eledmac
A presumptuous attempt to port
EDMAC, TABMAC and EDSTANZA to LaTeX*

Peter Wilson
Herries Press†
Maieul Rouquette‡
based on the original work by
John Lavagnino, Dominik Wujastyk, Herbert Breger and Wayne Sullivan

Abstract
EDMAC, a set of Plain TeX macros, was made at the beginning of 90’s for
typesetting critical editions in the traditional way, i.e., similar to the Oxford
Classical Texts, Teubner, Arden Shakespeare and other series. A separate
set of Plain TeX macros, TABMAC, provides for tabular material. Another
set of Plain TeX macros, EDSTANZA, assists in typesetting verse.

The eldedmac package makes the EDMAC, TABMAC and EDSTANZA facilities
available to authors who would prefer to use \LaTeX. The principal functions
provided by the package are marginal line numbering and multiple series of
foot- and endnotes keyed to line numbers.

In addition to the EDMAC, TABMAC and EDSTANZA functions the package also
provides for index entries keyed to both page and line numbers. Multiple
series of the familiar numbered footnotes are also available.

Other \LaTeX packages for critical editions include EDNOTES, and poemscol
for poetical works.

eledmac provides many tools and options. Normally, they are all docu-
mented in this file. Also provided is a help folder, examples. The folder
contains additional examples (although not for all cases).

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 report bugs in English or in French
(better).

You can subscribe to the eldedmac mail list in:
http://geekographie.maieul.net/146

*This file (eledmac.dtx) has version number v1.18.0, last revised 2015/02/23.
†herries dot press at earthlink dot net
‡maieul at maieul dot net
## Contents

1 Introduction  
   1.1 Overview .................................................. 6  
   1.2 History .................................................... 8  
      1.2.1 EDMAC .................................................... 8  
      1.2.2 eledmac ............................................. 9  
      1.2.3 List of works edited with (e)ledmac ............... 9  

2 The eledmac package  

3 Text lines and paragraphs numbering  
   3.1 Text lines numbering ........................................ 10  
   3.2 Paragraphs ................................................ 11  
      3.2.1 Basis .................................................. 11  
      3.2.2 Content before specific \pstart and after \pend .... 11  
      3.2.3 Content before every \pstart and after every \pend .. 11  
      3.2.4 Producing automatically \pstart...\pend ............ 12  
      3.2.5 Numbering paragraphs (\pstart) ....................... 12  
      3.2.6 Languages written in Right to Left .................. 12  
      3.2.7 Memory limits ....................................... 12  
   3.3 Lineation commands .......................................... 13  
      3.3.1 Disabling lineation .................................... 13  
      3.3.2 Setting lineation start and step ..................... 13  
      3.3.3 Setting lineation reset ................................ 14  
      3.3.4 Setting line number margin ........................... 14  
      3.3.5 Other settings ....................................... 14  
   3.4 Changing the line numbers ................................... 15  

4 The apparatus  
   4.1 Commands .................................................. 16  
   4.2 Disambiguation of identical words in the apparatus ........ 18  
   4.3 Alternate footnote formatting ................................ 19  
   4.4 Display options ............................................ 20  
      4.4.1 Control line number printing ......................... 20  
      4.4.2 Separator between the lemma and the note content .... 21  
      4.4.3 Font style ........................................... 22  
      4.4.4 Font of the lemma .................................... 22  
      4.4.5 Styles of notes content ............................... 22  
      4.4.6 Arbitrary code at the begininging of notes .......... 23  
      4.4.7 Options for notes in columns .......................... 23  
      4.4.8 Options for paragraphed footnotes ..................... 23  
      4.4.9 Options for block of notes ............................ 24  
   4.5 Page layout ............................................... 25  
   4.6 Fonts ..................................................... 25  
   4.7 Create a new series ....................................... 26
20 Sectioning commands

21 Line counting
  21.1 Choosing the system of lineation ......................... 56
  21.2 List macros ........................................... 60
  21.3 Line-number counters and lists .......................... 61
  21.4 Reading the line-list file ............................. 65
  21.5 Commands within the line-list file ..................... 67
  21.6 Writing to the line-list file ........................... 73

22 Marking text for notes
  22.1 \edtext (and \critext) itself ............................ 78
  22.2 Substitute lemma ...................................... 83
  22.3 Substitute line numbers ................................ 84
  22.4 Lemma disambiguation .................................. 84

23 Paragraph decomposition and reassembly
  23.1 Boxes, counters, \pstart and \pend ....................... 88
  23.2 Processing one line ................................... 92
  23.3 Line and page number computation ....................... 94
  23.4 Line number printing .................................. 97
  23.5 Pstart number printing in side ......................... 101
  23.6 Add insertions to the vertical list .................... 102
  23.7 Penalties ............................................ 103
  23.8 Printing leftover notes ............................... 104

24 Critical footnotes
  24.1 Fonts .................................................. 105
  24.2 Outer-level footnote commands ......................... 105
  24.3 Normal footnote formatting ............................ 107
  24.4 Standard footnote definitions ........................ 113
  24.5 Paragraphed footnotes ................................ 115
  24.6 Insertion of the footnotes separator .................. 120
  24.7 Columnar footnotes ................................... 121

25 Familiar footnotes
  25.1 Generality ............................................ 126
  25.2 Footnote formats ...................................... 127
  25.3 Two columns footnotes ................................ 131
  25.4 Three columns footnotes .............................. 133
  25.5 Paragraphed footnotes ................................ 134

26 Footnotes’ width for two columns .......................... 137

27 Footnotes’ order .......................................... 138

28 Footnotes’ rule ........................................... 139
29 Footnotes’ output

30 Endnotes

31 Generate series
   31.1 Test if series is still existing
   31.2 Create all commands to memorize display options
   31.3 Create inserts, needed to add notes in foot
   31.4 Create commands for critical apparatus, \footnote
   31.5 Create tools for familiar footnotes (\footnoteX)
   31.6 The endnotes
   31.7 Init standards series (A,B,C,D,E,Z)
   31.8 Some tools
   31.9 Old commands, kept for backward compatibility
   31.10 Hooks for a particular footnote
   31.11 Alias
   31.12 Line number printing

32 Output routine

33 Cross referencing

34 Side notes

35 Minipages and such

36 Indexing
   36.1 Memoir compatibility
   36.2 Normal setting
   36.3 Choose the right variant
   36.4 Hyperref compatibility

37 Macro as environment

38 Verse

39 Arrays and tables

40 Section’s title commands
   40.1 Deprecated commands
   40.2 New commands: \eledxxx

41 Page breaking or no page breaking depending of specific lines

42 Long verse: prevents being separated by a page break

43 The End
1 Introduction

The EDMAC macros \cite{LW90} for typesetting critical editions of texts have been available for use with TeX since 90’s. Since EDMAC was introduced there has been a small but constant demand for a version of EDMAC that could be used with LaTeX. The eledmac package is an attempt to satisfy that request.

eledmac would not have been possible without the amazing work by John Lavagnino and Dominik Wujastyk, the original authors of EDMAC. I, Peter Wilson, am very grateful for their encouragement and permission to use EDMAC as a base. The majority of both the code and this manual are by these two. The tabular material is based on the TABMAC code \cite{Bre96}, by permission of its author, Herbert Breger. The verse-related code is by courtesy of Wayne Sullivan, the author of EDSTANZA \cite{Sul92}, who has kindly supplied more than his original macros.

Since 2011’s Maïeul Rouquette begun to maintain and extend eledmac. As plain TeX is used by little people, and LaTeX by more people eledmac and original EDMAC are more and more distant.

1.1 Overview

The eledmac package, together with LaTeX, provides several important facilities for formatting critical editions of texts in a traditional manner. Major features include:

- automatic stepped line numbering, by page or by section;
- sub-lineation within the main series of line numbers;
- variant readings automatically keyed to line numbers;
- caters for both prose and verse;
- multiple series of the footnotes and endnotes;
- block or columnar formatting of the footnotes;
- simple tabular material may be line numbered;
- indexing keyed to page and line numbers.


1.1 Overview

eledmac allows the scholar engaged in preparing a critical edition to focus attention wholly on the task of creating the critical text and evaluating the variant readings, text-critical notes and testimonia. \LaTeX{} and eledmac will take care of the formatting and visual correlation of all the disparate types of information.

The original EDMAC can be used as a ‘stand alone’ processor or as part of a process. One example is its use as the formatting engine or ‘back end’ for the output of an automatic manuscript collation program. COLLATE, written by Peter Robinson, runs on the Apple Macintosh, can collate simultaneously up to a hundred manuscripts of any length, and provides facilities for the scholar to tailor the collation interactively. For further details of this and other related work, visit the EDMAC home page at http://www.homepages.ucl.ac.uk/~ucgadkw/edmac/index.html.

Apart from eledmac there are some other \LaTeX{} packages for critical edition typesetting. As Peter Wilson is not an author, or even a prospective one, of any critical edition work he could not provide any opinions on what authors in this area might feel comfortable with or how well any of the packages meet their needs.

EDNOTES [Luc03], by Uwe Lück and Christian Tapp, is another \LaTeX{} package being developed for critical editions. Unlike eledmac which is based on EDMAC, EDNOTES takes a different (internal) approach and provides a different set of features. For example it provides additional facilities for overlapping lemmas and for handling tables. For more information there is a web site at http://ednotes.sty.de.vu or email to ednotes.sty@web.de.

The poemscol package [Bur01] by John Burt is designed for typesetting critical editions of collections of poems. I do not know how, or whether, poemscol and eledmac will work together.

Critical authors may find it useful to look at EDMAC, EDNOTES, eledmac, and poemscol to see which best meets their needs.

At the time of writing Peter Wilson knows of two web sites, apart from the EDMAC home page, that have information on eledmac, and other programs.

• Jerónimo Leal pointed me to http://www.guit.sssup.it/latex/critical.html. This also mentions another package for critical editions called MauroTeX (http://www.maurolico.unipi.it/mtex/mtex.htm). These sites are both in Italian.

• Dirk-Jan Dekker maintains http://www.djdekker.net/ledmac which is a FAQ for typesetting critical editions and eledmac.

This manual contains a general description of how to use the \LaTeX{} version of EDMAC, namely eledmac (in sections 2 through Appendix A.1); the complete source code for the package, with extensive documentation (in sections 18 and following); and an Index to the source code. We do not suggest that you need to read the source code for this package in order to use it; we provide this code primarily for reference, and many of our comments on it repeat material that is also found in the earlier sections. But no documentation, however thorough, can cover every question that comes up, and many can be answered quickly by consultation of the code. On a first reading, we suggest that you should read only the general
1 Introduction

1.2 History

1.2.1 EDMAC

The original version of EDMAC was TEXTED.TEX, written by John Lavagnino in late 1987 and early 1988 for formatting critical editions of English plays.

John passed these macros on to Dominik Wujastyk who, in September–October 1988, added the footnote paragraphing mechanism, margin swapping and other changes to suit his own purposes, making the style more like that traditionally used for classical texts in Latin and Greek (e.g., the Oxford Classical Texts series). He also wrote some extra documentation and sent the files out to several people. This version of the macros was the first to be called EDMAC.

The present version was developed in the summer of 1990, with the intent of adding necessary features, streamlining and documenting the code, and further generalizing it to make it easily adaptable to the needs of editors in different disciplines. John did most of the general reworking and documentation, with the financial assistance of the Division of the Humanities and Social Sciences, California Institute of Technology. Dominik adapted the code to the conventions of Frank Mittelbach’s doc option, and added some documentation, multiple-column footnotes, cross-references, and crop marks. A description by John and Dominik of this version of EDMAC was published as ‘An overview of EDMAC: a Plain TEX format for critical editions’, TUGboat 11 (1990), pp. 623–643.

From 1991 through 1994, the macros continued to evolve, and were tested at a number of sites. We are very grateful to all the members of the (now defunct) edmac@mailbase.ac.uk discussion group who helped us with smoothing out bugs and infelicities in the macros. Ron Whitney and our anonymous reviewer at the TUG were both of great help in ironing out last-minute wrinkles, while Ron made some important suggestions which may help to make future versions of EDMAC even more efficient. Wayne Sullivan, in particular, provided several important fixes and contributions, including adapting the Mittelbach/Schöpf ‘New Font Selection Scheme’ for use with Plain TEX and EDMAC. Another project Wayne has worked on is a DVI post-processor which works with an EDMAC that has been slightly modified to output specials. This combination enables you to recover to some extent the text of each line, as ASCII code, facilitating the creation of concordances, an index verborum, etc.

At the time of writing (1994), we are pleased to be able to say that EDMAC is being used for real-life book production of several interesting editions, such as the Latin texts of Euclid’s Elements, an edition of the letters of Nicolaus Coperni-

---

1 This version of the macros was used to format the Sanskrit text in volume I of Metarules of Panini Grammar by Dominik Wujastyk (Groningen: Forsten, 1993).

1.2 History

Simon Bredon’s *Arithmetica*, a Latin translation by Plato of Tivoli of an Arabic astrolabe text, a Latin translation of part II of the Arabic *Algebra* by Abū Kāmil Shu‘jīb b. Aslam, the Latin *Rithmachia* of Werinher von Tegernsee, a middle-Dutch romance epic on the Crusades, a seventeenth-century Hungarian politico-philosophical tract, an anonymous Latin compilation from Hungary entitled *Sermones Complati in Studio Generali Quinqueeclesiensi in Regno Ungarie*, the collected letters and papers of Leibniz, Theodosius’s *Spheres*, the German *Algorismus* of Sacrobosco, the Sanskrit text of the *Kāśikāvyātti* of Vāmana and Jayāditya, and the English texts of Thomas Middleton’s collected works.

1.2.2 eledmac

Version 1.0 of *TABMAC* was released by Herbert Breger in October 1996. This added the capability for typesetting tabular material.

Version 0.01 of *EDSTANZA* was released by Wayne Sullivan in June 1992, to help a colleague with typesetting Irish verse.

In March 2003 Peter Wilson started an attempt to port *EDMAC* from TeX to LaTeX. The starting point was *EDMAC* version 3.16 as documented on 19 July 1994 (available from CTAN). In August 2003 the *TABMAC* functions were added; the starting point for these being version 1.0 of October 1996. The *EDSTANZA* (v0.01) functions were added in February 2004. Sidenotes and regular footnotes in numbered text were added in April 2004.

This port was called *ledmac*.

Since July 2011, ledmac is maintained by Maïeul Rouquette.

Important changes were put in version 1.0, to make eledmac more easily extensible (see 4.4 p. 20). These changes can trigger small problems with the old customization. That is why a new name was selected: *eledmac*. To migrate from ledmac to eledmac, please read Appendix Appendix A.2 (p. 224).

1.2.3 List of works edited with (e)ledmac

A collaborative list of works edited with (e)ledmac is available on [https://www.zotero.org/groups/criticalEditions_typeset_with_edmac_ledmac_and_eledmac/](https://www.zotero.org/groups/criticalEditions_typeset_with_edmac_ledmac_and_eledmac/)

---

3 Being prepared at the German Copernicus Research Institute, Munich.
4 Being prepared by Menso Folkerts et al., at the Institut für Geschichte der Naturwissenschaften in Munich.
5 Richard Lorch, Gerhard Brey et al., at the same Institute.
10 Being produced, as was the previous book, by Gyula Mayer in Budapest.
11 Leibniz, *Sämtliche Schriften und Briefe*, series I, III, VII, being edited by Dr. H. Breger, Dr. N. Gädeke and others, at the Leibniz-Archiv, Niedersächsische Landesbibliothek, Hannover. (see [http://www.nlb-hannover.de/leibniz](http://www.nlb-hannover.de/leibniz)).
12 Being prepared at Poona and Lausanne Universities.
items: Please add your own edition made with (c)ledmac.

2 The eledmac package

eledmac is a three-pass package like \TeX itself. Although your textual apparatus and line numbers will be printed even on the first run, it takes two more passes through \TeX to be sure that everything gets to its right place. Any changes you make to the input file may similarly require three passes to get everything to the right place, if the changes alter the number of lines or notes. eledmac will tell you that you need to make more runs, when it notices, but it does not expend the labor to check this thoroughly. If you have problems with a line or two misnumbered at the top of a page, try running \TeX once or twice more.

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 eledmac’s note commands with it: this is appropriate for introductions and other material added by the editor around the edited text.

3 Text lines and paragraphs numbering

3.1 Text lines numbering

\beginnumbering
\endnumbering

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

\beginnumbering
⟨text⟩
\endnumbering

The \beginnumbering macro resets the line number to zero, reads an auxiliary file called ⟨jobname⟩.nn (where ⟨jobname⟩ is the name of the main input file for this job, and nn is 1 for the first numbered section, 2 for the second section, and so on), and then creates a new version of this auxiliary file to collect information during this run. The first instance of \beginnumbering also opens a file called ⟨jobname⟩.end to receive the text of the endnotes. \endnumbering closes the ⟨jobname⟩.nn file.

If the line numbering of a text is to be continuous from start to end, then the whole text will be typed between one pair of \beginnumbering and \endnumbering commands. But your text will most often contain chapter or other divisions marking sections that should be independently numbered, and these will be appropriate places to begin new numbered sections. eledmac has to read and store in memory a certain amount of information about the entire section when it encounters a \beginnumbering command, so it speeds up the processing and reduces memory use when a text is divided into a larger number of sections (at the expense of multiplying the number of external files that are generated).
3.2 Paragraphs

3.2.1 Basis

Within a numbered section, each paragraph of numbered text must be marked using the \pstart and \pend commands:

\pstart
(paragraph of text)
\pend

Text that appears within a numbered section but isn’t marked with \pstart and \pend will not be numbered.

The following example shows the proper section and paragraph markup, and the kind of output that would typically be generated:

\beginnumbering
\pstart
This is a sample paragraph, with lines numbered automatically.
\pend

\pstart
This paragraph too has its lines automatically numbered.
\pend

\pstart
The lines of this paragraph are not numbered.
\pend

\pstart
And here the numbering begins again.
\pend
\endnumbering

1 This is a sample paragraph
2 with lines numbered
3 automatically.
4 This paragraph too
5 has its lines automatically
6 numbered.

7 And here the numbering
8 begins again.

3.2.2 Content before specific \pstart and after \pend

Both \pstart and \pend can take an optional argument, in brackets. Its content will be printed before the beginning of \pstart / after the end of \pend instead of the argument of \AtEveryPstart / \AtEveryPend. If you need to start a \pstart by brackets, or to add brackets after a \pend, just add a \relax between \pstart/\pend and the brackets.

This feature is not needed for normal use of eledmac, but it is needed when using verse (see [5] p. [26]) or eledpar (see [17.3] p. [43]).

A \noindent is automatically added before this argument.

3.2.3 Content before every \pstart and after every \pend

\AtEveryPstart \AtEveryPend You can use both \AtEveryPstart and \AtEveryPend. Their arguments will be printed before every \pstart begins / after every \pend ends.
3.2.4 Producting automatically \pstart...\pend

\autopar

You can use \autopar to avoid the nuisance of this paragraph markup and still have every paragraph automatically numbered. The scope of the \autopar command needs to be limited by keeping it within a group, as follows:

\begingroup
  \beginnumbering
  \autopar

  A paragraph of numbered text.
  
  Another paragraph of numbered text.

\endnumbering
\endgroup

\autopar fails, however, on paragraphs that start with a \{ or with any other command that starts a new group before it generates any text. Such paragraphs need to be started explicitly, before the new group is opened, using \indent, \nindent, or \leavevmode, or using \pstart itself.\footnote{\label{footnote:pstart}

3.2.5 Numbering paragraphs (\pstart)

It is possible to insert a number at every \pstart command. You must use the \numberpstarttrue command to have it. You can stop the numbering with \numberpstartfalse. You can redefine the command \thepstart to change style. You can change the value of the pstart number by using after \beginnumbering:

\setcounter{numberpstart}{value}

On each \beginnumbering the numbering restarts.

With the \sidepstartnumtrue command, the number of \pstart will be printed inside. In this case, the line number will be not printed.

With the \labelpstarttrue command, a \label added just after a \pstart will refer to the number of this pstart.

3.2.6 Languages written in Right to Left

If you use languages written in Right to Left, we Lua\LaTeX{} or Xe\LaTeX{}, you must switch text direction \before the \pstart command.

3.2.7 Memory limits

This paragraph is kept for history, but problem described below should not appear with eledmac. eledmac stores a lot of information about line num-
bers and footnotes in memory as it goes through a numbered section. But at the
eend of such a section, it empties its memory out, so to speak. If your text has a
very long numbered section it is possible that your \TeX may reach its memory
limit. There are two solutions to this. The first is to get a larger \TeX with
increased memory. The second solution is to split your long section into several
smaller ones. The trouble with this is that your line numbering will start again at
zero with each new section. To avoid this problem, we provide \pausenumbering
and \resumenumbering which are just like \endnumbering ... \beginnumbering,
except that they arrange for your line numbering to continue across the break.
Use \pausenumbering only between numbered paragraphs:
\beginnumbering
\pstart
Paragraph of text.
\pstart
Another paragraph.
\endnumbering

We have defined these commands as two macros, in case you find it necessary
to insert text between numbered sections without disturbing the line numbering.
But if you are really just using these macros to save memory, you might as well
say
\newcommand{\memorybreak}{\pausenumbering\resumenumbering}

and say \memorybreak between the relevant \pend and \pstart.

3.3 Lineation commands

3.3.1 Disabling lineation

\numberlinefalse
\numberlinetrue

Line numbering can be disabled with \numberlinefalse. It can be enabled again
with \numberlinetrue.

3.3.2 Setting lineation start and step

By default, \edmac numbers every 5th line. There are two counters, \firstlinenum
and \linenumincrement, that control this behaviour; they can be changed using
\firstlinenum{(num)} and \linenumincrement{(num)}. \firstlinenum specifies the first line that will have a printed number, and \linenumincrement is the

\footnote{For a detailed study of the reasons for this restriction, see Barbara Beeton, ‘Initiation rites’,
\textit{TUGboat} \textbf{12} (1991), pp. 257–258.}
difference between successive numbered lines. For example, to start printing numbers at the first line and to have every other line numbered:
\firstlinenum{1} \linenumincrement{2}
There are similar commands, \firstsublinenum{(num)} and \sublinenumincrement{(num)} for controlling sub-line numbering. You can define \linenumberlist to specify a non-uniform distribution of printed line numbers. For example:
\def\linenumberlist{1,2,3,5,7,11,13,17,19,23,29}
to have numbers printed on prime-numbered lines only. There must be no spaces within the definition which consists of comma-separated decimal numbers. The numbers can be in any order but it is easier to read if you put them in numerical order. Either omitting the definition of \linenumberlist or following the vacuous definition
\def\linenumberlist{}
the standard numbering sequence is applied. The standard sequence is that specified by the combination of the firstlinenum, linenumincrement, firstsublinenum and linenumincrement counter values.

3.3.3 Setting lineation reset

\lineation Lines can be numbered either by page, by pstart or by section; you specify this using the \lineation{(arg)} macro, where ⟨arg⟩ is either page, pstart or section. You may only use this command at places where numbering is not in effect; you can’t change the lineation system within a section. You can change it between sections: they don’t all have to use the same lineation system. The package’s standard setting is \lineation{section}. If the lineation is by pstart, the pstart number will be printed before the line number in the notes.

3.3.4 Setting line number margin

\linenummargin The command \linenummargin{location} specifies the margin where the line (or pstart) numbers will be printed. The permissible value for ⟨location⟩ is one out of the list left, right, inner, or outer, for example \linenummargin{inner}. The package’s default setting is \linenummargin{left} to typeset the numbers in the left hand margin. You can change this whenever you’re not in the middle of making a paragraph.

More precisely, the value of \linenummargin used is that in effect at the \pend of a numbered paragraph. Apart from an initial setting for \linenummargin, only change it after a \pend, whereupon it will apply to all following numbered paragraphs, until changed again (changing it between a \pstart and \pend pair will apply the change to all the current paragraph).

3.3.5 Other settings

\leftlinenum When a marginal line number is to be printed, there are a lot of ways to display it. You can redefine \leftlinenum and \rightlinenum to change the way marginal
line numbers are printed in the left and right margins respectively; the initial versions print the number in font \numlabfont (described below) at a distance \linenumsep (initially set to one pica) from the text.

### 3.4 Changing the line numbers

Normally the line numbering starts at 1 for the first line of a section and steps up by one for each line thereafter. There are various common modifications of this system, however; the commands described here allow you to put such modifications into effect.

**You insert the \startsub and \endsub commands in your text to turn sub-lineation on and off. In plays, for example, stage directions are often numbered with sub-line numbers: as line 10.1, 10.2, 10.3, rather than as 11, 12, and 13. Titles and headings are sometimes numbered with sub-line numbers as well.**

When sub-lineation is in effect, the line number counter is frozen and the sub-line counter advances instead. If one of these commands appears in the middle of a line, it doesn’t take effect until the next line; in other words, a line is counted as a line or sub-line depending on what it started out as, even if that changes in the middle.

**The \startlock command, used in running text, locks the line number at its current value, until you say \endlock. It can tell for itself whether you are in a patch of line or sub-line numbering. One use for line-number locking is in printing poetry: there the line numbers should be those of verse lines rather than of printed lines, even when a verse line requires several printed lines.**

When line-number locking is used, several printed lines may have the same line number, and you have to specify whether you want the number attached to the first printed line or the last, or whether you just want the number printed by them all. (This assumes that, on the basis of the settings of the previous parameters, it is necessary to display a line number for this line.) You specify your preference using \lockdisp{⟨arg⟩}; its argument is a word, either first, last, or all. The package initially sets this as \lockdisp{first}.

**In some cases you may want to modify the line numbers that are automatically calculated: if you are printing only fragments of a work but want to print line numbers appropriate to a complete version, for example.** The \setline{⟨num⟩} and \advanceline{⟨num⟩} commands may be used to change the current line’s number (or the sub-line number, if sub-lineation is currently on). They change both the marginal line numbers and the line numbers passed to the notes. \setline takes one argument, the value to which you want the line number set; it must be 0 or greater. \advanceline takes one argument, an amount that should be added to the current line number; it may be positive or negative.

**The \setline and \advanceline macros should only be used within a \pstart...\pend group.** The \setlinenum{⟨num⟩} command can be used outside such a group, for example between a pend and a \pstart. It sets the line number to ⟨num⟩. It has no effect if used within a \pstart...\pend group

**Line numbers are normally printed as arabic numbers. You can use \linenumberstyle{⟨style⟩} \sublinenumberstyle to change the numbering style. ⟨style⟩ must be one of:**
Alph Uppercase letters (A...Z).

alph Lowercase letters (a...z).

arabic Arabic numerals (1, 2, ...)

Roman Uppercase Roman numerals (I, II, ...)

roman Lowercase Roman numerals (i, ii, ...)

Note that with the Alph or alph styles, ‘numbers’ must be between 1 and 26 inclusive.

Similarly \sublinenumberstyle{⟨style⟩} can be used to change the numbering style of sub-line numbers, which is normally arabic numerals.

When inserted into a numbered line the macro \skipnumbering causes the numbering of that particular line to be skipped; that is, the line number is unchanged and no line number will be printed.

4 The apparatus

4.1 Commands

\edtext Within numbered paragraphs, all footnotes and endnotes are generated by the \edtext macro:

\edtext{⟨lemma⟩}{⟨commands⟩}

The ⟨lemma⟩ argument is the lemma in the main text: \edtext both prints this as part of the text, and makes it available to the ⟨commands⟩ you specify to generate notes.

For example:
I saw my friend \edtext{Smith}{\Afootnote{Jones C, D.}} on Tuesday.

The lemma Smith is printed as part of this sentence in the text, and is also made available to the footnote that specifies a variant, Jones C, D. The footnote macro is supplied with the line number at which the lemma appears in the main text.

The ⟨lemma⟩ may contain further \edtext commands. Nesting makes it possible to print an explanatory note on a long passage together with notes on variants for individual words within the passage. For example:
\edtext{I saw my friend \edtext{Smith}{\Afootnote{Jones C, D.}} on Tuesday.}{\Bfootnote{The date was July 16, 1954.}}

1 I saw my friend
2 Smith on Tuesday.
3 Smith]Jones C, D.
1–2 I saw my friend Smith on Tuesday.] The date was July 16, 1954.
However, \texttt{edtext} cannot handle overlapping but unnested notes—for example, one note covering lines 10–15, and another covering 12–18; a \texttt{edtext} that starts in the \texttt{⟨lemma⟩} argument of another \texttt{edtext} must end there, too. (The \texttt{lemma} and \texttt{linenum} commands may be used to generate overlapping notes if necessary.)

**Commands used in \texttt{edtext}'s second argument**  The second argument of the \texttt{edtext} macro, \texttt{(commands)}, may contain a series of subsidiary commands that generate various kinds of notes.

\texttt{\Afootnote{⟨text⟩}} \hspace{2cm} \texttt{\Bfootnote{⟨text⟩}} \hspace{2cm} \texttt{\Cfootnote{⟨text⟩}} \hspace{2cm} \texttt{\Dfootnote{⟨text⟩}} \hspace{2cm} \texttt{\Efootnote{⟨text⟩}}

Five separate series of the footnotes are maintained; each macro taking one argument like \texttt{\Afootnote{⟨text⟩}}. When all five are used, the \texttt{A} notes appear in a layer just below the main text, followed by the rest in turn, down to the \texttt{E} notes at the bottom. These are the main macros that you will use to construct the critical apparatus of your text. The package provides five layers of notes in the belief that this will be adequate for the most demanding editions. But it is not hard to add further layers of notes should they be required.

An optional argument can be added before the text of the footnote. Its value is a comma separated list of options. The available options are:

- \texttt{nonum} to disable line numbering for this note.
- \texttt{nosep} to disable the lemma separator for this note.

Example: \texttt{\Afootnote{nonum}\{⟨text⟩\}}.

\texttt{\Aendnote{⟨text⟩}} \hspace{2cm} \texttt{\Bendnote{⟨text⟩}} \hspace{2cm} \texttt{\Cendnote{⟨text⟩}} \hspace{2cm} \texttt{\Dendnote{⟨text⟩}} \hspace{2cm} \texttt{\Eendnote{⟨text⟩}}

The package also maintains five separate series of endnotes. Like footnotes each macro takes a single argument like \texttt{\Aendnote{⟨text⟩}}. Normally, none of them are printed: you must use the \texttt{doendnotes} macro described below (p.31) to call for their output at the appropriate point in your document.

By default, no paragraph can be made in the notes of critical apparatus. You can allow it by adding the options \texttt{parapparatus} when loading the package:

\texttt{\usepackage[parapparatus]{eledmac}}

\texttt{\lemma{⟨alternative⟩}}

If you want to change the lemma that gets passed to the notes, you can do this by using \texttt{\lemma{⟨alternative⟩}} within the second argument to \texttt{edtext}, before the note commands. The most common use of this command is to abbreviate the lemma that’s printed in the notes. For example:

\texttt{\edtext{I saw my friend}}

\texttt{\edtext{Smith}\{\Afootnote{Jones C, D.} on Tuesday.\}}

\texttt{\{\lemma{I \dota Tuesday.}\}}

\texttt{\Bfootnote{The date was July 16, 1954.}}

\texttt{1 I saw my friend}

\texttt{2 Smith on Tuesday.}

\texttt{2 Smith\[Jones C, D.\]}

\texttt{1–2 I . . . Tuesday.]

\texttt{The date was July 16, 1954.}

\texttt{\linenum{⟨arg⟩}}

You can use \texttt{\linenum{⟨arg⟩}} to change the line numbers passed to the notes. The notes are actually given seven parameters: the page, line, and sub-line number for the start of the lemma; the same three numbers for the end of the lemma;
and the font specifier for the lemma. As the argument to \linenum, you specify
those seven parameters in that order, separated by vertical bars (the | character).
However, you can retain the value computed by eledmac for any number by sim-
ply omitting it; and you can omit a sequence of vertical bars at the end of the
argument. For example, \linenum{|||23} changes one number, the ending page
number of the current lemma.

This command doesn’t change the marginal line numbers in any way; it just
changes the numbers passed to the footnotes. Its use comes in situations that
\edtext{} has trouble dealing with for whatever reason. If you need notes for
overlapping passages that aren’t nested, for instance, you can use \lemma and
\linenum to generate such notes despite the limitations of \edtext. If the
\langle lemma \rangle argument to \edtext is extremely long, you may run out of memory;
here again you can specify a note with an abbreviated lemma using \lemma and
\linenum. The numbers used in \linenum need not be entered manually; you can
use the ‘x-’ symbolic cross-referencing commands below (p. 31) to compute them
automatically.

Similarly, being able to manually change the lemma’s font specifier in the notes
might be important if you were using multiple scripts or languages. The form of
the font specifier is three separate codes separated by / characters, giving the
family, series, and shape codes as defined within NFSS.

Changing the names of these commands The commands for generating the
apparatus have been given rather bland names, because editors in different fields
have widely divergent notions of what sort of notes are required, where they should
be printed, and what they should be called. But this doesn’t mean you have to
type \Afootnote when you’d rather say something you find more meaningful, like
\variant. We recommend that you create a series of such aliases and use them
instead of the names chosen here; all you have to do is put commands of this form
at the start of your file 14:

\newcommandx{\variant}[2][1,usedefault]{\Afootnote[#1]{#2}}
\newcommandx{\explanatory}[2][1,usedefault]{\Bfootnote[#1]{#2}}
\newcommand{\trivial}[1]{\Aendnote{#1}}
\newcommandx{\testimonia}[2][1,usedefault]{\Cfootnote[#1]{#2}}

4.2 Disambiguation of identical words in the apparatus

Sometimes, the same word occurs twice (or more time) in the same line. eledmac
provides tools to disambiguage references in the critical notes. The lemma will be
followed by a reference number if a given word occurs more that once in the same
line.

\sameword To use this tool, you have to mark every occurrence of the potentially ambigu-

\footnote{We use \newcommand and \newcommandx instead of classical \let command because the edtab-
ular environments have to modify the notes definition, and we need to use the newest definition
of notes. Read the handbook of xargs to know more about \newcommandx.}
ous term with the \sameword command:

\lupus \sameword{aut} canis \edtext{\sameword{aut}}{\Afootnote{et}} felix

In this example, aut will be followed, in the critical note, by the exponent 2 if it is printed in the same line as the first aut, but it won’t if it is printed in a different line. The number is printed only after the second run.

If you use the \lemma command, eledmac assumes that the word marked with \sameword is not already found in \lemma. However, if it is actually found in it, you must use this method:

• In the first argument of \edtext, use \sameword with the optional argument ‘[inlemma]’.
• In the content of \lemma, use \sameword with no optional argument.

Like this:

\edtext{\sameword{inlemma}{sw}}{\lemma{\sameword{sw} some lemma}\Afootnote{some note}}

You can redefine the \showwordrank macro to change the way the number is printed. The default value is

\newcommand{\showwordrank}{2}{% #1\textsuperscript{#2}%
}

### 4.3 Alternate footnote formatting

If you just launch into eledmac using the commands outlined above, you will get a standard layout for your text and notes. You may be happy to accept this at the very beginning, while you get the hang of things, but the standard layout is not particularly pretty, and you will certainly want to modify it in due course. The package provides ways of changing the fonts and layout of your text, but these are not aimed at being totally comprehensive. They are enough to deal with simple variations from the norm, and to exemplify how you might go on to make more significant changes.

By default, all footnotes are formatted as a series of separate paragraphs in one column. Three other formats are also available for notes, and using these macros you can select a different format for a series of notes.

• \footparagraph formats all the footnotes of a series as a single paragraph;
• \foottwocol formats them as separate paragraphs, but in two columns;
• \footthreecol, in three columns.
Each of these macros takes one argument: a letter (between A and E) for the series of notes you want changed. So a text with three layers of notes might begin thus:

\footnormal{A}\footthreecol{B}\footparagraph{C}

This would make the A-notes ordinary, B-notes would be in three columns, and the bottom layer of notes would be formed into a paragraph on each page.

4.4 Display options

Since version 1.0, some commands can be used to change the display of the footnotes. All can have an optional argument \%(s\), which is the letter of the series — or a list of letters separated by comma — depending on which option is applied.

When a length, noted \%(l\), is used, it can be stretchable: \(a + b - c\). The final length \(m\) is calculated by \LaTeX{} to have: \(a - c \leq m \leq a + b\). If you use relative unity\(^{15}\), it will be relative to fontsize of the footnote.

4.4.1 Control line number printing

\texttt{\textbackslash numberonlyfirstinline} By default, the line number is printed in every note. If you want to print it only the first time for a value (i.e., one time for line 1, one time for line 2 etc.), you can use \texttt{\textbackslash numberonlyfirstinline\%(s\)}. Use \texttt{\textbackslash numberonlyfirstinline\%(s\)[false]} to cancel it (\%(s\) can be empty if you want to disable it for every series).

\texttt{\textbackslash numberonlyfirstintwolines} Suppose you have a lemma on line 2 and a lemma between line 2 and line 3. With \texttt{\textbackslash numberonlyfirstinline}, the second lemma is considered to be on the same line as the first lemma. But if you use both \texttt{\textbackslash numberonlyfirstinline\%(s\)} and \texttt{\textbackslash numberonlyfirstintwolines\%(s\)}, the distinction is made. Use \texttt{\textbackslash numberonlyfirstintwolines\%(s\)[false]} to cancel it (\%(s\) can be empty if you want to disable it for every series).

\texttt{\textbackslash symlinenum} For setting a particular symbol in place of the line number, you can use \texttt{\textbackslash symlinenum\%(s\)[\{symbol\}]} in combination with \texttt{\textbackslash numberonlyfirstinline\%(s\)}. From the second lemma of the same line, the symbol will be used instead of line number.

\texttt{\textbackslash nonumberinfootnote} You can use \texttt{\textbackslash nonumberinfootnote\%(s\)} if you don’t want to have the line number in a footnote. To cancel it, use \texttt{\textbackslash nonumberinfootnote\%(s\)[false]}.

\texttt{\textbackslash pstartinfootnote} You can use \texttt{\textbackslash pstartinfootnote\%(s\)} if you want to print the pstart number in the footnote, before the line and subline number. Use \texttt{\textbackslash pstartinfootnote\%(s\)[false]} to cancel it (\%(s\) can be empty if you want to disable it for every series). Note that when you change the lineation system, the option is automatically switched:

\begin{itemize}
  \item If you use lineation by pstart, the option is enabled.
  \item If you use lineation by section or by page, the option is disabled.
\end{itemize}

\texttt{\textbackslash pstartinfootnoteeverytime} By default, the pstart number is printed only in the part of text where you

\(^{15}\)Like \emph{en} which is the width of a M.
have called `\numberpstarttrue`. We don’t know why you would like to print the pstart number in the notes and not in the main text. However, if you want to do it, you can call `\pstartinfootnoteeverytime[⟨s⟩]`. In this case, the pstart number will be printed every time in footnote.

In combination with `\pstartinfootnote`, you can use `\onlypstartinfootnote[⟨s⟩]` if you want to print only the pstart number in the footnote, and not the line and subline number. Use `\onlypstartinfootnote[⟨s⟩][false]` to cancel it (⟨s⟩ can be empty if you want to disable it for every series).

With `\beforenumberinfootnote[⟨s⟩]{⟨l⟩}`, you can add some space before the line number in a footnote. If the line number is not printed, the space is not either. The default value is 0 pt.

With `\afternumberinfootnote[⟨s⟩]{⟨l⟩}` you can add some space after the line number in a footnote. If the line number is not printed, the space is not either. The default value is 0.5 em.

By default, the space defined by `\afternumberinfootnote` is breakable. With `\nonbreakableafternumber[⟨s⟩]` it becomes nonbreakable. Use `\nonbreakableafternumber[⟨s⟩][false]` to cancel it (⟨s⟩ can be empty if you want to disable it for every series).

With `\beforesymlinenum[⟨s⟩]{⟨l⟩}` you can add some space before the line symbol in a footnote. The default value is set by `\beforenumberinfootnote`. The default value is value set by `\afternumberinfootnote`. If no number or symbolic line number is printed, you can add a space, with `\inplaceofnumber[⟨s⟩]{⟨l⟩}`. The default value is 1 em.

