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

Peter Wilson
Herries Press†
Maïeul Rouquette‡

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

Abstract

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

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

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

Contents

1 Introduction .................................. 3

2 The ledpar package ............................... 4
   2.1 General ..................................., 4

3 Parallel columns .................................. 5

4 Facing pages ................................... 6

5 Left and right texts ............................. 6

*This file (ledpar.dtx) has version number v0.14, last revised 2012/08/14.
†herries dot press at earthlink dot net
‡maieul at maieul dot net
6 Numbering text lines and paragraphs 8
7 Verse 9
8 Implementation overview 12
9 Preliminaries 12
   9.1 Messages 13
10 Sectioning commands 13
11 Line counting 16
   11.1 Choosing the system of lineation 16
   11.2 Line-number counters and lists 19
   11.3 Reading the line-list file 20
   11.4 Commands within the line-list file 21
   11.5 Writing to the line-list file 28
12 Marking text for notes 31
13 Parallel environments 32
14 Paragraph decomposition and reassembly 34
   14.1 Boxes, counters, \pstart and \pend 35
   14.2 Processing one line 38
   14.3 Line and page number computation 40
   14.4 Line number printing 42
   14.5 Pstart number printing in side 44
   14.6 Add insertions to the vertical list 46
   14.7 Penalties 47
   14.8 Printing leftover notes 48
15 Footnotes 48
   15.1 Outer-level footnote commands 48
   15.2 Normal footnote formatting 51
16 Cross referencing 52
17 Side notes 53
18 Familiar footnotes 55
19 Verse 56
20 Naming macros 58
21 Counts and boxes for parallel texts 58
1 Introduction

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

## 2 The \texttt{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 \texttt{ledmac}’s note commands with it: this is appropriate for introductions and other material added by the editor around the edited text.

The \texttt{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

\texttt{ledmac} essentially puts each chunk of numbered text (the text within a \texttt{\pstart} \ldots \texttt{\pend}) into a box and then following the \texttt{\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.

\texttt{ledpar} similarly puts the left and right chunks into boxes but can’t immediately output the text after a \texttt{\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.

It is possible to have multiple chunks in the left and right texts before printing them. The macro \texttt{\maxchunks{(num)}} specifies the maximum number of chunks within the left or right texts. This is initially set as:
\maxchunks{10}
meaning that there can be up to 10 chunks in the left text and up to 10 chunks
in the right text, requiring a total of 20 boxes. If you need more chunks then you
can increase \maxchunks. The \maxchunks must be called in the preamble.

TeX has a limited number of boxes; if you get an error message along the lines
of ‘no room for a new box’, then load the package etex, which needs pdflatex or
xelatex. If you \maxchunks is too little you can get a ledmac error message along
the lines: ‘Too many \pstart without printing. Some text will be lost.’ then you
will have to either increase \maxchunks or use the parallel printing commands
(\Columns or \Pages) more frequently.

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

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

3 Parallel columns

pairs Numbered text that is to be set in columns must be within a \pairs 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.

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 ‘bunkier’ 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 \columnseparater if you want more control. When you use \stanza, the visible rule may shift when a verse has a hanging indent. To prevent shifting, use \setstanzaindents outside the Leftside or Rightside environment.

4 Facing pages

\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.

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

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.

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

When doing parallel pages \texttt{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 \texttt{\goalfraction} of a page has been filled, it finishes that page and starts on the other side's text. The definition is:

\texttt{\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:

\texttt{\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. The left text is put within the Leftside environment and the right text like-
wise in the \texttt{Rightside} environment. The number of \texttt{Leftside} and \texttt{Rightside}
environments must be the same.

Within these environments you can designate the line numbering scheme(s) to be used. The \texttt{ledmac} package originally used counters for specifying the numbering scheme; now both \texttt{ledmac} and the \texttt{ledpar} package use macros instead. Following \texttt{\firstlinenum\{\langle num\rangle\}} the first line number will be \langle num\rangle, and following \texttt{\linenumincrement\{\langle num\rangle\}} only every \langle num\rangle th line will have a printed number. Using these macros inside the \texttt{Leftside} and \texttt{Rightside} environments gives you independent control over the left and right numbering schemes. The \texttt{\firstsublinenum} and \texttt{\sublinenumincrement} macros correspondingly set the numbering scheme for sublines.

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

\begin{verbatim}
\begin{...side}
  \%
  \begin{numbering}
  \pstart first chunk \pend
  \pstart second chunk \pend
  ...
  \pstart last chunk \pend
  \%
  \end{numbering}
\end{...side}
\end{verbatim}

Numbering, via \texttt{\begin{numbering}} and \texttt{\end{numbering}}, may extend across several \texttt{Leftside} or \texttt{Rightside} environments. Remember, though, that the Left/Right sides are effectively independent of each other.

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

If you are using the \texttt{babel} package with different languages (via, say, \texttt{\selectlanguage}) for the left and right texts it is particularly important to select the appropriate language within the \texttt{Leftside} and \texttt{Rightside} environments. The initial language selected for the right text is the \texttt{babel} package’s default. Also, it is the last \texttt{\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

\footnote{when used with \texttt{ledpatch} v0.2 or greater.}
language selection for the side. The right side language is also applied to the right side line numbers.

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

6 Numbering text lines and paragraphs

\begin{numbering}
\end{numbering}

Each section of numbered text must be preceded by \begin{numbering} and followed by \end{numbering}, like:
\begin{numbering}
⟨text⟩
\end{numbering}

These have to be separately specified within \textsc{Leftside} and \textsc{Rightside} environments.

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

The command \texttt{\memorydump} effectively performs an \end{numbering} immediately followed by a \begin{numbering} 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}
\begin{numbering}
...
\end{numbering}
\begin{Rightside}
\begin{numbering}
...
\end{numbering}
\Pages
\begin{Leftside}
\memorydump
...
\end{Leftside}
\begin{Rightside}
\memorydump
\memorydump
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}{.}

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

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

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

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 \texttt{\textbackslash 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:

\begin{verbatim}
\newcommand*{\stanzanum}[2]{\hskip -#1\llap{\textbf{#2}}\hskip #1\ignorespaces}
\newcommand{\interstanza}{\par\mbox{}\skipnumbering}
\end{verbatim}

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

\begin{verbatim}
\setstanzaindents{1,0,1,0,1,0,1,0,1,0,1,0,1}
\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}
  ...
\end{Leftside}
\end{pairs}
\end{verbatim}

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

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

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

\renewcommand{\hangingsymbol}{[\,\,]}
8 Implementation overview

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

\texttt{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 \texttt{ledmac} code is concerned with handling this box and its contents.

\texttt{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 \texttt{ledmac}.

9 Preliminaries

Announce the name and version of the package, which is targeted for LaTeX2e. The package also requires the \texttt{ledmac} package, preferably at least version 0.13 (2011/11/08).

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

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

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

As noted above, much of the code is a duplication of the original \texttt{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 \texttt{Leftside} and \texttt{Rightside} environments set things up appropriately.
9.1 Messages

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

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

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

\lederr@LeftOnRightPage
\lederr@RightOnLeftPage
\newcommand{\lederr@LeftOnRightPage}{{%}
  \ledmac@error{The left page has ended on a right page}{\@ehc}}
\newcommand{\lederr@RightOnLeftPage}{{%}
  \ledmac@error{The right page has ended on a left page}{\@ehc}}

10 Sectioning commands

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.

\section@numR
\newcount{\section@numR}
\section@numR=\z@
\ifpst@rtedL \ifpst@rtedL is set FALSE at the start of left side numbering, and similarly for \ifpst@rtedR \ifpst@rtedR. \ifpst@rtedL is defined in ledmac.
\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} \\line@list@stuffR{\jobname.\extensionchars\the\section@numR} \l@dend@stuff \setcounter{pstartR}{1} }
\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 \iftoredL to FALSE. It is fully defined in \texttt{ledmac}.

