 We poets belong to the Pierides,\footnote{Muses} we are not your folk. &
2383 \edlabel{beginparadox}What if Venus should seize away the arms of
```

```
Minerva with the golden hair, &
 if Minerva with the golden hair should fan alight the kindled torch
of love? \&

\stanza
Who would approve of Ceres\footnote{Ceres was the Roman goddess of
the harvest.} reigning on the woodland ridges, &
 and of land tilled under the law of the Maid with the
quiver\footnote{By `\textit{Virgo}' (`Virgin') Ovid means Diana, the
Roman goddess of the hunt.}? &
Who would provide Phoebus with his beautiful hair with a sharp-pointed
spear, &
 while Mars stirs the \edtext{Aonian}{\Bfootnote{Mount Parnassus,
where the Muses live, is located in Aonia.}}
lyre?\edlabel{endparadox}\footnote{Lines
\xlineref{beginparadox}--\xlineref{endparadox} show some paradoxical
situations that would occur if the gods didn't stay with their own
business.} \&
\endnumbering
\end{Rightside}

\Pages
\end{pages}

\begin{pages}
\begin{Leftside}
\def\endstanzaextra{\interstanza}
\beginnumbering

\begin{astanza}
Arma gravi numero violentaque bella parabam &
 edere, materi\={a} conveniente modis. &
Par erat inferior versus---risisse Cupido &
 dicitur atque unum surripuisse pedem. \&
\end{astanza}

\begin{astanza}
``Quis tibi, saeve puer, dedit hoc in carmina iuris? &
 Pieridum vates, non tua turba \edtext{sumus}{\Afootnote{note lost}}. &
Quid si praeripiat flavae V\u{e}nus arma Minervae, &
 ventilet accensas flava Minerva faces? \&
\end{astanza}

\begin{astanza}
Quis probet in silvis Cererem regnare iugosis, &
 lege pharetratae Virginis arva coli? &
Crinibus insignem quis \edtext{acuta}{\Afootnote{acut\={a} (abl.\ abs.)}}
cuspide Phoebum &
 instruat, Aoniam Marte movente lyram? \&
\end{astanza}
```

```
2434
2435 \endnumbering
2436 \end{Leftside}
2437
2438 \begin{Rightside}
2439 \def\endstanzaextra{\interstanza}
2440 \beginnumbering
2441 \firstlinenum{1}
2442 \linenumincrement{1}
2443 \setstanzaindents{6,0,1,0,1,0}
2444
2445 \begin{astanza}
2446 I was preparing to sing of weapons and violent wars, &
2447 in heavy numbers, with the subject matter suited to the verse measure. &
2448 The even lines were as long as the odd ones, but Cupid laughed, &
2449 they said, and he stole away one foot.\footnote{I.e., the even lines,
2450 which were hexameters (with six feet) became pentameters
2451 (with five feet).} \&
2452 \end{astanza}
2453
2454 \begin{astanza}
2455 ``O cruel boy, who gave you the right over poetry? &
2456 We poets belong to the Pierides,\footnote{Muses} we are not your folk. &
2457 \edlabel{beginparadox}What if Venus should seize away the arms of
2458 Minerva with the golden hair, &
2459  if Minerva with the golden hair should fan alight the kindled torch
2460 of love? \&
2461 \end{astanza}
2462
2463 \begin{astanza}
2464 Who would approve of Ceres\footnote{Ceres was the Roman goddess of the
2465 harvest.} reigning on the woodland ridges, &
2466  and of land tilled under the law of the Maid with the
2467 quiver\footnote{By `\textit{Virgo}' (`Virgin') Ovid means Diana,
2468 the Roman goddess of the hunt.}? &
2469 Who would provide Phoebus with his beautiful hair with a sharp-pointed
2470 spear, &
2471  while Mars stirs the \edtext{Aonian}{\Bfootnote{Mount Parnassus, where
2472 the Muses live, is located in Aonia.}}
2473 lyre?\edlabel{endparadox}\footnote{Lines
2474 \xlineref{beginparadox}--\xlineref{endparadox} show some paradoxical
2475 situations that would occur if the gods didn't stay with their
2476 own business.} \&
2477 \end{astanza}
2478
2479 \endnumbering
2480 \end{Rightside}
2481
2482 \Pages
2483 \end{pages}
```

```
2484
2485 \end{document}
2486
2487 ⟨/djdpoems⟩
```

References

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

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

[Wil04] Peter Wilson. *ledmac A presumptuous attempt to port EDMAC,TABMAC and EDSTANZA to LaTeX*. December 2004. (Available from CTAN in `macros/latex/contrib/ledmac`)

Index

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

Symbols

A

B

C

D

E

L

R

S

T

U

V

W

X

Z

Change History