It could be useful to put the line number inside a fixed box: the content of the note will be printed after this box. You can use `\boxlinenum[⟨s⟩]{⟨l⟩}` to do that. To subsequently disable this feature, use `\boxlinenum` with length equal to 0 pt. One use of this feature is to print line number in a column, and the note in another column:

```
\Xhangindent{1em}
\afternumberinfootnote{0em}
\boxlinenum{1em}
```

`\boxsymlinenum[⟨s⟩]{⟨l⟩}` is the same as `\boxlinenum` but for the line number symbol.

### 4.4.2 Separator between the lemma and the note content

`\lemmaseparator` By default, in a footnote, the separator between the lemma and thenote is a right bracket (\rbracket). You can use `\lemmaseparator[⟨s⟩]{⟨lemmaseparator⟩}` to change it. The optional argument can be used to specify in which series it is applied. Note that there is a non-breakable space between lemma and separator, but breakable space between separator and lemma.

`\beforelemmaseparator` Using `\beforelemmaseparator[⟨s⟩]{⟨l⟩}` you can add some space between lemma and separator. If your lemma separator is empty, this space won’t be printed. The default value is 0 em.
The apparatus

Using \afterlemmaseparator\{s\}\{l\} you can add some space between separator and note. If your lemma separator is empty, this space won’t be printed. The default value is 0.5 em.

You can suppress the lemma separator, using \nolemmaseparator\{s\}, which is simply a alias of \lemmaseparator\{s\}\{}.

With \inplaceoflemmaseparator\{s\}\{l\} you can add a space if no lemma separator is printed. The default value is 1 em.

4.4.3 Font style

\notenumfont\{s\}\{\langle\text{command}\rangle\} is used to change the font style for line numbers in critical footnotes; \textbf{\textlangle\text{command}\rangle} must be one (or more) switching command, like \textbf{\texttt{bfseries}}.

End notenumfont: \Xendnotenumfont\{s\}\{\langle\text{command}\rangle\} is used to change the font style for line numbers in critical footnotes. \textbf{\textlangle\text{command}\rangle} must be one (or more) switching command, like \textbf{\texttt{bfseries}}.

\notefontsize\{s\}\{\langle\text{command}\rangle\} is used to define the font size of critical footnotes of the series. The default value is \texttt{footnotesize}. The \textbf{\textlangle\text{command}\rangle} must not be a size in pt, but a standard \LaTeX\ size, like \texttt{small}.

Familiar notefontsize: \notefontsizeX\{s\}\{\langle\text{command}\rangle\} is used to define the font size of critical footnotes of the series. The default value is \texttt{footnotesize}. The \textbf{\textlangle\text{command}\rangle} must not be a size in pt, but a standard \LaTeX\ size, like \texttt{small}.

End notefontsize: \Xendnotefontsize\{s\}\{\langle\text{command}\rangle\} is used to define the font size of end critical footnotes of the series. The default value is \texttt{footnotesize}. The \textbf{\textlangle\text{command}\rangle} must not be a size in pt, but a standard \LaTeX\ size, like \texttt{small}.

4.4.4 Font of the lemma

\lemmadisablefontselection By default, font of the lemma in footnote is the same as font of the lemma in the main text. For example, if the lemma is in italic in the main text, it is also in italic in note. The \texttt{\\lemmadisablefontselection\{s\}\{} command allows to disable it for a specific series.

End lemmadisablefontselection: By default, font of the lemma in endnote is the same as font of the lemma in the main text. For example, if the lemma is in italic in the main text, it is also in italic in note. The command allows \texttt{\Xendlemmadisablefontselection\{s\}\{l\}} to disable it for a specific series.

4.4.5 Styles of notes content

\xhangindent For critical notes NOT paragraphed you can define an indent with \texttt{\xhangindent\{s\}\{l\}}, which will be applied in the second line of notes. It can help to make distinction between a new note and a break in a note. The default value is 0 pt.

\xhangindentX For familiar notes NOT paragraphed you can define an indent with \texttt{\xhangindent\{s\}\{l\}},
which will be applied in the second line of notes. It can help to make a distinction between a new note and a break in anote.

### 4.4.6 Arbitrary code at the beginning of notes

The three next commands add an arbitrary code at the beginning of notes. As the name’s space is local to the notes, you can use it to redefine some style inside the notes. For example, if you don’t want the pstart number to be in bold, use:

\hooknote{(series)}{(code)} is to be used at the beginning of the critical footnotes.

\hooknoteX{(series)}{(code)} is to be used at the beginning of the familiar footnotes.

\hookXendnote{(series)}{(code)} is to be used at the beginning of the endnotes.

### 4.4.7 Options for notes in columns

For the following four macros, be careful that the columns are made from right to left.

\hsizetwocol{(s)}{(l)} is used to change width of a column when critical notes are displaying in two columns. Default value is .45 \hsize.

\hsizethreecol{(s)}{(l)} is used to change width of a column when critical notes are displaying in three columns. Default value is .3 \hsize.

\hsizetwocolX{(s)}{(l)} is used to change width of a column when familiar notes are displaying in two columns. Default value is .45 \hsize.

\hsizethreecolX{(s)}{(l)} is used to change width of a column when familiar notes are displaying in three columns. Default value is .3 \hsize.

### 4.4.8 Options for paragraphed footnotes

You can add some space after a note by using \afternote{(s)}{(l)}. The default value is 1em plus.4em minus.4em.

For paragraphed footnotes (see below), you can choose the separator between each note by \parafootsep{(s)}{(l)}. A common separator is a double pipe ($||$), which you can set by \parafootsep{$||$}.

\ragged{(s)} Text in paragraphed critical notes is justified, but you can use \raggedL{(s)} if you want it to be ragged left, or \raggedR{(s)} if you want it to be ragged right.

\raggedX{(s)} Text in paragraphed footnotes is justified, but you can use \raggedXL{(s)} if you want it to be ragged left, or \raggedXR{(s)} if you want it to be ragged right.
4.4.9 Options for block of notes

`\txtbeforeXnotes` You can add some text before critical notes with `\txtbeforeXnotes{⟨text⟩}`.

`\beforeXnotes` You can change the vertical space printed before the rule of the critical notes with `\beforeXnotes{⟨s⟩}{⟨l⟩}`. The default value is `1.2em plus .6em minus .6em`.

Be careful, the standard \LaTeX\ footnote rule, which is used by `eledmac`, decreases by 3pt. This 3pt decrease is not changed by this command.

`\beforenotesX` You can change the vertical space printed before the rule of the familiar notes with `\beforenotesX{⟨s⟩}{⟨l⟩}`. The default value is `1.2em plus .6em minus .6em`.

Be careful, the standard \LaTeX\ footnote rule, which is used by `eledmac`, decreases 3pt. These 3pt are not changed by this command.

`\afterXrule` You can change the vertical space printed after the rule of the critical notes with `\afterXrule{⟨s⟩}{⟨l⟩}`. The default value is `0pt`.

Be careful, the standard \LaTeX\ footnote rule, which is used by `eledmac`, decreases 3pt. These 3pt are not changed by this command.

`\afterruleX` You can change the vertical space printed after the rule of the familiar notes with `\afterruleX{⟨s⟩}{⟨l⟩}`. The default value is `0pt`.

Be careful, the standard \LaTeX\ footnote rule, which is used by `eledmac`, adds 2.6pt. These 2.6pt are not changed by this command.

`\preXnotes` You can set the space before the first series of critical notes printed on each page and set a different amount of space for subsequent the series on the page. You can do it with `\preXnotes{⟨l⟩}`. Default value is `0pt`. You can disable this feature by setting the length to `0pt`.

Be careful with this setting: it can place notes by the page number, at the bottom of the page.

`\prenotesX` You can want the space before the first printed (in a page) series of familiar notes not to be the same as before other series. Default value is `0pt`. You can do it with `\prenotesX{⟨l⟩}`. You can disable this feature by setting the length to `0pt`.

Be careful with this setting: it could make the notes be written on the bottom pages number. By default, one series of critical notes can take 80% of the page size, before being broken to the next page. If you want to change the size use `\maxhXnotes{⟨s⟩}{⟨l⟩}`. Be careful: the length can’t be flexible, and is relative to the current font. For example, if you want the note to take, at most, 33 of the text height, do `\maxhXnotes{.33\textheight}{⟨l⟩}`.

`\maxhnotesX` `\maxhnotesX{⟨s⟩}{⟨l⟩}` is the same as previous, but for familiar footnotes.

Be careful with the two previous commands. Actually, for technical purposes, one paragraphed note is considered as one block. Consequently, it can’t be broken between two pages, even if you used these commands. The debug is in the todolist.
4.5 Page layout

You should set up the page layout parameters, and in particular the \baselineskip of the footnotes (this is done for you if you use the standard \notefontsetup), before you call any of these macros because their action depends on these; too much or too little space will be allotted for the notes on the page if these macros use the wrong values.\footnote{There is one tiny proviso about using paragraphed notes: you shouldn’t force any explicit line-breaks inside such notes: do not use \par, \break, or \penalty=-10000. If you must have a line-break for some obscure reason, just suggest the break very strongly: \penalty=-9999 will do the trick. Page 117 explains why this restriction is necessary.}

If you use {eledpar} columns macro, you can call:

- \noteswidthliketwocolumns\[(s)\] to create critical notes with a two-column size width. Use \noteswidthliketwocolumns\[(s)\][false] to disable it.
- \notesXwidthliketwocolumns\[(s)\] to create familiar notes with a two-column size width. Use \notesXwidthliketwocolumns\[(s)\][false] to disable it.

4.6 Fonts

One of the most important features of the appearance of the notes, and indeed of your whole document, will be the fonts used. We will first describe the commands that give you control over the use of fonts in the different structural elements of the document, especially within the notes, and then in subsequent sections specify how these commands are used.

For those who are setting up for a large job, here is a list of the complete set of {eledmac} macros relating to fonts that are intended for manipulation by the user: \endashchar, \fullstop, \numlabfont, and \rbracket.

Line numbers for the main text are usually printed in a smaller font in the margin. The \numlabfont macro is provided as a standard name for that font: it is initially defined as \newcommand{\numlabfont}{\normalfont\scriptsize} You might wish to use a different font if, for example, you preferred to have these line numbers printed using old-style numerals.

A relatively trivial matter relates to punctuation. In your footnotes, there will sometimes be spans of line numbers like this: 12–34, or lines with sub-line numbers like this: 55.6. The en-dash and the full stop are taken from the same font as the numbers, and it all works nicely. But what if you wanted to use old-style numbers, like 12 and 34? These look nice in an edition, but when you use the fonts provided by {PLAIN TeX} they are taken from a math font which does not have the en-dash or full stop in the same places as a text font. If you (or your macros) just typed \oldstyle 12--34 or \oldstyle 55.6 you would get ‘12–34’ and ‘55–6’. So we define \endashchar and \fullstop, which produce an en-dash and a full stop respectively from the normal document font, whatever font you are using for the
numbers. These two macros are used in the macros which format the line numbers in the margins and footnotes, instead of explicit punctuation. We also define an \rbracket macro for the right square bracket printed at the end of the lemma in many styles of textual notes (including \eledmac's standard style). For polyglotica, when the lemma is RTL, the bracket automatically switches to a left bracket.

We will briefly discuss \select@lemmefont here because it is important to know about it now, although it is not one of the macros you would expect to change in the course of a simple job. Hence it is 'protected' by having the @-sign in its name.

When you use the \edtext macro to mark a word in your text as a lemma, that word will normally be printed again in your apparatus. If the word in the text happens to be in a font such as italic or bold you would probably expect it to appear in the apparatus in the same font. This becomes an absolute necessity if the font is actually a different script, such as Arabic or Cyrillic. \select@lemmefont does the work of decoding \eledmac's data about the fonts used to print the lemma in the main text and calling up those fonts for printing the lemma in the note.

\select@lemmefont is a macro that takes one long argument—the cluster of line numbers passed to the note commands. This cluster ends with a code indicating what fonts were in use at the start of the lemma. \select@lemmefont selects the appropriate font for the note using that font specifier.

\eledmac uses \select@lemmefont in a standard footnote format macro called \normalfootfmt. The footnote formats for each of the layers A to E are \let equal to \normalfootfmt. So all the layers of the footnotes are formatted in the same way.

4.7 Create a new series

If you need more than 5 series of critical footnotes you can create extra series, using \newseries command. For example to create G and H series \newseriesG,H.

5 Verse

In 1992 Wayne Sullivan\textsuperscript{17} wrote the EDSTANZA macros [Sul92] for typesetting verse in a critical edition. More specifically they were for handling poetry stanzas which use indentation to indicate rhyme or metre.

With Wayne Sullivan's permission the majority of this section has been taken from [Sul92]. Peter has made a few changes to enable his macros to be used in the L\TeX ledmac, and now in \eledmac package.

Use \stanza at the start of a stanza. Each line in a stanza is ended by an ampersand (&), and the stanza itself is ended by putting \& at the end of the last line.

Be careful: you must have NO space between the end of your verse and & or \&. In most cases, you will see no difference, but if your verse is exactly the same length as a line, then you will have an empty hanging verse.

\textsuperscript{17}Department of Mathematics, University College, Dublin 4, Ireland
5.1 Repeating stanza indents

\texttt{\textbackslash setstanzaindents} \hspace{1cm} Lines within a stanza may be indented. The indents are integer multiples of the length \texttt{\textbackslash stanzaindentbase}, whose default value is 20pt.

In order to use the stanza macros, \textbf{one must set the indentation values.} First the value of \texttt{\textbackslash stanzaindentbase} should be set, unless the default value 20pt is desired. Every stanza line indentation is a multiple of this.

To specify these multiples one invokes, for example \texttt{\textbackslash setstanzaindents\{3,1,2,1,2\}}.

The numerical entries must be whole numbers, 0 or greater, separated by commas without embedded spaces. The first entry gives the hanging indentation to be used if the stanza line requires more than one print line.

If it is known that each stanza line will fit on more than one print line, then this first entry should be 0. \TeX{} does less work in this case, but no harm ensues if the hanging indentation is not 0 but is never used.

If you want the hanging verse to be flush right, you can use \texttt{\textbackslash hanginsymbol}: see p. \textbf{28}.

Enumeration is by stanza lines, not by print lines. In the above example the lines are indented one unit, two units, one unit, two units, with 3 units of hanging indentation in case a stanza line is too long to fit on one print line.

5.1 Repeating stanza indents

Since version 0.13, if the indentation is repeated every $n$ verses of the stanza, you can define only the $n$ first indentations, and say they are repeated, defining the value of the \texttt{\textbackslash stanzaindentsrepetition} counter at $n$. For example:

\texttt{\setstanzaindents\{5,1,0\}}
\texttt{\setcounter{stanzaindentsrepetition}\{2\}}

is like

\texttt{\setstanzaindents\{0,1,0,1,0,1,0,1,0,1,0\}}

\textbf{Be careful: the feature change in eledmac 1.5.1. See Appendix A.3 p. 225.}

If you don’t use the \texttt{\textbackslash stanzaindentsrepetition} counter, make sure you have at least one more numerical entry in \texttt{\setstanzavalues} than the number of lines in the stanza.

If you want to disable this feature again, just put the counter to 0:

\texttt{\setcounter{stanzaindentsrepetition}\{0\}}

The macros make no restriction on the number of lines in a stanza. Stanza indentation values (and penalty values) obey \TeX{}’s grouping conventions, so if one stanza among several has a different structure, its indentations (penalties) may be set within a group; the prior values will be restored when the group ends.
5.2 Manual stanza indent

\stanzaindent\stanzaindent*

You can set the indent of some specific verse by calling \stanzaindent\{value\}\stanzaindent* at the beginning of the verse, before any other character. In this case, the indent defined by \setstanzaindents for this verse is skipped, and \{value\}\stanzaindent* is used instead.

If you use the mechanism of indent repetition, the next verse will be printed as it should be even if the current verse would have its normal indent value. In other words, using \stanzaindent in a verse does not shift the indent repetition.

However, if you want to shift the indent repetition, so the next verse has the indent normally used for the current verse, use \stanzaindent* instead of \stanzaindent.

5.3 Stanza breaking

When the stanzas run over several pages, it is often desirable that page breaks should arise between certain lines in the stanza, so a facility for including penalties after stanza lines is provided. If you are satisfied with the page breaks, you need not set the penalty values.

The command

\setstanzapenalties{1,5000,10100,5000,0}

results in a penalty of 5000 being placed after the first and third lines of the stanza, and a penalty of −100 after the second.

The first entry “1” is a control value. If it is zero, then no penalties are passed on to \TeX, which is the default. Values between 0 and 10000 are penalty values; values between 10001 and 20000 have 10000 subtracted and the result is given as a negative penalty. The mechanism used for indentations and penalties requires unsigned values less than 32768. No penalty is placed after the last line, so the final ,0 in the example above could be omitted. The control sequence \endstanzaextra can be defined to include a penalty. A penalty of 10000 will prevent a page break; such a penalty is included automatically where there is stanza hanging indentation. A penalty of −10000 (corresponding to the entry value 20000 in this context) forces a page break. Values in between act as suggestions as to the desirability of a page break at a given line. There is a subtle interaction between penalties and glue, so it may take some adjustment of skips and penalties to achieve the best results.

5.4 Hanging symbol

It’s possible to insert a symbol in each line of hanging verse, as in French typography for \[. To insert in eledmac, redefine macro \hangingsymbol with this code:

\renewcommand{\hangingsymbol}{\}

You can also use it to force hanging verse to be flush right:
If you want to prevent page breaks inside long verses, use the option `nopbinverse` when loading package, or use \texttt{\textbackslash{lednopbinverstrue}}. Read [16 p. 40] for further details.

\textbf{5.6 Various tools}

\texttt{\textbackslash{ampersand}} If you need to print an \& symbol in a stanza, use the \texttt{\textbackslash{ampersand}} macro, not \& which will end the stanza.

\texttt{\textbackslash{endstanzaeextra}} The macro \texttt{\textbackslash{endstanzaeextra}}, if it is defined, is called at the end of a stanza. You could define this, for example, to add extra space between stanzas (by default there is no extra space between stanzas); if you are using the \texttt{memoir} class, it provides a length \texttt{\textbackslash{stanzaskip}} which may come in handy.

\texttt{\textbackslash{startstanzahook}} Similarly, if \texttt{\textbackslash{startstanzahook}} is defined, it is called by \texttt{\textbackslash{stanza}} at the start. This can be defined to do something.

\texttt{\textbackslash{flagstanzanumberit}} Putting \texttt{\textbackslash{flagstanzanumberit}} at the start of a line in a stanza (or elsewhere) will typeset \langle\texttt{text}\rangle at a distance \langle\texttt{len}\rangle before the line. The default \langle\texttt{len}\rangle is \texttt{\textbackslash{stanzaindentbase}}.

For example, to put a verse number before the first line of a stanza you could proceed along the lines:

\begin{verbatim}
\newcounter{stanzanumberit}
\setcounter{stanzanumberit}{0}
\newcommand*{\startstanzahook}{\refstepcounter{stanzanumberit}}
\newcommand{\numberit}{\flagstanzanumberit{\thestanzanumberit}}

...\stanza
\numberit First line...&
    rest of stanza
&
\stanza
\numberit First line, second stanza...
\end{verbatim}

\textbf{5.7 Hanging symbol}

It’s possible to insert a symbol on each line of hanging verse, as in French typography for ‘='. To insert in \texttt{eledmac}, redefine macro \texttt{\textbackslash{hangingsymbol}} with this code:

\begin{verbatim}
\renewcommand{\hangingsymbol}{[\,\}\
\end{verbatim}
5.8 Text before/after verses

It is possible to add text, like a subtitle, before or after verse:

- \stanza command can take a optional argument (in brackets). Its content will be printed before the stanza.
- & can be replaced by \newverse with two optional arguments (in brackets). The first will be printed after the current verse, the second before the next verse.
- \& can take a optional argument (in brackets). Its content will be printed after the stanza.

6 Grouping

In a minipage environment \LaTeX{} changes \footnote numbering from arabic to alphabetic and puts the footnotes at the end of the minipage.

You can put numbered text with critical footnotes in a minipage and the footnotes are set at the end of the minipage.

You can also put familiar footnotes (see section \ref{sec:1}) in a minipage but unlike with \footnote the numbering scheme is unaltered.

Minipages, of course, aren’t broken across pages. Footnotes in a ledgroup environment are typeset at the end of the environment, as with minipages, but the environment includes normal page breaks. The environment makes no change to the textwidth so it appears as normal text; it just might be that footnotes appear in the middle of a page, with text above and below.

The \textwidth environment is similar to ledgroup except that you must specify a width for the environment, as with a minipage.

\begin{ledgroupsized}{\textwidth}
\end{ledgroupsized}

The required \textwidth argument is the text width for the environment. The optional \pos argument is for positioning numbered text within the normal textwidth. It may be one of the characters:

- l (left) numbered text is flush left with respect to the normal textwidth. This is the default.
- c (center) numbered text is in the center of the textwidth.
- r (right) numbered text is flush right with respect to the normal textwidth.

Note that normal text, footnotes, and so forth are all flush left.

\begin{ledgroupsized}{\textwidth} is effectively the same as \begin{ledgroup}

7 Crop marks

The ledmac package does not provide crop marks. These are available with either the memoir class \cite{Wil02} or the crop package.
\section{Endnotes}

\begingroup
\doendnotes{⟨letter⟩} \endprint 
\printnpnum{⟨num⟩} \endprint
\endgroup

\noendnotes \noendnotes\endgroup
\beginnumbering
\endgroup

\doendnotes{⟨letter⟩} \endprint 
\printnpnum{⟨num⟩} \endprint
\endgroup

\noendnotes \beginnumbering \noendnotes\endgroup
\endgroup

As endnotes may be printed at any point in the document they always start with the page number of where they were specified. The macro \printnpnum{⟨num⟩} is used to print these numbers. Its default definition is:
\newcommand*{\printnpnum}[1]{p.#1) }

If you aren’t going to have any endnotes, you can say \noendnotes in your file, before the first \beginnumbering, to suppress the generation of an unneeded .end file.

\section{Cross referencing}

The package provides a simple cross-referencing facility that allows you to mark places in the text with labels, and generate page and line number references to those places elsewhere using those labels.

First you place a label in the text using the command \edlabel{⟨lab⟩}. ⟨lab⟩ can be almost anything you like, including letters, numbers, punctuation, or a combination—anything but spaces; you might say \edlabel{toves-3}, for example.\footnote{More precisely, you should stick to characters in the \TeX categories of ‘letter’ and ‘other’.}

Elsewhere in the text, either before or after the \edlabel, you can refer to its location via \edpageref{⟨lab⟩}, or \edlineref{⟨lab⟩} \footnote{Previously, the \edlineref command was \lineref. But some packages also define \lineref. That is why you should use \edlineref instead of \lineref. \eledmac defines \lineref as equal to \edlineref, except if one package has also defined a \lineref command.}, \sublineref{⟨lab⟩}, or \pstartref{⟨lab⟩}. These commands will produce, respectively, the page, line, sub-line and pstart on which the \edlabel{⟨lab⟩} command occurred.

An \edlabel command may appear in the main text, or in the first argument of \edtext, but not in the apparatus itself. But \edpageref, \edlineref, \sublineref, \pstartref commands can also be used in the apparatus to refer to \edlabels in the text.

The \edlabel command works by writing macros to \LaTeX aux file. You will need to process your document through \LaTeX twice in order for the references to be resolved.

You will be warned if you say \edlabel{foo} and foo has been used as a label before. The ref commands will return references to the last place in the file marked with this label. You will also be warned if a reference is made to an undefined label. (This will also happen the first time you process a document after adding a new \edlabel command: the auxiliary file will not have been updated yet.)
If you want to refer to a word inside an `\edtext{...}{...}` command, the `\edlabel` should be defined inside the first argument, e.g.,

The `\edtext{creature\edlabel{elephant} was quite unafraid}{\Afootnote{Of the mouse, that is.}}`

However, there are situations in which you’ll want eledmac to return a number without displaying any warning messages about undefined labels or the like: if you want to use the reference in a context where \TeX is looking for a number, such a warning will lead to a complaint that the number is missing. This is the case for references used within the argument to `\linenum`, for example. For this situation, three variants of the reference commands, with the `x` prefix, are supplied: `\xpageref`, `\xlineref`, `\xsublineref` and `\xpstartref`. They have these limitations:

- They will not tell you if the label is undefined.
- They must be preceded in the file by at least one of the four other cross-reference commands—e.g., a `\edlabel{foo}` command, even if you never refer to that label—since those commands can all do the necessary processing of the `.aux` file, and the `\x...` ones cannot.
- When `hyperref` is loaded, the `hyperref` link won’t be added. (Indeed, it’s not a limitation, but a feature.

The macros `\xxref` and `\edmakelabel` let you manipulate numbers and labels in ways which you may find helpful in tricky situations.

The `\xxref{⟨lab1⟩}{⟨lab2⟩}` command generates a reference to a sequence of lines, for use in the second argument of `\edtext`. It takes two arguments, both of which are labels: e.g., `\xxref{mouse}{elephant}`. It calls `\linenum` (q.v., p.17 above) and sets the beginning page, line, and sub-line numbers to those of the place where `\edlabel{mouse}` was placed, and the ending numbers to those where `\edlabel{elephant}` occurs.

Sometimes the `\edlabel` command cannot be used to specify exactly the page and line desired—for example, if you want to refer to a page and line number in another volume of your edition. In such cases, you can use the `\edmakelabel{⟨lab⟩}{⟨numbers⟩}` macro so that you can ‘roll your own’ label. For example, if you say `\edmakelabel{elephant}{10|25|0}` you will create a new label, and a later call to `\edpageref{elephant}` would print ‘10’ and `\lineref{elephant}` would print ‘25’. The sub-line number here is zero. It is usually best to collect your `\edmakelabel` statements near the top of your document, so that you can see them at a glance.

The normal `\label`, `\ref` and `\pageref` macros may be used within numbered text, and operate in the familiar fashion.
10 Side notes

The \texttt{marginpar} command does not work in numbered text. Instead the package provides for non-floating sidenotes in either margin.

\texttt{\textbackslash ledinnernote\{text\}} will put \texttt{⟨text⟩} into the inner margin level with where the command was issued. Similarly, \texttt{\textbackslash ledouternote\{text\}} puts \texttt{⟨text⟩} in the outer margin.

\texttt{\textbackslash ledsidenote\{text\}} will put \texttt{⟨text⟩} into the margin specified by the current setting of \texttt{\textbackslash sidenotemargin\{\textit{location}\}. The permissible value for \texttt{⟨location⟩} is one out of the list \texttt{left}, \texttt{right}, \texttt{inner}, or \texttt{outer}, for example \texttt{\textbackslash sidenotemargin\{outer\}. The package's default setting is to typeset \texttt{\textbackslash ledsidenote} in the right hand margin. This is the opposite to the default margin for line numbers. The style for a \texttt{\textbackslash ledsidenote} follows that for a \texttt{\textbackslash ledleftnote} or a \texttt{\textbackslash ledrightnote} depending on the margin it is put in.

If two, say, \texttt{\textbackslash ledleftnote}, commands are called in the same line the second \texttt{⟨text⟩} will obliterate the first. There is no problem though with having both a left and a right sidenote on the same line.

The left sidenote text is put into a box of width \texttt{\textbackslash ledlsnotewidth} and the right text into a box of width \texttt{\textbackslash ledrsnotewidth}. These are initially set to the value of \texttt{\textbackslash marginparwidth}.

By default, Sidenotes are placed to align with the last line of the note to which it refers. If you want them to be placed to align with the first line of the note to which it refers, use \texttt{\textbackslash leftnoteupfalse} (for left note) and/or \texttt{\textbackslash rightnoteupfalse} (for right note).

The texts are put a distance \texttt{\textbackslash ledlsnotesep} (or \texttt{\textbackslash ledrsnotesep}) into the left margin. These lengths are initially set to the value of \texttt{\textbackslash linenumsep}.

These macros specify how the sidenote texts are to be typeset. The initial definitions are:

\begin{verbatim}
\newcommand*{\ledlsnotefontsetup}{\raggedleft\footnotesize}% left
\newcommand*{\ledrsnotefontsetup}{\raggedright\footnotesize}% right
\end{verbatim}

These can of course be changed to suit.

If you have two or more sidenotes for the same line, they are separated by a comma. But if you want to change this separator, you can redefine the macro \texttt{\textbackslash sidenotesep}.

11 Familiar footnotes

The \texttt{footmisc} package \cite{b-Fai03} by Robin Fairbairns has an option whereby sequential footnote marks in the text can be separated by commas\textsuperscript{3,4} like so. As a convenience \texttt{eledmac} provides this automatically.

\texttt{\textbackslash multfootsep} is used as the separator between footnote markers. Its default definition is:
As well as the standard \LaTeX\ footnotes generated via \footnote, the package also provides five series of additional footnotes called \footnoteA through \footnoteE. These have the familiar marker in the text, and the marked text at the foot of the page can be formatted using any of the styles described for the critical footnotes. Note that the ‘regular’ footnotes have the series letter at the end of the macro name whereas the critical footnotes have the series letter at the start of the name.

Each of the \foot...X macros takes one argument which is the series letter (e.g., B). \footnormalX is the typical footnote format. With \footparagraphX the series is typeset a one paragraph, with \foottwocolX the notes are in two columns, and are in three columns with \foothreecolX.

As well as using the \foot...X macros to specify the general footnote arrangement for a series, each series uses a set of macros for styling the marks. The mark numbering scheme is defined by the \thefootnoteA macro; the default is:

\renewcommand*{\thefootnoteA}{\arabic{footnoteA}}

The appearance of the mark in the text is controlled by \bodyfootmarkA which is defined as:

\newcommand*{\bodyfootmarkA}{\hbox{\textsuperscript{\@nameuse{@thefnmarkA}}}}

The command \footfootmarkA controls the appearance of the mark at the start of the footnote text. It is defined as:

\newcommand*{\footfootmarkA}{\textsuperscript{\@nameuse{@thefnmarkA}}}

There are similar command triples for the other series.

Additional footnote series can be easily defined: you just have to use \newseries, defined above (see \thesection p\pageref{sec:footnote}).

11.1 Position of the familiar footnotes

There is a historical incoherence in \evedmac. The familiar footnotes are before the critical footnotes in a normal page, but after in a minipage or in a ledgroup. However, it is possible to change the relative position of both types of footnotes. If you want to have familiar footnotes after critical footnotes in a normal page, use:

\fnpos{critical-familiar}

Or, if you want a minipage or ledgroup to have critical footnotes after familiar footnotes, use:

\mpfnpos{familiar-critical}

12 Indexing

\edindex \LaTeX\ provides the \index{\item} command for specifying that \item and
the current page number should be added to the raw index (idx) file. The \edindex{{item}} macro can be used in numbered text to specify that {item} and the current page & linenumber should be added to the raw index file.

If the \texttt{memoir} class or the \texttt{imakeidx} or \texttt{indextools} package is used then the macro takes an optional argument, which is the name of a raw index file. For example \edindex[\texttt{line}]{item} will use \texttt{line.idx} as the raw file instead of \texttt{\jobname.idx}.

The minimal version of \texttt{imakeidx} package to be used is the version 1.3a uploaded on CTAN on 2013/07/11.

Be careful with the order of package loading and index declaration. You must use this order:

1. Load \texttt{imakeidx} or \texttt{indextools}.
2. Load \texttt{eledmac}.
3. Declare the index with the macro \makeindex of \texttt{imakeidx}/\texttt{indextools}.

\texttt{\pagelinesep} The page & linenumber combination is written as \texttt{page\pagelinesep line}, where the default definition is \texttt{newcommand{\pagelinesep}{-}} so that an item on page 3, line 5 will be noted as being at 3-5. You can renew \texttt{\pagelinesep} to get a different separator (but it just so happens that - is the default separator used by the \texttt{MakeIndex} program).

\texttt{\edindexlab} The \texttt{\edindex} process uses a \texttt{\label}/\texttt{\ref} mechanism to get the correct line number. It automatically generates labels of the form \texttt{\label{\edindexlab N}}, where N is a number, and the default definition of \texttt{\edindexlab} is: \texttt{newcommand*{\edindexlab}{$\&$}} in the hopes that this will not be used by any other labels (\texttt{\edindex}’s labels are like \texttt{\label{$\&$27}}). You can change \texttt{\edindexlab} to something else if you need to.

13 Tabular material

\LaTeX{}’s normal \texttt{tabular} and \texttt{array} environments cannot be used where line numbering is being done; more precisely, they can be used but with odd results, so don’t use them. However, \texttt{eledmac} provides some simple tabulation environments that can be line numbered. The environments can also be used in normal unnumbered text.

There are six environments; the \texttt{edarray*} environments are for math and\texttt{edarrayl} \texttt{edarrayc} \texttt{edarrayr} \texttt{edtabularl} \texttt{edtabularc} \texttt{edtabularr} for text entries. The final l, c, or r in the environment names indicate that the entries will be flushleft (l), centered (c) or flushright (r). There is no means of specifying different formats for each column, nor for specifying a fixed width for a column. The environments are centered with respect to the surrounding text.
Entries in the environments are the same as for the normal \texttt{array} and \texttt{tabular} environments but there must be no ending $\backslash\backslash$ at the end of the last row. \textit{There must be the same number of column designators (the $\&$) in each row.} There is no equivalent to any line drawing commands (such as \texttt{\backslash hline}). However, unlike the normal environments, the \texttt{ed...} environments can cross page breaks.

Macros like \texttt{\texttt{edtext}} can be used as part of an entry.

For example:

\begin{verbatim}
\begin{edtabularl}
\textbf{Large I} & wish I was a little bug& \edindex{bug} & \textbf{Large I} & eat my peas with honey& \edindex{honey} \\
& With whiskers & \texttt{\texttt{edtext}{round}{\texttt{Afootnote}{around}}} & my tummy & & \\
& I've done it all my life. & & & & \\
& I'd climb into a honey& \edindex{honey} & pot & & \\
& It makes the peas taste funny & & & & \\
& And get my tummy gummy.& \edindex{gummy} & & & \\
\texttt{\texttt{edtext}{\edrowfill}{\texttt{\texttt{start}}}{}{\texttt{end}}} & & & & & \\
\texttt{\texttt{edtext}{\texttt{\texttt{pend}}}}
\end{edtabularl}
\end{verbatim}

produces the following parallel pair of verses.

1 I wish I was a little bug I eat my peas with honey
2 With whiskers round my tummy I've done it all my life.
3 I'd climb into a honey pot It makes the peas taste funny
4 And get my tummy gummy. But it keeps them on the knife.

\texttt{\texttt{\texttt{edtabcolsep}}} The distance between the columns is controlled by the length \texttt{\texttt{edtabcolsep}}.
\texttt{\texttt{spreadmath}} \texttt{\texttt{\texttt{\texttt{spreadmath}{\{math\}}} typesets \{\langle{\texttt{math}}\rangle\} but the \{\langle{\texttt{math}}\rangle\} has no effect on}}
\texttt{\texttt{\texttt{spreadtext}}} the calculation of column widths. \texttt{\texttt{\texttt{spreadtext}{\{text\}}} is the analogous command}}
for use in \texttt{edtabular} environments.
\begin{verbatim}
\begin{edarrayl}
1 & 2 & 3 & 4 \\\n& \texttt{\texttt{spreadmath}{F+G+C}} & & \\
1 & 2 & 3 & 4 \\
\end{edarrayl}
\end{verbatim}

\texttt{\texttt{\texttt{edrowfill}}} The macro \texttt{\texttt{\texttt{edrowfill}{\langle{\texttt{start}}}\rangle}{\langle{\texttt{end}}}\rangle{\langle{\texttt{fill}}}\rangle fills columns number $\langle{\texttt{start}}}\rangle$ to $\langle{\texttt{end}}}\rangle$ inclusive with $\langle{\texttt{fill}}}\rangle$. The $\langle{\texttt{fill}}}\rangle$ argument can be any horizontal ‘fill’. For example \texttt{\texttt{\texttt{\texttt{hrulefill}}} or \texttt{\texttt{upbracefill}}}.
Note that every row must have the same number of columns, even if some would not appear to be necessary.

The \texttt{\textbackslash edrowfill} macro can be used in both tabular and array environments. The typeset appearance of the following code is shown below.

\begin{verbatim}
\begin{edtabularr}
1 & 2 & 3 & 4 & 5 \\
Q & & fd & h & qwertziohg \\
v & & wptz & x & y & vb \\
g & nnn & \texttt{\textbackslash edrowfill\{3\}\{5\}\{\textbackslash upbracefill\}} & & \\
\texttt{\textbackslash edrowfill\{1\}\{3\}\{\textbackslash downbracefill\}} & & & & \\
k & & 1 & co & ghweropjklmnbvcxys \\
1 & & 2 & 3 & \texttt{\textbackslash edrowfill\{4\}\{5\}\{\textbackslash hrulefill\}}
\end{edtabularr}
\end{verbatim}

You can also define your own ‘fill’. For example:

\begin{verbatim}
\newcommand*{\upbracketfill}{%\vrule height 4pt depth 0pt\hrulefill\vrule height 4pt depth 0pt}
\end{verbatim}

is a fill like \texttt{\textbackslash upbracefill} except it has the appearance of a (horizontal) bracket instead of a brace. It can be used like this:

\begin{verbatim}
\begin{edarrayc}
1 & 2 & 3 & 4 & 5 \\
Q & & fd & h & qwertziohg \\
v & & wptz & x & y & vb \\
g & nnn \hline \upbracketfill \hline
k & & 1 & co & ghweropjklmnbvcxys \\
1 & 2 & 3 \hline \upbracketfill
\end{edarrayc}
\end{verbatim}

You can also define your own ‘fill’. For example:

\begin{verbatim}
\newcommand*{\upbracketfill}{%\vrule height 4pt depth 0pt\hrulefill\vrule height 4pt depth 0pt}
\end{verbatim}

is a fill like \texttt{\textbackslash upbracefill} except it has the appearance of a (horizontal) bracket instead of a brace. It can be used like this:

\begin{verbatim}
\begin{edarrayc}
1 & 2 & 3 & 4 & 5 \\
Q & & fd & h & qwertziohg \\
v & & wptz & x & y & vb \\
g & nnn \hline \upbracketfill \hline
k & & 1 & co & ghweropjklmnbvcxys \\
1 & 2 & 3 \hline \upbracketfill
\end{edarrayc}
\end{verbatim}

\begin{verbatim}
\edatleft[\langle math\rangle] \{\langle symbol\rangle}\{\langle halfheight\rangle}\} typesets the math \langle symbol\rangle as
\edatright \langle left\langle symbol\rangle\rangle with the optional \langle math\rangle centered before it. The \langle symbol\rangle is twice \langle halfheight\rangle tall. The \texttt{\textbackslash edatright} macro is similar and it typesets \langle right\langle symbol\rangle\rangle with \langle math\rangle centered after it.
\end{verbatim}
\begin{edarrayc}
  & 1 & 2 & 3 & \\
  & 4 & 5 & 6 & \\
  \end{edarrayc}

\begin{edarrayc}
  & 7 & 8 & 9 & \\
\end{edarrayc}

\begin{edarrayc}
left = \begin{bmatrix}
  1 & 2 & 3 \\
  4 & 5 & 6 \\
  7 & 8 & 9 \\
\end{bmatrix} = right
\end{edarrayc}

\begin{edarrayl}
A & 1 & 2 & 3 \\
\& 1 & 3 & 6 \\
C & 1 & 4 & \& 8 \\
D & 1 & 5 & 0
\end{edarrayl}

\begin{edarrayr}
a & b & C & d & \\
v & w & x & y & \\
m & n & o & p & \\
k & k & L & cvb & \end{edarrayr}

\begin{edarrayr}
a & b & C & d \\
v & w & x & y \\
m & n & o & p \\
k & L & cvb
\end{edarrayr}

The macro \texttt{edvertline}\{(height)\} draws a vertical line \texttt{height} high (contrast this with \texttt{edatright} where the size argument is half the desired height).

\begin{edarrayr}
a & b & C & d & \\
v & w & x & y & \\
m & n & o & p & \\
k & k & L & cvb & \end{edarrayr}

The \texttt{edvertdots} macro is similar to \texttt{edvertline} except that it produces a vertical dotted instead of a solid line.
14 Sectioning commands

The standard sectioning command (\chapter, \section etc.) can be used inside a numbered text. But the line which contains it won’t be numbered, and you can’t add critical notes inside. In the past (between versions 1.1.0 and 1.12.0), these following commands were provided:

- \ledchapter[⟨text⟩]{⟨critical text⟩}
- \ledchapter*
- \ledsection[⟨text⟩]{⟨critical text⟩}
- \ledsection*
- \ledsubsection[⟨text⟩]{⟨critical text⟩}
- \ledsubsection*
- \ledsubsubsection[⟨text⟩]{⟨critical text⟩}
- \ledsubsubsection*

These commands are deprecated, and won’t be maintained anymore, because of a bad conception. Since version 1.12.0, you have to use the following commands:

- \eledchapter[⟨text⟩]{⟨critical text⟩}
- \eledchapter*
- \eledsection[⟨text⟩]{⟨critical text⟩}
- \eledsection*
- \eledsubsection[⟨text⟩]{⟨critical text⟩}
- \eledsubsection*
- \eledsubsubsection[⟨text⟩]{⟨critical text⟩}
- \eledsubsubsection*

Which are equivalent to the \LaTeX commands. Each individual command must be called alone in a \pstart...\pend:

\pstart
\eledsection*{xxxx\ledsidenote{section}}
\pend
\pstart
\eledsubsection*{xxxx\ledsidenote{sub}}
\pend
\pstart
normal text
\pend
At the first run, you will see only the text. It’s normal. At the second run, you will see the formatting. And consequently, at the third run, you will see the table of contents.