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

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

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

\pausenumberingR\resumenumberingR

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

90 \newcommand*{\pausenumberingR}{%  
91 \endnumberingR\global\numberingRtrue}%
92 \newcommand*{\resumenumberingR}{%  
93 \ifnumberingR  
94 \global\pst@rtedRtrue  
95 \global\advance\section@numR \@one  
96 \let@mess@SectionContinued{\the\section@numR \texttt{\section@numR}}%  
97 \let@list@stuffR{\jobname.\extensionchars\the\section@numR \texttt{\section@numR}}%  
98 \le@d@stuff  
99 \else  
100 \let@err@numberingShouldHaveStarted  
101 \endnumberingR  
102 \beginnumberingR  
103 \fi}%

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

105 \newcommand*{\memorydumpL}{%  
106 \endnumbering}
11 Line counting

11.1 Choosing the system of lineation

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

The \ifbypageR and \ifbypstartR flag specify the current lineation system:

- line-of-page : bypstartR = false and bypageR = true.
- line-of-pstart : bypstartR = true and bypageR = false.

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

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

\ifnumbering
\led@err@LineationInNumbered
\else
\def\@tempa{#1}\def\@tempb{page}%
\ifx\@tempa\@tempb
\global\bypageRtrue
\global\bypstartRfalse
\else
\def\@tempb{pstart}%
\global\bypageRfalse
\global\bypstartRtrue
\fi
\else
\def\@tempb{pstart}%
\fi
\fi
11.1 Choosing the system of lineation

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

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

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

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

These are the user’s macros for changing (sub) line numbers. They are defined in \pseudocode{ledmac v0.7}, but just in case I have started by providing them.

\providecommand*{\firstlinenum}{}
\providecommand*{\linenumincrement}{}
\providecommand*{\firstsublinenum}{}
\providecommand*{\sublinenumincrement}{}
\renewcommand*{\firstlinenum}[1]{%}
\renewcommand*{\linenumincrement}[1]{%}
\renewcommand*{\firstsublinenum}[1]{%}
\renewcommand*{\sublinenumincrement}[1]{%}
\ifledRcol \setcounter{firstlinenumR}{#1}\
\else \setcounter{firstlinenum}{#1}\
\fi
\ifledRcol \setcounter{linenumincrementR}{#1}\
\else \setcounter{linenumincrement}{#1}\
\fi
\ifledRcol \setcounter{firstsublinenumR}{#1}\
\else \setcounter{firstsublinenum}{#1}\
\fi
\ifledRcol \setcounter{sublinenumincrementR}{#1}\
\else \setcounter{sublinenumincrement}{#1}\
\fi
\linenumrepR \linenumrepR{⟨ctr⟩} typesets the right line number ⟨ctr⟩, and similarly \sublinenumrepR for subline numbers.
\newcommand*{\linenumrepR}{\@arabic{#1}}
\newcommand*{\sublinenumrepR}{\@arabic{#1}}

\leftlinenumR \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 unmaintained in \pseudocode{\l@dlinenumR}.
\newcommand*{\leftlinenumR}{\l@dlinenumR}\
\newcommand*{\rightlinenumR}{\l@dlinenumR}\
\l@dlinenumR
\kern\linenumsep
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 regular or left text.

\newcommand*{\rightlinenumR}{}
\kern\linumsep
\@dlinenumR

\newcommand*{\@dlinenumR}{\numlabfont\linenumrepR{\line@numR}\@lineflag%}
\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 regular or left text.

\newcommand*{\line@numR}{}
\newcommand*{\subline@numR}{}
\newcommand*{\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 analogous 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 synchronise left and right chunks in parallel processing we need to know how many lines are in each left and right text chunk, and the maximum of these for each pair of chunks.

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

\page@numR The right text page number.
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@list\empty
\gdef\next@actionlineR{1000000}%
\else
\g@p\actionlines@list\to\next@actionlineR
\g@p\actions@list\to\next@actionR
\fi
\else
\global\page@num=\m@ne
\ifx\actionlines\empty
\gdef\next@actionline{1000000}%
\else
\g@p\actionlines\to\next@actionline
\g@p\actions\to\next@action
\fi
\fi
\else
\global\page@num=\m@ne
\ifx\actionlines\empty
\gdef\next@actionline{1000000}%
\else
\g@p\actionlines\to\next@actionline
\g@p\actions\to\next@action
\fi
\fi
11.4 Commands within the line-list file

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

11.4 Commands within the line-list file

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

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

\@l@regR \@l does everything related to the start of a new line of numbered text. Exactly 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
  \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
  \fi
  \ifcase\@lockR
    \or \@lockR \tw@
  \or
    \@lockR \z@
  \fi
  \orcase\sub@lockR
    \or \sub@lockR \tw@
  \or
    \sub@lockR \z@
  \fi
\fi
\iffalse\sublines@false\fi
\let\sub@change\relax
\let\sub@action\relax
\else
  \sublines@false
\fi
\orcase\@lockR
  \or \@lockR \tw@
  \or
    \@lockR \z@
  \fi
\orcase\sub@lockR
  \or \sub@lockR \tw@
  \or
    \sub@lockR \z@
  \fi
\fi
\iftfsublines@
We have to adjust \fix@page to handle parallel texts.

\begin{minipage}{\textwidth}
\last@page@num

\fix@page

newcount \last@page@num
\last@page@num=-10000
\renewcommand*{\fix@page}[1]{% \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@numR{#1}% \fi \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@numR{#1}% \fi \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@numR{#1}% \fi \fi \fi}
\end{minipage}

\adv The \adv{\langle num \rangle} macro advances the current visible line number by the amount specified as its argument. This is used to implement \advanceline.
The \texttt{\set{⟨num⟩}} macro sets the current visible line number to the value specified as its argument. This is used to implement \texttt{\setline}.

\begin{verbatim}
\renewcommand*{\@set}[1]{%\@set
  \ifledRcol
    \ifsublines@
      \subline@numR=#1\relax
    \else
      \line@numR=#1\relax
    \fi
    \set@line@action
  \else
    \ifsublines@
      \subline@num=#1\relax
    \else
      \line@num=#1\relax
    \fi
    \set@line@action
  \fi
\fi\fi\fi}
\end{verbatim}
\let\dchsetnum\relax
\page@action \page@action adds an entry to the action-code list to change the page number.
\set@line@action \set@line@action adds an entry to the action-code list to change the visible line number.
\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 \ifsuplines flag.

\renewcommand*{\sub@action}{%
\iffledRcol
\xright@appenditem{\the\absline@numR}\to\actionlines@listR
\ifsuplines
\xright@appenditem{-1001}\to\actions@listR
\else
\xright@appenditem{-1002}\to\actions@listR
\fi
\else
\xright@appenditem{\the\absline@num}\to\actionlines@list
\ifsuplines
\xright@appenditem{-1001}\to\actions@list
\else
\xright@appenditem{-1002}\to\actions@list
\fi
\fi}
\do@lockon \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
\ifsuplines
\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
\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
\lock@off adds an entry to the action-code list to turn line number locking off.
\do@lockoff
\lock@off
\do@lockoffR
\skip@lockoff
\renewcommand{\do@lockoffR}{% 
  \xright@appenditem{\the\absline@numR}{\to}\actionlines@listR 
  \ifsplibines0
    \xright@appenditem{-1006}{\to}\actions@listR 
    \ifnum\sub@lockR=\tw@
      \sub@lockR \thr@@R 
    \else
      \sub@lockR \z@R 
    \fi
    \else
    \xright@appenditem{-1004}{\to}\actions@listR 
    \ifnum\@lockR=\tw@
      \@lockR \thr@@R 
    \else
      \@lockR \z@R 
    \fi
  \fi
}\do@lockoffR
\renewcommand{\do@lockoff}{% 
  \ifledRcol
    \xright@appenditem{\the\absline@numR}{\to}\actionlines@listR 
  \else
    \xright@appenditem{-1007}{\to}\actions@listR 
  \fi
}\do@lockoffR
This macro implements the \skipnumbering command. It uses a new action code, namely 1007.
\skipnumbering
\n@num
\renewcommand{\n@num}{% 
  \ifsplibines0
    \xright@appenditem{-1004}{\to}\actions@listR 
  \else
    \xright@appenditem{-1007}{\to}\actions@listR 
  \fi
}\n@num
\@ref \@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
  \begingroup
  \let\@ref=\dummy@ref
  \let\page@action=\relax
  \let\sub@action=\relax
  \let\set@line@action=\relax
  \let\@lab=\relax
  \@lab=#2
  \global\endpage@num=\page@numR
  \global\endline@num=\line@numR
  \global\endsubline@num=\subline@numR
  \endgroup
  \#2
  \global\endpage@num=\page@numR
  \global\endline@num=\line@numR
  \global\endsubline@num=\subline@numR
  \endgroup
}

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
\@lab=\#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.

\right@appenditem%
{"\the\page@numR|\the\line@numR|\the\subline@numR|\the\endpage@num|
\the\endline@num|\the\endsubline@num \else 0|fi}%
\right@appenditem%
Finally, execute the second argument of \@ref again, to perform for real all the commands within it.

```latex
\@refreg{#1}{#2}\
\fi}
```

And when not in right text

```latex
\@ref{#1}{#2}\
\fi}
```

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

```latex
\providecommand*{\@pend}[1]{\xright@appenditem{#1}\to\linesinpar@listL}
\providecommand*{\@pendR}[1]{\xright@appenditem{#1}\to\linesinpar@listR}
```

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

```latex
\providecommand*{\@lopL}[1]{\xright@appenditem{#1}\to\linesonpage@listL}
\providecommand*{\@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 \texttt{ledmac} uses within the text of a section to write commands out to the line-list.

```latex
\linesonoutR \linesonoutR \linesonoutR \linesonoutR
\iffirst@linenum@out@Rtrue \first@linenum@out@R\false
\fi
```

The file for right texts will be opened on output stream \texttt{linenum@OutR}.

```latex
\newwrite{linenum@OutR}
```

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

```latex
\newif{\iffirst@linenum@out@R}
```

```latex
\first@linenum@out@Rfalse
```

```latex
\first@linenum@out@Rtrue
```
11.5 Writing to the line-list file

This is the right text version of the \line@list@stuff() macro. It is called by \beginnumbering 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@stuff}{\if\first\linenum@out@false\immediate\closeout\linenum@out\global\first\linenum@out@false\immediate\openout\linenum@out=#1\else\closeout\linenum@out\openout\linenum@out=#1\fi}

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

\newcommand*{\new@line}{\write\linenum@out{\string\@l[the\c@page][the\page]}}

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

\startsub \endsub and \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@out{\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}
\texttt{\texttt{\textbackslash \textbackslash advanceline}} You can use \texttt{\textbackslash \textbackslash advanceline\{\textit{num}\}} in running text to advance the current visible line-number by a specified value, positive or negative.

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

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

\texttt{\texttt{\textbackslash \textbackslash startlock}} or \texttt{\textbackslash \textbackslash 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.
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 accommodation for when right text is being processed.
And similarly for \texttt{edtext}.

```latex
\renewcommand{\texttt{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
```

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

```latex
\renewcommand*{\texttt{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
```

\section{Parallel environments}

The initial set up for parallel processing is deceptively simple.
pairs The *pairs* environment is for parallel columns and the *pages* environment for parallel pages.

\begin{environment}{pairs} \% \\l@dpairingtrue \\l@dpagingfalse \% \l@dpairingfalse \% \end{environment}

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

\begin{environment}{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}

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

\newif\ifinstanzaL \newif\ifinstanzaR

\begin{environment}{Leftside} \% \ledRcolfalse \let\beginnumbering\beginnumbering\setcounter{pstartL}{1} \let\pstart\pstartL \let\thepstart\thepstartL \let\pend\pendL \let\memorydump\memorydumpL \Leftsidehook \let\oldstanza\stanza

\ifinstanzaL 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\beginnumbering\beginnumbering\beginnumbering\setcounter{pstartL}{1} \let\pstart\pstartL \let\thePstart\thePstartL \let\pend\pendL \let\memorydump\memorydumpL \Leftsidehook \let\oldstanza\stanza}


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.
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 \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. The old counters are used to have the good reset of the pstart counters at the beginning of the \Pages command.

\newcounter{pstartL} \newcounter{pstartLold} \renewcommand{\thepstartL}{{\bfseries\@arabic\c@pstartL}. }
\newcounter{pstartR} \newcounter{pstartRold} \renewcommand{\thepstartR}{{\bfseries\@arabic\c@pstartR}. }
\newcommand{\pstartL}{\if@nobreak
\let@oldnobreak@nobreaktrue \else
\let@oldnobreak@nobreakfalse \fi
\nobreaktrue \ifnumbering \else
\if@nobreak@nobreakfalse \else
\let@oldnobreak@nobreaktrue \ifnumberedpar
\endnumbering \fi
\fi
\fi
\ifnumberedpar
\beginnumbering
\fi
\pstart
If this is the first `\pstart` in a numbered section, clear any inserts and set `\ifpstedL` to FALSE. Save the `pstartL` counter.

```latex
\ifpstedL\else
\setcounter{pstartLold}{\value{pstartL}}%
\list@clear{\inserts@list}%
\global\let\next@insert=\empty
\global\pstedLtrue
\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.

```latex
\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\ifautopar\else\ifnumberpstart\ifsidepstartnum\else\thepstartL\fi\fi\fi%
\hsize=Lcolwidth
\numberedpar@true}
\newcommand*{\pstartR}{
\if@nobreak
\let\@oldnobreak\@nobreaktrue
\else
\let\@oldnobreak\@nobreakfalse
\fi
\@nobreaktrue
\ifnumberingR \else
\led@err@PstartNotNumbered
\beginnumberingR
\fi
\ifpstedR\else
\setcounter{pstartRold}{\value{pstartR}}%
\list@clear{\inserts@listR}%
\global\let\next@insertR=\empty
\global\pstedRtrue
\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\ifautopar\else\ifnumberpstart\ifsidepstartnum\else\thepstartR\fi\fi\fi%
```

14  Paragraph decomposition and reassembly
14.1 Boxes, counters, \texttt{pstart} and \texttt{pend}

\def\hsize{R\textwidth}
\def\numberedpartrue}

\pendL \texttt{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 \texttt{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 \texttt{vbox} ends.

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

End the group that was begun in the \texttt{pstart}.

\endgroup
\ignorespaces
\@oldnobreak
\ifnumberpstart
  \addtocounter{pstartL}{1}
\fi}
\pendR \texttt{pend} needed for right texts.

\newcommand*{\pendR}{\ifnumberingR \else
  \led@err@PendNotNumbered
  \fi
  \ifnumberedparR \else
  \led@err@PendNoPstart
  \fi
  \l@dzeropenalties
  \endgraf\global\num@linesR=\prevgraf\egroup
  \global\par@lineR=0
  \endgroup
  \ignorespaces
  \@oldnobreak
  \ifnumberpstartR
    \addtocounter{pstartR}{1}
  \fi}

\pendR \texttt{pend} needed for right texts.
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 of right text.
\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).