For technical reasons, the page break before \elechapter can’t be added automatically. You have to insert it manually via \beforeelechapter, which must be called outside of a numbering section. If you aren’t going to have any elexxx commands, you can say \noeledsec in your file, before the first \beginnumbering, to suppress the generation of unneeded .eledsec file.

15 Quotation environments

The quotation and quote environment can be used so that same definition/note appears both inside and outside a numbered section. The typographical consequences will resemble the outside numbered sections, based on the styles of the book class. However, if you use a package that redefines these environments, these redefinitions won’t be available inside the numbering section. You must open any quotation environments inside a \start-\pend block, not outside. A quotation environment MUST not be opened immediately after a \pstart and MUST not be closed immediately before a \pend.

In some case, you don’t want these environments be redefined in numbered section. You can load the package with the option noquotation to prevent this redefinition.

16 Page breaks

Eledmac and elepar break pages automatically. However, you may sometimes want to either force page breaks or prevent them. The packages provide two macros:

- \ledpb adds a page break.
- \lednopb prevents a page break, by adding one line to the current page if needed.

These commands have effect only at the second run.

These two commands take effect at the beginning of line in which they are called. For example, if you call \ledpb at l. 444, the l. 443 will be at the p. n, and the l. 444 at the p. n + 1. However you can change the behavior, and decide they will have effect after the end of the line, adding \ledpsetting{after} at the beginning of your file (better: in your preamble). With the previous example, the l. 444 will be at the p. n and the l. 445 will be at the p. n + 1.

If you are using elepar to typeset parallel pages you must use \lednopb on both sides in the two corresponding lines. This is especially important when you are using stanzas; otherwise the pages will run out of sync. You can also decide to prevent page breaks between two lines of a long verse. To do this, use nopbinverse when loading package, or add \lednopbinversetrue in the beginning of your file.
This feature works only with verse of 2 lines, not more. It works at the third run, or at fourth run with \eledpar. By default, when a long verse runs normally between two pages, a page break will be placed at the beginning of the verse. However, if you have added \ledpbsetting{after}, the page break will be placed at the end of the long verse, and the page containing the long verse will have one extra line.

17 Miscellaneous

**\extensionchars** When the package assembles the name of the auxiliary file for a section, it prefixes \extensionchars to the section number. This is initially defined to be empty, but you can add some characters to help distinguish these files if you like; what you use is likely to be system-dependent. If, for example, you said \renewcommand{\extensionchars}{!}, then you would get temporary files called \jobname.1, \jobname.2, etc.

**\ifledfinal** The package can take options. The option ‘final’, which is the default is for final typesetting; this sets \ifledfinal to TRUE. The other option, ‘draft’, may be useful during earlier stages and sets \ifledfinal to FALSE.

**\showlemma** The lemma within the text is printed via \showlemma{lemma}. Normally, or with the ‘final’ option, the definition of \showlemma is:

\newcommand*{\showlemma}[1]{#1}

so it just produces its argument. With the ‘draft’ option it is defined as

\newcommand*{\showlemma}[1]{\textit{#1}}

so that its argument is typeset in an italic font, which may make it easier to check that all lemmas have been treated.

If you would prefer some other style, you could put something like this in the preamble:

\ifledfinal
  \renewcommand{\showlemma}[1]{\textbf{#1}}% or simply ...[1]{#1}
\fi

17.1 Known and suspected limitations

In general, \eledmac’s system for adding marginal line numbers breaks anything that makes direct use of the \TeX{} insert system, which includes marginpars, footnotes and floats.

However, you can use both \footnote and the familiar footnote series notes in numbered text. A \marginpar in numbered text will throw away its contents and send a warning message to the terminal and log file, but will do no harm.

\parshape cannot be used within numbered text, except in a very restricted way.

\ballast \TeX{} is a three-pass system, but even after a document has been processed three times, there are some tricky situations in which the page breaks decided by \TeX{} never settle down. At each successive run, \eledmac may oscillate between two
different sets of page decisions. To stop this happening, should it arise, Wayne Sullivan suggested the inclusion of the quantity \texttt{\textbackslash ballast}. The amount of \texttt{\textbackslash ballast} will be subtracted from the penalties which apply to the page breaks calculated on the previous run through \TeX, thus reinforcing these breaks. So if you find your page breaks oscillating, say
\texttt{\setcounter{ballast}{100}}
or some such figure, and with any luck the page breaks will settle down. Luckily, this problem doesn’t crop up at all often.

The restriction on explicit line-breaking in paragraphed footnotes, mentioned in a footnote\footnote{16} p. 25 and described in more detail on p. 117 really is a nuisance if that’s something you need to do. There are some possible solutions, described by Michael Downes, but this area remains unsatisfactory.

\LaTeX has a reputation for putting things in the wrong margin after a page break. The \texttt{eledmac} package does nothing to improve the situation — in fact it just makes it more obvious if numbered text crosses a page (or column) boundary and the numbers are meant to flip from side to side. Try and keep the numbers in the same margin all the time. Another aspect of TeX’s page breaking mechanism is that when numbering lines by the page, the first few numbers after a page break may continue as though the lines were still on the previous page.

If you can’t resist flipping the numbers or numbering by the page, then you might find that judicious use of \texttt{\pageparbreak} may help if numbering goes awry across a page (or column) break. It tries to force TeX into partitioning the current paragraph into two invisibly joined paragraphs with a page break between them. Insert the command between the last word on one page and the first word on the next page. If later you change something earlier in the document the natural page break may be in a different place, and you will have to adjust the location of \texttt{\pageparbreak} accordingly.

\texttt{\renewcommand{\footfudgefiddle}{68}}

Help, suggestions and corrections will be gratefully received.

17.2 Use with other packages

Because of \texttt{eledmac}’s complexity it may not play well with other packages. In particular \texttt{eledmac} is sensitive to commands in the arguments to the \texttt{\textbackslash edtext} and \texttt{\*footnote} macros (this is discussed in more detail in section\footnote{22} and in particular the discussion about \texttt{\noexpand} and \texttt{\morenoexpand}). You will have to see what works or doesn’t work in your particular case.

It is possible that \texttt{eledmac} and the \texttt{hyperref} package may work together. I have not tried this combination but past experience with \texttt{hyperref} suggests that cooperation is unlikely; \texttt{hyperref} changes many \LaTeX internals and \texttt{eledmac} does things that are not normally seen in \LaTeX.
17.3 Parallel typesetting

If you want to use the option \textit{bottom} of the \texttt{footmisc} package, you must load this package \textit{before} the \texttt{eledmac} package.

You can define the macro \texttt{\morenoexpands} to modify macros that you call within \texttt{\edtext}. Because of the way \texttt{eledmac} numbers the lines the arguments to \texttt{\edtext} can be processed more than once and in some cases a macro should only be processed once. One example is the \texttt{\colorbox} macro from the \texttt{color} package, which you might use like this:

\begin{verbatim}
... \edtext{\colorbox{mycolor}{lemma}}\{\Afootnote{...\colorbox...}}
\end{verbatim}

If you actually try this\footnote{Reported by Dirk-Jan Dekker in the CTT thread ‘Incompatibility of “color” package’ on 2003/08/28.} you will find \LaTeX{} whinging ‘\texttt{Missing \{} inserted’, and then things start to fall apart. The trick in this case is to specify either:

\begin{verbatim}
\newcommand{\morenoexpands}{\let\colorbox=0}
\end{verbatim}
or
\begin{verbatim}
\makeatletter
\newcommand{\morenoexpands}{\let\colorbox\@secondoftwo}
\makeatother
\end{verbatim}

(\texttt{\@secondoftwo} is an internal \LaTeX{} macro that takes two arguments and throws away the first one.) The first incantation lets color show in both the main text and footnotes whereas the second one shows color in the main text but kills it in the lemma and footnotes. On the other hand if you use \texttt{\textcolor} instead, like

\begin{verbatim}
... \edtext{\textcolor{mycolor}{lemma}}\{\Afootnote{...\textcolor...}}
\end{verbatim}

there is no need to fiddle with \texttt{\morenoexpands} as the color will naturally be displayed in both the text and footnotes. To kill the color in the lemma and footnotes, though, you can do:

\begin{verbatim}
\makeatletter
\newcommand{\morenoexpands}{\let\textcolor\@secondoftwo}
\makeatother
\end{verbatim}

It took me a little while to discover all this. If you run into this sort of problem you may have to spend some time experimenting before hitting on a solution.

17.3 Parallel typesetting

Peter Wilson has developed the Ledpar package as an extension to \texttt{eledmac} specifically for parallel typesetting of critical texts. This also cooperates with the \texttt{babel}
/ polyglossia packages for typesetting in multiple languages. The package has been called eledpar since September 2012.

He also developed the ledarab package for handling parallel Arabic text in critical editions. However, this package is not maintained by Maieul Rouquette. You should use the capabilities of a modern TeX processor, like Xe(La)TeX.
18 Implementation overview

We present the \texttt{eledmac} code in roughly the order in which it’s used during a run of \texttt{TeX}. The order is \textit{exactly} that in which it’s read when you load \texttt{eledmac} package, because the same file is used to generate this manual and to generate the \LaTeX{} package file. Most of what follows consists of macro definitions, but there are some commands that are executed immediately—especially at the start of the code. The documentation generally describes the code from the point of view of what happens when the macros are executed, though. As each macro is introduced, its name is printed in the margin.

We begin with the commands you use to start and stop line numbering in a section of text (Section \ref{sec:19}). Next comes the machinery for writing and reading the auxiliary file for each section that helps us count lines, and for creating list macros encoding the information from that file (Section \ref{sec:21}); this auxiliary file will be read at the start of each section, to create those list macros, and a new version of the file will be started to collect information from the body of the section.

Next are commands for marking sections of the text for footnotes (Section \ref{sec:22}), followed by the macros that take each paragraph apart, attach the line numbers and insertions, and send the result to the vertical list (Section \ref{sec:23}). The footnote commands (Section \ref{sec:24}) and output routine (Section \ref{sec:32}) finish the main part of the processing; cross-referencing (Section \ref{sec:33}) and endnotes (Section \ref{sec:30}) complete the story.

In what follows, macros with an @ in their name are more internal to the workings of \texttt{eledmac} than those made up just of ordinary letters, just as in \texttt{Plain TeX} (see \textit{The \TeX{}book}, p. 344). You are meant to be able to make free with ordinary macros, but the ‘@’ ones should be treated with more respect, and changed only if you are pretty sure of what you are doing.

19 Preliminaries

We try and use \texttt{\@d} in macro names to help avoid name clashes, but this is not a hard and fast rule. For example, if an original \texttt{EDMAC} macro includes \texttt{edmac} We will simply change that to \texttt{eledmac}.

Announce the name and version of the package, which is targetted for \LaTeX{}2e.

\begin{verbatim}
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{eledmac}[2015/02/23 v1.18.0 \LaTeX{} port of EDMAC]
\end{verbatim}

Generally, these are the modifications to the original. \texttt{EDMAC} code:

- Replace as many \texttt{\def}’s by \texttt{\newcommand}’s as possible to avoid overwriting \texttt{\LaTeX} macros.
- Replace user-level \texttt{\TeX} counts by \texttt{\LaTeX} counters.
- Use the \texttt{\LaTeX} font handling mechanisms.
- Use \texttt{\LaTeX} messaging and file facilities.
19.1 Package options

Use this to remember which option is used, set and execute the options with final as the default.

\begin{verbatim}
\ifledfinal
\ifparapparatus@
\ifnoquotation@
\iflednopbinverse
\ifparledgroup
\ifwidthliketwocolumns
\ifledsecnolinenumber
\newif\ifledfinal
\DeclareOption{noquotation}{\noquotation@true}
\DeclareOption{final}{\ledfinaltrue}
\DeclareOption{draft}{\ledfinalfalse}
\DeclareOption{parapparatus}{\parapparatus@true}
\DeclareOption{nopbinverse}{\lednopbinversetrue}
\DeclareOption{ledsecnolinenumber}{\ledsecnolinenumbertrue}
\DeclareOption{widthliketwocolumns}{\widthliketwocolumnstrue}%
\ExecuteOptions{final}
\end{verbatim}

Use the starred form of `\ProcessOptions` which executes options in the order listed in the source file: class options, then listed package options, so a package option can override a class option with the same name. This was suggested by Dan Luecking in the ctt thread *Class/package option processing*, on 27 February 2004.

\begin{verbatim}
\ProcessOptions*\relax
\end{verbatim}

Loading package `xargs` to declare commands with optional arguments. `Etoolbox` is also used to make code clearer - for example, in dynamic command names (which can replace `\csname` etc.). Use `suffix` to declare commands with a starred version, `xstring` to work with strings and `ifluatex` to test if LuaLaTeX is running, and `ragged2e` to manage ragging for paragraphed notes.

\begin{verbatim}
\RequirePackage{xargs}
\RequirePackage{etoolbox}
\reserveinserts{32}
\RequirePackage{suffix}
\RequirePackage{xstring}
\RequirePackage{ifluatex}
\RequirePackage{ragged2e}
\end{verbatim}

\begin{verbatim}
\if@RTL
\else
\fi
\end{verbatim}

The `\if@RTL` is defined by the bidi package, which is sometimes loaded by `polyglossia`. But we define it if the bidi package is not loaded.

\begin{verbatim}
\ifdef{\if@RTL}{\newif\if@RTL}
\end{verbatim}

`\showlemma` `\showlemma{\{lemma\}}` typesets the lemma text in the body. It depends on the option.

\begin{verbatim}
\ifledfinal
\end{verbatim}
Loading packages

Loading package \texttt{xargs} to declare commands with optional arguments. \texttt{Etoolbox} is also used to make code clearer - for example, in dynamic command names (which can replace \csname etc.). Use \texttt{suffix} to declare commands with a starred version, \texttt{xstring} to work with strings, \texttt{ifluatex} to test if LuaLaTeX is running, and \texttt{ragged2e} to manage ragged for paragraphed notes.

\ Reserve inserts\{32\}

\newcommand*{\showlemma}[1]{\underline{#1}}

\newif\l@dmemoir
\@ifclassloaded{memoir}{\l@dmemoirtrue}{\l@dmemoirfalse}

\newif\l@imakeidx
\@ifpackageloaded{imakeidx}{\l@imakeidxtrue}{}%False is the default value
\let\imki@wrindexentry\indtl@wrindexentry

\newif\l@indextools\@ifpackageloaded{indextools}{\l@indextoolstrue}{\l@indextoolsfalse}\l@imakeidxtrue\l@imakeidxtrue\let\imki@wrindexentry\indtl@wrindexentry

\ifdef{\if@RTL}{}{\newif\if@RTL}

\newif\l@imakeidx
\newif\l@indextools

19.4 Messages

All the messages are grouped here as macros. This saves TeX's memory when the same message is repeated and also lets them be edited easily.
\eledmac@warning Write a warning message.
55 \newcommand{\eledmac@warning}[1]{\PackageWarning{eledmac}{#1}}

\eledmac@error Write an error message.
56 \newcommand{\eledmac@error}[2]{\PackageError{eledmac}{#1}{#2}}

\led@err@NumberingStarted \led@err@NumberingNotStarted \led@err@NumberingShouldHaveStarted
57 \newcommand*{\led@err@NumberingStarted}{\eledmac@error{Numbering has already been started}{@ehc}}
58 \newcommand*{\led@err@NumberingNotStarted}{\eledmac@error{Numbering was not started}{@ehc}}
59 \newcommand*{\led@err@NumberingShouldHaveStarted}{\eledmac@error{Numbering should already have been started}{@ehc}}

\led@err@edtextoutsidepstart
63 \newcommand*{\led@err@edtextoutsidepstart}{\eledmac@error{\string\edtext space outside numbered paragraph (\pstart...\pend)}{@ehc}}

\led@mess@NotesChanged
65 \newcommand*{\led@mess@NotesChanged}{\typeout{eledmac reminder: }\typeout{ The number of the footnotes in this section has changed since the last run.}\typeout{ You will need to run LaTeX two more times before the footnote placement and line numbering in this section are correct.}}

\led@mess@SectionContinued
73 \newcommand*{\led@mess@SectionContinued}[1]{\message{Section #1 (continuing the previous section)}}

\led@err@LineationInNumbered
75 \newcommand*{\led@err@LineationInNumbered}{\eledmac@error{You can’t use \string\lineation space within a numbered section}{@ehc}}

\led@warn@BadLineation \led@warn@BadLinenummargin \led@warn@BadLockdisp \led@warn@BadSublockdisp
78 \newcommand*{\led@warn@BadLineation}{\eledmac@warning{Bad \string\lineation space argument}}
79 \newcommand*{\led@warn@BadLinenummargin}{\eledmac@warning{Bad \string\linenummargin space argument}}
80 \newcommand*{\led@warn@BadLockdisp}{\eledmac@warning{Bad \string\lockdisp space argument}}
81 \newcommand*{\led@warn@BadSublockdisp}{\eledmac@warning{Bad \string\sublockdisp space argument}}

\led@warn@NoLineFile
86 \newcommand*{\led@warn@NoLineFile}[1]{\eledmac@warning{Can’t find line-list file #1}}
19.4 Messages

\newcommand*{\led@warn@BadAdvancelineSubline}{ %
  \eledmac@warning{string\advanceline\space produced a sub-line \number less than zero.}}
\newcommand*{\led@warn@BadAdvancelineLine}{ %
  \eledmac@warning{string\advanceline\space produced a line \number less than zero.}}

\newcommand*{\led@warn@BadSetline}{ %
  \eledmac@warning{Bad \string\setline\space argument}}
\newcommand*{\led@warn@BadSetlinenum}{ %
  \eledmac@warning{Bad \string\setlinenum\space argument}}

\newcommand*{\led@err@PstartNotNumbered}{ %
  \eledmac@error{\string\pstart\space must be used within a numbered section}{\@ehc}}
\newcommand*{\led@err@PstartInPstart}{ %
  \eledmac@error{\string\pstart\space encountered while another \string\pstart\space was in effect}{\@ehc}}
\newcommand*{\led@err@PendNotNumbered}{ %
  \eledmac@error{\string\pend\space must be used within a numbered section}{\@ehc}}
\newcommand*{\led@err@PendNoPstart}{ %
  \eledmac@error{\string\pend\space must follow a \string\pstart}{\@ehc}}
\newcommand*{\led@err@AutoparNotNumbered}{ %
  \eledmac@error{\string\autopar\space must be used within a numbered section}{\@ehc}}

\newcommand*{\led@warn@BadAction}{ %
  \eledmac@warning{Bad action code, value \next@action.}}
\newcommand*{\led@warn@DuplicateLabel}[1]{ %
  \eledmac@warning{Duplicate definition of label ‘#1’ on page \the\pageno.}}
\newcommand*{\led@warn@RefUndefined}[1]{ %
  \eledmac@warning{Reference ‘#1’ on page \the\pageno\space undefined. Using ‘000’.}}
\newcommand*{\led@warn@NoMarginpars}{ %
  \eledmac@warning{You can’t use \string\marginpar\space in numbered text}}
\newcommand*{\led@warn@BadSidenotemargin}{ %
  \eledmac@warning{Bad \string\sidenotemargin\space argument}}
19.5 Gobbling

Here, we define some commands which gobble their arguments.

19.6 Miscellaneous commands
20 Sectioning commands

\section@num
You use \texttt{\begin{numbering}} and \texttt{\end{numbering}} to begin and end a line-numbered section of the text; the pair of commands may be used as many times as you like within one document to start and end multiple, separately line-numbered sections. \LaTeX{} will maintain and display a ‘section number’ as a count named \texttt{\section@num} that counts how many \texttt{\begin{numbering}} and \texttt{\resumenumbering} commands have appeared; it needn’t be related to the logical divisions of your text.

\extensionchars
Each section will read and write an associated ‘line-list file’, containing information used to do the numbering; the file will be called \texttt{⟨jobname⟩.nn}, where \texttt{nn} is the section number. However, you may direct that an extra string be added before the \texttt{nn} in that filename, in order to distinguish these temporary files from others: that string is called \texttt{\extensionchars}. Initially it’s empty, since different operating systems have greatly varying ideas about what characters are permitted in filenames. So \texttt{\renewcommand{\extensionchars}{-}} gives temporary files called \texttt{jobname.-1, jobname.-2}, etc.

\begin{verbatim}
186 \newcount\section@num
187 \section@num=0
188 \let\extensionchars=\empty
\end{verbatim}

\ifnumbering
The \texttt{\ifnumbering} flag is set to \texttt{true} if we’re within a numbered section (that is, between \texttt{\begin{numbering}} and \texttt{\end{numbering}}). You can use \texttt{\ifnumbering} in your own code to check whether you’re in a numbered section, but don’t change the flag’s value.

\begin{verbatim}
189 \newif\ifnumbering
190 \ifnumbering\else\fi
\end{verbatim}

\begin{verbatim}
191 \newif\ifl@dpairing
192 \l@dpairingfalse
\end{verbatim}

\begin{verbatim}
193 \newif\ifl@dprintingpages
194 \l@dprintingpagesfalse
\end{verbatim}

\begin{verbatim}
195 \newif\ifl@dprintingcolumns
196 \l@dprintingcolumnsfalse
\end{verbatim}

\begin{verbatim}
197 \newif\ifpst@rteL
198 \l@rteLfalse
\end{verbatim}

\begin{verbatim}
199 \newif\ifledRcol
200 \l@edRcolfalse
\end{verbatim}

\ifledRcol is set to true in the \texttt{Rightside} \texttt{\environnement}. It must be distinguished from \texttt{\ifledRcol} which is set to true when a right line is processed, in \texttt{\Pages} or \texttt{\Columns}.

\begin{verbatim}
201 \newif\ifledRcol
202 \ifledRcolfalse
\end{verbatim}

In preparation for the \texttt{eledpar} package, these are related to the ‘left’ text of parallel texts (when \texttt{\ifl@dpairing} is \texttt{true}). They are explained in the \texttt{eledpar} manual.
The \ifnumberingR flag is set to true if we’re within a right text numbered section.

\beginnumbering \beginnumbering begins a section of numbered text. When it’s executed we increment the section number, initialize our counters, send a message to your terminal, and call macros to start the lineation machinery and endnote files.

The initializations here are trickier than they look. \line@list@stuff will use all of the counters that are zeroed here when it assembles the line-list and other lists of information about the lineation. But it will do all of this locally and within a group, and when it’s done the lists will remain but the counters will return to zero. Those same counters will then be used as we process the text of this section, but the assignments will be made globally. These initializations actually apply to both uses, though in all other respects there should be no direct interaction between the use of these counters and variables in the two processing steps.

For parallel processing:

- zero \l@dnumpstartsL — the number of chunks to be processed.
- set \ifpst@rtedL to FALSE.

The tools for section’s title commands are called:

- Define old (deprecated) sectioning commands.
- Define an empty list of pstart number where sectioning commands are called.
- Input auxiliary file with the description of section titles.
- Open the same auxiliary file to write in.

\begin{verbatim}
\beginnumbering*{\beginnumbering}\
\begin{group}
\beginnumbering@reg
\message{Section \the\section@num }
\line@list@stuff{\jobname.\extensionchars\the\section@num}
\l@dend@stuff
\setcounter{pstart}{1}
\ifl@dpairing
\global\l@dnumpstartsL \z@ \global\pst@rtedLfalse
\else
\begingroup
\beginnumbering@sectcmd
\ifwidthliketwocolumns%
\end{verbatim}
must follow the last text for a numbered section. It takes care of notifying you when changes have been noted in the input that require running the file through again to move everything to the right place.

\endnumbering  \endnumbering must follow the last text for a numbered section. It takes care of notifying you when changes have been noted in the input that require running the file through again to move everything to the right place.
The \pausenumbering macro is just the same as \endnumbering, but with the \ifnumbering flag set to true, to show that numbering continues across the gap.\footnote{Our thanks to Wayne Sullivan, who suggested the idea behind these macros.}

The \resumenumbering macro is a bit more involved, but not much. It does most of the same things as \beginnumbering, but without resetting the various counters. Note that no check is made by \resumenumbering to ensure that \pausenumbering was actually invoked.
21 Line counting

21.1 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. \eledmac can do it either way, and you can switch from one to the other within one work. But you have to choose one or the other for all line numbers and line references within each section. Here we will define internal codes for these systems and the macros you use to select them.

\ifbypage@ and \ifbypstart@ flag specify the current lineation system:

- line-of-page: \bypage@ = false and \bypage@ = true.
- line-of-pstart: \bypage@ = true and \bypage@ = false.

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

\lineation{⟨word⟩} is the macro you use to select the lineation system. Its argument is a string: either page or section or pstart.

\newcommand*{\lineation}[1]{%
\ifnumbering
\eled@err@LineationInNumbered
\else
\def\@tempa{#1}\def\@tempb{page}%
\ifx\@tempa\@tempb
\global\bypage@true
\global\bypstart@false
\pstartinfootnote]\false%
\else
\def\@tempb{pstart}%
\ifx\@tempa\@tempb
\global\bypage@false
\global\bypstart@true
\pstartinfootnote\false%
\else
\def\@tempb{section}%
\ifx\@tempa\@tempb
\global\bypage@false
\global\bypstart@false
\pstartinfootnote][false]
\else
\eled@warn@BadLineation
\fi
\fi
\fi
\fi}
21.1 Choosing the system of lineation

You call \texttt{\linenummargin\{\textit{word}\}} to specify which margin you want your line numbers in; it takes one argument, a string. You can put the line numbers in the same margin on every page using \texttt{left} or \texttt{right}; or you can use \texttt{inner} or \texttt{outer} to get them in the inner or outer margins. (These last two options assume that even-numbered pages will be on the left-hand side of every opening in your book.) 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.

The selection is recorded in the count \texttt{\line@margin}: 0 for left, 1 for right, 2 for outer, and 3 for inner.

\begin{verbatim}
\newcount\line@margin
\newcommand*{\linenummargin}[1]{% \l@dgetline@margin{#1}% \ifnum\@l@dtempcntb\m@ne \global\line@margin=\@l@dtempcntb \fi}% \newcommand{\l@dgetline@margin}[1]{% \def\@tempa{#1}\def\@tempb{left}% \ifx\@tempa\@tempb \@l@dtempcntb \z@ \else \def\@tempb{right}% \ifx\@tempa\@tempb \@l@dtempcntb \@ne \else \def\@tempb{outer}% \ifx\@tempa\@tempb \@l@dtempcntb \tw@ \else \def\@tempb{inner}% \ifx\@tempa\@tempb \@l@dtempcntb \thr@@ \else \led@warn@BadLinenummargin \@l@dtempcntb \m@ne \fi \fi \fi \fi}
\end{verbatim}

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

\begin{verbatim}
\newcounter{firstlinenum}
\setcounter{firstlinenum}{5}
\end{verbatim}
The following parameters are just like `firstlinenum` and `linenumincrement`, but for sub-line numbers. `sublinenumincrement` must be at least 1.

These macros can be used to set the corresponding counters.

When line locking is being used, the \texttt{\lockdisp{⟨word⟩}} macro specifies whether a line number—of one is due to appear—should be printed on the first printed line or on the last, or by all of them. Its argument is a word, either \texttt{first}, \texttt{last}, or \texttt{all}. Initially, it is set to \texttt{first}.

\lock@disp encodes the selection: 0 for first, 1 for last, 2 for all.
21.1 Choosing the system of lineation

The same questions about where to print the line number apply to sub-lines, and these are the analogous macros for dealing with the problem.

\newcount\subblock@disp
\newcommand{\subblockdisp}[1]{% \@dgetlock@disp{#1}\% \ifnum\@dtempcntb>\m@ne \global\subblock@disp=\@dtempcntb \else \led@warn@BadSublockdisp \fi}

We provide a mechanism for using different representations of the line numbers, not just the normal arabic.

\newcommand*{\linenumberstyle}[1]{% \def\linenumrep##1{\@nameuse{@#1}{##1}}} \newcommand*{\sublinenumberstyle}[1]{% \def\sublinenumrep##1{\@nameuse{@#1}{##1}}}

Initialise the number styles to arabic.

\linenumberstyle{arabic} \let\linenumr@p\linenumrep
\sublinenumberstyle{arabic} \let\sublinenumr@p\sublinenumrep

\leftlinenum \rightlinenum are the macros that are called to print marginal line numbers on a page, for left- and right-hand margins respectively. They’re made easy to access and change, since you may often want to change the styling in some way. These standard versions illustrate the general sort of thing that will be needed; they’re based on the \leftheadline macro in The \TeX\book, p. 416.

Whatever these macros output gets printed in a box that will be put into the appropriate margin without any space between it and the line of text. You’ll generally want a kern between a line number and the text, and \linenumsep is provided as a standard way of storing its size. Line numbers are usually printed in a smaller font, and \numlabfont is provided as a standard name for that font.

When called, these macros will be executed within a group, so font changes and the like will remain local.

\ledlinenum typesets the line (and subline) number.

The original \numlabfont specification is equivalent to the \LaTeX\ \scriptsize for a 10pt document.
21.2 List macros

Reminder: compare these with the \LaTeX{} list macros in case they would be suitable instead.

We will make heavy use of lists of information, which will be built up and taken apart by the following macros; they are adapted from The TeXbook, pp. 378–379, which discusses their use in more detail.

These macros consume a large amount of the run-time of this code. We intend to replace them in a future version, and in anticipation of doing so have defined their interface in such a way that it is not sensitive to details of the underlying code.

\list@create

The \list@create macro creates a new list. In this version of eledmac this macro doesn’t do anything beyond initializing an empty list macro, but in future versions it may do more.

\list@clear

The \list@clear macro just initializes a list to the empty list; in this version of eledmac it is no different from \list@create.

\xright@appenditem

\xright@appenditem expands an item and appends it to the right end of a list macro. We want the expansion because we’ll often be using this to store the current value of a counter. \xright@appenditem creates global control sequences, like \xdef, and uses two temporary token-list registers, \@toksa and \@toksb.

\global\led@toksa={}

\global\led@toksb={}\the\led@toksa\expandafter{#1}
21.3 Line-number counters and lists

\xleft@appenditem \xleft@appenditem expands an item and appends it to the left end of a list macro; it is otherwise identical to \xright@appenditem.

\gl@p The \gl@p macro removes the leftmost item from a list and places it in a control sequence. You say \gl@p\l\to\z (where \l is the list macro, and \z receives the left item). \l is assumed nonempty: say \ifx\l\empty to test for an empty \l. The control sequences created by \gl@p are all global.

21.3 Line-number counters and lists

Footnote references using line numbers rather than symbols can’t be generated in one pass, because we don’t know the line numbers till we ship out the pages. It would be possible if footnotes were never keyed to more than one line; but some footnotes gloss passages that may run for several lines, and they must be tied to the first line of the passage glossed. And even one-line passages require two passes if we want line-per-page numbering rather than line-per-section numbering.

So we run LATEX over the text several times, and each time save information about page and line numbers in a ‘line-list file’ to be used during the next pass. At the start of each section—whenever \beginnumbering is executed—the line-list file for that section is read, and the information from it is encoded into a few list macros.

We need first to define the different line numbers that are involved in these macros, and the associated counters.

\line@num The count \line@num stores the line number that’s used in marginal line numbering and in notes: counting either from the start of the page or from the start of the section, depending on your choice for this section. This may be qualified by \subline@num.

\subline@num The count \subline@num stores a sub-line number that qualifies \line@num. For example, line 10 might have sub-line numbers 1, 2 and 3, which might be printed as lines 10.1, 10.2, 10.3.

\ifsublines\sublines@true \sublines@false \sublines@false \sublines@false We maintain an associated flag, \ifsublines, to tell us whether we’re within a sub-line range or not.

You may wonder why we don’t just use the value of \subline@num to determine this—treating anything greater than 0 as an indication that sub-lineation is on. We need a separate flag because sub-lineation can be used together with line-number
locking in odd ways: several pieces of a logical line might be interrupted by pieces of sub-lineated text, and those sub-line numbers should not return to zero until the next change in the major line number. This is common in the typesetting of English Renaissance verse drama, in which stage directions are given sub-line numbers: a single line of verse may be interrupted by several stage directions.

457 \newif\ifsublines@ \absline@num  \line@num  \subline@num  The count \absline@num stores the absolute number of lines since the start of the section: that is, the number we’ve actually printed, no matter what numbers we attached to them. This value is never printed on an output page, though \line@num will often be equal to it. It is used internally to keep track of where notes are to appear and where new pages start: using this value rather than \line@num is a lot simpler, because it doesn’t depend on the lineation system in use.

458 \newcount\absline@num  

We’ll be calling \absline@num numbers ‘absolute’ numbers, and \line@num and \subline@num numbers ‘visible’ numbers.

\@lock  \sub@lock  The counts \@lock and \sub@lock tell us the state of line-number and sub-line-number locking. 0 means we’re not within a locked set of lines; 1 means we’re at the first line in the set; 2, at some intermediate line; and 3, at the last line.

459 \newcount\@lock  460 \newcount\sub@lock  

\line@list  \insertlines@list  \actionlines@list  \actions@list  Now we can define the list macros that will be created from the line-list file. We will maintain the following lists:

- \line@list: the page and line numbers for every lemma marked by \edtext. There are seven pieces of information, separated by vertical bars:

  1. the starting page,
  2. line, and
  3. sub-line numbers, followed by the
  4. ending page,
  5. line, and
  6. sub-line numbers, and then the
  7. font specifier for the lemma.

These line numbers are all visible numbers. The font specifier is a set of four codes for font encoding, family, series, and shape, separated by / characters. Thus a lemma that started on page 23, line 35 and went on until page 24, line 3 (with no sub-line numbering), and was typeset in a normal roman font would have a line list entry like this:

23|35|0|24|3|0|OT1/cmr/m/n.

There is one item in this list for every lemma marked by \edtext, even if there are several notes to that lemma, or no notes at all. \edtext reads the data in this list, making it available for use in the text of notes.
• \insertlines@list: the line numbers of lines that have footnotes or other insertions. These are the absolute numbers where the corresponding lemmas begin. This list contains one entry for every footnote in the section; one lemma may contribute no footnotes or many footnotes. This list is used by \add@inserts within \do@line, to tell it where to insert notes.

• \actionlines@list: a list of absolute line numbers at which we are to perform special actions; these actions are specified by the \actions@list list defined below.

• \actions@list: action codes corresponding to the line numbers in \actionlines@list. These codes tell eledmac what action it’s supposed to take at each of these lines. One action, the page-start action, is generated behind the scenes by eledmac itself; the others, for specifying sub-lineation, line-number locking, and line-number alteration, are generated only by explicit commands in your input file. The page-start and line-number-alteration actions require arguments, to specify the new values for the page or line numbers; instead of storing those arguments in another list, we have chosen the action-code values so that they can encode both the action and the argument in these cases. Action codes greater than $-1000$ are page-start actions, and the code value is the page number; action codes less than $-5000$ specify line numbers, and the code value is a transformed version of the line number; action codes between these two values specify other actions which require no argument.

Here is the full list of action codes and their meanings:

Any number greater than $-1000$ is a page-start action: the line number associated with it is the first line on a page, and the action number is the page number. (The cutoff of $-1000$ is chosen because negative page-number values are used by some macro packages; we assume that page-number values less than $-1000$ are not common.) Page-start action codes are added to the list by the \page@action macro, which is (indirectly) triggered by the workings of the \page@start macro; that macro should always be called in the output routine, just before the page contents are assembled. eledmac calls it in \pagecontents.

The action code $-1001$ specifies the start of sub-lineation: meaning that, starting with the next line, we should be advancing \subline@num at each start-of-line command, rather than \line@num.

The action code $-1002$ specifies the end of sub-lineation. At the next start-of-line, we should clear the sub-line counter and start advancing the line number. The action codes for starting and ending sub-lineation are added to the list by the \sub@action macro, as called to implement the \startsub and \endsub macros.

The action code $-1003$ specifies the start of line number locking. After the number for the current line is computed, it will remain at that value through the next line that has an action code to end locking.

The action code $-1004$ specifies the end of line number locking.
The action code \(-1005\) specifies the start of sub-line number locking. After
the number for the current sub-line is computed, it will remain at that value
through the next sub-line that has an action code to end locking.

The action code \(-1006\) specifies the end of sub-line number locking.

The four action codes for line and sub-line number locking are added to the
list by the \texttt{\do@lockon} and \texttt{\do@lockoff} macros, as called to implement the
\texttt{\startlock} and \texttt{\endlock} macros.

An action code of \(-5000\) or less sets the current visible line number (either
the line number or the sub-line number, whichever is currently being ad-
vanced) to a specific positive value. The value of the code is \(-(5000 + n),\)
where \(n\) is the value (always \(\geq 0\)) assigned to the current line number. Ac-
tion codes of this type are added to the list by the \texttt{\set@line@action}
macro, as called to implement the \texttt{\advanceline} and \texttt{\setline} macros: this action
only occurs when the user has specified some change to the line numbers
using those macros. Normally \texttt{eledmac} computes the visible line numbers
from the absolute line numbers with reference to the other action codes and
the settings they invoke; it doesn’t require an entry in the action-code list
for every line.

Here are the commands to create these lists:

\begin{verbatim}
\list@create{\line@list}
\list@create{\insertlines@list}
\list@create{\actionlines@list}
\list@create{\actions@list}
\end{verbatim}

We’ll need some counts while we read the line-list, for the page number and the
ending page, line, and sub-line numbers. Some of these will be used again later
on, when we are acting on the data in our list macros.

\begin{verbatim}
\newcount{\page@num}
\newcount{\endpage@num}
\newcount{\endline@num}
\newcount{\endsubline@num}
\end{verbatim}

\ifnoteschanged@ We’ll need some counts while we read the line-list, for the page number and the
\else ending page, line, and sub-line numbers. Some of these will be used again later
\fi on, when we are acting on the data in our list macros.

\begin{verbatim}
\newcount{\page@num}
\newcount{\endpage@num}
\newcount{\endline@num}
\newcount{\endsubline@num}
\end{verbatim}

If the number of the footnotes in a section is different from what it was during
the last run, or if this is the very first time you’ve run \texttt{\LaTeX}, on this file, the
information from the line-list used to place the notes will be wrong, and some
notes will probably be misplaced. When this happens, we prefer to give a single
error message for the whole section rather than messages at every point where we
notice the problem, because we don’t really know where in the section notes were
added or removed, and the solution in any case is simply to run \texttt{\LaTeX} two more
times; there’s no fix needed to the document. The \texttt{\ifnoteschanged@} flag is set
if such a change in the number of notes is discovered at any point.

\begin{verbatim}
\newif{\ifnoteschanged@}
\end{verbatim}
Inside the apparatus, at each note, the line number is stored in a macro called \resetprevline, where X is the letter of the current series. This macro is called when using \numberonlyfirstinline. This macro must be reset at the same time as the line number. The \resetprevline does this resetting for every series.