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

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

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

\do@lineRend
14.3 Line and page number computation

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}{%
\ifnumberline
 \global\advance\absline@numR \@ne
 \fi
 \do@actionsR
 \do@ballastR
 \ifnumberline
 \ifsutlines@\fi
 \ifnum\sublockR<\tw@
 \global\advance\subline@numR \@ne
 \fi
 \else\fi
 \fi
\}
\newcommand*{\getline@numL}{%
\ifnumberline
 \global\advance\absline@num \@ne
 \fi
 \do@actions
 \do@ballast
 \ifnumberline
 \ifsutlines@\fi
 \ifnum\lockR<\tw@
 \global\advance\line@numR \@ne
 \global\subline@num \z@
 \fi
 \else\fi
 \fi
\}
The real work in the line macros above is done in \do@actions, but before we plunge into that, let’s get \do@ballast out of the way.

\begin{verbatim}
\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}
\end{verbatim}

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.

\begin{verbatim}
\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
    \l@dskipnumbertrue
  \else
    \led@warn@BadAction
  \fi
%
\newcommand*{\do@actionsR}{%
  \global\let\do@actions@nextR=\relax
  \@l@dtempcntb=\absline@numR
\end{verbatim}
14.4 Line number printing

\l@dcalcnum \affixline@numR is the right text version of the \affixline@num macro.
\ch@cksub@lockR; \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@lockR}{%
14.4 Line number printing

\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@
    \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
  \fi}
\newcommand*{\f@x@l@cksR}{% 
  \ifcase\@lockR
    \or
      \global\@lockR \tw@
    \or
      \global\@lockR \z@
  \fi
  \ifcase\sub@lockR
    \or
      \global\sub@lockR \tw@
    \or
      \global\sub@lockR \z@
  \fi}
\newcommand*{\affixline@numR}{% 
  \ifnumberline
    \ifl@dskipnumber

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

\newcommand*{\affixline@numR}{% 
  \ifnumberline
  \ifl@dskipnumber

\newcommand*{\affixline@numR}{% 
  \ifnumberline
  \ifl@dskipnumber

14.5 Pstart number printing in side

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

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

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

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

\add@insertsR The right text version.
\add@inserts@nextR
14.7 Penalties

\add@penaltiesL \add@penaltiesL is the last macro used by \do@lineL. It adds up the club, widow, and interline penalties, and puts a single penalty of the appropriate size back into the paragraph; these penalties get removed by the \vspli 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 not really relevant. I think that it is likely with parallel text that there is no real need to add back any penalties; even if there was, they would have to match across the left and right lines. So, I end up with the following.

\newcommand*{\add@penaltiesL}{}
\newcommand*{\add@penaltiesR}{}
14.8 Printing leftover notes

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 
    \@loop 
    \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{\vAfootnote{A}\@tag{#1}}\to\inserts@listR 
    \global\advance\insert@countR 1 
    \else 
      \xright@appenditem{\vAfootnote{A}\@tag{#1}}\to\inserts@list 
    \global\advance\insert@count 1 
    \fi 
  \else 
    \vAfootnote{A}\@tag{#1}% 
  \fi\ignorespaces}

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

\Bfootnote

We need similar commands for the other footnote series.

\Cfootnote, \Dfootnote, \Efootnote
15.1 Outer-level footnote commands

\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
  \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
  \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
  \else
  \vEfootnote{E}{0|0|0|0|0|0|0}{#1}%
  \fi\ignorespaces
\fi
\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
  \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
  \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
  \else
  \vEfootnote{E}{0|0|0|0|0|0|0}{#1}%
  \fi\ignorespaces
\fi
\fi
\else
\vBfootnote{B}{0|0|0|0|0|0|0}{#1}\
\fi\ignorespaces
\mpAfootnote  For footnotes in minipages and the like, we need a similar series of commands.
\mpBfootnote \renewcommand*{\mpAfootnote}[1]{% 
\ifnumberedpar@ \ifledRcol \xright@appenditem{\noexpand\mpvAfootnote{A} {{\l@d@nums}{\@tag}{#1}}} \to\inserts@listR \global\advance\insert@countR 1 \else \xright@appenditem{\noexpand\mpvAfootnote{A} {{\l@d@nums}{\@tag}{#1}}} \to\inserts@list \global\advance\insert@count 1 \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 1 \else \xright@appenditem{\noexpand\mpvBfootnote{B} {{\l@d@nums}{\@tag}{#1}}} \to\inserts@list \global\advance\insert@count 1 \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 1 \else \xright@appenditem{\noexpand\mpvCfootnote{C} {{\l@d@nums}{\@tag}{#1}}} \to\inserts@list \global\advance\insert@count 1 \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 1 \else \xright@appenditem{\noexpand\mpvDfootnote{D} {{\l@d@nums}{\@tag}{#1}}} \to\inserts@list \global\advance\insert@count 1 \fi \else \mpvDfootnote{D}{{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. Just in case, \ledsavedprintlines is a copy of the original \printlines.

Just a reminder of the arguments:
\printlinesR #1 | #2 | #3 | #4 | #5 | #6 | #7 \printlinesR start-page | line | subline | end-page | line | subline | font

1415 \def\printlinesR#1|#2|#3|#4|#5|#6|#7|{egingroup
1416 \setprintlines{#1}{#2}{#3}{#4}{#5}{#6}\
1417 \ifl@d@pnum #1\fullstop\fi
1418 \ifledplinenum\linenumr@p{#2}\Rlineflag\else\symplinenum\fi
1419 \ifl@dd@sub \fullstop \sublinenumr@p{#3}\fi
1420 \ifl@dd@dash \endashchar\fi
1421 \ifl@dd@pnum #4\fullstop\fi
1422 \ifl@dd@elin \linenumr@p{#5}\Rlineflag\fi
1423 \ifl@dd@esl \ifl@dd@elin \fullstop\fi \sublinenumr@p{#6}\fi
1424 \endgroup}
1425
1426 \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.
1428 \list@create{\labelref@listR}
1429
\edlabel The \edlabel command first writes a \@lab macro to the \linenum@out file. It then checks to see that the \labelref@list actually has something in it (if not, it creates a dummy entry), and pops the next value for the current label, storing it in \label@refs. Finally it defines the label to be \empty so that any future check will turn up the fact that it has been used.
1430 \renewcommand*{\edlabel}[1]{\@bsphack
1431 \ifledRcol
1432 \write\linenum@outR{\string\@lab}\
1433 \ifx\labelref@list\empty
1434 \edef\label@refs{zz\null}\%
1435 \else
1436 \gl@p\labelref@listR\to\label@refs
1437 \fi
1438 \ifvmode
1439 \advancelabel@refs
1440 \fi
1441 \protected@write\@auxout{}%
1442 {\string\l@dmake@labelsR\space\thepage|\label@refs|{#1}}%
This is the right text version of \l@dmake@labelsR, taking account of \Rlineflag.

\l@dmake@labelsR 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.

\l@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.

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.
\texttt{\csname sidenote@margin\endcsname} \hspace{1em} Specifies which margin sidenotes can be in.
\texttt{\csname sidenotemargin\endcsname} \hspace{1em} \texttt{\newcount\csname sidenote@margin\endcsname}
\texttt{\renewcommand*{\csname sidenotemargin\endcsname}[1]{%}
\texttt{\ifnum\csname Sidebar@margin\endcsname<\m@ne}
\texttt{\ifledRcol}
\texttt{\global\csname sidenote@margin\endcsname=\csname Sidebar@margin\endcsname}
\texttt{\else}
\texttt{\global\csname sidenote@margin\endcsname=\csname Sidebar@margin\endcsname}
\texttt{\fi}}
\texttt{\sidenotemargin{right}}
\texttt{\global\csname sidenote@margin\endcsname=\onedimen}
\texttt{\global\csname sidenote@margin\endcsname=\onedimen}
\texttt{\l@dlsnote}
\texttt{\l@drsnote}
\texttt{\l@dcsnote} The ‘footnotes’ for left, right, and moveable sidenotes. The whole scheme is reminiscent of the critical footnotes code.
\texttt{\renewcommand*{\csname l@dlnote\endcsname}[1]{%}
\texttt{\ifnumberedpar}%
\texttt{\ifledRcol}
\texttt{\xright@appenditem{\noexpand\vl@dlsnote{#1}}%}
\texttt{\to\inserts@listR}
\texttt{\global\advance\insert@countR \onedimen}
\texttt{\else}
\texttt{\xright@appenditem{\noexpand\vl@dlsnote{#1}}%}
\texttt{\to\inserts@list}
\texttt{\global\advance\insert@count \onedimen}
\texttt{\fi}}
\texttt{\renewcommand*{\csname l@drsnote\endcsname}[1]{%}
\texttt{\ifnumberedpar}%
\texttt{\ifledRcol}
\texttt{\xright@appenditem{\noexpand\vl@drsnote{#1}}%}
\texttt{\to\inserts@listR}
\texttt{\global\advance\insert@countR \onedimen}
\texttt{\else}
\texttt{\xright@appenditem{\noexpand\vl@drsnote{#1}}%}
\texttt{\to\inserts@list}
\texttt{\global\advance\insert@count \onedimen}
\texttt{\fi}}
\texttt{\renewcommand*{\csname l@dcsnote\endcsname}[1]{%}
\texttt{\ifnumberedpar}%
\texttt{\ifledRcol}
\texttt{\xright@appenditem{\noexpand\vl@dcsnote{#1}}%}
\texttt{\to\inserts@listR}
\texttt{\global\advance\insert@countR \onedimen}
\texttt{\else}
\texttt{\xright@appenditem{\noexpand\vl@dcsnote{#1}}%}
\texttt{\to\inserts@list}
\texttt{\global\advance\insert@count \onedimen}
\texttt{\fi}}
\affixside@note \ 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\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}
19 Verse