\resetprevline\newcommand*{\resetprevline}{%\def\do##1{\global\csundef{prevline##1}}%\dolistloop{\@series}%;}\resetprevline

Inside the apparatus, at each note, the page number is stored in a macro called \resetprevpage\num, where X is the letter of the current series. This macro is called when using \parafootsep. This macro must be reset at the beginning of each numbered section. The \resetprevpage\num command resets this macro for every series.

\resetprevpage\num\newcommand*{\resetprevpage\num}{%\def\do##1{\ifcsdef{prevpage##1\num}{\global\csname prevpage##1\num\endcsname=0}{}}%\dolistloop{\@series};}\resetprevpage\num

21.4 Reading the line-list file

\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.

When the file is there we start a new group and make some special definitions we’ll need to process it: it’s a sequence of \TeX\ commands, but they require a few special settings. We make [ and ] become grouping characters: they’re used that way in the line-list file, because we need to write them out one at a time rather than in balanced pairs, and it’s easier to just use something other than real braces. @ must become a letter, since this is run in the ordinary \TeX\ context. We ignore carriage returns, since if we’re in horizontal mode they can get interpreted as spaces to be printed.

Our line, page, and line-locking counters were already zeroed by \line@list@stuff if this is being called from within \beginnumbering; sub-lineation will be turned off as well in that case. On the other hand, if this is being called from \resumenumbering, those things should still have the values they had when \pausenumbering was executed.

If the file is not there, we print an informative message.

Now, after these preliminaries, we start interpreting the file.

\get\linelistfile{#1}
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.

```latex
\global\page@num=`m\one
\ifx\actionlines@list\empty
  \gdef\next@actionline{1000000}%
\else
  \gl@p\actionlines@list\to\next@actionline
  \gl@p\actions@list\to\next@action
\fi
```

This version of `\read@linelist` creates list macros containing data for the entire section, so they could get rather large. It would be no more difficult to read the line-list file incrementally rather than all at once: we could read, at the start of each paragraph, only the commands relating to that paragraph. But this would require that we have two line-lists open at once, one for reading, one for writing, and on systems without version numbers we’d have to do some file renaming outside of \LaTeX for that to work. We’ve retained this slower approach to avoid that sort of hacking about, but have provided the `\pausenumbering` and `\resumenumbering` macros to help you if you run into macro memory limitations (see p. 12 above).

\list@clearing@reg Clear the lists for `\read@linelist`.

```
\newcommand*{\list@clearing@reg}{% 
  \list@clear{\line@list}%
  \list@clear{\insertlines@list}%
  \list@clear{\actionlines@list}%
  \list@clear{\actions@list}%
  \list@clear{\sw@list}%
  \list@clear{\sw@list@inedtext}%
}%
```

\get@linelistfile \eledma\ can take advantage of the \LaTeX `safe file input’ macros to get the line-list file.

```
\newcommand*{\get@linelistfile}{% 
  \InputIfFileExists(#1){% 
    \global\noteschanged@false
    \begingroup
    \catcode'[^=1 \catcode'\^^M=9}{% 
    \led@warn@NoLineFile(#1)%
    \global\noteschanged@true
    \begingroup)%
  }%
}%
```

This version of `\read@linelist` creates list macros containing data for the entire section, so they could get rather large. It would be no more difficult to read the line-list file incrementally rather than all at once: we could read, at the start of each paragraph, only the commands relating to that paragraph. But this would require that we have two line-lists open at once, one for reading, one for writing, and on systems without version numbers we’d have to do some file renaming outside of \LaTeX for that to work. We’ve retained this slower approach to avoid that sort of hacking about, but have provided the `\pausenumbering` and `\resumenumbering` macros to help you if you run into macro memory limitations (see p. 12 above).
21.5 Commands within the line-list file

This section defines the commands that can appear within a line-list file. They all have very short names because we are likely to be writing very large numbers of them out. One macro, \@nl, is especially short, since it will be written to the line-list file once for every line of text in a numbered section. (Another of these commands, \@lab, will be introduced in a later section, among the cross-referencing commands it is associated with.)

When these commands modify the various page and line counters, they deliberately do not say \global. This is because we want them to affect only the counter values within the current group when nested calls of \@ref occur. (The code assumes throughout that the value of \globaldefs is zero.)

The macros with \texttt{action} in their names contain all the code that modifies the action-code list: again, this is so that they can be turned off easily for nested calls of \@ref.

\@nl \@nl does everything related to the start of a new line of numbered text.
\@nl@reg In order to get the \texttt{setlinenum} to work Peter Wilson had to slip in some new code at the start of the macro, to get the timing of the actions correct. The problem was that his original naive implementation of \texttt{setlinenum} had an unfortunate tendency to change the number of the last line of the \texttt{preceeding} paragraph. The new code is sort of based on the page number handling and \texttt{setline} It seems that a lot of fiddling with the line number internals is required.

In November 2004 in order to accurately determine page numbers Peter Wilson added these to the macro. It is now:
\@nl{⟨page counter number⟩}{⟨printed page number⟩}
I don’t (yet) use the printed number (i.e., the \texttt{thepage}) but it may come in handy later. The macro \texttt{fix@page} checks if a new page has started.

\begin{verbatim}
\newcommand{\@nl}[2]{% \
\fix@page{#1}% 
\@nl@reg}
\newcommand*{\@nl@reg}{% 
\ifx\l@dchset@num\relax \else 
\advance\absline@num \@ne 
\set@line@action 
\let\l@dchset@num=\relax 
\advance\line@num \m@ne 
\fi
\advance\absline@num \@ne 
\ifx\next@page@num\relax \else 
\page@action 
\let\next@page@num=\relax
\fi
\iftx\sub@change\relax \else 
\advance\absline@num \@ne
\iftx\next@page@num\relax
\page@action
\let\next@page@num=\relax
\fi
\fi
\end{verbatim}

First increment the absolute line-number, and perform deferred actions relating to page starts and sub-lines.
Fix the lock counters, if necessary. A value of 1 is advanced to 2; 3 advances
to 0; other values are unchanged.

\ifcase\@lock
  \or \@lock \tw@
  \or \or \@lock \z@
\fi
\ifcase\sub@lock
  \or \sub@lock \tw@
  \or \or \sub@lock \z@
\fi

Now advance the visible line number, unless it’s been locked.

\ifsublines@
  \ifnum\sub@lock<\tw@
  \advance\subline@num \one
\fi
\else
  \ifnum\@lock<\tw@
  \advance\line@num \one \subline@num \z@
\fi
\fi

\listxadd{\normal@page@break}{\the\absline@num}

\fix@page basically replaces \@page. It determines whether or not a new page
has been started, based on the page values held by \@nl.

\newcount\last@page@num
\last@page@num=-10000
\newcommand*{\fix@page}{\ifnum #1=\last@page@num
  \else
    \ifbypage@
      \line@num=\z@ \subline@num=\z@
    \fi
    \page@num=#1\relax
    \last@page@num=#1\relax
    \def\next@page@num{#1}\
    \listxadd{\normal@page@break}{\the\absline@num}
21.5 Commands within the line-list file

\@pend These don’t do anything at this point, but will have been added to the auxiliary file(s) if the \eledpar package has been used. They are just here to stop \eledmac from moaning if the \eledpar is used for one run and then not for the following one.
\@pendR
\@lopL
\@lopR
These don’t do anything at this point, but will have been added to the auxiliary file(s) if the \eledpar package has been used. They are just here to stop \eledmac from moaning if the \eledpar is used for one run and then not for the following one.
\newcommand*{\@pend}{1}{}
\newcommand*{\@pendR}{1}{}
\newcommand*{\@lopL}{1}{}
\newcommand*{\@lopR}{1}{}

\sub@on The \sub@on and \sub@off macros turn sub-lineation on and off: but not directly, since such changes don’t really take effect until the next line of text. Instead they set a flag that notifies \@nl of the necessary action.
\newcommand*{\sub@on}{\ifsublines@ \let\sub@change=\relax \else \def\sub@change{1}\fi}
\newcommand*{\sub@off}{\ifsublines@ \def\sub@change{-1}\else \let\sub@change=\relax \fi}
\@adv The \@adv{⟨num⟩} macro advances the current visible line number by the amount specified as its argument. This is used to implement \advanceline.
\newcommand*{\@adv}{1}{⟨ifsublines0 \let\sub@change=\relax \else \def\sub@change{1}\fi}{⟨ifsublines0 \let\sub@change=\relax \else \def\sub@change{-1}\fi}}{⟨ifsublines0 \let\sub@change=\relax \else \def\sub@change=\relax \fi}}
\@set The \@set{⟨num⟩} macro sets the current visible line number to the value specified as its argument. This is used to implement \setline.
\newcommand*{\@set}{1}{⟨ifsublines0 \set@line@action}
The \texttt{\l@d@set{(num)}} macro sets the line number for the next \texttt{\pstart...} to the value specified as its argument. This is used to implement \texttt{\setlinenum}. \texttt{\l@dchset@num} is a flag to the \texttt{\@l} macro. If it is not \texttt{\relax} then a linenumber change is to be done.

\newcommand*{\l@d@set}{[1]}{% 
  \line@num=#1\relax 
  \advance\line@num@ by 1 
  \def\l@dchset@num{#1}}

\let\l@dchset@num\relax

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

\newcommand*{\page@action}{% 
  \xright@appenditem{\the\absline@num}\to\actionlines@list 
  \xright@appenditem{\next@page@num}\to\actions@list}

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

\newcommand*{\set@line@action}{% 
  \xright@appenditem{\the\absline@num}\to\actionlines@list 
  \ifsublines@ 
    \xright@appenditem{-1001}\to\actions@list 
  \else 
    \xright@appenditem{-1002}\to\actions@list 
  \fi}

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

\newcommand*{\sub@action}{% 
  \xright@appenditem{\the\absline@num}\to\actionlines@list 
  \ifslines@ 
    \xright@appenditem{-1001}\to\actions@list 
  \else 
    \xright@appenditem{-1002}\to\actions@list 
  \fi}

\lock@on \ \texttt{\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.
Adding commands to the action list is slow, and it’s very often the case that a lock-on command is immediately followed by a lock-off command in the line-list file, and therefore really does nothing. We use a look-ahead scheme here to detect such pairs, and add nothing to the line-list in those cases.

\newcommand*{\lock@on}{\futurelet\next\do@lockon}
\newcommand*{\do@lockon}{
  \ifx\next\lock@off
    \global\let\lock@off=\skip@lockoff
  \else
    \do@lockonL
  \fi}
\newcommand*{\do@lockonL}{
  \xright@appenditem{\the\absline@num}\to\actionlines@list
  \ifsublines@
    \xright@appenditem{-1005}\to\actions@list
    \ifnum\sub@lock=\z@
      \sub@lock \@ne
    \else
      \ifnum\sub@lock=\thr@@
        \sub@lock \@ne
      \fi
    \fi
  \else
    \xright@appenditem{-1003}\to\actions@list
    \ifnum\@lock=\tw@
      \@lock \thr@@
    \else
      \sub@lock \z@
    \fi
  \fi
  \fi}
\lock@off \lock@off adds an entry to the action-code list to turn line number locking off.
\newcommand*{\do@lockoff}{%}
\do@lockoffL \newcommand*{\do@lockoffL}{%}
\skip@lockoff \newcommand*{\skip@lockoff}{%}
\skip@lockoffL \newcommand*{\skip@lockoffL}{%}
\newcommand*{\do@lockoff}{%}
\newcommand*{\do@lockoffL}{%}
\newcommand*{\skip@lockoff}{%}
\newcommand*{\skip@lockoffL}{%}
\newcommand*{\lock@off}{%}
\lock@off adds an entry to the action-code list to turn line number locking off.
This macro implements the `\skipnumbering` command. It uses a new action code, namely 1007. This value, here and within `\edtext`, which computes it and writes it to the line-list file, will be stored in the count `\insert@count`.

- #1, the number of entries to add to `\insertlines@list` for this reference. The first thing `\@ref` (i.e. `\@ref@reg`) itself does is to add the specified number of items to the `\insertlines@list` list.

- #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`.)

When nesting of `\@ref` commands does occur, it’s necessary to temporarily redefine `\@ref` within `\@ref`, so that we’re only doing one of these at a time.

The first thing `\@ref` (i.e. `\@ref@reg`) itself does is to add the specified number of items to the `\insertlines@list` list. 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.
21.6 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 \eledmac uses within the text of a section to write commands out to the line-list.

The file will be opened on output stream \linenum@out.

Once any file is opened on this stream, we keep it open forever, or else switch to another file that we keep open. The reason is that we want the output routine to write the page number for every page to this file; otherwise we’d have to write it at the start of every line. But it’s not very easy for the output routine to tell whether an output stream is open or not. There’s no way to test the status of a particular output stream directly, and the asynchronous nature of output routines makes the status hard to determine by other means.

We can manage pretty well by means of the \iffirst@linenum@out@ flag; its inelegant name suggests the nature of the problem that made its creation necessary. It’s set to be true before any \linenum@out file is opened. When such a file is opened for the first time, it’s done using \immediate, so that it will at once be safe for the output routine to write to it; we then set this flag to false.
The \line@list@stuff{⟨file⟩} macro, which is called by \beginnumbering, performs all the line-list operations needed at the start of a section. Its argument is the name of the line-list file.

First, use the commands of the previous section to interpret the line-list file from the last run.

Now close the current output line-list file, if any, and open a new one. The first time we open a line-list file for output, we do it using \immediate, and clear the \iffirst@linenum@out@ flag.

If we get here, then this is not the first line-list we’ve seen, so we don’t open or close the files immediately.

The \new@line macro sends the \@nl command to the line-list file, to mark the start of a new text line, and its page number.
21.6 Writing to the line-list file

\ifnoneedFootnote

\ifnoneedFootnote is a boolean to check if we have to print a error message when a \texttt{\edtext} is called without any footnotes.

\flagstart \flagend We enclose a lemma marked by \texttt{\edtext} in \texttt{\flagstart} and \texttt{\flagend}: these send the \texttt{\@ref} command to the line-list file. \texttt{\edtext} is responsible for setting the value of \texttt{\insertcount} appropriately; it actually gets done by the various footnote macros.

\newif\ifnoneedFootnote
\newcommand*{\flagstart}{\ifledRcol
\edef\next{\writeline{
\ifnum\insertcount<1
\ifnoneedFootnote
\Led@err@EdtextWithoutFootnote
\fi
\fi}
\edef\next{\writeline{
\ifnum\insertcount<1
\ifnoneedFootnote
\Led@err@EdtextWithoutFootnote
\fi
\fi}

\pagestart

Originally the commentary was: \texttt{\pagestart} writes a command to the line-list file noting the current page number; when used within an output routine, this should be called so as to place its \texttt{\write} within the box that gets shipped out, and as close to the top of that box as possible.

However, in October 2004 Alexej Krukov discovered that when processing long paragraphs that included Russian, Greek and Latin texts \texttt{eledmac} would go into an infinite loop, emitting thousands of blank pages. This was caused by being unable to find an appropriate place in the output routine. A different algorithm is now used for getting page numbers.
\newcommand*{\page@start}\{}

\startsub \starts and \endsub turn sub-lineation on and off, by writing appropriate instructions to the line-list file. When sub-lineation is in effect, the line number counter is frozen and the sub-line counter advances instead. If one of these commands appears in the middle of a line, it doesn’t take effect until the next line; in other words, a line is counted as a line or sub-line depending on what it started out as, even if that changes in the middle.

We tinker with \lastskip because a command of either sort really needs to be attached to the last word preceding the change, not the first word that follows the change. This is because sub-lineation will often turn on and off in mid-line—stage directions, for example, often are mixed with dialogue in that way—and when a line is mixed we want to label it using the system that was in effect at its start. But when sub-lineation begins at the very start of a line we have a problem, if we don’t put in this code.

\newcommand*{\startsub}{\dimen0\lastskip}
\ifdim\dimen0>0pt \unskip \fi
\write\linenum@out{\string\sub@on}\
\ifdim\dimen0>0pt \hskip\dimen0 \fi}
\def\endsub{\dimen0\lastskip
\ifdim\dimen0>0pt \unskip \fi
\write\linenum@out{\string\sub@off}\
\ifdim\dimen0>0pt \hskip\dimen0 \fi}
\advanceline You can use \advanceline{⟨num⟩} in running text to advance the current visible line-number by a specified value, positive or negative.
\newcommand*{\advanceline}[1]{\write\linenum@out{\string\@adv[#1]}}
\setline You can use \setline{⟨num⟩} in running text (i.e., within \pstart...\pend) to set the current visible line-number to a specified positive value.
\newcommand*{\setline}[1]{\ifnum#1<\z@ \led@warn@BadSetline
\else \write\linenum@out{\string\@set[#1]}\fi}
\setlinenum You can use \setlinenum{⟨num⟩} before a \pstart to set the visible line-number to a specified positive value. It writes a \l@d@set command to the line-list file.
\newcommand*{\setlinenum}[1]{\ifnum#1<\z@ \led@warn@BadSetlinenum
\else \write\linenum@out{\string\l@d@set[#1]}\fi}
You can use `\startlock` or `\endlock` in running text to start or end line number locking at the current line. They decide whether line numbers or sub-line numbers are affected, depending on the current state of the sub-lineation flags.

\begin{verbatim}
\ifl@dskipnumber
  \l@dskipnumbertrue
\else
  \l@dskipnumberfalse
\fi
\skipnumbering
\newcommand*{\skipnumbering}{\skipnumbering@reg}
\newcommand*{\skipnumbering@reg}{% \\
  \write\linenum@out{\string\n@num}\
  \advanceline{-1}}
\end{verbatim}

In numbered text `\skipnumbering` will suspend the numbering for that particular line.

22 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.

For convenience, I will use `\*text` when I do not need to distinguish between `\edtext` and `\critext`. The `\*text` macros take two arguments, the only difference between `\edtext` and `\critext` is how the second argument is delineated.

`\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}`

- **#1** is the piece of the main text being glossed; it gets added to the main text, and is also used as a lemma for notes to it.
- **#2** is a series of subsidiary macros that generate various kinds of notes. With `\critext` the `/ after #2 must` appear: it marks the end of the macro. (*The TeXbook*, p. 204, points out that when additional text to be matched follows the arguments like this, spaces following the macro are not skipped, which is very desirable since this macro will never be used except within text. Having an explicit terminator also helps keep things straight when nested calls to `\critext` are used.) Braces around **#2** are optional with `\critext` and required for `\edtext`. 
The \*text macro may be used (somewhat) recursively; that is, \*text may be used within its own first argument. The code would be much simpler without this feature, but nested notes will commonly be necessary: it’s quite likely that we’ll have an explanatory note for a long passage and notes on variants for individual words within that passage. The situation we can’t handle is overlapping notes that aren’t nested: for example, one note covering lines 10–15, and another covering 12–18. You can handle such cases by using the \lemma and \linenum macros within #2: they alter the copy of the lemma and the line numbers that are passed to the notes, and hence allow you to overcome any limitations of this system, albeit with extra effort.

The recursive operation of \*text will fail if you try to use a copy that is called something other than \*text. In order to handle recursion, \*text needs to redefine its own definition temporarily at one point, and that doesn’t work if the macro you are calling is not actually named \*text. There’s no problem as long as \*text is not invoked in the first argument. If you want to call \*text something else, it is best to create instead a macro that expands to an invocation of \*text, rather than copying \*text and giving it a new name; otherwise you will need to add an appropriate definition for your new macro to \morenoexpands.

Side effects of our line-numbering code make it impossible to use the usual footnote macros directly within a paragraph whose lines are numbered (see comments to \do@line, p. 92). Instead, the appropriate note-generating command is appended to the list macro \inserts@list, and when \pend completes the paragraph it inserts all the notes at the proper places.

Note that we don’t provide previous-note information, although it’s often wanted; your own macros must handle that. We can’t do it correctly without keeping track of what kind of notes have gone past: it’s not just a matter of remembering the line numbers associated with the previous invocation of \*text, because that might have been for a different kind of note. It is preferable for your footnote macros to store and recall this kind of information if they need it.

### 22.1 \edtext (and \critext) itself

The various note-generating macros might want to request that commands be executed not at once, but in close connection with the start or end of the lemma. For example, footnote numbers in the text should be connected to the end of the lemma; or, instead of a single macro to create a note listing variants, you might want to use several macros in series to create individual variants, which would each add information to a private macro or token register, which in turn would be formatted and output when all of #2 for the lemma has been read.

To accomodate this, we provide a list macro to which macros may add commands that should subsequently be executed at the end of the lemma when that lemma is added to the text of the paragraph. A macro should add its contribution to \end@lemmas by using \xleft@appenditem. (Anything that needs to be done at the start of the lemma may be handled using \aftergroup, since the commands
specified within \edtext’s second argument are executed within a group that ends just before the lemma is added to the main text.)

\end@lemmas is intended for the few things that need to be associated with the end of the lemma, like footnote numbers. Such numbers are not implemented in the current version, and indeed no use is currently made of \end@lemmas or of the \aftergroup trick. The general approach would be to define a macro to be used within the second argument of \edtext that would add the appropriate command to \end@lemmas.

Commands that are added to this list should always take care not to do anything that adds possible line-breaks to the output; otherwise line numbering could be thrown off.

\list@create{\end@lemmas}

\dummy@text We now need to define a number of macros that allow us to weed out nested instances of \edtext, and other problematic macros, from our lemma. This is similar to what we did in reading the line-list file using \dummy@ref and various redefinitions—and that’s because nested \edtext macros create nested \@ref entries in the line-list file.

Here’s a macro that takes the same arguments as \critext but merely returns the first argument and ignores the second.

\long\def\dummy@text#1#2/{#1}

\dummy@edtext\Latex users are not used to delimited arguments, so we provide a \edtext macro as well.

\newcommand{\dummy@edtext}[2]{#1}

\dummy@edtext@showlemma Some time, we want to obtain only the first argument of \edtext, while also wrapping it in \showlemma. For example, when printing a \eledsection.

\newcommand{\dummy@edtext@showlemma}[2]{\showlemma{#1}}%

\no@expands We’re going to need another macro that takes one argument and ignores it entirely. This is supplied by the \LaTeX \@gobble{⟨arg⟩}.

\morenoexpands We need to turn off macro expansion for certain sorts of macros we’re likely to see within the lemma and within the notes.

The first class is font-changing macros. We suppress expansion for them by letting them become equal to zero.\footnote{Since ‘control sequences equivalent to characters are not expandable’—\textit{The \TeX \book}, answer to Exercise 20.14.} This is done because we want to pass into our notes the generic commands to change to roman or whatever, and not their expansions that will ask for a particular style at a specified size. The notes may well be in a smaller font, so the command should be expanded later, when the note’s environment is in effect.

A second sort to turn off includes a few of the accent macros. Most are not a problem: an accent that’s expanded to an \accent command may be harder to read but it works just the same. The ones that cause problems are: those that use...
alignments—\TeX\ seems to get confused about the difference between alignment
parameters and macro parameters; those that use temporary control sequences;
and those that look carefully at what the current font is.

(The \copyright macro defined in Plain \TeX\ has this sort of problem as
well, but isn’t used enough to bother with. That macro, and any other that
causes trouble, will get by all right if you put a \protect in front of it in your
file.)

We also need to eliminate all \texttt{eledmac} macros like \texttt{edlabel} and \texttt{setline}
that write things to auxiliary files: that writing should be done only once. And
we make \texttt{edtext} itself, if it appears within its own argument, do nothing but
copy its first argument.

Finally, we execute \texttt{morenoexpands}. The version of \texttt{morenoexpands} defined
here does nothing; but you may define a version of your own when you need to add
more expansion suppressions as needed with your macros. That makes it possible
to make such additions without needing to copy or modify the standard \texttt{eledmac}
code. If you define your own \texttt{morenoexpands}, you must be very careful about
spaces: if the macro adds any spaces to the text when it runs, extra space will
appear in the main text when \texttt{edtext} is used.

(A related problem, not addressed by these two macros, is that of characters
whose category code is changed by any of the macros used in the arguments to
\texttt{edtext}. Since the category codes are set when the arguments are scanned, macros
that depend on changing them will not work. We have most often encountered
this with characters that are made ‘active’ within text in some, but not all, of the
languages used within the document. One way around the problem, if it takes
this form, is to ensure that those characters are \texttt{always} active; within languages
that make no special use of them, their associated control sequences should simply
return the proper character.)

\begin{macrocode}
\newcommand*{\no@expands}{%
\let\select@@lemmafont=0%
\let\startsub=\relax \let\endsub=\relax
\let\startlock=\relax \let\endlock=\relax
\let\edlabel=\@gobble
\let\setline=\@gobble \let\advanceline=\@gobble
\let\critext=\dummy@text
\let\sameword\sameword@inedtext%
\let\edtext=\dummy@edtext
\l@dtabnoexpands
\morenoexpands}
\let\morenoexpands=\relax
\end{macro}
% \begin{macro}{\if@edtext@}
\end{macrocode}
\edtext (and \critext) itself

This boolean is set to TRUE inside a \cs{edtext} (or \cs{critext}). That is useful for some commands which can have a different behavior if called inside or outside of \marg{lemma}.

\begin{macrocode}
\newif\if@edtext@
\end{macrocode}

Now we begin \critext itself. The definition requires a / after the arguments: this eliminates the possibility of problems about knowing where \#2 ends. This also changes the handling of spaces following an invocation of the macro: normally such spaces are skipped, but in this case they’re significant because \#2 is a ‘delimited parameter’. Since \critext is always used in running text, it seems more appropriate to pay attention to spaces than to skip them.

Since v.1.17.0, \critext only refers to \edtext.

\begin{macrocode}
\long\def\critext#1#2/{\edtext{#1}{#2}}\end{macrocode}

When executed, \edtext first ensures that we’re in horizontal mode.

\begin{macrocode}
\newcommand{\edtext}[2]{\leavevmode\@edtext@true\@tag

## Our normal lemma is just argument \#1; but that argument could have further invocations of \edtext within it. We get a copy of the lemma without any \edtext macros within it by temporarily redefining \edtext to just copy its first argument and ignore the other, and then expand \#1 into \@tag, our lemma.

This is done within a group that starts here, in order to get the original \edtext restored; within this group we’ve also turned off the expansion of those control sequences commonly found within text that can cause trouble for us.

## Prepare more data for the benefit of note-generating macros: the line references and font specifier for this lemma go to \l@d@nums.

## \insert@count will be altered by the note-generating macros: it counts the number of deferred footnotes or other insertions generated by this instance of \edtext. If we are in a right column (eledpar), we use \insert@countR instead of \insert@count.

Now process the note-generating macros in argument \#2 (i.e., \Afootnote, \lemma, etc.). \ignorespaces is here to skip over any spaces that might appear at the start of \#2; otherwise they wind up in the main text. Footnote and
other macros that are used within \#2 should all end with \ignorespaces as well, to skip any spaces between macros when several are used in series.

Finally, we’re ready to admit the first argument into the current paragraph.

It’s important that we generate and output all the notes for this chunk of text before putting the text into the paragraph: notes that are referenced by line number should generally be tied to the start of the passage they gloss, not the end. That should all be done within the expansion of \#2 above, or in \aftergroup commands within that expansion.

\@ifundefined{xpg@main@language}{{%if not polyglossia
\flag@start}\%}{%if@RTL\flag@end\else\flag@start\fi% With polyglossia, you must track whether
\%}\}
\endgroup%\%\%\showlemma{#1}%

Finally, we add any insertions that are associated with the end of the lemma. Footnotes that are identified by symbols rather than by where the lemma begins in the main text need to be done here, and not above.

\ifx\end@lemmas\empty\else%\gl@p\end@lemmas\to\x@lemma%\x@lemma%\global\let\x@lemma=\relax%
\fi%\@ifundefined{xpg@main@language}{{%if not polyglossia
\flag@end}\%}{%if@RTL\flag@start\else\flag@end\fi% With polyglossia, you must track whether
%}%
\global\@noneed@Footnotefalse%\@edtext@false%\global\@lemmacommand@false%

If we are outside of a numbered paragraph, we send error message and print the first argument.

\else%\showlemma{#1} ({\textbf{\textsc{Edtext outside numbered paragraph}}})\led@err@edtextoutsidepstart%
\fi%}
\fi%}
\newcommand*{\flag@end}{%
22.2 Substitute lemma

The \texttt{numberline} option can be set to FALSE to disable line numbering.

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

One instance of \texttt{critext} may generate several notes, or it may generate none—it’s legitimate for argument \texttt{#2} to \texttt{critext} to be empty. But \texttt{flag@start} and \texttt{flag@end} induce the generation of a single entry in \texttt{line@list} during the next run, and it’s vital to also remove one and only one \texttt{line@list} entry here.

If no more lines are listed in \texttt{line@list}, something’s wrong—probably just some change in the input. We set all the numbers to zeros, following an old publishing convention for numerical references that haven’t yet been resolved.

The macro \texttt{edfont@info} returns coded information about the current font.

22.2 Substitute lemma

The \texttt{lemma}\{\textit{text}\} macro allows you to change the lemma that’s passed on to the notes.

This boolean is set to TRUE inside a \texttt{edtext} (or \texttt{critext}) when a \texttt{lemma} command is called. That is useful for some commands which can have a different behavior if the lemma in the note is different from the lemma in the main text.
22.3 Substitute line numbers

\linenum The \linenum macro can change any or all of the page and line numbers that are passed on to the notes.

As argument \linenum takes a set of seven parameters separated by vertical bars, in the format used internally for \l@d@nums (see p. [52]: 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. However, you can omit any parameters you don’t want to change, and you can omit a string of vertical bars at the end of the argument. Hence \linenum{18\mid4\mid0\mid18\mid7\mid1\mid0} is an invocation that changes all the parameters, but \linenum{13} only changes the starting line number, and leaves the rest unaltered.

We use \ as an internal separator for the macro parameters.

\newcommand*{\linenum}{%\xdef\@tempa{#1|||||||\protect\l@d@nums}%\global\let\l@d@nums=\empty\expandafter\line@set\@tempa|\ignorespaces}

\line@set \linenum calls \line@set to do the actual work; it looks at the first number in the argument to \linenum, sets the corresponding value in \l@d@nums, and then calls itself to process the next number in the \linenum argument, if there are more numbers in \l@d@nums to process.

\def\line@set#1|#2\#3|#4\#5\#6\#7\{\gdef\l@d@nums#1\expandafter\line@set#2\#3\#4\#5\#6\#7}

\l@d@add \line@set uses \l@d@add to tack numbers or vertical bars onto the right hand end of \l@d@nums.

\newcommand{\l@d@add}[1]{\xdef\l@d@nums{\l@d@nums#1}}

22.4 Lemma disambiguation

The mechanism which counts the occurrence of a same word in a same line is quite complex, because, when \TeX reads a command between a \pstart and a \pend, it does not know yet which are the line numbers.

The general mechanism is the following:

- At the first run, each \sameword command increments an etoolbox counter the name of which contains the argument of the \sameword commands.
22.4 Lemma disambiguation

• Then this counter, associated with the argument of \texttt{sameword} is stored (\texttt{@sw} command) in the auxiliary file of the current \texttt{eledmac} section (the .1, .2... file).

• \textbf{When this auxiliary file is read at the second run}, different operations are achieved:
  
  – For each paired \texttt{sameword} argument and absolute line number, a counter is defined. Its value corresponds to the number of times \texttt{sameword{argument}} is called from the beginning of the lineation to the end of the current line. We also store the same data for the preceding absolute line number, if it does not have \texttt{sameword{argument}}.

  – A \texttt{sw@list} list is filled with the values stored in the auxiliary file. But before doing this we transform these values: we subtract from them the number stored for the paired \texttt{sameword} argument and previous absolute line number.

• At the second run, when the \texttt{sameword} command is called, new operations happen. We first read the first element of the \texttt{sw@list}, then delete it from this list, and, if we are inside a \texttt{edtext} command, we store it in a \texttt{sw@list@inedtext} list.

• At the second run, when the critical notes are built, the \texttt{sameword@inedtext} command is used instead of \texttt{sameword}. Then, we read the next value of \texttt{sw@list@inedtext} list and remove it from this list. We send it to the \texttt{showwordrank} to be printed after the lemma, but only if the current line has more than one value for the argument of \texttt{sameword}. Otherwise, we just print the lemma, with no number.

\texttt{\sw@list} So, first, the lists.
\texttt{\sw@list@inedtext} 923 \texttt{list@create{sw@list}}% 924 \texttt{list@create{sw@list@inedtext}}%

\texttt{\sameword} The high level macro \texttt{sameword}, used by the editor.
925 \texttt{newcommandx{\sameword}[2][1,usedefault]{\leavevmode%}
  First, increment the counter corresponding to the argument.
926 \texttt{unless\ifledRcol%\csnumgdef{sw@#2}{\csuse{sw@#2}+1}%
  Then, write its value to the numbered file
928 \texttt{protected@write\linenum@out{}{\string@sw{#2}{\csuse{sw@#2}}}}%
  At the second run, read the \texttt{sw@list} next item and, if we are in \texttt{edtext}, put it to \texttt{sw@list@inedtext}.
929 \texttt{unless\ifx\sw@list\empty%}
930 \texttt{gl@p\sw@list@to@tempb%}
931 \texttt{if@edtext0%}
932 \texttt{unless\if@lemmacommand0%}
First, define a counter which stores the second argument as value for each paired absolute line number/first argument.

Then, calculate the position of the word in the line, and put it in `\sw@list`.

And do the same thing for the right side.
22.4 Lemma disambiguation

The command called when \sameword is called in an edtext.
\def\sameword@inedtext#1{%}
  \unless\ifledRcol%
  \iffalse\edef\sw@listR\empty%
  \else\edef\sw@listR\sw@list@inedtext{}%
  \fi
  \edef\sw@list@inedtext\empty%
  \def\the@sw{999}%
  \else
  \edef\sw@listR\sw@list@inedtext{}%
  \ifx\sw@listR\empty%
    \def\the@sw{999}%
  \else
    \global\sw@list@inedtext\the@sw%
  \fi
  \fi
  \ifnum\sw@atthisline>1%
    \showwordrank{#1}{\the@sw}%
  \else
    \fi
  \fi

Then, print the rank, but only if there is more than one occurrence of the word in the current line.
\ifnum\sw@atthisline>1%
  \showwordrank{#1}{\the@sw}%
\else
  \fi

And the same for right side.
\else
  \iffalse\edef\sw@listR\empty%
  \else\edef\sw@listR\sw@list@inedtext{}%
  \fi
  \edef\sw@list@inedtext\empty%
  \def\the@sw{999}%
  \else
    \edef\sw@listR\sw@list@inedtext{}%
    \ifx\sw@listR\empty%
      \def\the@sw{999}%
    \else
      \global\sw@list@inedtext\the@sw%
    \fi
    \ifnum\sw@atthisline>1%
      \showwordrank{#1}{\the@sw}%
    \else
      \fi
    \fi
  \fi
\fi
23 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.

23.1 Boxes, counters, \pstart and \pend

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, \raw@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 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 register, and \par@line will be the number of that line within the paragraph.

\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 \raw@text box. \pstart needs to appear at the start of every paragraph that’s to be numbered; the \autopar command below may be used to insert these commands automatically.

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.
\newcounter{pstart}
\renewcommand{\the\pstart}{\bfseries\@arabic\c@pstart}. }
\newif\ifnumberpstart
\numberpstartfalse
\newif\iflabelpstart
\labelpstartfalse
\newcommandx*{\pstart}*{1}[1][]{% 
  \ifstrempty{#1}{\at@every\pstart}{\noindent#1}%%
  \ifluatex%
  \edef\l@luatextextdir{\the\luatextextdir}\% 
  \fi%
  \if@nobreak%
  \let\@oldnobreak\@nobreaktrue%
  \else%
  \let\@oldnobreak\@nobreakfalse%
  \fi%
  \@nobreaktrue%
  \ifnumbering
  \else%
  \led@err@PstartNotNumbered%
  \beginnumbering%
  \fi%
  \ifnumberedpar%
  \led@err@PstartInPstart%
  \ pend%
  \fi%
  \fi%
  \list@clear{\inserts@list}%%
  \global\let\next@insert=\empty%
  \begingroup\pagebreak=alse%
  \global\advance \l@dnumpstarts\@ne
  \global\setbox\raw@text\vbox{
    \ifautopar%
      \ifnumberpstart
        \ifinstanza
          \ifsidepstartnum
            \the\pstart
          \fi
        \fi
      \fi
    \fi
    \fi%
    \numberedpar=true
    \ifnumbering
      \else%
      \led@err@PendNumbered%
      \fi%
  \fi%
  \fi%
  \fi%
  \fi%
  \fi%
  \fi%
  \fi%
  \fi%
  \fi%
  \fi%
  \fi%
  \fi%
  \fi%
  \fi%
  \fi%
  \fi%
  \fi%

\pstart \ pend must be used to end a numbered paragraph.
\newcommandx*{\pend}*{1}[1][]{\ifnumbering
  \else%
  \led@err@PendNotNumbered%
  \fi%
We set all the usual interline penalties to zero and then immediately call `\endgraf` to end the paragraph; this ensures that there'll be no large interline penalties to prevent us from slicing the paragraph into pieces. These penalties revert to the values that you set when the group for the `\vbox` ends. Then we call `\do@line` to slice a line off the top of the paragraph, add a line number and footnotes, and restore it to the page; we keep doing this until there aren’t any more lines left.

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

We check if lineation is by pstart: in this case, we reset line number, but only in the second line of the pstart, to prevent some trouble. We can’t reset line number at the beginning of `\pstart \setline` is parsed at the end of previous `\pend`, and so, we must do it at the end of first line of pstart.

```
\csnumdef{pstartline}{0}%
\loop\ifvbox\raw@text%
\csnumdef{pstartline}{\pstartline+1}%
\do@line%
\ifbypstart@%
\ifnumequal{\pstartline}{1}{\setline{1}\resetprevline@}{}%
\fi%
\repeat%
```

Deal with any leftover notes, and then end the group that was begun in the `\pstart`.

```
\flush@notes%
\endgroup%
\ignorespaces%
\ifnumberpstart%
\pstartnumtrue%
\fi%
\@oldnobreak%
\addtocounter{pstart}{1}%
\ifstrempty{#1}{\at@every@pend}{\noindent#1}%
```

```
\AtEveryPend
\at@every@pend{\xdef\at@every@pend{\noindent\unexpanded{#1}}}%
```

```
\l@dzeropenalties  A macro to zero penalties for `\pend`.
\newcommand*{\l@dzeropenalties}{%
In most cases it’s only an annoyance to have to label the paragraphs to be numbered with \pstart and \pend. \autopar will do that automatically, allowing you to start a paragraph with its first word and no other preliminaries, and to end it with a blank line or a \par command. The command should be issued within a group, after \beginnumbering has been used to start the numbering; all paragraphs within the group will be affected.

A few situations can cause problems. One is a paragraph that begins with a begin-group character or command: \pstart will not get invoked until after such a group beginning is processed; as a result the character that ends the group will be mistaken for the end of the \vbox that \pstart creates, and the rest of the paragraph will not be numbered. Such paragraphs need to be started explicitly using \indent, \noindent, or \leavevmode—or \pstart, since you can still include your own \pstart and \pend commands even with \autopar on.

Prematurely ending the group within which \autopar is in effect will cause a similar problem. You must either leave a blank line or use \par to end the last paragraph before you end the group.

The functioning of this macro is more tricky than the usual \everypar: we don’t want anything to go onto the vertical list at all, so we have to end the paragraph, erase any evidence that it ever existed, and start it again using \pstart. We remove the paragraph-indentation box using \lastbox and save the width, and then skip backwards over the \parskip that’s been added for this paragraph.

Then we start again with \pstart, restoring the indentation that we saved, and locally change \par so that it’ll do our \pend for us.
We also define a macro which we can rely on to turn off the \autopar definitions at various important places, if they are in force. We’ll want to do this within a footnotes, for example.

\newcommand*{\normal@pars}{\everypar{}\let\par\endgraf}

We define a boolean test switched to true at the beginning of the \pausenumbering command if the autopar is enabled. This boolean will be tested at the beginning of \resumenumbering to continue the autopar if needed.

\newif{\ifautopar@pause}

### 23.2 Processing one line

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

\newcommand*{\l@dunhbox@line}[1]{\unhbox #1}
\newcommand*{\do@line}{\vbadness=10000\splittopskip=\z@\l@demptyd@ta\global\setbox\one@line=\vsplit\raw@text to\baselineskip}
\unvbox\one@line \global\setbox\one@line=\lastbox\getline@num\IfStrEq{\led@pb@setting}{before}{\led@check@pb\led@check@nopb}{\inserthangingsymboltrue\check@pb@in@verse\affixline@num}%
\ifnum\@lock>\@ne\inserthangingsymboltrue\else\inserthangingsymbolfalse\fi\check@pb@in@verse\affixline@num\xifinlist{\the\l@dnumpstartsL}{\eled@sections@@}{\print@eledsection}{\print@line}{\IfStrEq{\led@pb@setting}{after}{\led@check@pb\led@check@nopb}{\print@line}}\print@line\print@line

\def\print@line{\check@pb@innverse\affixline@num\xifinlist{\the\l@dnumpstartsL}{\eled@sections@@}{\print@eledsection}{\print@line}{\IfStrEq{\led@pb@setting}{after}{\led@check@pb\led@check@nopb}{}}}

\print@line is for normal line, i. e line without sectioning command.
\def\print@line{\affixpstart@num\xifinlist{\the\l@dnumpstartsL}{\eled@sections@@}{\print@eledsection}{\print@line}{\IfStrEq{\led@pb@setting}{after}{\led@check@pb\led@check@nopb}{}}}

Insert the pstart number in side, if we are in the first line of a pstart.

\affixpstart@num\xifinlist{\the\l@dnumpstartsL}{\eled@sections@@}{\print@eledsection}{\print@line}{\IfStrEq{\led@pb@setting}{after}{\led@check@pb\led@check@nopb}{}}

The line will be boxed, to have the good width.
23.2 Processing one line

User hook.
1156  \do@insidelinehook%

Left line number
1157  \l@dld@ta%

Restore marginal and footnotes.
1158  \add@inserts\affixside@note%

Print left notes.
1159  \l@dlsn@te

Boxes the line, writes information about new line in the numbered file.
1160  {\ledllfill\hb@xt@ \wd\one@line{\new@line%

If we use Lua\TeX then restore the direction.
1161  \ifluatex%
1162  \luatextextdir\l@luatextextdir@L%
1163  \fi%

Insert, if needed, the hanging symbol.
1164  \inserthangingsymbol \%Space keep for backward compatibility

And so, print the line.
1165  \l@dunhbox@line{\one@line}%%

Right line number
1166  \ledrlfill\l@drd@ta%

Print right notes.
1167  \l@drsn@te
1168  }%

And reinsert penalties (for page breaking)...
1169  \add@penalties%
1170 }

\print@eledsection \print@eledsection to print sectioning command with line number. It sets the correct spacing, depending whether a sectioning command was called at previous \pstart, calls the sectioning command, prints the normal line outside of the paper, to be able to have critical footnotes. Because of how this prints, a vertical spacing correction is added.
1171 \def\print@eledsection{%
1172 \add@inserts\affixside@note%
1173 \numdef{\temp@}{\l@dnumpstartsL-1}%
1174 \xifinlist{\temp@}{\eled@sections@@}{\@nobreaktrue}{\@nobreakfalse}%
1175 \eled@sectioningtrue%
1176 \csuse{eled@sectioning\l@dnumpstartsL}%
1177 \eled@sectioningfalse%
1178 \global\csundef{eled@sectioning\l@dnumpstartsL}%
1179 \if@RTL%
1180 \hspace{-3}\paperwidth}%
1181 {\hbox{\l@dunhbox@line{\one@line}}} \new@line%
23 Paragraph decomposition and reassembly

These high level commands just redefine the low level commands. They have to be used by user, without \makeatletter.

Two hooks into \do@line. The first is called at the beginning of \do@line, the second is called in the line box. The second can, for example, have a \markboth command inside, the first can't.

Nulls the \ld@emptyd@ta, which may later hold line numbers. Similarly for \ld@csnotetext, \ld@csnotetext@l, \ld@csnotetext@r for the texts of the sidenotes, left and right notes.

Zero width boxes of the left and right side notes, together with their kerns.

These macros are called at the left (\ledllfill) and the right (\ledrlfill) of each numbered line. The initial definitions correspond to the original code for \do@line.

The \getline@num macro determines the page and line numbers for the line we're about to send to the vertical list.
23.3 Line and page number computation

\newcommand*{\getline@num}{%
  \global\advance\absline@num \@ne
  \do@actions
  \do@ballast
  \ifnumberline
    \ifnum\subline@num < \@two@num
      \global\advance\subline@num \@ne
    \fi
  \else
    \ifnum\@lock < \@two@num
      \global\advance\line@num \@ne
      \global\subline@num \z@
    \fi
  \fi
}\fi
\fi}
\do@ballast

The real work in the macro above is done in \do@actions, but before we plunge into that, let's get \do@ballast out of the way. This macro looks to see if there is an action to be performed on the next line, and if it is going to be a page break action, \do@ballast decreases the count \ballast@count counter by the amount of ballast. This means, in practice, that when \add@penalties assigns penalties at this point, \TeX will be given extra encouragement to break the page here (see p. 103).

\ballast@count First we set up the required counters; they are initially set to zero, and will remain so unless you say \setcounter{ballast}{⟨some figure⟩} in your document.

1225 \newcount\ballast@count
1226 \newcounter{ballast}
1227 \setcounter{ballast}{0}

And here is \do@ballast itself. It advances \absline@num within the protection of a group to make its check for what happens on the next line.

1228 \newcommand*{\do@ballast}{\global\ballast@count \z@
1229 \begingroup
1230 \advance\absline@num \@ne
1231 \ifnum\next@actionline=\absline@num
1232 \ifnum\next@action>-1001\relax
1233 \global\advance\ballast@count by -\c@ballast
1234 \fi
1235 \fi
1236 \fi
1237 \endgroup}

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

It may call itself recursively, and to do this efficiently (using \TeX's optimization for tail recursion), we define a control-sequence called \do@actions@next that is always the last thing that \do@actions does. If there could be more actions to

1238 \do@actions@next
process for this line, \do@actions@next is set equal to \do@actions; otherwise it’s just \relax.\newcommand*{\do@actions}{% \global\let\do@actions@next=\relax \ifnum\absline@num<\next@actionline\else
First, page number changes, which will generally be the most common actions. If we’re restarting lineation on each page, this is where it happens.
\ifnum\next@action>-1001
  \global\page@num=\next@action
  \ifbypage@
    \global\line@num=\z@ \global\subline@num=\z@
    \resetprevline@
  \fi
\else
  \ifnum\next@action<-4999
    \@l@dtempcnta=-\next@action
    \advance\@l@dtempcnta by -5001
    \ifsublines@
      \global\subline@num=\@l@dtempcnta
    \else
      \global\line@num=\@l@dtempcnta
    \fi
  \else
    \@l@dtempcnta=-\next@action
    \advance\@l@dtempcnta by -1000
    \do@actions@fixedcode
  \fi
\fi
Next, we handle commands that change the line-number values. (We subtract 5001 rather than 5000 here because the line number is going to be incremented automatically in \getline@num.)\else
  \ifnum\next@action<-4999
    \@l@dtempcnta=-\next@action
    \advance\@l@dtempcnta by -5001
    \ifsublines@
      \global\subline@num=\@l@dtempcnta
    \else
      \global\line@num=\@l@dtempcnta
    \fi
  \else
    \@l@dtempcnta=-\next@action
    \advance\@l@dtempcnta by -1000
    \do@actions@fixedcode
  \fi
\fi
It’s one of the fixed codes. We rescale the value in \@l@dtempcnta so that we can use a case statement.
\else
  \@l@dtempcnta=-\next@action
  \advance\@l@dtempcnta by -1000
  \do@actions@fixedcode
\fi
Now we get information about the next action off the list, and then set \do@actions@next so that we’ll call ourself recursively: the next action might also be for this line.
There’s no warning if we find \actionlines@list empty, since that will always happen near the end of the section.
23.4 Line number printing

Make the recursive call, if necessary.

\newcommand*{\do@actions@fixedcode}{% \ifcase\@l@dtempcnta \or% % 1001 \global\sublines@true \or% % 1002 \global\sublines@false \or% % 1003 \global@lock=\@ne \or% % 1004 \ifnum\@lock=\tw@ \global\@lock=\thr@@ \else \global\@lock=\z@ \fi \or% % 1005 \global\sub@lock=\@ne \or% % 1006 \ifnum\sub@lock=\tw@ \global\sub@lock=\thr@@ \else \global\sub@lock=\z@ \fi \or% % 1007 \l@dskipnumbertrue \else \led@warn@BadAction \fi}

\affixline@num \affixline@num originally took a single argument, a series of commands for printing the line just split off by \do@line; it put that line back on the vertical list, and added a line number if necessary. It now just puts a left line number into \l@dld@ta or a right line number into \l@drd@ta if required.

To determine whether we need to affix a line number to this line, we compute the following:

\[ n = \text{int}
\left(\frac{\text{linenum} - \text{firstlinenum}}{\text{linenumincrement}}\right) \]

\[ m = \text{firstlinenum} + (n \times \text{linenumincrement}) \]
(where \textit{int} truncates a real number to an integer). \(m\) will be equal to \texttt{linenum} only if we're to paste a number on here. However, the formula breaks down for the first line to number (and any before that), so we check that case separately: if \(\texttt{line@num} \leq \texttt{firstlinenum}\), we compare the two directly instead of making these calculations.

We compute, in the scratch counter \texttt{@l@dtempcnta}, the number of the next line that should be printed with a number \(m\) in the above discussion, and move the current line number into the counter \texttt{@l@dtempcntb} for comparison.

First, the case when we're within a sub-line range.

\begin{verbatim}
\newcommand*{\affixline@num}{%}
\ifledgroupnotesL@false\else\ifnumberline\fi\fi\global\l@dskipnumberfalse\else\ifsublines@\global\l@dtempcntb=\subline@num\global\l@dtempcnta=\subline@num\advance\l@dtempcnta by-\c@firstsublinenum\divide\l@dtempcnta by\c@sublinenumincrement\multiply\l@dtempcnta by\c@sublinenumincrement\advance\l@dtempcnta by\c@firstsublinenum\else\global\l@dtempcnta=\c@firstsublinenum\fi\fi\fi\fi\chi@cksub@l@ck
\end{verbatim}

That takes care of computing the values for comparison, but if line number locking is in effect we have to make a further check. If this check fails, then we disable the line-number display by setting the counters to arbitrary but unequal values.

\begin{verbatim}
\chi@cksub@l@ck
\end{verbatim}

Now the line number case, which works the same way.

\begin{verbatim}
\else
\global\l@dtempcntb=\line@num\fi
\end{verbatim}

Check on the \texttt{linenumberlist} If it's \texttt{empty} use the standard algorithm.
The `\linenumberlist` wasn’t `\empty`, so here’s Wayne’s numbering mechanism. This takes place in TeX’s mouth.

```latex
\@l@dtempcnta=\line@num
\edef\rem@inder{},\linenumberlist,\number\line@num,}%
\edef\sc@n@list{\def\noexpand\sc@n@list####1,\number\@l@dtempcnta,####2|\{\def\noexpand\rem@inder{####2}}%}
\sc@n@list\expandafter\sc@n@list\rem@inder|%
\ifx\rem@inder\empty\advance\@l@dtempcnta\@ne\fi

A locking check for lines, just like the version for sub-line numbers above.
```

The following test is true if we need to print a line number.

```latex
\ch@ck@l@ck
\fi
```

The following test is true if we need to print a line number.

```latex
\ifnum\@l@dtempcnta=\@l@dtempcntb
If we got here, we’re going to print a line number; so now we need to calculate a number that will tell us which side of the page will get the line number. We start from `\line@margin`, which asks for one side always if it’s less than 2; and then if the side does depend on the page number, we simply add the page number to this side code—because the values of `\line@margin` have been devised so that this produces a number that’s even for left-margin numbers and odd for right-margin numbers.

For \LaTeX{} we have to consider two column documents as well. In this case I think we need to put the numbers at the outside of the column — the left of the first column and the right of the second. Do the two-column stuff before going on with the original code.

A left line number is stored in `\l@dld@ta` and a right one in `\l@drd@ta`.

```latex
\l@dld@ta\ A left line number is stored in `\l@dld@ta` and a right one in `\l@drd@ta`.\l@drd@ta
```

```latex
@if@twocolumn
\l@dld@ta\ A left line number is stored in `\l@dld@ta` and a right one in `\l@drd@ta`.\l@drd@ta
```

Continuing the original code ...

```latex
\@l@dtempcntb=\line@margin
\ifnum\@l@dtempcntb>\@ne
\advance\@l@dtempcntb \page@num
\fi
```

Now print the line (#1) with its page number.

```latex
\ifodd\@l@dtempcntb
\gdef\l@drd@ta{\rlap{\rightlinenum}}%
```

```latex
\l@dld@ta\ A left line number is stored in `\l@dld@ta` and a right one in `\l@drd@ta`.\l@drd@ta
```

```latex
\l@dld@ta\ A left line number is stored in `\l@dld@ta` and a right one in `\l@drd@ta`.\l@drd@ta
```
As no line number is to be appended, we just print the line as is.

Now fix the lock counters, if necessary. A value of 1 is advanced to 2; 3 advances to 0; other values are unchanged.

These macros handle line number locking for \texttt{\texttt{affixline@num}}. \texttt{\texttt{ch@cksub@l@ck}} checks subline locking. If it fails, then we disable the line-number display by setting the counters to arbitrary but unequal values.

Similarly for line numbers.
Fix the lock counters. A value of 1 is advanced to 2; 3 advances to 0; other values are unchanged.

\newcommand*{\f@x@l@cks}{% 
\ifcase\@lock 
\or \global\@lock=\tw@
\or \or \global\@lock=\z@
\fi
\ifcase\sub@lock 
\or \global\sub@lock=\tw@
\or \or \global\sub@lock=\z@
\fi}

Because of TeX’s asynchronous page breaking mechanism we can never be sure just where it will make a break and, naturally, it has already decided exactly how it will typeset any remainder of a paragraph that crosses the break. This is disconcerting when trying to number lines by the page or put line numbers in different margins. This macro tries to force an invisible paragraph break and a page break.

\newcommand{\pageparbreak}{\pend\newpage\pstart\noindent}

\pageparbreak

23.5 Pstart number printing in side

In side, the printing of pstart number is running like the printing of line number. There is only some differences:

\affixpstart@num
\pstartnum

• The pstarts counter is upgrade in the \pstart command. Consequently, the \affixpstart@num command has not to upgrade it, unlike the \affixline@num which upgrades the lines counter.

• To print the pstart number only at the beginning of a pstart, and not in every line, a boolean test is made. The \pstartnum boolean is set to TRUE at every \pstart. It’s tried in the \leftpstartnum and \rightstartnum commands. After the try, it is set to FALSE.
23.6 Add insertions to the vertical list

\add@inserts \add@inserts is the penultimate macro used by \do@line; it takes insertions saved in a list macro and sends them onto the vertical list.

It may call itself recursively, and to do this efficiently (using \TeX’s optimization for tail recursion), we define a control-sequence called \add@inserts@next that is always the last thing that \add@inserts does. If there could be more inserts to
process for this line, \add@inserts@next is set equal to \add@inserts; otherwise it's just \relax.

1453 \newcommand*{\add@inserts}{%
1454 \global\let\add@inserts@next=\relax

If \inserts@list is empty, there aren’t any more notes or insertions for this paragraph, and we needn’t waste our time.

1455 \ifx\inserts@list\empty \else

The \next@insert macro records the number of the line that receives the next footnote or other insert; it’s empty when we start out, and just after we’ve affixed a note or insert.

1456 \ifx\next@insert\empty
1457 \ifx\insertlines@list\empty
1458 \global\noteschanged@true
1459 \gdef\next@insert{100000}%
1460 \else
1461 \gl@p\insertlines@list\to\next@insert
1462 \fi
1463 \fi

If the next insert’s for this line, tack it on (and then erase the contents of the insert macro, as it could be quite large). In that case, we also set \add@inserts@next so that we’ll call ourself recursively: there might be another insert for this same line.

1464 \ifnum\next@insert=\absline@num
1465 \gl@p\inserts@list\to@insert
1466 \@insert
1467 \global\let@insert=\undefined
1468 \global\let\next@insert=\empty
1469 \global\let\add@inserts@next=\add@inserts
1470 \fi
1471 \fi

Make the recursive call, if necessary.

1472 \add@inserts@next}
1473

23.7 Penalties

\add@penalties \add@penalties is the last macro used by \do@line. 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 t operation. \displaywidowpenalty and \brokenpenalty are not restored, since we have no easy way to find out where we should insert them.

In this code, \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 \texttt{ballast\@count}, which has been worked out in \texttt{do\@ballast} above (p. 95). Finally, the penalty is checked to see that it doesn’t go below $-10000$.

\begin{verbatim}
\newcommand*{\add@penalties}{\@l@dtempcnta=\ballast\@count
\ifnum\num@lines>\@ne
  \global\advance\par@line \@one
  \ifnum\par@line=\@one
    \advance\@l@dtempcnta \clubpenalty
  \fi
  \@l@dtempcntb=\par@line \advance\@l@dtempcntb \@one
  \ifnum\@l@dtempcntb=\num@lines
    \advance\@l@dtempcnta \widowpenalty
  \fi
  \ifnum\par@line<\num@lines
    \advance\@l@dtempcnta \interlinepenalty
  \fi
  \fi
\ifnum\@l@dtempcnta=\z@\relax
\else
  \ifnum\@l@dtempcnta>-10000
    \penalty\@l@dtempcnta
  \else
    \penalty -10000
  \fi
\fi
}\end{verbatim}

\section{Printing leftover notes}

The \texttt{flush\@notes} macro is called after the entire paragraph has been sliced up and sent on to the vertical list. If the number of notes to this paragraph has increased since the last run of \TeX, then there can be leftover notes that haven’t yet been printed. An appropriate error message will be printed elsewhere; but it’s best to go ahead and print these notes somewhere, even if it’s not in quite the right place. What we do is dump them all out here, so that they should be printed on the same page as the last line of the paragraph. We can hope that’s not too far from the proper location, to which they’ll move on the next run.

\begin{verbatim}
\newcommand*{\flush@notes}{\%\@xloop
  \@xloop
\else
  \ifnum\@l@dtempcnta=-10000
    \penalty\@l@dtempcnta
  \else
    \penalty -10000
  \fi
\fi
\}
\end{verbatim}

\texttt{@xloop} is a variant of the \texttt{Plain \TeX\ loop} macro, useful when it’s hard to construct a positive test using the \texttt{\TeX\ if} commands—as in \texttt{flush\@notes} above.
One says \@xloop ... \if ... \else ... \repeat, and the action following \else is repeated as long as the \if test fails. (This macro will work wherever the Plain TeX \loop is used, too, so we could just call it \loop; but it seems preferable not to change the definitions of any of the standard macros.)

This variant of \loop was introduced by Alois Kabelschacht in TUGboat 8 (1987), pp. 184–5.

24 Critical footnotes

The footnote macros are adapted from those in Plain TeX, but they differ in these respects: the outer-level commands must add other commands to a list macro rather than doing insertions immediately; there are five separate levels of the footnotes, not just one; and there are options to reformat footnotes into paragraphs or into multiple columns.

24.1 Fonts

Before getting into the details of formatting the notes, we set up some font macros. It is the notes that present the greatest challenge for our font-handling mechanism, because we need to be able to take fragments of our main text and print them in different forms: it is common to reduce the size, for example, without otherwise changing the fonts used.

\select@lemmafont \select@lemmafont is provided to set the right font for the lemma in a note. This macro extracts the font specifier from the line and page number cluster, and issues the associated font-changing command, so that the lemma is printed in its original font.

24.2 Outer-level footnote commands

\footnotefont The \footnotefont\{\langle side\}\{\langle options\}\{\langle value\}\} change the value of on options of Xfootnote, to switch between true and false.
Critical footnotes

\footnotelang@lua \footnotelang@lua is called to remember the information about the language of a lemma when LuaLaTeX is used.

\newcommandx*{\footnotelang@lua}[1][1=L, usedefault]{%}
\ifstrequal{#1}{L}{%
\xright@appenditem{{\csxdef{footnote@luatextextdir}{\the\luatextextdir}}}\to\inserts@list\% Know the dir of lemma
\global\advance\insert@count \@ne%
\xright@appenditem{{\csxdef{footnote@luatexpardir}{\the\luatexpardir}}}\to\inserts@listR% Know the dir of lemma
\global\advance\insert@countR \@ne%
}%
\notblank{#2}{\docsvlist{#2}}{}% Parsing all options
}

\footnotelang@poly \footnotelang@poly is called to remember the information about the language of a lemma when Polyglossia is used.

\newcommandx*{\footnotelang@poly}[1][1=L, usedefault]{%}
\ifstrequal{#1}{L}{%
\if@RTL\%
\xright@appenditem{{\csxdef{footnote@dir}{@RTLtrue}}}\to\inserts@list% Know the language used in the lemma
\global\advance\insert@count \@ne%
\else\%
\xright@appenditem{{\csxdef{footnote@dir}{@RTLfalse}}}\to\inserts@list% Know the language of lemma
\global\advance\insert@count \@ne%
\fi%
\xright@appenditem{{\csxdef{footnote@lang}{\expandonce\languagename}}}\to\inserts@list% Know the language of lemma
\global\advance\insert@count \@ne%
}%
\notblank{#2}{\docsvlist{#2}}{}% Parsing all options
}

\footnotelang@lua \footnotelang@lua \footnotelang@poly \footnotelang@poly
24.3 Normal footnote formatting

The processing of each note is done by four principal macros: the \vfootnote macro takes the text of the footnote and does the \insert; it calls on the \footfmt macro to select the right fonts, print the line number and lemma, and do any other formatting needed for that individual note. Within the output routine, the two other macros, \footstart and \footgroup, are called; the first prints extra vertical space and a footnote rule, if desired; the second does any reformating of the whole set of the footnotes in this series for this page—such as paragraphing or division into columns—and then sends them to the page.

These four macros, and the other macros and parameters shown here, are distinguished by the ‘series letter’ that indicates which set of the footnotes we’re dealing with—A, B, C, D, or E. The series letter always precedes the string foot in macro and parameter names. Hence, for the A series, the four macros are called \vAfootnote, \Afootfmt, \Afootstart, and \Afootgroup.

\normalvfootnote We now begin a series of commands that do ‘normal’ footnote formatting: a format much like that implemented in Plain \TeX, in which each footnote is a separate paragraph.

\normalvfootnote takes the series letter as #1, and the entire text of the footnote is #2. It does the \insert for this note, calling on the \footfmt macro for this note series to format the text of the note.

\footstart \footgroup Some setup code that is common for a variety of the footnotes.

\mpnormalvfootnote And a somewhat different version for minipages.
The fonts that are used for printing notes might not have the character mapping we expect: for example, the Computer Modern font that contains old-style numerals does not contain an en-dash or square brackets, and its period and comma are in odd locations. To allow use of the standard footnote macros with such fonts, we use the following macros for certain characters.

The `\endashchar` macro is simply an en-dash from the normal font and is immune to changes in the surrounding font. The same goes for the full stop. These two are used in `\printlines`. The right bracket macro is the same again;

The fonts that are used for printing notes might not have the character mapping we expect: for example, the Computer Modern font that contains old-style numerals does not contain an en-dash or square brackets, and its period and comma are in odd locations. To allow use of the standard footnote macros with such fonts, we use the following macros for certain characters.

The `\endashchar` macro is simply an en-dash from the normal font and is immune to changes in the surrounding font. The same goes for the full stop. These two are used in `\printlines`. The right bracket macro is the same again;
24.3 Normal footnote formatting

It crops up in `\normalfootfmt` and the other footnote macros for controlling the format of the footnotes.

With polyglossia, each critical note has a `\footnote@lang` which shows the language of the lemma, and which can be used to switch the bracket from right to left.

```latex
\newcommand*{\fullstop}{\textnormal{.}}
\newcommand*{\rbracket}{\textnormal{\csuse{text\csuse{footnote@lang}}}%}
```

The `\footnote@lang` shows the language of the lemma, and which can be used to switch the bracket from right to left.

The `\printpstart` macro prints the pstart number for a note.

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 62: 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.

The original EDMAC code used several counters at this point, saying:

To simplify the logic, we use a lot of counters to tell us which numbers need to get printed (using 1 for yes, 0 for no, so that `\ifodd` tests for ‘yes’). The counter assignments are:

- `\@pnum` for page numbers;
- `\@ssub` for starting sub-line;
- `\@elin` for ending line;
- `\@esl` for ending sub-line; and
- `\@dash` for the dash between the starting and ending groups.

There’s no counter for the line number because it’s always printed.
\LaTeX{} tends to use a lot of counters and packages should try and minimise the number of new ones they create. In line with this Peter Wilson have reverted to traditional booleans.

\begin{verbatim}
\ifl@d@pnum
\ifl@d@ssub
\ifl@d@elin
\ifl@d@esl
\ifl@d@dash
\newif\ifl@d@pnum
\l@d@pnumfalse
\newif\ifl@d@ssub
\l@d@ssubfalse
\newif\ifl@d@elin
\l@d@elinfalse
\newif\ifl@d@esl
\l@d@eslfalse
\newif\ifl@d@dash
\l@d@dashfalse
\l@dparsefootspec \l@dparsefootspec{⟨spec⟩}{⟨lemma⟩}{⟨text⟩} parses a footnote specification. ⟨lemma⟩ and ⟨text⟩ are the lemma and text respectively. ⟨spec⟩ is the line and page number and lemma font specifier in \l@d@nums style format. The real work is done by \l@dparsefootspec which defines macros holding the numeric values.
\newcommand*{\l@dparsefootspec}{\l@dparsefootspec{⟨spec⟩}{⟨lemma⟩}{⟨text⟩}}
\def{\l@dparsefootspec}{\l@dparsefootspec{⟨spec⟩}{⟨lemma⟩}{⟨text⟩}}
\setprintlines
First of all, we print the page numbers only if: 1) we're doing the lineation by page, and 2) the ending page number is different from the starting page number.
Just a reminder of the arguments:
\setprintlines #1 | #2 | #3 | #4 | #5 | #6 | #7
\setprintlines start-page | line | subline | end-page | line | subline | font
The macro \setprintlines does the work of deciding what numbers should be printed. Its arguments are the same as the first 6 of \printlines.
\newcommand*{\setprintlines}{\setprintlines{⟨spec⟩}{⟨lemma⟩}{⟨text⟩}}
\end{verbatim}
24.3 Normal footnote formatting

We print the ending line number if: (1) we’re printing the ending page number, or (2) it’s different from the starting line number.

We print the starting sub-line if it’s nonzero.

We print the ending sub-line if it’s nonzero and: (1) it’s different from the starting sub-line number, or (2) the ending line number is being printed.

Now we’re ready to print it all. If the lineation is by pstart, we print the pstart.

One subtlety left here is when to print a period between numbers. But the only instance in which this is tricky is for the ending sub-line number: it could come after the starting sub-line number (in which case we want only the dash) or after an ending line number (in which case we need to insert a period).
\normalfootstart is a standard footnote-starting macro, called in the output routine whenever there are footnotes of this series to be printed: it skips a bit and then draws a rule.

Any footstart macro must put onto the page something that takes up space exactly equal to the \skipfootins value for the associated series of notes. \TeX makes page computations based on that \skip value, and the output pages will suffer from spacing problems if what you add takes up a different amount of space.

But if the skip \preXnotes@ is greater than 0 pt, it’s used instead of \skipfootins for the first printed series.

The \leftskip and \rightskip values are both zeroed here. Similarly, these skips are cancelled in the vfootnote macros for the various types of notes. Strictly speaking, this is necessary only if you are using paragraphed footnotes, but we have put it here and in the other vfootnote macros too so that the behavior of \eledmac in this respect is general across all footnote types (you can change this). What this means is that any \leftskip and \rightskip you specify applies to the main text, but not the footnotes. The footnotes continue to be of width \hsize.

\newcommand*{\normalfootstart}[1]{%  
  \ifdimequal{0pt}{\preXnotes@}{}{  
    \iftoggle{preXnotes@}{\togglefalse{preXnotes@}\skip\csname #1footins\endcsname=\csuse{preXnotes@}}{}  
  }%  
  \vskip\skip\csname #1footins\endcsname%  
  \leftskip0pt \rightskip0pt  
  \ifl@dpairing\else%  
    \hsize=\old@hsize%  
  \fi%  
  \setXnoteswidthliketwocolumns@{#1}%  
  \setXnotespositionliketwocolumns@{#1}%  
  \print@Xfootnoterule{#1}%%  
  \vskip\csuse{afterXrule@#1}%  
  \noindent\leavevmode}%

\normalfootnoterule is a standard footnote-rule macro, for use by a footstart macro: just the same as the PLAIN \TeX footnote rule.

\newcommand*{\normalfootnoterule}{\footnoterule}

\normalfootgroup is a standard footnote-grouping macro: it sends the contents of the footnote-insert box to the output page without alteration.

\newcommand*{\normalfootgroup}[1]{%  
  {\csuse{Xnotefontsize@#1}\noindent\csuse{txtbeforeXnotes@#1}}%  
  \unvbox\csname #1footins\endcsname%  
  \hsize=\old@hsize%  
}  

\normalfootnoterule

\mpnormalfootgroup A somewhat different version for minipages.
24.4 Standard footnote definitions

\footnormal We can now define all the parameters for the five series of footnotes; initially they use the ‘normal’ footnote formatting, which is set up by calling \footnormal. You can switch to another type of formatting by using \footparagraph, \foottwocol, or \footthreecol.

Switching to a variation of ‘normal’ formatting requires changing the quantities defined in \footnormal. The best way to proceed would be to make a copy of this macro, with a different name, make your desired changes in that copy, and then invoke it, giving it the letter of the footnote series you wish to control.

(We have not defined baseline skip values like \abaselineskip, since this is one of the quantities set in \notefontsetup.)

What we want to do here is to say something like the following for each footnote series. (This is an example, not part of the actual eledmac code.)

\skip\Afootins=12pt plus5pt minus5pt
\count\Afootins=1000
\dimen\Afootins=0.8\vsize
\let\Afootnote=\normalvfootnote \let\Afootfmt=\normalfootfmt
\let\Afootstart=\normalfootstart \let\Afootgroup=\normalfootgroup
\let\Afootnoterule=\normalfootnoterule
Instead of repeating ourselves, we define a \footnormal macro that makes all these assignments for us, for any given series letter. This also makes it easy to change from any different system of formatting back to the normal setting.

\ledfootinsdim Have a constant value for the \dimen of \footins.

\preXnotes If user redefines \preXnotes to a value greater than 0 pt, this skip will be added before first series notes instead of the notes skip.

The same, but for familiar footnotes.

Now we set up the \footnormal macro itself. It takes one argument: the footnote series letter.

Some of these values deserve comment: the \dimen setting allows 80% of the page to be occupied by notes; the \skip setting is deliberately flexible, since pages with lots of notes attached to many of the lines can be a bit hard for \TeX to make.
24.5 Paragraphed footnotes

The paragraphed-footnote option reformat all the footnotes of one series for a page into a single paragraph; this is especially appropriate when the notes are numerous and brief. The code is based on *The TeXbook*, pp. 398–400, with alterations for our environment. This algorithm uses a considerable amount of save-stack space: a \TeX of ordinary size may not be able to handle more than about 100 notes of this kind on a page.

\footparagraph

The \texttt{\footparagraph} macro sets up everything for one series of the footnotes so that they'll be paragraphed; it takes the series letter as argument. We include the setting of \texttt{\count\footins} to 1000 for the footnote series just in case you are switching to paragraphed footnotes after having columnar ones, since they change this value (see below).

It is important to call \texttt{\footparagraph} only after \texttt{\hsize} has been set for the pages that use this series of notes; otherwise \TeX will try to put too many or too few of these notes on each page. If you need to change the \texttt{\hsize} within the document, call \texttt{\footparagraph} again afterwards to take account of the new value. The argument of \texttt{\footparagraph} is the letter (A–E) denoting the series of notes to be paragraphed.

\footfudgefiddle

For paragraphed footnotes \TeX has to estimate the amount of space required. If it underestimates this then the notes may get too long and run off the bottom of the text block. \texttt{\footfudgefiddle} can be increased from its default 64 (say to 70) to increase the estimate.

\para@footsetup

\footparagraph calls the \texttt{\para@footsetup} macro to calculate a special fudge factor, which is the ratio of the \texttt{\baselineskip} to the \texttt{\hsize}. We assume that the proper value of \texttt{\baselineskip} for the footnotes (normally 9 pt) has been set
already, in \notefontsetup. The argument of the macro is again the note series letter.

Peter Wilson thinks that \columnwidth should be used here for \LaTeX not \hsize. I've also included \footfudgefiddle.

\begin{verbatim}
\newcommand*{\paramfootsetup}[1]{{\csuse{Xnotefontsize@#1}
\setXnoteswidthliketwocolumns@{#1}\%
\dimen0=\baselineskip
\multiply\dimen0 by 1024
\divide \dimen0 by \columnwidth \multiply\dimen0 by \footfudgefiddle\relax
\csxdef{#1footfudgefactor}{%
\expandafter\strip@pt\dimen0 }}}
\end{verbatim}

EDMAC defines \en@number which does the same as the \LaTeX kernel \strip@pt, namely strip the characters pt from a dimen value. Eledmac use \strip@pt.

\begin{verbatim}
\parafootstart
\parafootstart is the same as \normalfootstart, but we give it again to ensure that \rightskip and \leftskip are zeroed (this needs to be done before \paragroup in the output routine). You might have decided to change this for other kinds of note, but here it should stay as it is. The size of paragraphed notes is calculated using a fudge factor which in turn is based on \hsize. So the paragraph of notes needs to be that wide.

The argument of the macro is again the note series letter.

\begin{verbatim}
\newcommand*{\parafootstart}[1]{{\rightskip=0pt \leftskip=0pt \parindent=0pt
\ifdimequal{0pt}{\preXnotes@}{}%
\iftoggle{preXnotes@}{%\togglefalse{preXnotes@}\skip\csname #1footins\endcsname=\csuse{preXnotes@}}%
\vskip\skip\csname #1footins\endcsname%\setXnoteswidthliketwocolumns@{#1}%
\setXnotespositionliketwocolumns@{#1}%
\print@Xfootnoterule{#1}%%
\vskip\csuse{afterXrule@#1}%
\noindent\leavevmode}
\end{verbatim}

\begin{verbatim}
\para@vfootnote
\para@vfootnote is a version of the \vfootnote command that's used for paragraphed notes. It gets appended to the \inserts@list list by an outer-level footnote command like \Afootnote. The first argument is the note series letter; the second is the full text of the printed note itself, including line numbers, lemmata, and footnote text.

The initial model for this insertion is, of course, the \insert\footins definition in The \TeX\book, p. 398. There, the footnotes are first collected up in hboxes, and these hboxes are later unpacked and stuck together into a paragraph.

However, Michael Downes has pointed out that because text in hboxes gets typeset in restricted horizontal mode, there are some undesirable side-effects if
24.5  Paragraphed footnotes

you later want to break such text across lines. In restricted horizontal mode, where \TeX does not expect to have to break lines, it does not insert certain items like \texttt{\textit{discretionary}}s. If you later unbox these hboxes and stick them together, as the \TeXbook macros do to make these footnotes, you lose the ability to hyphenate after an explicit hyphen. This can lead to overfull \hboxes when you would not expect to find them, and to the uninitiated it might be very hard to see why the problem had arisen.\footnote{Michael Downes, ‘Line Breaking in unboxed Text’, \textit{TUGboat} \textbf{11} (1990), pp. 605–612.}

Wayne Sullivan pointed out to us another subtle problem that arises from the same cause: \TeX also leaves the \texttt{\textit{language}} whatsit nodes out of the horizontal list.\footnote{See \textit{The TeXbook}, p. 455 (editions after January 1990).} So changes from one language to another will not invoke the proper hyphenation rules in such footnotes. Since critical editions often do deal with several languages, especially in a footnotes, we really ought to get this bit of code right.

To get around these problems, Wayne suggested emendations to the \TeXbook versions of these macros which are broadly the same as those described by Michael: the central idea (also suggested by Donald Knuth in a letter to Michael) is to avoid collecting the text in an \hbox in the first place, but instead to collect it in a \vbox whose width is (virtually) infinite. The text is therefore typeset in unrestricted horizontal mode, as a paragraph consisting of a single long line. Later, there is an extra level of unboxing to be done: we have to unpack the \vbox, as well as the hboxes inside it, but that’s not too hard. For details, we refer you to Michael’s article, where the issues are clearly explained.\footnote{Wayne supplied his own macros to do this, but since they were almost identical to Michael’s, we have used the latter’s \texttt{\textit{unv}xh} macro since it is publicly documented.}

Michael’s unboxing macro is called \texttt{\textit{unv}xh}: unvbox, extract the last line, and unhbox it.

Doing things this way has an important consequence: as Michael pointed out, you really can’t put an explicit line-break into a note built in a \vbox the way we are doing.\footnote{‘Line Breaking’, p. 610.} In other words, be very careful not to say \texttt{\textit{break}}, or \texttt{\textit{penalty}−10000}, or any equivalent inside your para-footnote. If you do, most of the note will probably disappear. You are allowed to make strong suggestions; in fact \texttt{\textit{penalty}−9999} will be quite okay. Just don’t make the break mandatory. We haven’t applied any of Michael’s solutions here, since we feel that the problem is exiguous, and \texttt{eledmac} is quite baroque enough already. If you think you are having this problem, look up Michael’s solutions.

One more thing; we set \texttt{\textit{leftskip}} and \texttt{\textit{rightskip}} to zero. This has the effect of neutralizing any such skips which may apply to the main text (cf. p. \textbf{112} above). We need to do this, since \texttt{footfudgefactor} is calculated on the assumption that the notes are \texttt{\textit{hsize}} wide.

So, finally, here is the modified foot-paragraph code, which sets the footnote in vertical mode so that language and discretionary nodes are included.

\begin{verbatim}
1813 \newcommand*{\para@vfootnote}[2]{% 
1814 \insert\csname #1footins\endcsname
1815 \bgroup
1816 \csuse{bhookXnote@#1}
1817 \endgroup
\endverbatim

Here we produce the contents of the footnote from box 0, and add a penalty of 0 between boxes in this insert.

The final penalty of 0 was added here at Wayne’s suggestion to avoid a weird page-breaking problem, which occurs on those occasions when TeX attempts to split foot paragraphs. After trying out such a split (see The TeXbook, p. 124), TeX inserts a penalty of −10000 here, which nearly always forces the break at the end of the whole footnote paragraph (since individual notes can’t be split) even when this leads to an overfull vbox. The change above results in a penalty of 0 instead which allows, but doesn’t force, such breaks. This penalty of 0 is later removed, after page breaks have been decided, by the \unpenalty macro in \makeboxofhboxes. So it does not affect how the footnote paragraphs are typeset (the notes still have a penalty of −10 between them, which is added by \parafootfmt).

\mppara@vfootnote This version is for minipages.

\unvxh Here is (modified) Michael’s definition of \unvxh, used above. Michael’s macro also takes care to remove some unwanted penalties and glue that TeX automatically attaches to the end of paragraphs. When TeX finishes a paragraph, it throws away any remaining glue, and then tacks on the following items: a \penalty of 10000,


\parfillskip and a \rightskip (The TeXbook, pp. 99–100). \unvxx cancels these unwanted paragraph-final items using \unskip and \unpenalty.