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

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

\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.
\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 The maximum number of chunk pairs before printing has to be called for. The
default is 10 chunk pairs.

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

\l@dnumpstartsL The numbers of left and right chunks. \l@dnumpstartsL is defined in ledmac.
\l@dnumpstartsR
\newcount\l@dnumpstartsR
\l@ncL
\l@ncR A couple of scratch counts for use in left and right texts, respectively.
\l@ncL
\newcount\l@ncL
\l@ncR
\newcount\l@ncR
\l@dsetuprawboxes This macro creates \maxchunks pairs of boxes for left and right chunks. The boxes
are called \l@dLcolrawbox1, \l@dLcolrawbox2, etc.
\l@dsetupmaxlinecounts
\l@dzeromaxlinecounts To be able to synchronise left and right texts we need to know the maximum num-
ber of text lines there are in each pair of chunks. \l@dsetupmaxlinecounts creates
\maxchunks new counts called \l@dmaxlinesinpar1, etc., and \l@dzeromaxlinecounts
zeros all of them.

\l@dsetupmaxlinecounts
\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}

\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.
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.

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

\newif\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.

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

\selectlanguage
\label{10duselanguage}
\selectlanguage is a \texttt{babel} command. \texttt{\theledlanguageL} and \texttt{\theledlanguageR} are the names of the languages of the left and right texts. \texttt{\10duselanguage} is similar to \texttt{\selectlanguage}.

Now do the \texttt{babel} fix or \texttt{polyglossia}, if necessary.

\AtBeginDocument{% 
\10dusedbabelfalse
\renewcommand*{\selectlanguage}[1]{% 
Here we deal with the case where \texttt{babel} has been used. \texttt{\selectlanguage} has to be redefined to use our version of \texttt{\bb@set@language} and to store the left or right language.

\10dusedbabeltrue
\let\10oldselectlanguage\selectlanguage
\let\10oldbb@set@language\bb@set@language
\let\bb@set@language\l@dbbl@set@language
\renewcommand{\selectlanguage}[1]{% 
\10oldselectlanguage{#1}% 
\if-ledRcol \gdef\theledlanguageR{#1}% 
\else \gdef\theledlanguageL{#1}% 
\fi%
\10duselanguage simply calls the original \texttt{\selectlanguage} so that \texttt{\theledlanguageL} and \texttt{\theledlanguageR} are unaltered.

\renewcommand*{\10duselanguage}[1]{% 
\10oldselectlanguage{#1}
Lastly, initialise the left and right languages to the current \texttt{babel} one.

\begin{macrocode}
\gdef\theledlanguageL{\bbl@main@language}\
\gdef\theledlanguageR{\bbl@main@language}\
\end{macrocode}

If on Polyglossia

\begin{macrocode}
\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}
\end{macrocode}

\begin{macrocode}
\begin{macrocode}
\end{macrocode}

\section{Parallel columns}

The \texttt{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.

\begin{macrocode}
\newcommand*{\Columns}{%\
\setcounter{pstartL}{\value{pstartLold}}\
\setcounter{pstartR}{\value{pstartRold}}\
\ifnum\l@dnumpstartsL=\l@dnumpstartsR\else\
\led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}\
\fi\
\begin{group}
\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@pstarts\
\loop\if@pstarts\
\global\pstartnumtrue\
\global\pstartnumRtrue\
\checkp@start\
\loop\if\p@startnum\
\p@startp@startR=\p@startp@startL\
\p@startnum=\p@startnum-1\
\repeat\
\p@startnum=\z@\
\fi\
\fi\
\p@startnum=\z@\
\fi\
\end{group}\
\end{macrocode}

\begin{macrocode}
\end{macrocode}

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

\begin{macrocode}
\check@pstarts\
\loop\if@pstarts\
\global\pstartnumtrue\
\global\pstartnumRtrue\
\global\p@startnum=\p@startnum-1\
\repeat\
\p@startnum=\z@\
\fi\
\end{macrocode}
Increment \@dpaeL and \@dpaeR which here count the numbers of left and right chunks.
\global\advance\@dpaeL \@ne
\global\advance\@dpaeR \@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
  \ifraw@text
    \do@lineL
    \do@lineR
    \else
    \l@duselanguage{\theledlanguageL} \%
    \do@lineL
    \l@duselanguage{\theledlanguageR} \%
    \do@lineR
  \fi
  \hb@xt@ \hsize{\hfill \unhbox\l@dleftbox \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. Increment pstart counters and reset line numbering if it’s by pstart.
\@writelinesinparL
\@writelinesinparR
\check@pstarts
{ \ifbypstart\%
  \write\linenum@out\{\string\@set[1]\}
  \fi
  \ifbypstart\R \%
  \write\linenum@out\R\{\string\@set[1]\}
  \fi
  \addtocounter{pstartL}{1}
  \addtocounter{pstartR}{1}
} %

Having output all chunks, make sure all notes have been output, then zero counts ready for the next set of texts. The boolean tests for stanza are switched to false.
\flush@notes
\flush@notesR
\endgroup
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@
\@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}{% 
  \setcounter{pstartL}{\value{pstartRold}} 
  \setcounter{pstartR}{\value{pstartLold}} 
  \typeout{} 
  \typeout{*************************** PAGES ***************************} 
  \ifnum\l@dnumpstartsL=\l@dnumpstartsR\else 
    \led@err@BadLeftRightPstarts{\the\l@dnumpstartsL}{\the\l@dnumpstartsR}% 
  \fi 
  \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 
  \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.

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

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

Zero the counts again, ready for the next bit.

Get the number of lines on the first pair of pages and store the minimum in \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.

Increment the chunk counts to get the first pair.

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

Start a loop over the boxes (chunks).
See if there is more that can be done for the left page and set up the left language.

\checkpageL
\@duselanguage{\thedlanguageL}%
\%
\begingroup
\%
\loop\ifl@dsamepage
\do@lineL
\advance\numpagelinesL \@ne
\ifshiftedverses
\ifdim\ht\l@dleftbox>0pt \hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}\fi%
\else
\hb@xt@ \hsize{\ledstrutL\unhbox\l@dleftbox}%
\fi
\get@nextboxL
\checkpageL
\repeat

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.
\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
\@duselanguage{\thedlanguageR}%
\%
\loop\ifl@dsamepage
\do@lineR
\advance\numpagelinesR \@ne
\ifshiftedverses
\ifdim\ht\l@drightbox>0pt \hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}\fi%
\else
\hb@xt@ \hsize{\ledstrutR\unhbox\l@drightbox}%
\fi
\get@nextboxR
The page is full, so move onto the next (left, odd) page and repeat left text processing.

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.

We have now output the text from all the chunks.

Make sure that there are no inserts hanging around.