\newcommandx*{\unvxx}[2][2=2]\{% 2th is optional for retro-compatibility
  \setbox0=\vbox{\unvbox#1}%
  \global\setbox1=\lastbox%
  \unhbox1
  \unskip % remove \rightskip,
  \unskip % remove \parfillskip,
  \unpenalty % remove \penalty of 10000,
  \hskip\csuse{afternote@#2}} % but add the glue to go between the notes

\parafootfmt \parafootfmt is \normalfootfmt adapted to do the special stuff needed for paragraphed notes—leaving out the \endgraf at the end, sticking in special penalties and kern, and leaving out the \footstrut. The first argument is the line and page number information, the second is the lemma, the third is the text of the footnote, and the fourth is the series (optional, for backward compatibility).

\newcommandx*{\parafootfmt}[4][4=2]\{%
  \insertparafootsep{#4}%
  \ledsetnormalparstuff%
  \printlinefootnote{#1}{#4}%
  \ifcempty{lemmaseparator@#4}{\hskip\csuse{inplaceoflemmaseparator@#4}}{
obreak\hskip\csuse{beforelemmaseparator@#4}\csuse{lemmaseparator@#4}\hskip\csuse{afterlemmaseparator@#4}%%}
  \ifcempty{lemmaseparator@#4}{\hskip\csuse{inplaceoflemmaseparator@#4}}{}
  \ifcempty{lemmaseparator@#4}{\hskip\csuse{inplaceoflemmaseparator@#4}}{}
  \ifcempty{lemmaseparator@#4}{\hskip\csuse{inplaceoflemmaseparator@#4}}{}
  \ifcempty{lemmaseparator@#4}{\hskip\csuse{inplaceoflemmaseparator@#4}}{}
  \iftoggle{nosep@}{\hskip\csuse{inplaceoflemmaseparator@#4}}{\ifcempty{lemmaseparator@#4}{\hskip\csuse{inplaceoflemmaseparator@#4}}{\hskip\csuse{beforelemmaseparator@#4}\csuse{lemmaseparator@#4}\hskip\csuse{afterlemmaseparator@#4}%%}
  \hskip\csuse{afternote@#2}}\%

Note that in the above definition, the penalty of −10 encourages a line break between notes, so that notes have a slight tendency to begin on new lines. The \insertparafootsep command is used to insert the \parafootsep@series between each note in the same page.

This footgroup code is modelled on the macros in The TeXbook, p. 399. The only difference is the \unpenalty in \makeboxofhboxes, which is there to remove the penalty of 0 which was added to the end of each footnote by \para@vfootnote.

The call to \notefontsetup is to ensure that the correct \baselineskip for the footnotes is used. The argument is the note series letter.
24.6 Insertion of the footnotes separator

The command \texttt{\insertparafootsep{(series)}} must be called at the beginning of \texttt{\parafootftm} (and like commands).
24.7 Columnar footnotes

We will now define macros for three-column notes and two-column notes. Both sets of macros will use \rigidbalance, which splits a box (#1) into a number (#2) of columns, each with a space (#3) between the top baseline and the top of the \vbox. The \rigidbalance macro is taken from *The TeXbook*, p. 397, with a slight change to the syntax of the arguments so that they don't depend on white space. Note also the extra unboxing in \splitoff, which allows the new \vbox to have its natural height as it goes into the alignment.

The \LaTeX \line macro has no relationship to the TeX \line. The \LaTeX equivalent is \@@line.

Three columns

You say \footthreecol{A} to have the A series of the footnotes typeset in three columns. It is important to call this only after \hsize has been set for the document.
The \threecolfootsetup macro calculates and sets some numbers for three-column footnotes.

We set the \count of the foot insert to 333. Each footnote can be thought of as contributing only one third of its height to the page, since the footnote insertion has been made as a long narrow column, which then gets trisected by the \rigidbalance routine (inside \threecolfootgroup). These new, shorter columns are saved in a box, and then that box is put back into the footnote insert, replacing the original collection of the footnotes. This new box is, therefore, only about a third of the height of the original one.

The \dimen value for this note series has to change in the inverse way: it needs to be three times the actual limit on the amount of space these notes are allowed to fill on the page, because when \TeX{} is accumulating material for the page and checking that limit, it doesn’t apply the \count scaling.

The call to \notefontsetup ensures that the \splittopskip and \splitmaxdepth take their values from the right \strutbox: the one used in a footnotes. Note especially the importance of temporarily reducing the \hsize to 0.3 of its normal value. This determines the widths of the individual columns. So if the normal \hsize is, say, 10 cm, then each column will be $0.3 \times 10 = 3$ cm wide, leaving a gap of $1$ cm spread equally between columns (i.e., $0.5$ cm between each).

The arguments are 1) the note series letter and 2) the full text of the note (including numbers, lemma and text).


\threecolfootfmt

\threecolfootfmt is the command that formats one note. It uses \raggedright, which will usually be preferable with such short lines. Setting the \parindent to zero means that, within each individual note, the lines begin flush left.

The arguments are 1) the line numbers, 2) the lemma and 3) the text of the -footnote command 4) optional (for backward compatibility): the series.

And here is the footgroup macro that’s called within the output routine to re-group the notes into three columns. Once again, the call to \notefontsetup is there to ensure that it is the right \splittopskip—the one used in footnotes—which is used to provide the third argument for \rigidbalance. This third argument (\@h) is the topskip for the box containing the text of the footnotes, and does the job of making sure the top lines of the columns line up horizontally. In The TeXbook, p. 398, Donald Knuth suggests retrieving the output of \rigidbalance, putting it back into the insertion box, and then printing the box. Here, we just print the line which comes out of \rigidbalance directly, without any re-boxing.
Two columns

Two columns

You say \texttt{\footnotewcol{A}} to have the A series of the footnotes typeset in two columns. It is important to call this only after \texttt{\hsize} has been set for the document.

The additional setup for minipages.

Here is a series of macros which are very similar to their three-column counterparts. In this case, each note is assumed to contribute only a half a line of text. And the notes are set in columns giving a gap between them of one tenth of the \texttt{\hsize}.

\footnotewcol

\footnotewcolfootnote

\footnotewcolfootfmt

\footnotewcolfootgroup

\twocolfootsetup

\twocolvfootnote

\twocolfootfmt

\twocolfootgroup
\section{Columnar footnotes}

\notbool{parapparatus*}{\newcommandx*}{\newcommandx}{\twocolfootfmt} [4=Z] \% 4th arg is optional, for backward compatibility
\normal\pars
\hsize \csuse{hsizetwocol@#4}
\parindent=0pt
\tolerance=5000
\raggedright
\hangindent=\csuse{Xhangindent@#4}
\leavevmode
\strut{\printlinefootnote{#1}{#4}}\%
\iftoggle{nosep@}{\hskip\csuse{inplaceoflemmaseparator@#4}}{
\ifcsempty{lemmaseparator@#4}{\hskip\csuse{inplaceoflemmaseparator@#4}}{
\nobreak\hskip\csuse{beforelemmaseparator@#4}\csuse{lemmaseparator@#4}\hskip\csuse{afterlemmaseparator@#4}}%
#3\strut\par\allowbreak}

\newcommand*{\twocolfootgroup} [1] {\csuse{Xnotefontsize@#1}{\csuse{Xnotefontsize@#1}\noindent\csuse{txtbeforeXnotes@#1}}\par
\splittopskip=\ht\strutbox
\expandafter
\rigidbalance\csname #1footins\endcsname \tw@ \splittopskip}

\mptwocolfootsetup The versions for minipages.
\mptwocolfootgroup
\newcommand*{\mptwocolfootsetup} [1] {%
\count\csname mp#1footins\endcsname 500
\multiply\dimen\csname mp#1footins\endcsname \tw@}
\newcommand*{\mptwocolfootgroup} [1] {%
\vskip\skip\@nameuse{mp#1footins}
\ifl@dpairing\ifparledgroup%
\leavevmode\marks\parledgroup@{begin}%
\marks\parledgroup@series{#1}%
\marks\parledgroup@type{Xfootnote}%
\fi\fi\normalcolor
\ifparledgroup%
\ifl@dpairing%
\else%
\setXnoteswidthliketwocolumns@{#1}%
\setXnotepositionliketwocolumns@{#1}%
\print@Xfootnoterule{#1}%%
\vskip\csuse{afterXrule@#1}%%
\fi%
\else%
\setXnoteswidthliketwocolumns@{#1}%
\setXnotepositionliketwocolumns@{#1}%
\print@Xfootnoterule{#1}%%
\vskip\csuse{afterXrule@#1}%%
\fi%
\csuse{Xnotefontsize@#1}\noindent\csuse{txtbeforeXnotes@#1}}\par
\splittopskip=\ht\strutbox
25 Familiar footnotes

25.1 Generality

The original EDMAC provided users with five series of critical footnotes (\Afootnote \Bfootnote \Cfootnote \Dfootnote \Efootnote), and \LaTeX{} provides a single numbered footnote. The \eledmac package uses the EDMAC mechanism to provide five series of numbered footnotes.

First, though, the \footmisc package has an option whereby two or more consecutive \footnote{s} have their marks separated by commas. This seems such a useful ability that it is provided automatically by \eledmac.

\multiplefootnotemarker \multfootsep

These macros may have been defined by the \memoir class, are provided by the \footmisc package and perhaps by other footnote packages.

\providecommand*{\multiplefootnotemarker}{3sp}
\providecommand*{\multfootsep}{\textsuperscript{\normalfont,}}
\m@mmf@prepare

A pair of self-cancelling kerns. This may have been defined in the \memoir class.

\providecommand*{\m@mmf@prepare}{\kern-\multiplefootnotemarker \kern\multiplefootnotemarker\relax}
\m@mmf@check

This may have been defined in the \memoir class. If it recognises the last kern as \multiplefootnotemarker it typesets \multfootsep.

\providecommand*{\m@mmf@check}{\ifdim\lastkern=\multiplefootnotemarker \relax
\edef@x@sf{\the\spacefactor}\spacefactor\@x@sf\relax\fi}
\m@mmf@check

\renewcommand*{\@footnotemark}{
\if@filesw \def\@thm@sw{\fi}
\fi}
\renewcommand*{\@footnotemark}{
\if@filesw \def\@thm@sw{\fi}
\fi}

We have to modify \@footnotetext and \@footnotemark. However, if \memoir is used the modifications have already been made.

\ifclassloaded{\memoir}{
\let\@footnotetext\@footnotetext
\let\@footnotemark\@footnotemark
}
\let\@footnotemark\@footnotemark

Add \m@mmf@prepare at the end of \footnotetext.
\apptocmd{\footnotetext}{\m@mmf@prepare}{\{\}}
\renewcommand*{\@footnotemark}{
\if@filesw \def\@thm@sw{\fi}
\fi}
\renewcommand*{\@footnotemark}{
\if@filesw \def\@thm@sw{\fi}
\fi}
\l@dbfnote \l@dbfnote adds the footnote to the insert list, and \vl@dbfnote calls the original \@footnotetext.
\vl@dbfnote Two convenience macros for use by \ldots\@footnotemark\ldots macros.

25.2 Footnote formats

Some of the code for the various formats is remarkably similar to that in section 24.3.

The following macros generally set things up for the ‘standard’ footnote format.

\l@doldold\@footnotetext In order to enable the regular \footnotes in numbered text we have to play around
with its \@footnotetext, using different forms for when in numbered or regular
text.

\l@dbfnote \l@dbfnote adds the footnote to the insert list, and \vl@dbfnote calls the original
\vl@dbfnote Two convenience macros for use by \ldots\@footnotemark\ldots macros.

\l@doldold\@footnotetext 25.2 Footnote formats

Some of the code for the various formats is remarkably similar to that in section 24.3.

The following macros generally set things up for the ‘standard’ footnote format.
Familiar footnotes

\normal@footnotemarkX \normal@footnotemarkX{(series)} sets up the typesetting of the marker at the point where the footnote is called for.

\normalbodyfootmarkX \normalbodyfootmarkX{(series)} really typesets the in-text marker. The style is the normal superscript.

\normalvfootnoteX \normalvfootnoteX{(series)}{(text)} does the \texttt{\insert} for the \texttt{(series)} and calls the series’ \texttt{\footfmt...} to format the \texttt{(text)}.

\mpnormalvfootnoteX The minipage version.
25.2 Footnote formats

\normalfootfmtX\normalfootfmtX\langle series\rangle\{(text)\} typesets the footnote text, prepended by the marker.

\newcommand*{\normalfootfmtX}[2]{%
\protected@edef\@currentlabel{\@nameuse{@thefnmark#1}}%
\ledsetnormalparstuff
\hangindent=\csuse{hangindentX@#1}%
{{\csuse{notenumfontX@#1}\@nameuse{footfootmark#1}}
\strut\enspace
#2\strut\par}
%

\normalfootfootmarkX\normalfootfootmarkX\langle series\rangle is called by \normalfootfmtX to typeset the footnote marker in the footer before the footnote text.

\newcommand*{\normalfootfootmarkX}[1]{%
\textsuperscript{\@nameuse{@thefnmark#1}}}

\normalfootstartX\normalfootstartX\langle series\rangle is the \langle series\rangle footnote starting macro used in the output routine.

\newcommand*{\normalfootstartX}[1]{%
\ifdimequal{0pt}{\prenotesX@}{}%
{% \iftoggle{prenotesX@}{\togglefalse{prenotesX@} \skip\csname footins#1\endcsname=\csuse{prenotesX@}}% 
\vskip\skip\csname footins#1\endcsname%
\leftskip=\z@ \rightskip=\z@
\ifl@dpairing\else \hsize=\old@hsize\fi%
\setnotesXwidthliketwocolumns@{#1}%
\setnotesXpositionliketwocolumns@{#1}%
\print@footnoteXrule{#1}%
\vskip\csuse{afterruleX@#1}}%

\normalfootnoteruleX The rule drawn before the footnote series group.
\let\normalfootnoteruleX=\footnoterule

\normalfootgroupX\normalfootgroupX\langle series\rangle sends the contents of the \langle series\rangle insert box to the output page without alteration.
\mpnormalfootgroupX  \ The minipage version.
\newcommand*{\mpnormalfootgroupX}[1]{%
  \vskip\skip\@nameuse{mpfootins#1}
  \if@dpairing\ifparledgroup%
    \leavevmode\marks\parledgroup@{begin}%
    \marks\parledgroup@series{#1}%
    \marks\parledgroup@type{footnoteX}%
  \fi\fi\normalcolor
  \ifparledgroup%
  \if@dpairing%
    \else%
    \setnotesXwidthliketwocolumns@{#1}%
    \setnotesXpositionliketwocolumns@{#1}%
    \print@footnoteXrule{#1}%
    \vskip\csuse{afterruleX@#1}%
  \else%
    \else%
    \setnotesXwidthliketwocolumns@{#1}%
    \setnotesXpositionliketwocolumns@{#1}%
    \print@footnoteXrule{#1}%
    \vskip\csuse{afterruleX@#1}%
  \fi%
  \else%
  \setnotesXwidthliketwocolumns@{#1}%
  \setnotesXpositionliketwocolumns@{#1}%
  \print@footnoteXrule{#1}%
  \vskip\csuse{afterruleX@#1}%
  \fi%
  \unvbox\@nameuse{mpfootins#1}%
\}

\normalbfnoteX
\newcommand{\normalbfnoteX}[2]{%
  \ifnumberedpar@
    \protected@xdef\thisfootnote{\csuse{thefootnote#1}%%}
    \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\expandonce\thisfootnote}}%
    \edtext{}{\normalbfnoteX{#1}{#2}{}{\expandonce\thisfootnote}}%
    \to\inserts@list
  \global\advance\insert@count \@ne
  \fi\ignorespaces}

\vbfnoteX
\newcommand{\vbfnoteX}[3]{%
  \ifnumberedpar@
    \edef\thisfnmark{#1}{}
    \@namedef{@thefnmark#3}{#2}%
    \@nameuse{regvfootnote#1}{#1}{#3}%
  \else
    \@nameuse{regvfootnote#1}{#1}{#2}%
  \fi}

\vnumfootnoteX
\newcommand{\vnumfootnoteX}[2]{%
  \ifnumberedpar@
    \edtext{\normalbfnoteX{#1}{#2}{}{\expandonce\thisfootnote}}%
  \else
    \@nameuse{regvfootnote#1}{#1}{#2}%
  \fi}
The following macros set footnotes in two columns. It is assumed that the length of each footnote is less than the column width.

\twocolfootsetupX  \twocolfootsetupX\{⟨series⟩\}
\mptwocolfootsetupX
\twocolfootsetupX\{⟨series⟩\}
\twocolvfootnoteX \twocolvfootnoteX\{(series)\}
\twocolfootfmtX \twocolfootfmtX\{(series)\}
\twocolfootgroupX \twocolfootgroupX\{(series)\}
25.4 Three columns footnotes

The following macros set footnotes in three columns. It is assumed that the length of each footnote is less than the column width.

\footthreecolX \footthreecolX{(series)}
\newcommand*{\footthreecolX}[1]{
  \csgdef{series@displayX#1}{threecol}
  \expandafter\let\csname regvfootnote#1\endcsname=\threecolvfootnoteX
  \expandafter\let\csname footfmt#1\endcsname=\threecolfootfmtX
  \expandafter\let\csname footgroup#1\endcsname=\threecolfootgroupX
  \threecolfootsetupX{#1}
  \expandafter\let\csname mpvfootnote#1\endcsname=\mpnormalvfootnoteX
  \expandafter\let\csname mpfootgroup#1\endcsname=\mpthreecolfootgroupX
  \mpthreecolfootsetupX{#1}}

\threecolfootsetupX \threecolfootsetupX{(series)}
\mpthreecolfootsetupX \mpthreecolfootsetupX{(series)}{(text)}
\newcommand*{\threecolfootsetupX}[1]{
  \count\csname footins#1\endcsname 333
  \multiply\dimen\csname footins#1\endcsname by \thr@@}
\newcommand*{\mpthreecolfootsetupX}[1]{
  \count\csname mpfootins#1\endcsname 333
  \multiply\dimen\csname mpfootins#1\endcsname by \thr@@}

\threecolvfootnoteX \threecolvfootnoteX{(series)}{(text)}
\newcommand*{\threecolvfootnoteX}[2]{
  \insert\csname footins#1\endcsname\bggroup
  \csuse{notefontsizeX@#1}
  \footspillskips
  \@nameuse{footfmt#1}{#1}{#2}\egroup}

\threecolfootfmtX \threecolfootfmtX{(series)}
25.5 Paragraphed footnotes

The following macros set footnotes as one paragraph.
25.5 Paragraphe footnotes

\footparagraphX \footparagraphX{⟨series⟩}
\newcommand*{\footparagraphX}[1]{%  
\csgdef{series@displayX#1}{paragraphX}%
\expandafter\newcount\csname prevpage#1\endcsname
\expandafter\let\csname footstart#1\endcsname=\parafootstartX
\expandafter\let\csname regvfootnote#1\endcsname=\para@vfootnoteX
\expandafter\let\csname footfmt#1\endcsname=\parafootfmtX
\expandafter\let\csname footgroup#1\endcsname=\para@footgroupX
\expandafter\let\csname footnoterule#1\endcsname=\normalfootnoteruleX
\count\csname footins#1\endcsname=1000
\dimen\csname footins#1\endcsname=\csuse{maxhnotesX@#1}
\skip\csname footins#1\endcsname=\csuse{beforenotesX@#1}
\expandafter\let\csname mpvfootnote#1\endcsname=\mppara@vfootnoteX
\expandafter\let\csname mpfootgroup#1\endcsname=\mppara@footgroupX
\count\csname mpfootins#1\endcsname=1000
\dimen\csname mpfootins#1\endcsname=\csuse{maxhnotesX@#1}
\skip\csname mpfootins#1\endcsname=\csuse{beforenotesX@#1}
\skip\csname footins#1\endcsname=\csuse{beforenotesX@#1}
\para@footsetupX{#1}}

\para@footsetupX \para@footsetupX{⟨series⟩}
\newcommand*{\para@footsetupX}[1]{%  
\setnotesXwidthliketwocolumns@{#1}%
\dimen0=\baselineskip
\multiply\dimen0 by 1024
\divide\dimen0 by \columnwidth \multiply\dimen0 by \footfudgefactorfiddle\relax%
\expandafter
\xdef\csname footfudgefactor#1\endcsname{\strip@pt\dimen0 }
}

\parafootstartX \parafootstartX{⟨series⟩}
\newcommand*{\parafootstartX}[1]{%  
\ifdimequal{0pt}{\prenotesX@}{}%
\iftoggle{prenotesX@}{}{
\skip\csname footins#1\endcsname=\csuse{prenotesX@}}%
\vskip\skip\@nameuse{footins#1}%
\setnotesXwidthliketwocolumns@{#1}%
\setnotesXpositionliketwocolumns@{#1}%
\print@footnoteXrule{#1}%
\vskip\csuse{afterruleX@#1}%
}
\para@vfootnoteX \para@vfootnoteX{(series)}{⟨text⟩}
\mppara@vfootnoteX
\newcommand*{\para@vfootnoteX}[2]{%
  \insert\csname footins#1\endcsname
  \bgroup
  \csuse{bhooknoteX@#1}
  \csuse{notefontsizeX@#1}
  \footsepskips
  \setbox0=\vbox{\hsize=\maxdimen
    \noindent\nameuse{footfmt#1}{#1}{#2}}%
  \setbox0=\hbox{\unvxh0[#1]}%
  \dp0=\z@%
  \ht0=\csname footfudgefactor#1\endcsname\wd0
  \box0
  \penalty0
  \egroup}
\newcommand*{\mppara@vfootnoteX}[2]{%
  \global\setbox\@nameuse{mpfootins#1}\vbox{%}
  \unvbox\@nameuse{mpfootins#1}
  \csuse{bhooknoteX@#1}
  \csuse{notefontsizeX@#1}
  \footsepskips
  \setbox0=\vbox{\hsize=\maxdimen
    \noindent\color@begingroup\nameuse{footfmt#1}{#1}{#2}\color@endgroup}%
  \setbox0=\hbox{\unvxh0[#1]}%
  \dp0=\z@%
  \ht0=\csname footfudgefactor#1\endcsname\wd0
  \box0
  \penalty0}}%
\parafootfmtX \parafootfmtX{(series)}
\newcommand*{\parafootfmtX}[2]{%
  \protected@edef\@currentlabel{\nameuse{@thefnmark#1}}%
  \insertparafootsep{#1}%
  \ledsetnormalparstuff{\csuse{notenumfontX@#1}\csuse{notenumfontX@#1}\@nameuse{footfootmark#1}\strut\enspace\ifmpfootins\#1\else\penalty-10\fi}}%
\para@footgroupX \para@footgroupX{(series)}
\newcommand*{\para@footgroupX}[1]{%
  \unvbox\csname footins#1\endcsname
  \ifcsstring{raggedX@#1}{L}{\RaggedLeft}{}%
  \ifcsstring{raggedX@#1}{R}{\RaggedRight}{}%
  \makehboxofhboxes
  \setbox0=\hbox{\unhbox0 \removehboxes}%
  \csuse{notefontsizeX@#1}
26 Footnotes’ width for two columns

We define here some commands which make sense only with eledpar, but must be called when defining notes parameters. These commands change the width of block notes to allow them to have the same size than two parallel columns.

These two commands are called at the beginning of critical or familiar notes groups. They set, if the option is enabled, the \hsize. They are also called at the on the setup for paragraphed notes.
These two commands set the position of the critical / familiar footnotes, depending on the hooks `Xnoteswidthliketwocolumns` and `notesXwidthliketwocolumns`. They call commands which are defined only in `eledpar`, because this feature has no sense without `eledpar`.

The `\fnpos` and `\mpfnpos` simply place their arguments in `\@fnpos` and `\@mpfnpos`, which will be used later in the output routine.

```
\setnotespositionliketwocolumns@  \setnotesXpositionliketwocolumns@
\setnotespositionliketwocolumns@  \setnotesXpositionliketwocolumns@
```

### 27 Footnotes’ order

```
\fnpos  The `\fnpos` and `\mpfnpos` simply place their arguments in `\@fnpos` and `\@mpfnpos`, which will be used later in the output routine.
\mpfnpos
\@fnpos  \def\@fnpos{familiar-critical}
\@mpfnpos  \def\@mpfnpos{critical-familiar}
```
28 Footnotes’ rule

Because the footnotes’ rules can be shifted to the right when footnotes are set like two columns, we don’t print them directly, but we put them in a \vbox.

\printXfootnoterule
\printfootnoteXrule

29 Footnotes’ output

\doextrafeeti We have to add all the new kinds of familiar footnotes to the output routine. These are the class 1 feet.
\doreinxtrafeeti
\addfootinsX Juste for backward compatibility: print a warning message.

\newcommand{\fnpos}{1}{\xdef@fnpos{#1}}
\newcommand{\mpfnpos}{1}{\xdef@mpfnpos{#1}}

\newcommand\printXfootnoterule{\nointerlineskip\moveleft-\leftskip\vbox{\csuse{#1footnoterule}}\nointerlineskip}
\newcommand\printfootnoteXrule{\nointerlineskip\moveleft-\leftskip\vbox{\csuse{footnoterule#1}}\nointerlineskip}

\newcommand*{\doextrafeeti}{\setbox\@outputbox \vbox{\unvbox\@outputbox\def\do##1{\ifvoid\csuse{footins##1}\else\csuse{footstart##1}{##1}\csuse{footgroup##1}{##1}\fi}\dolistloop\@series}}
\newcommand*{\doreinxtrafeeti}{\def\do##1{\ifvoid\csuse{footins##1}\else\insert\csuse{footins##1}{\unvbox\csuse{footins##1}}\fi}\dolistloop\@series}

\newcommand*{\addfootinsX}{\led@warn@AddfootinsXObsolete\footnormalX{#1}\g@addto@macro{\doextrafeeti}{\setbox\@outputbox \vbox{\unvbox\csuse{footins##1}\ifvoid\csuse{footins##1}\else\insert\csuse{footins##1}{\unvbox\csuse{footins##1}}\fi}\dolistloop\@series}}
30 Endnotes

Endnotes of all varieties are saved up in a file, typically named \jobname\end. \if1\dend@ is the output stream number for this file, and \if1\dend@ is a flag that’s true when the file is open.
\dend@open is used by \beginnumbering to do everything that’s necessary for the endnotes at the start of each section: it opens the \dend@ file, if necessary, and writes the section number to the endnote file.
\dend@stuff is used by \endprint to display the endnotes after they’ve been written. The endnote file also contains \section commands, which supply the section numbers from the main text; standard \eledmac does nothing with this information, but it’s there if you want to write custom macros to do something with it.

\endprint The \endprint here is nearly identical in its functioning to \normalfootfmt.
\section The endnote file also contains \section commands, which supply the section numbers from the main text; standard \eledmac does nothing with this information, but it’s there if you want to write custom macros to do something with it.
\setprintendlines  The \texttt{printendlines} macro is similar to \texttt{printlines} but is for printing endnotes rather than footnotes.

The principal difference between foot- and endnotes is that footnotes are printed on the page where they are specified but endnotes are printed at a different point in the document. We need an indication of the source of an endnote; \texttt{setprintendlines} provides this by always printing the page number. The coding is slightly simpler than \texttt{setprintlines}.

First of all, we print the second page number only if the ending page number is different from the starting page number.

\begin{verbatim}
\newcommand*{\setprintendlines}[6]{% 
  \l@dpnumfalse \l@d@dashfalse 
  \ifnum#4=#1 \else 
    \l@dpnumtrue 
    \l@d@dashtrue 
  \fi 
  \ifl@d@pnum \l@d@elintrue \else \l@d@elinfalse \fi 
  \ifnum#2=#5 \else 
    \l@d@elintrue 
    \l@d@dashtrue 
  \fi 
  \l@d@ssubfalse 
  \ifnum#3=0 \else 
    \l@d@ssubtrue 
  \fi 
  \l@d@eslfalse 
  \ifnum#6=0 \else 
    \ifl@d@elin \l@d@esltrue \else \l@d@eslfalse \fi 
    \else 
      \l@d@esltrue 
    \fi 
    \l@d@dashtrue 
    \fi 
  \fi}
\end{verbatim}

We print the ending line number if: (1) we’re printing the ending page number, or (2) it’s different from the starting line number.

\begin{verbatim}
\def\printendlines#1|#2|#3|#4|#5|#6|#7|{
  \begingroup 
  \setprintendlines{#1}{#2}{#3}{#4}{#5}{#6}\
  \endgroup 
}\end{verbatim}

The only subtlety left here is when to print a period between numbers. But the only instance in which this is tricky is for the ending sub-line number: it could be coming after the starting sub-line number (in which case we want only the dash) or after an ending line number (in which case we need to insert a period).
\printnpnum A macro to print a page number in an endnote.
\newcommand*{\printnpnum}[1]{p.#1) }
\doendnotes
\doendnotes is the command you use to print one series of endnotes; it takes one argument: the series letter of the note series you want to print.
\newcommand*{\doendnotes}[1]{\l@end@close
\begingroup
\makeatletter
\expandafter\let\csname #1end\endcsname=\endprint
\input\jobname.end
\endgroup}
\noendnotes
You can say \noendnotes before the first \beginumbering in your file if if you will not use any of the endnote commands: this will suppress the creation of an .end file. If you do have some lingering endnote commands in your file, the notes will be written to your terminal and to the log file.
\newcommand*{\noendnotes}{\global\let\l@end@stuff=\relax
\global\chardef\l@end=16 }

31 Generate series

In this section, X means the name of the series (A, B etc.)
\series\series creates one more newseries. It’s the public command, which just loops on the private command \newseries@.
\newcommand{\newseries}[1]{%
\def\do##1{\newseries@{##1}}%
\docsvlist{#1}
}%
\@series The \series@ macro is an etoolbox list, which contains the name of all series.
\newcommand{\@series}{ }

The command \newseries@\series creates a new series of the footnote.
31.1 Test if series is still existing

\ifinlist{#1}{\@series}{\led@warn@SeriesStillExist{#1}}\%

\newtoggle{Xlemmadisablefontselection@#1}
\newtoggle{Xendlemmadisablefontselection@#1}
\csgdef{Xhangindent@#1}{0pt}
\csgdef{hangindentX@#1}{0pt}
\csgdef{Xragged@#1}{}
\csgdef{raggedX@#1}{}
\csgdef{hsizetwocol@#1}{0.45 \hsize}
\csgdef{hsizetwocolX@#1}{0.45 \hsize}
\csgdef{hsizethreecol@#1}{0.3 \hsize}
\csgdef{hsizethreecolX@#1}{0.3 \hsize}
\csgdef{Xnotenumfont@#1}{}\notenumfont
\csgdef{Xendnotenumfont@#1}{}\notenumfont
\csgdef{notenumfontX@#1}{}\notenumfont
\csgdef{Xnotefontsize@#1}{\notefontsetup}
\csgdef{notefontsizeX@#1}{\notefontsetup}
\csgdef{Xendnotefontsize@#1}{\notefontsetup}
\csgdef{bhooknoteX@#1}{}
\csgdef{bhookXnote@#1}{}
\csgdef{bhookXendnote@#1}{}
\csgdef{boxlinenum@#1}{0pt}
\csgdef{boxsymlinenum@#1}{0pt}
\newtoggle{numberonlyfirstinline@#1}
\newtoggle{numberonlyfirstintwolines@#1}
\newtoggle{onlypstartinfootnote@#1}
\newtoggle{pstartinfootnoteeverytime@#1}
\newtoggle{pstartinfootnote@#1}
\csgdef{symlinenum@#1}{\symplinenum}
\newtoggle{nonumberinfootnote@#1}
\csgdef{beforenumberinfootnote@#1}{0pt}
\csgdef{afternumberinfootnote@#1}{0.5em}
\newtoggle{nonbreakableafternumber@#1}
\csgdef{beforesymlinenum@#1}{\csuse{beforenumberinfootnote@#1}}
\csgdef{aftersymlinenum@#1}{\csuse{afternumberinfootnote@#1}}
\csgdef{inplaceofnumber@#1}{1em}
\global\cslet{lemmaseparator@#1}{\rbracket}
\csgdef{beforelemmaseparator@#1}{0em}
\csgdef{afterlemmaseparator@#1}{0.5em}
\csgdef{inplaceoflemmaseparator@#1}{1em}
\csgdef{beforeXNotes@#1}{1.2em \@plus .6em \@minus .6em}
\csgdef{beforenotesX@#1}{1.2em \@plus .6em \@minus .6em}
\csgdef{afterXrule@#1}{Opt}
\csgdef{afterruleX@#1}{Opt}
31.3 Create inserts, needed to add notes in foot

Concerning inserts, see chapter 15 of the TeXBook by D. Knuth

31.4 Create commands for critical apparatus, \texttt{Xfootnote}

Note the double # in command: it’s because command is made inside another command.
\section{Create tools for familiar footnotes (\texttt{\footnotex})}

First, create the \texttt{\footnotex} command.

\begin{verbatim}
\global\expandafter\newcommand\csname footnote#1\endcsname[1]{% 
  \begingroup 
  \newcommand{\content}{##1} 
  \stepcounter{footnote#1} 
  \protected@csxdef{@thefnmark#1}{\csuse{thefootnote#1}} 
  \csuse{@footnotemark#1} 
  \csuse{vfootnote#1}{#1}{\expandonce\content}\m@mmf@prepare 
  \endgroup } 
\end{verbatim}

The counters.

\begin{verbatim}
\newcounter{footnote#1} 
\global\expandafter\renewcommand\csname thefootnote#1\endcsname{\arabic{footnote#1}} 
\end{verbatim}

\section{The endnotes}

The \texttt{\Xendnote} macro functions to write one endnote to the .end file. We change \texttt{\newlinechar} so that in the file every space becomes the start of a new line; this generally ensures that a long note doesn’t exceed restrictions on the length of lines in files.

\begin{verbatim}
\global\expandafter\newcommand\csname #1endnote\endcsname[2]{{\newlinechar='40 
  \global\@noneed\Xfootnotetrue% 
  \newcommand{\content}{##1}{\@noneed\Xfootnotetrue% 
    \immediate\write\l@d@end{\expandafter\string\csname #1end\endcsname% 
      \ifnumberedpar@\l@d@nums\fi} %3 
      \ifnumberedpar@\expandonce\@tag\fi} %3 
    \expandonce\content}{#1}}\ignorespaces% 
\end{verbatim}
\texttt{\textbackslash Xendnote} commands called \texttt{\textbackslash Xend} commands on to the endnote file; these are analogous to the various \texttt{footfmt} commands above, and they take the same arguments. When we process this file, we’ll want to pick out the notes of one series and ignore all the rest. To do that, we equate the \texttt{end} command for the series we want to \texttt{\textbackslash endprint}, and leave the rest equated to \texttt{\@gobblethree}, which just skips over its three arguments.\textsuperscript{27}

\begin{verbatim}
\global\cslet{#1end}{\@gobblefour}
\% We need to be able to modify \Eledmac’s footnote macros and restore their
\global\csletcs{#1@@footnote}{#1footnote}
\% \cs{Stock series in \cs{@series}
\begin{macrocode}
\listxadd{\@series}{#1}
\end{macrocode}
\% End of \newseries
\end{verbatim}

### 31.7 Init standards series (A,B,C,D,E,Z)

\begin{verbatim}
\newseries{A,B,C,D,E,Z}
\end{verbatim}

### 31.8 Some tools

\texttt{\textbackslash firstseries \textbackslash seriesatbegin\{\langle s\rangle\}} changes the order of series, to put the series \langle s\rangle at the beginning of the list. The series can be the result of a command.

\begin{verbatim}
\newcommand{\seriesatbegin}[1]{
  \edef\series{#1}
  \def\new{}
  \listeadd{\new}{\series}
  \def\dolistloop{\@series}
  \xdef\@series{\new}
}
\end{verbatim}

\texttt{\textbackslash seriesatend} And \texttt{\textbackslash seriesatend} moves the series to the end of the list.

\begin{verbatim}
\newcommand{\seriesatend}[1]{
  \edef\series{#1}
  \def\new{}
  \dolistloop{\@series}
  \xdef\@series{\new}
}
\end{verbatim}

\textsuperscript{27}Christophe Hebeisen (christophe.hebeisen@a3.epfl.ch) emailed on 2003/11/05 to say he had found that \texttt{\@gobblethree} was also defined in the amsfonts package.
31.8 Some tools

\subsection{Display}
\changes{v1.0}{2012/09/15}{New generic commands to customize footnote display.}
\subsection{Options}
\changes{v1.1}{2012/09/25}{\cs{settoggle@series} switch the global value of the toggle, not only the local value.}
\changes{v1.13.0}{2014/09/16}{\cs{settoggle@series} can take an optional arguments to reload series setup.}
\settoggle@series\marg{series}\marg{toggle}\marg{value} is a generic command to switch toggles for some series.

\begin{macrocode}
\newcommandx{\settoggle@series}[4][4]{%\def\do##1{\global\settoggle{#2@##1}{#3}\ifstrequal{#4}{reload}{\csuse{foot\csuse{series@display##1}}{##1}\csuse{foot\csuse{series@displayX##1}}{##1}}{}\ifstrempty{#1}{\dolistloop{\@series}}{\docsvlist{#1}}}}%
\newhookcommand@series\newhookcommand@series\command names is a generic command to add new commands for hooks, like \hsizetwocol.
\newcommandx{\newhookcommand@series}[1]{%\def\do##1{\global\expandafter\newcommand\expandafter*{#1}[2][2]{\setcommand@series{##1}{#1}{##2}}}}%
\end{macrocode}
\newhookcommand@series{Xhangindent}
\newhookcommand@series{hangindentX}
\newhookcommand@series{Xragged}
\newhookcommand@series{raggedX}
\newhookcommand@series{hsizetwocol}
\newhookcommand@series{hsizethreecol}
\newhookcommand@series{hsizetwocolX}
\newhookcommand@series{hsizethreecolX}
\newhookcommand@series{Xnotenumfont}
\newhookcommand@series{notenumfontX}
\newhookcommand@series{Xendnotenumfont}
\newhookcommand@series{bhooknoteX}
\newhookcommand@series{bhookXnote}
\newhookcommand@series{bhookXendnote}
\newhookcommand@series{Xnotefontsize}
\newhookcommand@series{notefontsizeX}
\newhookcommand@series{Xendnotefontsize}
\newhookcommand@series{boxlinenum}
\newhookcommand@series{boxsymlinenum}
\newhookcommand@series{parafootsep}
\newhookcommand@series{symlinenum}
\newhookcommand@series{beforenumberinfootnote}
\newhookcommand@series{afternumberinfootnote}
\newhookcommand@series{beforesymlinenum}
31.8 Some tools

\newhookcommand@series{aftersymlinenum}
\newhookcommand@series{inplaceofnumber}
\newhookcommand@series{lemmaseparator}
\newhookcommand@series{beforelemmaseparator}
\newhookcommand@series{afterlemmaseparator}
\newhookcommand@series{inplaceoflemmaseparator}
\newhookcommand@series{afternote}
\newhookcommand@series{txtbeforeXnotes}
\newhookcommand@series{afterruleX}
\newhookcommand@series{afterXrule}
\newhookcommand@series@reload
\newhookcommand@series@reload does the same thing as \newhookcommand@series but the commands created by this macro also reload the series displaying (normal, paragraph, twocol, threecol).
\newcommand{\newhookcommand@series@reload}{\%\global\expandafter\newcommand\expandafter\*{\csname #1\endcsname}{\%\setcommand@series{##1}{#1}{##2}{\%\{reload\}\%})}
\newhookcommand@series@reload{beforeXnotes}
\newhookcommand@series@reload{beforenotesX}
\newhookcommand@series@reload{maxhnotesX}
\newhookcommand@series@reload{maxhXnotes}
\end{macrocode}
\begin{macro}{\newhooktoggle@series}
\cs{newhooktoggle@series}\cs{command names} is a generic command to add new commands for new toggle
\newcommand{\newhooktoggle@series}{\%\global\expandafter\newcommandx\expandafter{\csname #1\endcsname}{\%\settoggle@series{##1}{#1}{##2}{\%\{#1\{##1\endcsname}{\%\{#2\}}}\begin{macrocode}
\newhooktoggle@series{numberonlyfirstinline}
\newhooktoggle@series{numberonlyfirstintwolines}
\newhooktoggle@series{nonumberinfootnote}
\newhooktoggle@series{pstartinfootnote}
\newhooktoggle@series \newhookcommand@toggle@reload does the same thing as \newhooktoggle@series but the commands created by this macro also reload the series displaying (normal, paragraph, twocol, threecol).

\newhooktoggle@series@reload
\newhooktoggle@series@reload

31.9 Old commands, kept for backward compatibility
The next commands are kept for ascendant compatibility, but should not be used anymore.

\notenumfont
\notefontsetup
\ifledplinenum
\symplinenum
\newif\ifledplinenum
\ledplinenumtrue
\newcommand*{\symplinenum}{ }

31.10 Hooks for a particular footnote
\nonum@ \nonum@ toggle is used to disable line number printing in a particular footnote.
\newtoggle{nonum@}
\nosep@
\nosep@ toggle is used to disable the lemma separator in a particular footnote.
\newtoggle{nosep@}

31.11 Alias
\nolemmaseparator \nolemmaseparator[(series)] is just an alias for \lemmaseparator[(series)]{}
\newcommand*{\nolemmaseparator}{(series)[]}
\interparanoteglue
\ipn@skip The \ipn@skip skip and \interparanoteglue command are kept for backward compatibility, but should not be used anymore.
\newskip{\ipn@skip}
\newcommand*{\interparanoteglue}{%
31.12 Line number printing

The \parafootfmtsep macro is kept for backward compatibility. It is default value of \parafootsep\textregistered.

\newcommand{\parafootfmtsep}{

31.12 Line number printing

The \printlinefootnote macro is called in each \texttt{footfmt} command. It controls whether the line number is printed or not, according to the previous options. Its first argument is the information about lines; its second is the series of the footnote. The printing of the line number is shared in \printlinefootnotenumbers.

\newcommand{\printlinefootnote}[2]{
\def\extractline@##1|##2|##3|##4|##5|##6|##7|{##2}\
\def\extractsubline@##1|##2|##3|##4|##5|##6|##7|{##3}\
\def\extractendline@##1|##2|##3|##4|##5|##6|##7|{##5}\
\def\extractendsubline@##1|##2|##3|##4|##5|##6|##7|{##6}\
\iftoggle{numberonlyfirstintwolines@#2}{%
 \edef\lineinfo@{\extractline@ #1| - \extractsubline@ #1| - \extractendline@ #1| - \extractendsubline@ #1|}%
}{%
 \edef\lineinfo@{\extractline@ #1| - \extractsubline@ #1|}%
}\
\iftoggle{nonum@}{%Try if the line number must printed for this specific not (by default, yes)
 \hspace{\csuse{inplaceofnumber@#2}}%
}{%
}{%
\iftoggle{nonumberinfootnote@#2}%Try if the line number must printed (by default, yes)
 \hspace{\csuse{inplaceofnumber@#2}}%
}{%
\iftoggle{numberonlyfirstinline@#2}% If for this series the line number must be printed only in the first time.
 \ifcdef{prevline#2}{%Be sure the \prevline exists.
 \ifcsequal{prevline#2}{lineinfo@}%Try it
 {%
 \hspace{\csuse{inplaceofnumber@#2}}%
 \hspace{\csuse{beforesymlinenum@#2}}\csuse{Xnotenumfont@#2}\
 \ifdimequal{\csuse{boxsymlinenum@#2}}{0pt}%
 \hspace{\csuse{symlinenum@#2}}\
 \hbox to \csuse{boxsymlinenum@#2}{\csuse{symlinenum@#2}\hfill}%
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
}{%}
\texttt{\printlinefootnotearea}  This macro prints the space before the line number, changes the font, then prints the line number and the space after it. It is called by \texttt{\printlinefootnotenote} depending of the options about repeating line numbers. The first argument is line information, the second is the notes series (A, B, C, etc.)
% This macro prints, if needed, the pstart number and the line number.
% The first argument is line information, the second is the notes series (A, B, C. etc.)
% The previous \cs{boxlinefootnote} calls it.
\begin{macrocode}
\newcommand{\printlinefootnotenumbers}[2]{% 
  \ifboolexpr{\bgroup
    (\togg{pstartinfootnote@#2} and \bool{numberpstart})%
  \egroup or \togg{pstartinfootnoteeverytime@#2}}{%
    \printpstart{}% 
  \iftoggle{onlypstartinfootnote@#2}{}{\printlines#1|}%
  }% 
\printbeforenumberinfootnote
\end{macrocode}

\printbeforenumberinfootnote
This macro prints a space (before the line number) in footnote. It is called by \printlinefootnotearea. Its only argument is the series
\begin{macrocode}
\newcommand{\printbeforenumberinfootnote}[1]{% 
  \hspace{\csuse{beforenumberinfootnote@#1}}% 
\end{macrocode}

\printafternumberinfootnote
This macro prints the space, adding eventually a \nobreak, after the line number, in footnote. It is called by \printlinefootnotearea. Its only argument is the series
\begin{macrocode}
\newcommand{\printafternumberinfootnote}[1]{% 
  \iftoggle{nonbreakableafternumber@#1}{\nobreak}{}% 
  \hspace{\csuse{afternumberinfootnote@#1}}% 
\end{macrocode}

32 Output routine

Now we begin the output routine and associated things.

\begin{macrocode}
\pageno\newcommand{\pageno}{1}\advancepageno\makeheadline\makefootline\dosupereject\pageno\newcommand{\pageno}{0}\ifnum\pageno<\z@\global\advance\pageno\@ne\else\global\advance\pageno\m@ne\fi
\output{\edmac@output}
\def\edmac@output{\shipout\vbox{\normalpars
\end{macrocode}
\def\pagecontents{\page@start
  \ifvoid\topins\else\unvbox\topins\fi
  \dimen@=\dp@cclv \unvbox\@cclv \% open up \box255
  \do@feet
  \ifr@ggedbottom \kern-\dimen@ \vfil \fi
}

\do@feet ships out all the footnotes. Standard EDMAC has only five feet, but there is nothing in principal to prevent you from creating an arachnoid or centipede edition; straightforward modifications of EDMAC are all that’s required. However, the myriapedal edition is ruled out by cTeX limitations: the number of insertion classes is limited to $2^{16}$.

With luck we might only have to change \@makecol and \@reinserts. The kernel definition of these, and perhaps some other things, is:

\gdef \@makecol {%
  \ifvoid\footins
    \setbox\@outputbox \box\@cclv
  \else
    \setbox\@outputbox \vbox {%
      \boxmaxdepth \@maxdepth
      \@tempdima\dp@cclv
      \unvbox \@cclv
      \vskip \skip\footins
      \color@begingroup
      \normalcolor
      \footnoterule
      \unvbox \footins
      \color@endgroup
    }%
  \fi
  \xdef\@freelist{\@freelist\@midlist}%
  \global \let \@midlist \@empty
  \@combinefloats
  \ifvbox\@kludgeins
    \@makespecialcolbox
  \else
    \setbox\@outputbox \vbox to\@colht {%
      \@texttop
      \dimen@ \dp\@outputbox
      \unvbox\@outputbox
      \vskip -\dimen@\@textbottom
    }%
  \fi
}
Now we start actually changing things.

These macros are defined in the \texttt{memoir} class and form part of the definition of \texttt{@makecol}.

This is a partitioned version of the ‘standard’ \texttt{@makecol}, with the initial code put into another macro.

The \texttt@ifFN@bottom macro is defined by the \texttt{footmisc} package. If this package is not loaded, we define it.

This macro essentially holds the initial portion of the kernel \texttt{@makecol} code.
That's the end of the copy of the kernel code. We finally call a macro to handle all the additional *EDMAC* feet.

```latex
\l@ddoxtrafeet
```

\textbf{\textbackslash doxtrafeet} \textbf{\textbackslash doxtrafeet} is the code extending \texttt{\@makecol} to cater for the extra \texttt{eledmac} feet. We have two classes of extra footnotes. By default, we order the footnote inserts so that the regular footnotes are first, then class 1 (familiar footnotes) and finally class 2 (critical footnotes).

```latex
\newcommand*{\l@ddoxtrafeet}{% 
  \IfStrEq{familiar-critical}{\@fnpos} {\doxtrafeeti\doxtrafeetii}% 
  {\IfStrEq{critical-familiar}{\@fnpos}% 
    {\doxtrafeetii\doxtrafeeti}% 
    {\doxtrafeeti\doxtrafeetii}% 
  }% 
}%;
```

\textbf{\textbackslash doxtrafeetii} \textbf{\textbackslash doxtrafeetii} is the code extending \texttt{\@makecol} to cater for the extra critical feet (class 2 feet). NOTE: the code is likely to be ‘featurefull’.

```latex
\newcommand*{\doxtrafeetii}{% 
  \dolistloop{\@series}
}
```