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

Struts inserted into left and right text lines.
and \clearleftpage 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{} \cleartoevenpage \fi
\ifdim\pagetotal<\topskip% on an empty page
\else
\clearpage
\ifodd\c@page \hbox{} \cleardoublepage \fi
\fi
\newcommand*{\cleartoleftpage}{\clearpage \ifodd\c@page \else \led@err@LeftOnRightPage \hbox{} \cleardoublepage \fi}
\newcommand*{\clearrightpage}{\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.
\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.
These macros output the number of lines on a page to the section file in the form of \lopL or \lopR macros.

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

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

\if\l@dsamepage \checkpageL tests if the space and lines already taken on the page by text and footnotes is less than the constraints. If so, then \if\l@pagefull is set FALSE and if \l@dsamepage is set TRUE. If the page is spatially full then \if\l@pagefull is set TRUE and \if\l@dsamepage is set FALSE. If it is not spatially full but the maximum number of lines have been output then both \if\l@pagefull and \if\l@dsamepage are set FALSE.
\texttt{\textbackslash ledthegoal} \texttt{\textbackslash goalfraction} \texttt{\textbackslash check@goal}

\texttt{\textbackslash 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 \texttt{\textbackslash goalfraction}. \texttt{\textbackslash ledthegoal} is calculated via \texttt{\textbackslash check@goal}.

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

The current box is not empty; do nothing.

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

Sufficient lines have been output.

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

There are still unprocessed boxes. Recalculate the maximum number of lines needed, and move onto the next box (by incrementing \ldpsc). If needed, restart the line numbering. Increment the pstart counter.
\ifnum\l@dnumpstartsR>\l@dpscR
\writtenlinesRfalse
\ifbypstart@R
\ifnum\value{pstartR}<\value{pstartRold}
\else
\global\line@numR=0
\fi
\fi
\fi
\addtocounter{pstartR}{1}
\global\pstartnumRtrue
\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

\end
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,  
Qui n’avait que peu de religion.  
Il dit: ‘Quant `a moi,  
Je déteste tous les trois,  
Le Père, et le Fils, et le Pigeon.’

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

Dieu mercy et Tacque Thibault,  
Qui tant d’eau froid m’a fait boire,  
Mis en bas lieu, non pas en hault,  
Mengier d’angoisse maints poire,  
Je Prie pour luy et reliqua.


---

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 Thibaut | A favourite of Jean, Duc de Berry and loathed for his exactions and debauchery. Villon uses his name as an insulting nickname for Thibaud d’Auxigny, the Bishop of Orléans.
2r cold water | Can either refer to the normal prison diet of bread and water or to a common medieval torture which involved forced drinking of cold water.
1 De ecclesia S. Stephani Novimagensi

Nobilis itaque Comes Otto imperio et dominio Novimagensi si bi, ut praeferetur, impignoratis et commissis prouinciae praecessse cупiens, anno LIII superius descripto, mense Junio, 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 desrueretur et infra muros transferetur ac de novo construeretur, a reverendo patre domino Conrado de Hofsteden, archiepiscopo Colonensi, licentiam, et a venerabilibus dominis decano et capitulo sanctorum Apostolorum Colonensi, ipsius ecclesiae ab antico veris et pacificis patronis, consensum, citra tamen praeindicium, damnum aut gravamen iurium et bonorum eorumdem, 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 re compensationem illius areae infra castrum et portam, quae fuit dos ecclesiae, in qua plebanus habitare solet—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


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 indulgens...”

11–19 Cf. Sloet, Oorkondenboek nr. 762 (June 1254)

2

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 the judge, the sheriffs and other citizens of Nijmegen obtained permission to demolish the parish church that lay outside the town walls,2 to move it inside the walls and to rebuild it new. This operation was necessary and useful both for Otto himself and for the inhabitants of the town. The reverend father Conrad of Hochstaden, archbishop of Cologne,3 gave his permission. So did the reverend dean and canons of the chapter of St. Apostles’ in Cologne, who had long4 been the true and benevolent patrons of the church—but they did not allow Otto to do anything without their knowledge, nor to infringe their rights, nor to damage their property.

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

// One additional line to show synchronization. //

1In 1247 William II (1227–1256) count of Holland needed money to fight his way to Aachen to be crowned King of the Holy Roman Empire. He gave the town of Nijmegen in pledge to Otto II (1229–1271) count of Guelders.
2Since the early seventh century old St. Stephen’s church had been located close to the castle, at today’s Kelfkensbos square. Traces of the church and the presbytery were found during excavations in 1998–1999.
3Conrad of Hochstaden († 1261) was archbishop of Cologne in 1238–1261. Nijmegen belonged to the archdiocese of Cologne until 1559.
4They probably became the patrons when the chapter was established in the early eleventh century. About the church and the chapter, see Gottfried Stracke, Köln: St. Aposteln, Stadtspuren – Denkmäler in Köln, vol. 19, Köln: J. P. Bachem, 1992.
5Nowadays, the exact location of the medieval ditch—and of two Roman ones—can be seen in the pavement of Kelfkensbos square.
6The original letter is lost. A 15th century transcription of it is kept at the Historisches Archiv der Stadt Köln (HASTK).
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.)
I was preparing to sing of weapons and violent wars,
in heavy numbers, with the subject matter suited to the verse measure.
The even lines were as long as the odd ones, but Cupid laughed,
they said, and he stole away one foot.\footnote{1}

\begin{verse}
“O cruel boy, who gave you the right over poetry?
We poets belong to the Pierides,\footnote{2} we are not your folk.
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?
Who would approve of Ceres\footnote{3} reigning on the woodland ridges,
and of land tilled under the law of the Maid with the quiver?\footnote{4}
Who would provide Phoebus with his beautiful hair with a sharp-pointed spear,
while Mars stirs the Aonian lyre?\footnote{5}
\end{verse}

\footnotetext[1]{I.e., the even lines, which were hexameters (with six feet) became pentameters (with five feet).}
\footnotetext[2]{Muses}
\footnotetext[3]{Ceres was the Roman goddess of the harvest.}
\footnotetext[4]{By ‘Virgo’ (‘Virgin’) Ovid means Diana, the Roman goddess of the hunt.}
\footnotetext[5]{Lines 7R–12R show some paradoxical situations that would occur if the gods didn’t stay with their own business.}

\footnote{12R Aonian} Mount Parnassus, where the Muses live, is located in Aonia.

Figure 5: First right page output from \texttt{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 Vēnus 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.
I was preparing to sing of weapons and violent wars,
in heavy numbers, with the subject matter suited to the verse measure.

The even lines were as long as the odd ones, but Cupid laughed,
they said, and he stole away one foot.6

"O cruel boy, who gave you the right over poetry?
We poets belong to the Pierides,7 we are not your folk.

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?

Who would approve of Ceres8 reigning on the woodland ridges,
and of land tilled under the law of the Maid with the quiver9?

Who would provide Phoebus with his beautiful hair with a sharp-pointed spear,
while Mars stirs the Aonian lyre?10

---

6I.e., the even lines, which were hexameters (with six feet) became pentameters (with five feet).
7Muses
8Ceres was the Roman goddess of the harvest.
9By ‘Virgo’ (‘Virgin’) Ovid means Diana, the Roman goddess of the hunt.
10Lines 7R–12R show some paradoxical situations that would occur if the gods didn’t stay with their own business.

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

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.

\begin{verbatim}
\documentclass{article}
\setstanzaindents{0,0,0,0,0,0,0,0,0}
\renewcommand{\Lcolwidth}{0.46\textwidth}
\renewcommand{\Rcolwidth}{\Lcolwidth}
\renewcommand{\Rlineflag}{0}
\renewcommand{\Rlineflag}{r}
\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.
\begin{pairs}
\stanza
Il y avait un jeune homme de Dijon, &
Qui n’avait que peu de religion. &
Il dit: ‘Quant à moi, &
Je déteste tous les trois, &
Le Père, et le Fils, et le Pigeon.’ &
\endstanza
\end{pairs}
\end{document}
\end{verbatim}
A.1 Parallel column example

\begin{Rightside}
\% different right text line numbering sequence
\firstlinenum{1}
\linenumincrement{2}
\linenummargin{right}
\beginnumbering
\stanza
There was a young man of Dijon, &
Who had only a little religion, &
He said: ‘As for me, &
I detest all the three, &
The Father, the Son, and the Pigeon.’ &
\endnumbering
\end{Rightside}

The following is verse \textsc{lxxiii} of Fran\c{c}ois Villon’s \textit{Le Testament} (The Testament), composed in 1461.

\setstanzaindents{1,0,0,0,0,0,0,0,0}
\setlength{\Lcolwidth}{0.46\textwidth}
\setlength{\Rcolwidth}{\Lcolwidth}
\vspace*{\baselineskip}
\begin{pairs}
\begin{Leftside}
\firstlinenum{2}
\linenumincrement{2}
\linenummargin{left}
\beginnumbering
\stanza
Dieu mercy et Tacque Thibault, &
Qui tant d’eau froid m’a fait boire, &
Mis en bas lieu, non pas en haut, &
Mengier d’angoisse maints \edtext{poire}{\lemma{poire d’angoisse}}%\Afootnote{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.}, &
Enferr\’e \ldots Quant j’en ay memoire, &
Je Prie pour luy \edtext{et reliqua}{\Afootnote{and so on}}, &
Que Dieu luy doint, et voire, voire! &
Ce que je pense \ldots \textit{et cetera}. &
\endnumbering
\end{pairs}
\end{Leftside}
Thanks to God --- and to 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. &

Who made me drink so much 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. &

Put me underground instead of higher up &

And made me eat such bitter fruit, &

In chains &

When I think of this, &

I pray for him——&

May God grant him (yes, by God) &

What I think &

The translation and notes are by Anthony Bonner,

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.
\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\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}\% to here\setbox0=\vbox{\hsize=\maxdimen\noindent\csname #1footfmt\endcsname#2}\setbox0=\hbox{\unvxh0}\dp0=0pt\ht0=\csname #1footfudgefactor\endcsname\wd0\box0\fi}
\newcommand*{\previous@A@number}{-1}
\newcommand*{\previous@B@number}{-1}
\newcommand*{\previous@C@number}{-1}
\newcommand*{\previous@page}{-1}
\newcommand{\abb}{\let\rbracket\nobrak\relax}
\newcommand{\nobrak}{\textnormal{}}
\newcommand{\morenoexpands}{\let\abb=0\}}
\newcommand{\Aparafootfmt}[3]{\letsetnormalparstuff
\scriptsize\notenumfont\printlines#1|\enspace\lemmafont#1|#2\enskip\notetextfont#3\penalty-10\hskip 1em plus 4em minus.4em\relax}
\newcommand{\Bparafootfmt}[3]{\letsetnormalparstuff\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]{\letsetnormalparstuff\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}
Nobilis itaque comes Otto imperio et dominio Novimagensi sibi, ut praefertur, impignoratis praeesse cupiens, anno \textsc{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 \textit{extra civitatem} sita destrueretur et \textit{infra} muros transferetur, a reverendo patre domino Conrado de Hofsteden, archiepiscopo Coloniensi, licentiam obtinuimus.

Nos devotionis tue precibus annuentes, ut ipsam ecclesiam faciens demoliiri transferas in locum alium competenter, tibi auctoritatem presentium indulgimus, et a venerabilibus dominis decano et capitulo sanctorum Apostolorum permissionem in Church in Cologne had been obtained as early as 1249. Cf. Sloet’s Oorkondenboek nr. 707 (14 November 1249): ‘Nos devotionis tue precibus annuentes, ut ipsam ecclesiam faciens demoliiri transferas in locum alium competenter, tibi auctoritatem presentium indulgimus, et a venerabilibus dominis decano et capitulo sanctorum Apostolorum permissionem in Church in Cologne had been obtained as early as 1249. Cf. Sloet’s Oorkondenboek nr. 707 (14 November 1249): ‘Nos devotionis tue precibus annuentes, ut ipsam ecclesiam faciens demoliiri transferas in locum alium competenter, tibi auctoritatem presentium indulgimus, et a venerabilibus dominis decano et capitulo sanctorum Apostolorum permissionem in Church in Cologne had been obtained as early as 1249. Cf. Sloet’s Oorkondenboek nr. 707 (14 November 1249): ‘Nos devotionis tue precibus annuentes, ut ipsam ecclesiam faciens demoliiri transferas in locum alium competenter, tibi auctoritatem presentium indulgimus, et a venerabilibus dominis decano et capitulo sanctorum Apostolorum permissionem in Church in Cologne had been obtained as early as 1249. Cf. Sloet’s Oorkondenboek nr. 707 (14 November 1249): ‘Nos devotionis tue precibus annuentes, ut ipsam ecclesiam faciens demoliiri transferas in locum alium competenter, tibi auctoritatem
dendum

dominii

\textit{Coloniensi}, ipsius ecclesiae ab
antiquo veris et pacificis patronis, consensum, citra tamen praejudicium, damnum aut gravamen iurium et bonorum eorundem, impetravit.

Et exinde liberum locum eiusdem civitatis qui dicitur Hundisburg, de praelibati Wilhelmi Romanorum regis fundi do mini, consensu, ad aedificandum ecclesiam et coemeterium, eiusdem decano et capitulo de expresso 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 destructa---siam aream competentem et ecclesiae novae, satis contiguum, ipsi plebano darent et assignarent.}

Et desuper dictam ecclesiam sanctorum Apostolorum littera sigillis ipsorum Otonis comitum et civitatis Novimagensis sigillata sigillis communita.}

St. Stephen’s Church in Nijmegen
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\ledisidenote{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.\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{Köln: St. Aposteln}, Stadtspuren -- Denkm\"{a}ler in Köln, vol. 19, Köln: 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.

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.\protect\index{St. Apostles’ (Cologne)} 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.\protect\index{St. Apostles’ (Cologne)}.
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 \endstanzaeextra to give a short line after each stanza.
\begin{document}
\newcommand{\interstanza}{\pstart\centering\longdash\skipnumbering\pend}
\begin{pages}
\begin{Leftside}
\beginnumbering
\stanza
Arma gravi numero violentaque bella parabam \&
edere, materi\={a} conveniente modis. \&
Par erat inferior versus---risisse Cupido \&
dicitur atque unum surripuisse pedem. \&
\stanza
'Quis tibi, saeve puer, dedit hoc in carmina iuris? \&
Pieridum vates, non tua turba \edtext{sumus}{\Afootnote{note lost}}. \&
Quid si praeripiat flavae V\={e}nus arma Minervae, \&
ventilet accensas flava Minerva faces? \&
\stanza
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? \&
\endnumbering
\end{Leftside}
\begin{Rightside}
\beginnumbering
\firstlinenum{1}
\linenumincrement{1}
\setstanzaindents{6,0,1,0,1,0}
\stanza
I was preparing to sing of weapons and violent war, \&
in heavy numbers, with the subject matter suited to the verse measure. \&
The even lines were as long as the odd ones, but Cupid laughed, \&
they said, and he stole away one foot. \footnote{I.e., the even lines, which were hexameters (with six feet) became pentameters} \&
\stanza
'O cruel boy, who gave you the right over poetry? \&
We poets belong to the Pierides, \footnote{Muses} we are not your folk. \&
\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.}}? &

\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\endstanzaxtra{\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 \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, Aoni\=am Marte movente lyram? &

\end{astanza}
A.3 Example poetry on parallel facing pages

I was preparing to sing of weapons and violent wars, &
in heavy numbers, with the subject matter suited to the verse measure. &
The even lines were as long as the odd ones, but Cupid laughed, &
they said, and he stole away one foot. \footnote{I.e., the even lines,}
which were hexameters (with six feet) became pentameters
(with five feet). &

'O cruel boy, who gave you the right over poetry? &
We poets belong to the Pierides, \footnote{Muses} we are not your folk. &
\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? &
\end{astanza}

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.? &
\end{astanza}

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.} &
\end{astanza}
A Examples

\end{document}

⟨/djpoeems⟩
References


[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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;</td>
<td>1609, 1610, 1614, 1631, 1645, 2192, 2207, 2244, 2267, 2559, 2595, 2602, 2619, 2627, 2642, 2658, 2665, 2673, 2692, 2701, 2717</td>
</tr>
<tr>
<td>@M</td>
<td>1620</td>
</tr>
<tr>
<td>@MM</td>
<td>2318</td>
</tr>
<tr>
<td>@adv</td>
<td>337, 606, 607</td>
</tr>
<tr>
<td>@afterindentfalse</td>
<td>717</td>
</tr>
<tr>
<td>@arabic</td>
<td>194, 195, 766, 769</td>
</tr>
<tr>
<td>@astanza@line</td>
<td>1630, 1636, 1639</td>
</tr>
<tr>
<td>@auxout</td>
<td>1441, 1453, 1719</td>
</tr>
<tr>
<td>@chapter</td>
<td>718</td>
</tr>
<tr>
<td>@cs@linesinparL</td>
<td>1893, 2004</td>
</tr>
<tr>
<td>@cs@linesinparR</td>
<td>1893, 2004</td>
</tr>
<tr>
<td>@cs@linesonpageL</td>
<td>1901, 1961, 2017</td>
</tr>
<tr>
<td>@cs@linesonpageR</td>
<td>1901, 1961, 2017</td>
</tr>
<tr>
<td>@donetotallinesL</td>
<td>872, 901, 1855, 1857, 1907</td>
</tr>
<tr>
<td>@donetotallinesR</td>
<td>872, 901, 1855, 1857, 1907</td>
</tr>
<tr>
<td>@lock</td>
<td>529, 1432, 1444, 1468</td>
</tr>
<tr>
<td>@lockR</td>
<td>888, 975</td>
</tr>
<tr>
<td>@reg</td>
<td>309</td>
</tr>
<tr>
<td>@regR</td>
<td>260</td>
</tr>
<tr>
<td>@l</td>
<td>410, 412, 414, 415, 419, 421, 423, 424, 994, 1033, 1034, 1036, 1038, 1041, 1042, 1059–1063, 1065, 1072, 1077, 1081, 1089, 1094, 1098, 1131, 1134, 1136, 1140</td>
</tr>
<tr>
<td>@l@dtempcnta</td>
<td>153, 155, 157, 1024, 1025, 1072, 1077, 1081, 1089, 1094, 1098, 1123, 1127, 1140, 1148–1150, 1152, 1172–1174, 1176, 1193–1195, 1197, 1482, 1484, 1486, 1539–1541, 1543, 1673–1677, 1681–1684, 1688–1691</td>
</tr>
<tr>
<td>@l@dtempcntb</td>
<td>153, 155, 157, 1024, 1025, 1072, 1077, 1081, 1089, 1094, 1098, 1123, 1127, 1140, 1148–1150, 1152, 1172–1174, 1176, 1193–1195, 1197, 1482, 1484, 1486, 1539–1541, 1543, 1673–1677, 1681–1684, 1688–1691</td>
</tr>
<tr>
<td>@l@reg</td>
<td>309</td>
</tr>
<tr>
<td>@l@regR</td>
<td>260</td>
</tr>
<tr>
<td>@l@reg</td>
<td>309</td>
</tr>
<tr>
<td>@l@regR</td>
<td>260</td>
</tr>
<tr>
<td>@l</td>
<td>410, 412, 414, 415, 419, 421, 423, 424, 994, 1033, 1034, 1036, 1038, 1041, 1042, 1059–1063, 1065, 1072, 1077, 1081, 1089, 1094, 1098, 1131, 1134, 1136, 1140</td>
</tr>
<tr>
<td>@lock</td>
<td>529, 1432, 1444, 1468</td>
</tr>
<tr>
<td>@lockR</td>
<td>888, 975</td>
</tr>
<tr>
<td>@lab</td>
<td>529, 1432, 1444, 1468</td>
</tr>
<tr>
<td>@lock</td>
<td>58, 282, 284, 286, 299, 444, 460, 461, 463, 464, 492, 493, 495, 923, 956, 1000, 1002, 2116, 2118</td>
</tr>
</tbody>
</table>
Change History

v0.1
  General: First public release . . . . . 1

v0.10
  General: \edlabel commands on the right side are now correctly indicated. 1
  \edlabel commands which start a paragraph are now put in the right place. 1

v0.11
  General: Change \do@lineR and \do@lineL to allow line numbering by pstart (like in ledmac 0.15). 36
  Lineation can be by pstart (like in ledmac 0.15). 14
  New management of hanging symbol insertion, preventing undesirable insertions. 54
  Prevent shift of column separator when a verse is hanged. 54

\affixline@numR: Changed
\affixline@numR to allow to disable line numbering (like in ledmac 0.15) ................. 40
\Columns: Line numbering by pstart. ............................... 61
\get@nextboxR: Change \get@nextboxL and \get@nextboxR to allow to disable line numbering (like in ledmac 0.15). ..................... 70
Pstart number can be printed in side .............................. 70
v0.12  
General: New new management of hangingsymbol insertion, preventing undesirable insertions. 54
v0.13  
General: Report ledmac 0.14 debug of \lineation ..................... 1
v0.2  
General: Added section of babel related code ...................... 58
Fix babel problems ............................................. 1
\Columns: Added \@dchecklang and \@duselanguage to \Columns ........................................ 61
\Pages: Added \@duselanguage to \Pages ................................ 65
v0.3  
General: Reorganize for ledarab ... 1
\affixline@numR: Changed \affixline@numR to match new ledmac ...................... 40
\do@actions@nextR: Used \do@actions@fixedcode in \do@actionsR ...................... 39
\do@lineL: Added \do@lineLhook to \do@lineL ...................... 36
Simplified \do@lineL by using macros for some common code 36
\do@lineR: Changed \do@lineR similarly to \do@lineL .......... 37
\do@lineRhook: Added \do@lineRhook and \do@lineRhookhook .......... 37
Leftside: Added hooks into Leftside environment .............. 31
\flag@end: Removed extraneous spaces from \flag@end .......... 27
\ifledRcol: Moved \ifledRcolpaireing to ledmac .................. 11
v0.3a  
General: Minor \linenummargin fix ........................................ 1
\line@marginR: Don't just set \line@marginR in \linenummargin ........................... 15
v0.3b  
General: Improved parallel page balancing ....................... 1
\Pages: Added \@dminpagelines calculation for succeeding page pairs ......................................... 66
v0.3c  
General: Compatibility with Polyglossia ............................. 1
v0.4  
General: No more ledparpatch. All patches are now in the main file. .............................. 1
v0.5  
General: Corrections about \section and other titles in numbered sections ........... 1
v0.6
General: Be able to use \chapter in parallel pages. ................. 1
v0.7
General: Option ‘shiftedverses’ which make there is no blank between two parallel verses with inequal length. ................. 1
v0.8
General: Possibility to have a symbol on each hanging of verses, like in the french typography. Redefine the commande \hangingsymbol to define the character. ................. 1
v0.9
General: Possibility to number \pstart. ....................... 7
Possibility to number the pstart with the commands \numberpstarttrue. ............. 1
\ifledRcol: Moved \ifledRcol and \ifnumberingR to ledmac 11
v0.9.1
General: The numbering of the pstarts restarts on each \beginnumbering. ................. 1
v0.9.2
General: Debug : with \Columns, the hanging indentation now runs on the left columns and the hanging symbol is shown only when \stanza is used. ............. 1
v0.9.3
General: \thepstartL and \thepstartR use now \textbfseries and not \textbf, which is deprecated and makes conflicts with memoir class. ...... 1