\textbf{\texttt{\@opxtrafeetii}} The extra critical feet to be added to the output.

```latex
\newcommand*{\@opxtrafeetii}{% 
  \def\do##1{\ifvoid\csuse{##1footins}\else\csuse{##1footstart}{##1}\csuse{##1footgroup}{##1}}\csuse{##1footstart}{##1}\csuse{##1footgroup}{##1}}
  \dolistloop{\@series}
}
\doreinxtrafeet \doreinxtrafeet is the code for catering for the extra footnotes within \reinserts. The implementation may well have to change. We use the same classes and ordering as in \doreinxtrafeet.

\doreinxtrafeetii \doreinxtrafeetii is the code for catering for the class 2 extra critical footnotes within \reinserts. The implementation may well have to change.

\l@doreinserts And here is the modified version of \reinserts.

\addfootins \addfootins is for backward compatibility, but should’nt be used anymore.
It turns out that `\@doclearpage` also needs modifying.

We have to check if there are any leftover feet. `\@led@extranofeet` is a hook for handling further footnotes.

As `memoir` is not loaded we have to do it all here.

As `memoir` is not loaded we have to do it all here.
33 Cross referencing

Peter Wilson have rewritten portions of the code in this section so that the LaTeX .aux file is used. This will also handle \included files.

Further, I have renamed some of the original EDMAC macros so that they do not clash with the LaTeX label/ref commands (EDMAC and LaTeX use very different mechanisms). In particular, the original EDMAC \label and \pageref have been renamed as \edlabel and \edpageref respectively.

You can mark a place in the text using a command of the form \edlabel{foo}, and later refer to it using the label foo by saying \edpageref{foo}, or \lineref{foo} or \sublineref{foo}. These reference commands will produce, respectively, the page, line and sub-line on which the \edlabel{foo} command occurred.

The reference macros warn you if a reference is made to an undefined label. If foo has been used as a label before, the \edlabel{foo} command will issue a complaint; subsequent \edpageref and \lineref commands will refer to the latest occurrence of \label{foo}.

\labelref@list Set up a new list, \labelref@list, to hold the page, line and sub-line numbers
for each label.
\list@create{\labelref@list}

\zz@@@ A convenience macro to zero two labeling counters in one go.
\newcommand*{\zz@@@}{000|000|000} % set three counters to zero in one go
\newcommand*{\zz@@@}{000|000} % set two counters to zero in one go

\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.\footnote{The remaining macros in this section were kindly revised by Wayne Sullivan, who substantially improved their efficiency and flexibility.}

This version of the original EDMAC \label uses \@bsphack and \@esphack to eliminate extra space problems and also the LaTeX write methods for the .aux file.

Jesse Billett\footnote{\texttt{jdb43@cam.ac.uk} via the ctt thread ‘ledmac cross referencing’, 25 August 2003.} found that the original code could be off by several pages. This version, hopefully cures that, and also allows for non-arabic page numbering.

\newcommand*{\edlabel}{[1]}{%
  \ifl@dpairing\ifautopar\fi\fi\fi% 
  \@bsphack% 
  \ifledRcol% 
    \write\linenum@outR{\string\@lab}% 
  \else% 
    \g@p\labelref@listR\to\label@refs% 
  \fi% 
  \ifvmode% 
    \advancelabel@refs% 
  \fi% 
}

Use code from the kernel \label command to write the correct page number (it seems possible that the original EDMAC’s \page@num scheme might also have had problems in this area). Also define an hypertarget if \hyperref package is loaded.

\protected@write@auxout{}% 
\string\l@dmake@labelsR\space\thepage|\label@refs|\the\c@pstartR|{#1}% 
\ifndef{\hypertarget}{\hypertarget{#1}{}}{}% 
\else% 
  \write\linenum@out{\string\@lab}% 
  \ifx\labelref@list\empty% 
    \xdef\label@refs{\zz@@@}% 
  \else% 
    \edef{\xdef}{\label@refs{\zz@@@}}% 
  \fi% 
\fi%
In cases where \label is the first element in a paragraph, we have a problem with line counts, because line counts change only at the first horizontal box of the paragraph. Hence, we need to test \label if it occurs at the start of a paragraph. To do so, we use ifvmode. If the test is true, we must advance by one unit the amount of text we write into the .aux file. We do so using \advancelabel@refs command.

\newcounter{line}
\newcounter{subline}
\newcommand{\advancelabel@refs}{%\setcounter{line}{\expandafter\labelrefsparseline\label@refs}%\stepcounter{line}%\ifsublines@%\setcounter{subline}{\expandafter\labelrefsparsesubline\label@refs}%\stepcounter{subline}{1}%\def\label@refs{\theline|\thesubline}%\else%\def\label@refs{\theline|0}%\fi%}
\def\labelrefsparseline#1|#2{#1}
\def\labelrefsparsesubline#1|#2{#2}
\l@dmake@labels
In cases where \label is the first element in a paragraph, we have a problem with line counts, because line counts change only at the first horizontal box of the paragraph. Hence, we need to test \label if it occurs at the start of a paragraph. To do so, we use ifvmode. If the test is true, we must advance by one unit the amount of text we write into the .aux file. We do so using \advancelabel@refs command.

\newcounter{line}
\newcounter{subline}
\newcommand{\advancelabel@refs}{%\setcounter{line}{\expandafter\labelrefsparseline\label@refs}%\stepcounter{line}%\ifsublines@%\setcounter{subline}{\expandafter\labelrefsparsesubline\label@refs}%\stepcounter{subline}{1}%\def\label@refs{\theline|\thesubline}%\else%\def\label@refs{\theline|0}%\fi%}
\def\labelrefsparseline#1|#2{#1}
\def\labelrefsparsesubline#1|#2{#2}
\l@dmake@labels
The \l@dmake@labels macro gets executed when the labels file is read. For each label it defines a macro, whose name is made up partly from the label you supplied, that contains the page, line and sub-line numbers. But first it checks to see whether the label has already been used (and complains if it has).

The initial use of \newcommand is to catch if \l@dmake@labels has been previously defined (by a class or package).
LaTeX reads the aux file at both the beginning and end of the document, so we have to switch off duplicate label checking after the first time the file is read.

\AtBeginDocument{\def\l@dmake@labels#1|#2|#3|#4|#5{}%}

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, \nl, and the \sub@on and \sub@off commands appearing in the \linenum@out file.

LaTeX uses the page counter for page numbers. However, it appears that this is not the right place to grab the page number. That task is now done in the \edlabel macro. This version of \lab appends just the current line and sub-line numbers to \labelref@list.

\newcommand*{\lab}{\xright@appenditem{\linenumrep{\line@num}|\ifsplines@\sublinenumrep{\subline@num}\else0\fi}\to\labelref@list}

\wrap@edcrossref is called around all edlemac crossref commands, except those which start with x. It adds the hyperlink.

\newrobustcmd{\wrap@edcrossref}[2]{\ifdef{\hyperlink}{\hyperlink{#1}{#2}}{#2}}

If the specified label exists, \edpageref gives its page number. For this reference command, as for the other two, a special version with prefix x is provided for use in places where the command is to be scanned as a number, as in \linenum. These special versions have two limitations: they don’t print error messages if the reference is unknown, and they can’t appear as the first label or reference command in the file; you must ensure that a \edlabel or a normal reference command appears first, or these x-commands will always return zeros. LaTeX already defines a \pageref, so changing the name to \edpageref.

\edlineref is the \lineref command, but with prefix x.

If the specified label exists, \lineref gives its line number.
\sublineref If the specified label exists, \sublineref gives its sub-line number.
\xsublineref
\newcommand*{\sublineref}[1]{\l@dref@undefined{#1}\wrap@edcrossref{#1}{\l@getref@num{3}{#1}}}
\newcommand*{\xsublineref}[1]{\l@getref@num{3}{#1}}

\pstartref If the specified label exists, \pstartref gives its pstart number.
\xpstartref
\newcommand*{\pstartref}[1]{\l@dref@undefined{#1}\wrap@edcrossref{#1}{\l@getref@num{4}{#1}}}
\newcommand*{\xpstartref}[1]{\l@getref@num{4}{#1}}

The next three macros are used by the referencing commands above, and do the
job of extracting the right numbers from the label macro that contains the page,
line, and sub-line number.

\l@getref@undefined The \l@getref@undefined macro is called when you refer to a label with the normal
referencing macros. Its argument is a label, and it just checks that the label has
been defined.
\newcommand*{\l@getref@undefined}[1]{%
\expandafter\ifx\csname the@label#1\endcsname\relax
\l@warn@RefUndefined{#1}%
\else
\expandafter\expandafter\expandafter
\l@dlabel@parse\csname the@label#2\endcsname|1%
\fi}

\l@dlabel@parse Notice that we slipped another | delimiter into the penultimate line of \l@getref@num,
to keep the ‘switch-number’ separate from the reference numbers. This | is used
as another parameter delimiter by \l@dlabel@parse, which extracts the appro-
priate number from its first arguments. The |-delimited arguments consist of the
expanded label-macro (three reference numbers), followed by the switch-number
(1, 2, or 3) which defines which of the earlier three numbers to pick out. (It was
earlier given as the first argument of \l@getref@num.)
\newcommand*{\l@dlabel@parse}{%}
\def\l@dlabel@parse#1|#2|#3|#4|#5{%
\ifcase #5%
The `\texttt{\textbackslash xxref}` command takes two arguments, both of which are labels, e.g., `\texttt{\textbackslash xxref\{mouse\}\{elephant\}}`. It first does some checking to make sure that the labels do exist (if one doesn’t, those numbers are set to zero). Then it calls `\texttt{\textbackslash linenum}` and sets the beginning page, line, and sub-line numbers to those of the place where `\texttt{\textbackslash label\{mouse\}}` was placed, and the ending numbers to those at `\texttt{\textbackslash label\{elephant\}}`. The point of this is to be able to manufacture footnote line references to passages which can’t be specified in the normal way as the first argument to `\texttt{\textbackslash critext}` for one reason or another. Using `\texttt{\textbackslash xxref}` in the second argument of `\texttt{\textbackslash critext}` lets you set things up at least semi-automatically.

\begin{verbatim}
\newcommand*{\texttt{\textbackslash xxref}}[2]{% 
\expandafter\ifx\csname the@label#1\endcsname \relax% 
\expandafter\let\csname the@@label#1\endcsname\zz@@% 
\else% \expandafter\def\csname the@@label#1\endcsname{\l@dgetref@num{1}{#1}|\l@dgetref@num{2}{#1}|\l@dgetref@num{3}{#1}}% \fi% \expandafter\ifx\csname the@label#2\endcsname \relax% \expandafter\let\csname the@@label#2\endcsname\zz@@% \else% \expandafter\def\csname the@@label#2\endcsname{\l@dgetref@num{1}{#2}|\l@dgetref@num{2}{#2}|\l@dgetref@num{3}{#2}}% \fi% \linenum{\csname the@@label#1\endcsname|\csname the@@label#2\endcsname}}
\end{verbatim}

\texttt{\textbackslash edmakelabel}Sometimes the `\texttt{\textbackslash edlabel}` command cannot be used to specify exactly the page and line desired; you can use the `\texttt{\textbackslash edmakelabel}` macro make your own label. For example, if you say `\texttt{\textbackslash edmakelabel\{elephant\}\{10|25|0\}}` you will have created a new label, and a later call to `\texttt{\textbackslash edpageref\{elephant\}}` would print ‘10’ and `\texttt{\textbackslash lineref\{elephant\}}` would print ‘25’. The sub-line number here is zero. `\texttt{\textbackslash edmakelabel}` takes a label, followed by a page and a line number(s) as arguments. LaTeX defines a `\texttt{\textbackslash makelabel}` macro which is used in lists. I’ve changed the name to `\texttt{\textbackslash edmakelabel}`.

\begin{verbatim}
\newcommand*{\texttt{\textbackslash edmakelabel}}[2]{\expandafter\xdef\csname the@label#1\endcsname#2}
\end{verbatim}

(If you are only going to refer to such a label using `\texttt{\textbackslash xxref}`, then you can omit entries in the same way as with `\texttt{\textbackslash linenum}` (see pp. 84 and 62), since `\texttt{\textbackslash xxref}` makes a call to `\texttt{\textbackslash linenum}` in order to do its work.)
34 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.

Changing \@xympar a little at least ensures that \marginpars in numbered text do not disturb the flow.

We provide side notes as replacement for \marginpar in numbered text.

These are the sidenote equivalents to \line@margin and \linenummargin for specifying which margin. The default is the right margin (opposite to the default for line numbers). \l@dgetsidenote@margin returns the number associated to side note margin:

left : 0
right : 1
outer : 2
inner : 3
\def\@tempb{outer}\
\ifx\@tempa\@tempb
\@l@dtempcntb \tw@
\else
\def\@tempb{inner}\
\ifx\@tempa\@tempb
\@l@dtempcntb \thr@@
\else
\led@warn@BadSidenotemargin
\@l@dtempcntb \m@ne
\fi
\fi
\fi
\sidenotemargin{right}
\l@dlp@rbox \l@drp@rbox
We need two boxes to store sidenote texts.
\newbox\l@dlp@rbox
\newbox\l@drp@rbox
\ledlsnotewidth \ledrsnotewidth
\ledlsnotesep \ledrsnotesep
\ledlsnotefontsetup \ledrsnotefontsetup
These specify the width of the left/right boxes (initialised to \marginparwidth, their distance from the text (initialised to \linenumsep, and the fonts used.
\newdimen\ledlsnotewidth \ledlsnotewidth=\marginparwidth
\newdimen\ledrsnotewidth \ledrsnotewidth=\marginparwidth
\newdimen\ledlsnotesep \ledlsnotesep=\linenumsep
\newdimen\ledrsnotesep \ledrsnotesep=\linenumsep
\newcommand*{\ledlsnotefontsetup}{\raggedleft\footnotesize}
\newcommand*{\ledrsnotefontsetup}{\raggedright\footnotesize}
\ledleftnote \ledrightnote \ledinnernote \ledouternote
\ledsidenote are the user commands for left, right, inner and outer sidenotes. The two last one are just alias for the two first one, depending of the page number. \ledsidenote{(text)}
is the command for a moveable sidenote.
\ledleftnote \ledrightnote \ledinnernote \ledouternote
\ledsidenote
\newcommand*{\ledleftnote}[1]{\edtext{}{\l@dlsnote{#1}}}
\newcommand*{\ledrightnote}[1]{\edtext{}{\l@drsnote{#1}}}
\newcommand*{\ledinnernote}[1]{\ifodd\c@page% Do not use \page@num, because it is not yet calculated when command is called
\ledleftnote{#1}\
\else\ledrightnote{#1}\fi}
\newcommand*{\ledouternote}[1]{%
The ‘footnotes’ for left, right, and moveable sidenotes. The whole scheme is reminiscent of the critical footnotes code.
\newif{\rightnoteup}
\ifrightnoteup
\rightnoteuptrue
\newcommand*{\l@dlnote}[1]{% 
\begingroup
\newcommand{\content}{#1}% \ifnumberedpar
\ifledRcol
\xright@appenditem{\noexpand\vl@dlnote{\expandonce\content}}% \to\inserts@listR 
\global\advance\insert@countR \@ne%
\else
\xright@appenditem{\noexpand\vl@dlnote{\expandonce\content}}% \to\inserts@list 
\global\advance\insert@count \@ne%
\fi 
\fi\ignorespaces\endgroup}
\newcommand*{\l@drsnote}[1]{% 
\begingroup
\newcommand{\content}{#1}% \ifnumberedpar
\ifledRcol
\xright@appenditem{\noexpand\vl@drsnote{\expandonce\content}}% \to\inserts@listR 
\global\advance\insert@countR \@ne%
\else
\xright@appenditem{\noexpand\vl@drsnote{\expandonce\content}}% \to\inserts@list 
\global\advance\insert@count \@ne%
\fi 
\fi\ignorespaces\endgroup}
\newcommand*{\l@dcsnote}[1]{% 
\begingroup
\newcommand{\content}{#1}% \ifnumberedpar
\ifledRcol
\xright@appenditem{\noexpand\vl@dcsnote{\expandonce\content}}% \to\inserts@listR 
\global\advance\insert@countR \@ne%
\else
\xright@appenditem{\noexpand\vl@dcsnote{\expandonce\content}}% \to\inserts@list 
\global\advance\insert@count \@ne%
\fi 
\fi\ignorespaces\endgroup}
\vl@dlsnote Put the left/right text into boxes, but just save the moveable text. \l@dcsnotetext, \l@dcsnotetext@l and \l@dcsnotetext@r are etoolbox lists which will store the content of side notes. We store the content in lists, because we need to loop later on them, in case many sidenote co-exist for the same line. That is there some special test to do, in order to:

- Store the content of \ledsidenote to \l@dcsnotetext in any cases.
- Store the content of \rightsidenote to:
  - \l@dcsnotetext if \ledsidenote is to be put on right.
  - \l@dcsnotetext@r if \ledsidenote is to be put on left.
- Store the content of \leftsidenote to:
  - \l@dcsnotetext if \ledsidenote is to be put on left.
  - \l@dcsnotetext@l if \ledsidenote is to be put on right.

\newcommand*{\vl@dlnote}[1]{%
  \ifledRcol\%
  \@l@dtempcntb=sidenote@marginR\%
  \ifnum\@l@dtempcntb>\@ne\
    \adv Page@numR by\page@numR\%
  \fi\%
  \else\%
  \@l@dtempcntb=sidenote@margin\%
  \ifnum\@l@dtempcntb>\@ne\
    \adv Page@num by\page@numR\%
  \fi\%
  \fi\%
  \else\%
  \@l@dtempcntb=sidenote@margin\%
  \ifnum\@l@dtempcntb>\@ne\
    \adv Page@num by\page@numR\%
  \fi\%
  \fi\%
  \ifodd\@l@dtempcntb\%
    \listgadd{\l@dcsnotetext@l}{#1}\%
  \else\%
    \listgadd{\l@dcsnotetext}{#1}\%
  \fi\%
\}
This macro is used to separate sidenotes of the same line.

\sidenotesep

This macro puts any moveable sidenote text into the left or right sidenote box, depending on which margin it is meant to go in. It’s a very much stripped down version of \affixlin@num.

Before do it, we concatenate all moveable sidenotes of the line, using \sidenotesep as separator. It’s the result that we put on the sidenote.

\affixside@note
And we do the same for left and right notes (not movable).

And we do the same for left and right notes (not movable).
35 Minipages and such

We can put footnotes into minipages. The preparatory code has been set up earlier, all that remains is to ensure that it is available inside a minipage box. This requires some alteration to the kernel code, specifically the \@iiiminipage and \endminipage macros. We’ll arrange this so that additional series can be easily added.

\l@feetbeginmini These will be the hooks in \@iiiminipage and \endminipage They can be extended to handle other things if necessary.
\l@feetendmini

\l@dedbeginmini These handle the initiation and closure of critical footnotes in a minipage environment.
\l@dedendmini
\@dfambeginmini These handle the initiation and closure of familiar footnotes in a minipage environment.
\newcommand*{\@dfambeginmini}{\def\do##1{\csletcs{vfootnote##1}{mpvfootnote##1}}}\dolistloop{\@series}
\newcommand*{\@dfamendmini}{\def\do##1{\ifvoid\csuse{mpfootins##1}\else\csuse{mpfootgroup##1}{##1}\fi}}\dolistloop{\@series}

\@iiiminipage This is our extended form of the kernel \@iiiminipage defined in ltboxes.dtx.
\def\@iiiminipage#1#2[#3]#4{\leavevmode\@pboxswfalse\setlength\@tempdima{#4}\def\@mpargs{{#1}{#2}[#3]{#4}}\setbox\@tempboxa\vbox\bgroup\color@begingroup\hsize\@tempdima\textwidth\hsize\columnwidth\hsize\@parboxrestore\color@begingroup\hsize\tempdima\textwidth\hsize\columnwidth\hsize\@parboxrestore\def\@mpfn{mpfootnote}\def\thempfn{\thempfootnote}\c@mpfootnote\z@\let\@footnotetext\@mpfootnotetext}

The next line is our addition to the original.
\let\@feetbeginmini% added\let\@listdepth\@mplistdepth\@mplistdepth\z@\let\@minipagerestore\@setminipage}

\endminipage This is our extended form of the kernel \endminipage defined in ltboxes.dtx.
\def\endminipage{%\par\unskip\ifvoid\@mpfootins\else\l@dunboxmpfoot\fi}

The next line is our addition to the original.
\l@dfeetendmini% added\@minipagefalse\@minipagemgroup\endgroup\expandafter\@iiiparbox\@mpargs{\unvbox\@tempboxa}}

\l@dunboxmpfoot
\newcommand*{\l@dunboxmpfoot}{\vskip\skip\@mpfootins}
This environment puts footnotes at the end, even if that happens to be in the middle of a page, or crossing a page boundary. It is a sort of unboxed, fixed width minipage.

\begin{ledgroup}
\[\langle pos \rangle\]{\langle width \rangle}

This environment puts footnotes at the end, even if that happens to be in the middle of a page, or crossing a page boundary. It is a sort of unboxed, variable \langle width \rangle minipage. The optional \langle pos \rangle controls the sideways position of numbered text.

\begin{ledgroupsized}{\langle pos \rangle}{\langle width \rangle}

This environment puts footnotes at the end, even if that happens to be in the middle of a page, or crossing a page boundary. It is a sort of unboxed, variable \langle width \rangle minipage. The optional \langle pos \rangle controls the sideways position of numbered text.

\begin{ledgroupsized}{2\hsize}{1\relax}

Set the various text measures.

\let\ledllfill\hfil
\let\ledrlfill\hfil
\def\@tempa{#1}\def\@tempb{l}

Left adjusted numbered lines
These boolean tests check if we are in the notes of a ledgroup. If we are, we don’t number the lines.

\newif{\ifledgroupnotesL}
\newif{\ifledgroupnotesR}

36 Indexing

Here’s some code for indexing using page & line numbers. First, ensure that imakeidx or indextools is loaded before eledmac.

\AtBeginDocument{%
\unless\if@imakeidx%
\@ifpackageloaded{imakeidx}{\led@error@ImakeidxAfterEledmac}{}%
\fi%
\unless\if@indextools%
\@ifpackageloaded{indextools}{\led@error@indextoolsAfterEledmac}{}%
\fi%
}%

\newcommand{\pagelinesep}{-}
\newcommand{\edindexlab}{$&}
\newcounter{labidx}
\setcounter{labidx}{0}

\ifledgroupnotesL This in order to get a correct line number we have to use the label/\ref mechanism. These macros are for that.
\ifledgroupnotesR
\doedindexlabel  This macro sets an \edlabel.

\newcommand{\doedindexlabel}{\stepcounter{labidx}\edlabel{\edindexlab\thelabidx}}

\thepageline  This macro makes up the page/line number combo from the label/ref.

\newcommand{\thepageline}{\thepage\pagelinesep\xlineref{\edindexlab\thelabidx}}

\thestartpageline\theendpageline  These macros make up the page/line start/end number when the \edindex command is called in critical notes.

\newcommand{\thestartpageline}{\l@dparsedstartpage\pagelinesep\l@dparsedstartline}
\newcommand{\theendpageline}{\l@dparsedendpage\pagelinesep\l@dparsedendline}

\if@edindex@fornote@true  This boolean test is switching at the beginning of each critical note, to allow indexing in this note.
\newif{\if@edindex@fornote@}

\prepare@edindex@fornote  This macro is called at the beginning of each critical note. It switches some parameters, to allow indexing in this note, with reference to page and line number.

\newcommand{\prepare@edindex@fornote}{\l@dp@rsefootspec#1|\@edindex@fornote@true}

\get@index@command  This macro is used to analyse if a text to be indexed has a command after a |.

\def{\get@index@command}{#1}{#2+}{\gdef{\index@txt}{#1}\gdef{\index@command}{#2}\xdef{\index@parenthesis}{}\IfBeginWith{\index@command}{(}{\StrGobbleLeft{\index@command}{1}[\index@command@]{\global\let{\index@command@}{\index@command}}\xdef{\index@parenthesis}{(}\}{}}\IfBeginWith{\index@command}{)}{\StrGobbleLeft{\index@command}{1}[\index@command@]{\global\let{\index@command@}{\index@command}}\xdef{\index@parenthesis}{)}\}{}}

\ledinnote\ledinnotehyperpage  These macros are used to specify that an index reference points to a note.

\newcommand{\ledinnote}{\csuse{#1}{#2\emph{n}}}\newcommand{\ledinnotehyperpage}{\csuse{#1}{\hyperpage{#2}\emph{n}}}

The \texttt{memoir} class provides more flexible indexing than the standard classes. We need different code if the \texttt{memoir} class is being used, except if \texttt{imakeidx} or \texttt{indextools} is used.

\texttt{\edindex} 36.1 Memoir compatibility

\texttt{\create@edindex@for@memoir} define the \texttt{\edindex} command and related tool when:

1. Memoir class is used.
2. AND \texttt{imakeidx} is not used.
3. AND \texttt{indextools} is not used.

Need to add the definition of \texttt{\edindex} to \texttt{\makeindex}, and initialise \texttt{\edindex} to do nothing. In this case \texttt{\edindex} has an optional argument. We use the hook provided in \texttt{memoir} v1.61.

\begin{verbatim}
\def\create@edindex@for@memoir{
  \g@addto@macro{\makememindexhook}{%}
  \def\edindex{\@bsphack%\@ifnextchar {\l@d@index}{\l@d@index\jobname}}%
  \newcommand{\edindex}[2]{\@bsphack\@esphack}
}
\end{verbatim}

\texttt{\l@d@index[\textit{file}]} is the first stage of \texttt{\edindex}, handling the \texttt{idx} file. This is a virtually a verbatim copy of \texttt{memoir}'s \texttt{\@index}, the change being calling \texttt{\l@d@wrindexm@m} instead of \texttt{\@wrindexm@m}.

\begin{verbatim}
\def\l@d@wrindexm@m{\l@d@@wrindexhyp\item}
\def\l@d@@wrindexhyp##1|##2|##3\{%
  \ifshowindexmark\@showidx{##1}\fi
  \ifx\#\texttt{\@idxfile}##1\%
  \def\@idxfile{##1}\doedindexlabel
  \begingroup
    \@sanitize
    \l@d@wrindexm@m
\end{verbatim}

\texttt{\l@d@wrindexm@m\item} writes the \texttt{idx} file name and the indexed item to the \texttt{aux} file. These are almost verbatim copies of \texttt{memoir}'s \texttt{\@wrindexm@m} and \texttt{\@wrindexhyp}.

\begin{verbatim}
\newcommand{\l@d@wrindexhyp}{$\l@d@wrindexhyp\texttt{\item}}{%}
\def\l@d@wrindexhyp##1|##2|##3\{%
  \ifshowindexmark\@showidx{##1}\fi
  \ifx\#\texttt{\@idxfile}##1\%
  \def\@idxfile{##1}\doedindexlabel
  \begingroup
    \@sanitize
    \l@d@wrindexm@m
\end{verbatim}
36.2 Normal setting

\create@edindex@notfor@memoir define the \edindex command and related tool when:

1. Memoir class is NOT used.
2. OR imakeidx is used.
3. OR indextools is used.

This finishes the memoir-specific code.
\wredindex Write the index information to the idx file.
\newcommand{\wredindex}[2]{[i=!\expandonce\jobname, usedefault]{%#1 = the index name, #2 = the text}
\global\let\old@Rlineflag\Rlineflag%
\def\Rlineflag{}
\ifl@imakeidx%
\if@edindex@fornote%
\IfSubStr[1]{##2}{|}{\get@index@command##2+}{\get@index@command##2|+}%
\expandafter\imki@wrindexentry{##1}{\@index@txt|(ledinnote{\@index@command}}{\thestartpageline}%
\expandafter\imki@wrindexentry{##1}{\@index@txt|)ledinnote{\@index@command}}{\theendpageline}%
\else%
\get@edindex@hyperref{##2}%
\imki@wrindexentry{##1}{\@index@txt@edindex@hyperref}{\thepageline}%
\fi%
\else%
\if@edindex@fornote%
\IfSubStr[1]{##2}{|}{\get@index@command##2+}{\get@index@command##2|+}%
\protected@write\@indexfile{}%\string\indexentry{\@index@txt|(ledinnote{\@index@command}}{\thestartpageline}%
\expandafter\protected@write\@indexfile{}%\string\indexentry{\@index@txt|)ledinnote{\@index@command}}{\theendpageline}%
\else%
\protected@write\@indexfile{}%
\{\string\indexentry{##2}{\thepageline}
\}
\fi%
\fi%
\endgroup
\global\let\Rlineflag\old@Rlineflag%
\@esphack}

Need to add the definition of \edindex to \makeindex, and initialise \edindex to do nothing.
\pretocmd{\makeindex}{%\def\edindex{\@bsphack\@esphack}}%\doedindexlabel
\begingroup\@sanitize\wredindex\protected@write\@indexfile{}%\string\indexentry{\@index@txt|(ledinnote{\@index@command}}{\thestartpageline}%
\expandafter\protected@write\@indexfile{}%\string\indexentry{\@index@txt|)ledinnote{\@index@command}}{\theendpageline}%
\endgroup
\global\let\Rlineflag\old@Rlineflag%
\@esphack}

36.3 Choose the right variant

Then call \create@edindex@for@memoir or \create@edindex@not@for@memoir depending on the use of memoir and imakeidx
\ifarclassloaded{memoir}{%
36.4 Hyperref compatibility

\hyperlinkformat command is to be used to have both an internal hyperlink and a format, when indexing.
\newcommand{\hyperlinkformat}[3]{
  \ifstrempty{#1}
    {\hyperlink{#2}{#3}}
  {\csuse{#1}{\hyperlink{#2}{#3}}}
}

\hyperlinkR command is to be used to create a internal hyperlink and \ledRflag, when indexing.
\newcommand{\hyperlinkR}[2]{
  \hyperlink{#1}{#2\Rlineflag}
}

\hyperlinkformatR command is to be used to create a internal hyperlink, a format and a \Rlineflag, when indexing.
\newcommand{\hyperlinkformatR}[3]{
  \hyperlinkformat{#1}{#2}{#3\Rlineflag}
}

\get@edindex@hyperref is to be used to define the \@edindex@hyperref macro, which, in index, links to the point where the index was called (with hyperref).
\newcommand{\get@edindex@hyperref}[1]{%
  \ifdef{\hyperlink}{%}
  \edef\temp@{\catcode\' =9 %space need for catcode}
  \edef\temp@{\catcode\' =10 %space need for catcode}
  \IfSubStr{\temp@}{|}{}
  {\get@index@command#1+%
    \ifledRcol pomys
37 Macro as environment

The following is borrowed, and renamed, from the amsmath package. See also the
CTT thread ‘eeq and amstex’, 1995/08/31, started by Keith Reckdahl and ended
definitively by David M. Jones.

Several of the [math] macros scan their body twice. This means we must collect
all text in the body of an environment form before calling the macro.

\@emptytoks This is actually defined in the amsgen package.

\newtoks\@emptytoks

The rest is from amsmath.

\l@denvbody A token register to contain the body.

\addtol@denvbody \addtol@denvbody{arg} adds arg to the token register \l@denvbody.

\l@dcollect@body The macro \l@dcollect@body starts the scan for the \end{...} command of
the current environment. It takes a macro name as argument. This macro is
supposed to take the whole body of the environment as its argument. For example, given \texttt{cenv#1{...}} as a macro that processes \#1, then the environment form, \texttt{\begin{env}} would call \texttt{\l@dcollect@body\cenv}.

\newcommand{\l@dcollect@body}[1]{%  
\l@denvbody{\expandafter#1\expandafter{\the\l@denvbody}}%  
\edef\processl@denvbody{\the\l@denvbody\noexpand\end{\@currenvir}}%  
\l@denvbody\@emptytoks \def\l@dbegin@stack{b}%  
\begingroup  
\expandafter\let\csname\@currenvir\endcsname\l@dcollect@@body  
\edef\processl@denvbody{\expandafter\noexpand\csname\@currenvir\endcsname}  
\processl@denvbody%  
}\l@dpush@begins

\l@dpush@begins When adding a piece of the current environment’s contents to \texttt{\l@denvbody}, we scan it to check for additional \texttt{\begin} tokens, and add a ‘b’ to the stack for any that we find.

\def\l@dpush@begins#1\begin#2{%  \ifx\end#2\else b\expandafter\l@dpush@begins\fi%
}\l@dcollect@@body \l@dcollect@@body takes two arguments: the first will consist of all text up to the next \texttt{\end} command, and the second will be the \texttt{\end} command’s argument. If there are any extra \texttt{\begin} commands in the body text, a marker is pushed onto a stack by the \texttt{\l@dpush@begins} function. Empty state for this stack means we have reached the \texttt{\end} that matches our original \begin. Otherwise we need to include the \texttt{\end} and its argument in the material we are adding to the environment body accumulator.

\def\l@dcollect@body#1\end#2{%  \expandafter\@gobble\l@dbegin@stack}%
\ifx\@empty\l@dbegin@stack  
\endgroup  
\@checkend{#2}  
\addtol@denvbody{#1}\processl@denvbody % A little tricky! Note the grouping%
\else
\addtol@denvbody{#1\end{#2}}%
\fi%
\l@dcollect@body\l@dcollect@body

\l@dcollect@body There was a question on CTT about how to use \texttt{\collect@body} for a macro taking an argument. The following is part of that thread.

From: Heiko Oberdiek \texttt{<oberdiek@uni-freiburg.de>}
Newsgroups: comp.text.tex
Subject: Re: Using \texttt{\collect@body} with commands that take \textgreater1 argument
Date: Fri, 08 Aug 2003 09:03:20 +0200
eed132@psu.edu (Evan) wrote:
> I’m trying to make a new LaTeX environment that acts like the \colorbox command that is part of the color package. I looked through the FAQ and ran across this bit about using the \collect@body command that is part of AMSLaTeX:
> http://www.tex.ac.uk/cgi-bin/texfaq2html?label=cmdasenv
>
> It almost works. If I do something like the following:
> \newcommand{\redbox}[]{}{\colorbox{red}{#1}}
> \makeatletter
> \newenvironment{redbox}{\collect@body \redbox}{}
>
> \makeatletter
> \newenvironment{coloredbox}{}{\collect@body \redbox}
>
> \begin{coloredbox}{blue}
> Yadda yadda yadda... this is on a blue background...
> \end{coloredbox}
>
> and can’t figure out how to make the \collect@body take this.

> \collect@body \colorbox{red}
> \collect@body {\colorbox{red}}

The argument of \collect@body has to be one token exactly.

\documentclass{article}
\usepackage{color}
\usepackage{amsmath}
\newcommand{\redbox}[]{}{\colorbox{red}{#1}}
\makeatletter
\newenvironment{coloredbox}[]{}{\collect@body \redbox}
\makeatletter
\newenvironment{coloredboxII}[]{}{\collect@body \mycoloredbox}
\newcommand\mycoloredbox[]{}{\colorbox{#1}{\ignorespaces#2\unskip}}

% ignore spaces at begin and end of environment
\newenvironment{coloredboxIII}[]{}{\collect@body\next@}
\newcommand\mycoloredboxII[]{}{\colorbox{#1}{\ignorespaces#2\unskip}}

% support of optional color model argument
\newcommand{\coloredboxIII}[]{}
\def\coloredboxIII#1{% 
@coloredboxIII[#1]% 
} \def\@coloredboxIII#1#2{% 
\def\next@{\mycoloredboxIII[#1]{#2}}% 
\collect@body\next@ 
} \newcommand{\mycoloredboxIII}[3]{% 
\colorbox[#1]{#2}{\ignorespaces#3\unskip}% 
} \makeatother

\begin{document}
Black text before
\begin{coloredbox}{blue}
Hello World
\end{coloredbox}
Black text after

Black text before
\begin{coloredboxII}{blue}
Hello World
\end{coloredboxII}
Black text after

Black text before
\begin{coloredboxIII}[rgb]{0,0,1}
Hello World
\end{coloredboxIII}
Black text after

\end{document}

Yours sincerely
Heiko <oberdiek@uni-freiburg.de>

38 Verse

This is principally Wayne Sullivan’s code and commentary from EDSTANZA [Su92].

The macro \texttt{\textbackslash hangingsymbol} is used to insert a symbol on each hanging of verses. For example, in French typographie the symbol is ‘[’. We obtain it by the next code:

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

The \texttt{\textbackslash ifininstanza} boolean is used to be sure that we are in a stanza part.
The boolean \ifinserthangingsymbol is set to TRUE when \@lock is greater than 1, i.e. when we are not in the first line of a verse. The switch of \ifinserthangingsymbol is made in \do@line before the printing of line but after the line number calculation.

\ampersand Within a stanza the \& macro is going to be usurped. We need an alias in case an & needs to be typeset in a stanza. Define it rather than letting it in case some other package has already defined it.

\stanza@count, \stanzaindentbase

A count register is allocated for counting lines in a stanza; also allocated is a dimension register which is used to specify the base value for line indentation; all stanza indentations are multiples of this value. The default value of \stanzaindentbase is 20pt.

\strip@szacnt
\setstanzavalues

The indentations of stanza lines are non-negative integer multiples of the unit called \stanzaindentbase. To make it easier for the user to specify these numbers, some list macros are defined. These take numerical values in a list separated by commas and assign the values to special control sequences using \mathchardef. Though this does limit the range from 0 to 32767, it should suffice for most applications, including penalties, which will be discussed below.
\newcommand*{\setstanzavalues}[2]{\def\@tempa{#2,\,|}\stanza@count\z@\def\next{\expandafter\strip@szacnt\@tempa\ifx\@tempb\empty\let\next\relax\else\expandafter\mathchardef\csname #1@\number\stanza@count\endcsname\@tempb\relax\advance\stanza@count\@ne\fi\next}\next}

\setstanzaindents \setstanzapenalties \managestanza@modulo

In the original \setstanzavalues{sza}{...} had to be called to set the indents, and similarly \setstanzavalues{szp}{...} to set the penalties. These two macros are a convenience to give the user one less thing to worry about (misspelling the first argument). Since version 0.13, the \stanzaindentsrepetition counter can be used when the indentation is repeated every n verses. The \managestanza@modulo is a command which modifies the counter stanza@modulo. The command adds 1 to stanza@modulo, but if stanza@modulo is equal to the stanzaindentsrepetition counter, the command restarts it.

\newcommand*{\setstanzaindents}{\setstanzavalues{sza}{#1}} \newcommand*{\setstanzapenalties}{\setstanzavalues{szp}{#1}}

\newcounter{stanzaindentsrepetition} \newcount\stanza@modulo

\newcommand*{\managestanza@modulo}[0]{\advance\stanza@modulo\@ne\ifnum\stanza@modulo>\value{stanzaindentsrepetition}\stanza@modulo\@ne\fi}

\stanzaindent \stanzaindent*

The macro \stanzaindent, when called at the beginning of a verse, changes the indentation normally defined for this verse by \setstanzaindent. The starred version skips the current verse for the repetition of stanza indent.

\newcommand{\stanzaindent}{[1]{\hspace{\dimexpr#1\stanzaindentbase-\parindent\relax}}\ignorespaces} \newcommand\stanzaindent*[1]{\stanzaindent{#1}\global\advance\stanza@modulo-\@ne\ifnum\stanza@modulo=0\global\stanza@modulo=\value{stanzaindentsrepetition}\fi\ignorespaces}

\stanza@line \stanza@hang \sza@penalty

Now we arrive at the main works. \stanza@line sets the indentation for the line and starts a numbered paragraph—each line is treated as a paragraph.
\stanza@hang sets the hanging indentation to be used if the stanza line requires more than one print line. If it is known that each stanza line will fit on one print line, it is advisable to set the hanging indentation to zero. \sza@penalty places the specified penalty following each stanza line. By default, this facility is turned off so that no penalty is included. However, the user may initiate these penalties to indicate good and bad places in the stanza for page breaking.

\newcommandx{\stanza@line}[1][1]{{\ifnum\value{stanzaindentsrepetition}=0\parindent=\csname sza@\number\stanza@count\endcsname\stanzaindentbase\else\parindent=\csname sza@\number\stanza@modulo\endcsname\stanzaindentbase\fi}pstart[#1]\stanza@hang\ignorespaces}{\xdef\stanza@hang{\leavevmode\startlock\hangindent\expandafter\csname sza@0@\endcsname\stanzaindentbase\hangafter\@ne}}\def\sza@penalty{\count@\csname szp@\number\stanza@count@\endcsname\ifnum\count@>\@M\advance\count@-\@M\penalty-\else\penalty\fi\count@}

Now we have the components of the \stanza macro, which appears at the start of a group of lines. This macro initializes the count and checks to see if hanging indentation and penalties are to be included. Hanging indentation suspends the line count, so that the enumeration is by verse line rather than by print line. If the print line count is desired, invoke \let\startlock=\relax and do the same for \endlock. Here and above we have used \xdef to make the stored macros take up a bit less space, but it also makes them more obscure to the reader. Lines of the stanza are delimited by ampersands &. The last line of the stanza must end with \&. For convenience the macro \endstanzaxtra is included. The user may use this to add vertical space or penalties between stanzas.

As a further convenience, the macro \startstanzahook is called at the beginning of a stanza. This can be defined to do something useful.
\flagstanza Use \flagstanza[len]{text} at the start of a line to put text a distance len before the start of the line. The default for len is \stanzaindentbase.

The ampersand & is used to mark the end of each stanza line, except the last, which is marked with \\&. This means that \halign may not be used directly within a stanza line. This does not affect macros involving alignments defined outside \stanza \&. Since these macros usurp the control sequence \&, the replacement \ampersand is defined to be used if this symbol is needed in a stanza. Also we reset the modified category codes and initialize the penalty default.

\catcode'&=\next
\catcode'@=\body
\% \let\ampersand=&
\setstanzavalues{szp}{0}

39 Arrays and tables

This is based on the work by Herbert Breger in developing tabmac.tex.
The original tabmac.tex file was void of comments or any explanatory text other than the above notice. The algorithm is Breger’s. I have made some cosmetic changes to the original code and reimplemented some things so they are more LaTeX-like. All the commentary is mine, as are any mistakes or errors.

\l@tabnoexpands  An extended and modified version of the original additional no expansions.

\disable@familiarnotes  Macros to disable and restore familiar notes, to prevent them from printing multiple times in edtabularx and edarrayx environments.
\newcommand{\disable@sidenotes}{\let\@@ledrightnote\ledrightnote\let\@@ledleftnote\ledleftnote\let\@@ledsidenote\ledsidenote\let\ledrightnote\@gobble\let\ledleftnote\@gobble\let\ledsidenote\@gobble}
\newcommand{\restore@sidenotes}{\let\ledrightnote\@@ledrightnote\let\ledleftnote\@@ledleftnote\let\ledsidenote\@@ledsidenote}
\newcommand{\disable@notes}{\disable@sidenotes\disable@familiarnotes}
\newcommand{\restore@notes}{\restore@sidenotes\restore@familiarnotes}
\newcount\l@dampcount
\l@dampcount=1\relax
\newcount\l@dcolcount
\l@dcolcount=0\relax
\l@dampcount \l@dampcount is a counter for the & column dividers and \l@dcolcount is a counter for the columns. These were \Undcount and \stellencount respectively.
Some (temporary) helper items.

30 columns should be adequate (compared to the original 60). These are
the column widths. (Originally these were German spelled numbers e.g., \eins,
\zwei, etc).

This is a cunning way of storing the columnwidths indexed by the column number
\@dcolcount, like an array. (was \Dimenzuordnung)

\newcommand{\@dcolwidth}{\ifcase \the\@dcolcount \dcoli \or \dcolii \or \dcoliii
\or \dcoliv \or \dcolv \or \dcolvi
\or \dcolvii \or \dcolviii \or \dcolix \or \dcolx
\or \dcolxi \or \dcolxii \or \dcolxiii
\or \dcolxiv \or \dcolxv \or \dcolxvi
\or \dcolxvii \or \dcolxviii \or \dcolxix
\or \dcolxx \or \dcolxxi \or \dcolxxii
\or \dcolxxiii \or \dcolxxiv \or \dcolxxv
\or \dcolxxvi \or \dcolxxvii \or \dcolxxviii
\or \dcolxxix \or \dcolxxx \or \dcoli
\else \dcoli \fi}
\stepl@dcolcount  This increments the column counter, and issues an error message if it is too large.

\l@dsetmaxcolwidth  Sets the column width to the maximum value seen so far. (was \dimenzuordnung)

\EDTEXT  We need to be able to modify the \edtext and \critext macros and also restore their original definitions.
\CRITEXT
\EDLABEL  We need to be able to modify and restore the \edlabel macro.
\nedindex  Macros supporting modification and restoration of \edindex.

\ifl@dmemoir
\newcommand{\nedindex}{\@bsphack
\doedindexlabel
\begingroup
\@sanitize
\@wredindex}
\fi
\@line@@num Macro supporting restoration of \linenum.
4214 \let\@line@@num=\linenum
\l@dgobbledarg \l@dgobbledarg replaces its delineated argument by \relax (was \verschwinden).
\l@dgobblearg \l@dgobbleoptarg{⟨arg⟩}{⟨arg⟩} replaces these two arguments (first is optional) by \relax.
4216 \def\l@dgobbleoptarg[#1]{\relax}
4217 \newcommand*{\l@dgobbleoptarg}[2]{\relax}\
\Relax\NEXT\let\Relax=\relax\@hilfs@count\let\NEXT=\next
4220 \newcount\@hilfs@count
\measuremcell Measure (recursively) the width required for a math cell. (was \messen)
4222 \def\measuremcell #1&{%  \ifx #1\%\ifnum\l@dcolcount=0\let\NEXT=\relax%  \else\l@dcheckcols%  \l@dcolcount=0%  \let\NEXT=\measuremcell%  \fi%  \else\setbox\hilfsbox=\hbox{$\displaystyle{#1}$}%  \stepl@dcolcount%  \l@dsetmaxcolwidth%  \let\NEXT=\measuremcell%  \fi\NEXT}
\measuretcell Measure (recursively) the width required for a text cell. (was \messentext)
4224 \def\measuretcell #1&{%  \ifx #1\%\ifnum\l@dcolcount=0\let\NEXT=\relax%  \else\l@dcheckcols%  \l@dcolcount=0%  \let\NEXT=\measuretcell%  \fi%  \else\setbox\hilfsbox=\hbox{#1}%  \stepl@dcolcount%  \l@dsetmaxcolwidth%  \let\NEXT=\measuretcell%  \fi\NEXT}
\measuremrow Measure (recursively) the width required for a math row. (was \Messen)
4226 \def\measuremrow #1\{%  \ifx #1\%\ifnum\l@dcolcount=0\let\NEXT=\relax%  \else\measuremcell #1\&\&\&\&\&\%  \fi\NEXT%  \fi\NEXT}
\measuretrow  Measure (recursively) the width required for a text row. (was \Messentext)
\edef\measuretrow #1\{\%
  \ifx #1&\let\NEXT\relax\%
  \else\measuretcell #1&\&\&\%
  \let\NEXT\measuretrow\%
  \fi\NEXT\}

\edtabcolsep  The length \edtabcolsep controls the distance between columns. (was \abstand)
\newskip\edtabcolsep
\global\edtabcolsep=10pt
\let\Next=\next
\newcommand{\variab}{\relax}
\l@dcheckcols  Check that the number of columns is consistent. (was \tabfehlemeldung)
\newcommand*{\l@dcheckcols}{\%
  \ifnum\l@dcolcount=1\relax\%
  \else\%
    \ifnum\l@dampcount=1\relax\%
    \else\%
      \ifnum\l@dcolcount=\l@dampcount\relax\%
      \else\%
        \l@d@err@UnequalColumns
      \fi\%
      \l@dampcount=\l@dcolcount\%
    \fi\%
  \fi\%
  \fi\%
}
\l@dmodforcritext  Modify and restore various macros for when \critext is used.
\l@drestoreforcritext  
\newcommand{\l@dmodforcritext}{\%
  \let\critext\relax\%
  \def\do##1{\global\csletcs{##1footnote}{l@dgobbledarg}}\%
  \dolistloop{\@series}\%
  \let\edindex\nullediindex\%
  \let\linenum\gobble\%
  \newcommand{\l@drestoreforcritext}{\%
    \def\do##1{\csdef{##1footnote}##1##2/{\csuse{##1@@footnote}{##1}{##2}}}\%
    \dolistloop{\@series}\%
    \let\edindex\xediindex\%
  }\%
}
Modify and restore various macros for when \texttt{\textbackslash edtext} is used.

\newcommand{\modforedtext}{% \let\edtext\relax \def\do##1{\global\csletcs{##1footnote}{\gobbleoptarg}}\dolistloop{\@series}\let\edindex\nullifiedindex \let\linenum\gobble} \newcommand{\restoreforedtext}{% \def\do##1{\global\csletcs{##1footnote}{\@footnote}}\dolistloop{\@series}\let\edindex\index} \nullify and restore some column fillers, etc.

\newcommand{\nullfills}{% \def\edlabel##1{} \def\edrowfill##1##2##3{} } \newcommand{\restorefills}{% \def\edrowfill##1##2##3{\EDROWFILL@{##1}{##2}{##3}} } The original definition of \texttt{\textbackslash rverteilen} and friends (‘verteilen’ is approximately ‘distribute’) was along the lines:

\texttt{\def\rverteilen #1&{\def\label##1{}% \ifx #1! \ifnum\l@dcolcount=0\removelastskip \let\Next\relax% \else\l@dcolcount=0\let\Next=rverteilen% \fi% \else% \footnoteferschwert% \step\l@dcolcount% \setbox\hilfsbox=\hbox{$\displaystyle{#1}$}\% \let\critext=\xcritext \let\Dfootnote=\D@@footnote \let\Afootnote=\A@@footnote \let\Bfootnote=\B@@footnote \let\Cfootnote=\C@@footnote \let\linenum=\@line@@num \hilfsskip=\Dimenzuordnung% \advance\hilfsskip by -\wd\hilfsbox \def\l@dlabel##1{\xlabel{##1}}% \hskip\hilfsskip\displaystyle{#1}\% \hskip\edtabcolsep% \let\Next=rverteilen% \fi\Next}%}

where the lines

\texttt{\let\critext=\xcritext \let\Dfootnote=\D@@footnote}
were common across the several *verteilen* macros, and also

\def\footnoteverschw{\let\critext\relax
\let\Afootnote\verschwinden
\let\Bfootnote\verschwinden
\let\Cfootnote\verschwinden
\let\Dfootnote\verschwinden
\let\linenum\@gobble}

\let\forverteilen
Gathers some lets and other code that is common to the *verteilen* macros.

\newcommand{\forverteilen}{
\let\critext\xicritext
\let\edtext\xedtext
\let\edindex\xedindex
\def\do##1{\global\csletcs{##1footnote}{##1@@footnote}}
\dolistloop{\@series}
\let\linenum\@line@@num
\hilfsskip=\ld@colwidth
\advance\hilfsskip by -\wd\hilfsbox
\def\edlabel##1{\xedlabel{##1}}}

\setmcellright
Typeset (recursively) cells of display math right justified. (was \rverteilen)

\def\setmcellright #1&{
\ifx #1\fi
{\ifnum\l@dcolcount=0%
\removelastskip
\let\Next\relax%
\else\l@dcolcount=0%
\let\Next=\setmcellright%
\fi%
\else%
\disablel@dtabfeet%
\step@dcolcount%
\disable@notes%
\setbox\hilfsbox=\hbox{$\displaystyle{#1}$}
\restore@notes%
\letsforverteilen%
\hskip\hilfsskip$\displaystyle{#1}$
\hskip\edtabcolsep%
\let\Next=\setmcellright%
\fi\Next}
\settcellright Typeset (recursively) cells of text right justified. (was \rverteilentext)
\def\settcellright #1&{\def\edlabel##1{}% \
\let\edindex\nulledindex
\ifx #1\ifnum\l@dcolcount=0\removelastskip
\let\Next\relax%
\else\l@dcolcount=0%
\let\Next=\settcellright%
\fi%
\else%
\disable@dtabfeet%
\stepl@dcolcount%
\disable@notes%
\setbox\hilfsbox=\hbox{#1}%
\restore@notes%
\letsforverteilen%
\hskip\hilfsskip\hskip\edtabcolsep%
\let\Next=\settcellright%
\fi\Next}
\setmcellleft Typeset (recursively) cells of display math left justified. (was \lverteilen)
\def\setmcellleft #1&{\def\edlabel##1{}% \
\let\edindex\nulledindex
\ifx #1\ifnum\l@dcolcount=0 \let\Next\relax%
\else\l@dcolcount=0%
\let\Next=\setmcellleft%
\fi%
\else \disable@dtabfeet%
\stepl@dcolcount%
\disable@notes%
\setbox\hilfsbox=\hbox{$\displaystyle{#1}$}%
\restore@notes%
\letsforverteilen%
$\displaystyle{#1}$\hskip\hilfsskip\hskip\edtabcolsep%
\let\Next=\setmcellleft%
\fi\Next}
\settcelleft Typeset (recursively) cells of text left justified. (was \lverteilentext)
\def\settcelleft #1&{\def\edlabel##1{}% \
\let\edindex\nulledindex
\ifx #1\ifnum\l@dcolcount=0 \let\Next\relax%
\else\l@dcolcount=0%
\let\Next=\settcelleft%
\fi%
\else \disable@dtabfeet%
\stepl@dcolcount%
\disable@notes%
\setbox\hilfsbox=\hbox{#1}%
\fi\Next}
\setmcellcenter  Typeset (recursively) cells of display math centered. (was \zverteilen)
\def\setmcellcenter #1&{\def\edlabel##1{}%  
\let\edindex\nulledindex  
\ifx #1\\ \ifnum\l@dcolcount=0\let\Next\relax%  
\else\l@dcolcount=0%  
\let\Next=\settcellcenter%  
\fi%  
\else %  
\disablel@dtabfeet%  
\stepl@dcolcount%  
\disable@notes%  
\setbox\hilfsbox=\hbox{$\displaystyle{#1}$}%  
\restore@notes%  
\letsforverteilen%  
\hskip 0.5\hilfsskip\displaystyle{#1}$\hskip0.5\hilfsskip%  
\hskip\edtabcolsep%  
\let\Next=\settcellcenter%  
\fi\Next}
\settcellcenter  Typeset (recursively) cells of text centered. (new)
\def\settcellcenter #1&{\def\edlabel##1{}%  
\let\edindex\nulledindex  
\ifx #1\\ \ifnum\l@dcolcount=0\let\Next\relax%  
\else\l@dcolcount=0%  
\let\Next=\settcellcenter%  
\fi%  
\else %  
\disablel@dtabfeet%  
\stepl@dcolcount%  
\disable@notes%  
\setbox\hilfsbox=\hbox{#1}%  
\restore@notes%  
\letsforverteilen%  
\hskip 0.5\hilfsskip\displaystyle{#1}$\hskip0.5\hilfsskip%  
\hskip\edtabcolsep%  
\let\Next=\settcellcenter%  
\fi\Next}
\NEXT
\let\NEXT=\relax
\setmrowright  Typeset (recursively) rows of right justified math. (was \rsetzen)
\settrowright Typeset (recursively) rows of right justified text. (was \rsetzentext)
\settrowleft Typeset (recursively) rows of left justified math. (was \lsetzen)
\settrowcenter Typeset (recursively) rows of centered math. (was \zsetzen)
\settrowcenter Typeset (recursively) rows of centered text. (new)
\nullsetzen (was \nullsetzen)
\edatleft \edatleft[(\math)\{\langle symbol\rangle\}\{\langle len\rangle\} (combination and generalisation of original \seklam and \seklaml). Left \langle symbol\rangle, 2\langle len\rangle high with prepended \langle math\rangle vertically centered.
\newcommand{\edatleft}[3][\@empty]{% 
  \ifx#1\@empty 
    \vbox to 10pt{\vss\hbox{$\left#2\vrule width0pt height #3$\hss}\vfil}
  \else 
    \vbox to 4pt{\vss\hbox{$#1\left#2\vrule width0pt height #3$\hss}\vfil}
  \fi}
\edatright \edatright[(\math)\{\langle symbol\rangle\}\{\langle len\rangle\} (combination and generalisation of original \seklam and \seklaml). Right \langle symbol\rangle, 2\langle len\rangle high with appended \langle math\rangle vertically centered.
\newcommand{\edatright}[3][\@empty]{% 
  \ifx#1\@empty 
    \vbox to 10pt{\vss\hbox{$\left.\vrule width0pt height #3$\hss}\vfil}
  \else 
    \vbox to 4pt{\vss\hbox{$\left.\vrule width0pt height #3 #1$\hss}\vfil}
  \fi}
\edvertline \edvertline{\langle len\rangle} vertical line \langle len\rangle high. (was \sestrich)
\newcommand{\edvertline}[1]{\vbox to 8pt{\vss\hbox{\vrule height #1}\vfil}}
\edvertdots \edvertdots{\langle len\rangle} vertical dotted line \langle len\rangle high. (was \sepunkte)
\newcommand{\edvertdots}[1]{\vbox to 1pt{\vss\vbox to #1\hss\vbox to 0.5em{ }\vfil}}

I don’t know if this is relevant here, and I haven’t tried it, but the following appeared on CTT.

From: mdw@msict.org (Mark Wooding)
Newsgroups: comp.text.tex
Subject: Re: Dotted line
Date: 13 Aug 2003 13:51:14 GMT

Alexis Eisenhofer <alexis@eisenhofer.de> wrote:
> Can anyone provide me with the LaTeX command for a vertical dotted line?
How dotted? Here’s the basic rune.  
\newbox\linedotbox
\setbox\linedotbox=\vbox{...}
\leaders\copy\linedotbox\vskip2in

For just dots, this works:  
\setbox\linedotbox=\vbox{\hbox{\normalfont.}\kern2pt}

For dashes, something like  
\setbox\linedotbox=\vbox{\leaders\vrule\vskip2pt\vskip2pt}
is what you want. (Adjust the ‘2pt’ values to taste. The first one is  
the length of the dashes, the second is the length of the gaps.)

For dots in mid-paragraph, you need to say something like  
\lower10pt\vbox{\leaders\copy\linedotbox\vskip2in}
which is scungy but works.

-- [mdw]

\edfilldimen  A length. (was \klamdimen)
\newdimen\edfilldimen
\edfilldimen=0pt
\c@addcolcount  A counter to hold the number of a column. We use a roman number so that we
\theaddcolcount  can grab the column dimension from \dcol....
\newcounter{addcolcount}
\renewcommand{\theaddcolcount}{\roman{addcolcount}}
\l@dtabaddcols  \l@dtabaddcols{(startcol)}{(endcol)} adds the widths of the columns \((startcol) through \(endcol)\) to \edfilldimen. It is a \LaTeX \ style reimplementation of the
\original \@add@.
\newcommand{\l@dtabaddcols}[2]{%
  \l@dcheckstartend{#1}{#2}{%
    \if\l@dstartendok
      \setcounter{addcolcount}{#1}{%
        \@whilenum \value{addcolcount}<#2\relax \do
          \advance\edfilldimen by \the \csname dcol\theaddcolcount\endcsname
        \advance\edfilldimen by \edtabcolsep
        \stepcounter{addcolcount}}%
      \advance\edfilldimen by \the \csname dcol\theaddcolcount\endcsname
    \fi
    \} }%
\if\l@dstartendok
  \l@dcheckstartend{%
\newif\ifdstartendok\newcommand{\dcheckstartend}[2]{\ifnum #1<\@ne \dstartendokfalse\else\ifnum #2>30\relax \dstartendokfalse\fi\ifnum #1>#2\relax \dstartendokfalse\fi\fi}\edrowfill{⟨\text{startcol}⟩}{⟨\text{endcol}⟩}fill fills columns ⟨\text{startcol}⟩ to ⟨\text{endcol}⟩ inclusive with ⟨\text{fill}⟩ e.g. \hrulefill, \upbracefill. This is a LaTeX style reimplementation and generalization of the original \waklam, \Waklam, \waklamec, \wastricht and \wapunktel macros.

\newcommand*{\edrowfill}[3]{\l@dtabaddcols{#1}{#2} \hb@xt\the\l@dcolwidth{\hb@xt\the\edfilldimen{#3}\hss}}\let\@edrowfill@=\edrowfill\def\@EDROWFILL@#1#2#3{\@edrowfill@{#1}{#2}{#3}}

The macro \edbeforetab{⟨text⟩}{⟨math⟩} puts ⟨text⟩ at the left margin before array cell entry ⟨math⟩. Conversely, the macro \edaftertab{⟨math⟩}{⟨text⟩} puts ⟨text⟩ at the right margin after array cell entry ⟨math⟩. \edbeforetab should be in the first column and \edaftertab in the last column. The following macros support these.

\leftltab \leftltab{⟨text⟩} for \edbeforetab in \ltab. (was \linksltab)
\leftrtab \leftrtab{⟨text⟩}{⟨math⟩} for \edbeforetab in \rtab. (was \linksrtab)
\leftctab \leftctab{⟨text⟩}{⟨math⟩} for \edbeforetab in \ctab. (was \linksztab)
\rightctab{\langle math\rangle}{\langle text\rangle} for \texttt{edaftertab} in \texttt{ctab}. (was \texttt{rechtsztab})

\rightltab{\langle math\rangle}{\langle text\rangle} for \texttt{edaftertab} in \texttt{ltab}. (was \texttt{rechtsltab})

\rightrtab{\langle math\rangle}{\langle text\rangle} for \texttt{edaftertab} in \texttt{rtab}. (was \texttt{rechtsrtab})
\rtab \rtab{⟨body⟩} typesets ⟨body⟩ as an array with the entries right justified. (was \rtabtext)
\edbeforetab \rtab (Here and elsewhere, \edbeforetab and \edaftertab were originally \davor and \danach) The original \rtab and friends included a fair bit of common code which I have extracted into macros.

The process is first to measure the ⟨body⟩ to get the column widths, and then in a second pass to typeset the body.

\newcommand{\rtab}{%  \\
\l@dnullfills  \\
\def\edbeforetab##1##2{\leftrtab{##1}{##2}}%  \\
\def\edaftertab##1##2{\rightrtab{##1}{##2}}%  \\
\measurembody{#1}%  \\
\l@drestorefills  \\
\variab  \\
\settrowright #1\&\%  \\
\enablel@dtabfeet}

\measurembody \measurembody{⟨body⟩} measures the array ⟨body⟩.

\newcommand{\measurembody}{%  \\
\disablel@dtabfeet%  \\
\l@dcolcount=0%  \\
\nullsetzen%  \\
\l@dcolcount=0  \\
\measuretrrow #1\&\%  \\
\global\l@dampcount=1}

\rtabtext \rtabtext{⟨body⟩} typesets ⟨body⟩ as a tabular with the entries right justified. (was \rtabtext)

\newcommand{\rtabtext}{%  \\
\l@dnullfills  \\
\measuretbody{#1}%  \\
\l@drestorefills  \\
\variab  \\
\settrowright #1\&\%  \\
\enablel@dtabfeet}

\measuretbody \measuretbody{⟨body⟩} measures the tabular ⟨body⟩.

\newcommand{\measuretbody}{%  \\
\disable@notes%  \\
\disablel@dtabfeet%  \\
\l@dcolcount=0%  \\
\nullsetzen%  \\
\l@dcolcount=0  \\
\measuretrow #1\&\%  \\
\restore@notes%  \\
\global\l@dampcount=1}
\ltab  Array with entries left justified. (was \ltab)
\edef\edbeforetab{\newcommand{\ltab}[1]{{%}
\edef\edaftertab{%} \l@dnullfills
\def\edbeforetab##1##2{\leftltab{##1}{##2}}% 
\def\edaftertab##1##2{\rightltab{##1}{##2}}% 
\measurembody{#1}%
\l@drestorefills
\variab
\settrowleft #1\&\%
\enablel@dtabfeet}
\ltabtext  Tabular with entries left justified. (was \ltabtext)
\edef\edbeforetab{\newcommand{\ltabtext}[1]{{%}
\edef\edaftertab{%} \l@dnullfills
\measuretbody{#1}%
\l@drestorefills
\variab
\settrowleft #1\&\%
\enablel@dtabfeet}
\ctab  Array with centered entries. (was \ztab)
\edef\edbeforetab{\newcommand{\ctab}[1]{{%}
\edef\edaftertab{%} \l@dnullfills
\def\edbeforetab##1##2{\leftctab{##1}{##2}}% 
\def\edaftertab##1##2{\rightctab{##1}{##2}}% 
\measurembody{#1}%
\l@drestorefills
\variab
\settrowcenter #1\&\%
\enablel@dtabfeet}
\ctabtext  Tabular with entries centered. (new)
\edef\edbeforetab{\newcommand{\ctabtext}[1]{{%}
\edef\edaftertab{%} \l@dnullfills
\measuretbody{#1}%
\l@drestorefills
\variab
\settrowcenter #1\&\%
\enablel@dtabfeet}
\spreadtext  (was \breitertext)
\edef\edbeforetab{\newcommand{\spreadtext}[1]{{%}
\l@dcolcount=\l@dampcount%
\hb@xt@ \the\l@dcolwidth\{\hbox{#1}\hs}}
\spreadmath  (was \breiter, ‘breiter’ = ‘broadly’)
I have left the remaining `TABMAC` alone, apart from changing some names. I'm not yet sure what they do or how they do it. Authors should not use any of these as they are likely to be mutable.

\texttt{\textbackslash tabellzwischen} (was `\texttt{\textbackslash tabellzwischen}`)

\begin{verbatim}
\def\tabellzwischen #1&{\ifx #1\relax \let\NEXT\relax \l@dcolcount=0 \else \stepl@dcolcount\l@dcolwidth = #1 mm \let\NEXT=\tabellzwischen \fi \NEXT }
\end{verbatim}

\texttt{\textbackslash edatabell} For example `\texttt{\edatabell 4 & 19 & 8 \textbackslash}` specifies 3 columns with widths of 4, 19, and 8mm. (was `\texttt{\atabell}`)

\begin{verbatim}
\def\edatabell #1\textbackslash{\tabellzwischen #1&\textbackslash&}
\end{verbatim}

\texttt{\textbackslash Setzen} (was `\texttt{\textbackslash Setzen}`, ‘setzen’ = ‘set’)

\begin{verbatim}
\def\Setzen #1&{\ifx #1\relax \let\NEXT\relax \else \stepl@dcolcount\let\tabelskip=\l@dcolwidth \EDTAB #1| \let\NEXT=\Setzen \fi \NEXT}
\end{verbatim}

\texttt{\textbackslash EDATAB} (was `\texttt{\ATAB}`)

\begin{verbatim}
\def\EDATAB #1\textbackslash{\ifx #1\Relax \centerline{\Setzen #1\relax&} \let\Next\relax \else \centerline{\Setzen #1\relax&} \let\Next=\EDATAB \fi \Next}
\end{verbatim}

\texttt{\textbackslash edatab} (was `\texttt{\atab}`)

\begin{verbatim}
\newcommand{\edatab}[1]{{\varia% \EDATAB #1\\Relax\textbackslash}}
\end{verbatim}

\texttt{\textbackslash HILFSkip} More helpers.

\texttt{\textbackslash Hilfsskip}
\EDTABINDENT  (was \TABINDENT)
\newcommand{\EDTABINDENT}{%
  \ifnum\l@dcolcount=30\let\NEXT\relax\l@dcolcount=0%
  \else\stepl@dcolcount%
  \advance\Hilfsskip by\l@dcolwidth%
  \ifdim\l@dcolwidth=0pt\advance\hilfsscount\One
  \else\advance\Hilfsskip by \the\hilfsscount\edtabcolsep%
  \hilfsscount=1\fi%
  \let\NEXT=\EDTABINDENT%
  \fi\NEXT}%
\edef\EDTABINDENT{\EDTABINDENT}  (was \TABINDENT)
\edef\EDTABINDENT{
  \l@dcolcount=0\relax
  \Hilfsskip=0pt%
  \hilfscount=1\relax
  \EDTABINDENT%
  \hilfsskip=\hsize%
  \advance\hilfsskip -\Hilfsskip%
  \Hilfsskip=0.5\hilfsskip%}
\edef\EDTAB  (was \TAB){%
\def\EDTAB #1|#2|{
\setbox	abhilfbox=\hbox{$\displaystyle{#1}$}%
\setbox	abHilfbox=\hbox{$\displaystyle{#2}$}%
\advance\tabelskip -\wd\tabhilfbox%
\advance\tabelskip -\wd\tabHilfbox%
\unhbox\tabhilfbox\hskip\tabelskip%
\unhbox\tabHilfbox}%
\edef\EDTABtext (was \TABtext){%
\def\EDTABtext #1|#2|{
\setbox\tabhilfbox=\hbox{#1}%
\setbox\tabHilfbox=\hbox{#2}%
\advance\tabelskip -\wd\tabhilfbox%
\advance\tabelskip -\wd\tabHilfbox%
\unhbox\tabhilfbox\hskip\tabelskip%
\unhbox\tabHilfbox}%
\edef\tabhilfbox Further helpers.
\edef\tabHilfbox

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% That finishes tabmac
edarrayl  The ‘environment’ forms for \ltab, \ctab and \rtab.
edarrayc  \newenvironment{edarrayl}{\l@dcollect@body\ltab}{}
edarrayr  \newenvironment{edarrayc}{\l@dcollect@body\ctab}{}
edarrayr  \newenvironment{edarrayr}{\l@dcollect@body\rtab}{}
edtabular1  The ‘environment’ forms for \ltabtext, \ctabtext and \rtabtext.
edtabularc  \newenvironment{edtabularl}{\l@dcollect@body\ltabtext}{}
edtabularr  \newenvironment{edtabularc}{\l@dcollect@body\ctabtext}{}
edtabularr  \newenvironment{edtabularr}{\l@dcollect@body\rtabtext}{}

Here's the code for enabling \edtext (instead of \critext).
\usingcritext  Declarations for using \critext{}.../ or using \edtext{}{} inside tabulars.
\disablel@dtabfeet  The default at this point is for \edtext.
\enablel@dtabfeet
\usingedtext
\newcommand{\usingcritext}{{}
  \def\disablel@dtabfeet{\l@dmodforcritext}%
  \def\enablel@dtabfeet{\l@drestoreforcritext}%
}\newcommand{\usingedtext}{{}
  \def\disablel@dtabfeet{\l@dmodforedtext}%
  \def\enablel@dtabfeet{\l@drestoreforedtext}%

40  Section’s title commands

40.1  Deprecated commands

\initnumbering@sectcmd  \initnumbering@sectcmd defines \ledxxx commands. These commands are deprecated. It also defines quotation environment. Note: this assumes that the user didn’t change \chapter. If he did, he should redefine \initnumbering@sectcmd.
\ledsection  \newcommand{\initnumbering@sectcmd}{%
  \newcommand{\ledsection}[2][]{}%
  \led@war@ledxxxDeprecated{section}%
  \leavevmode\pend\vspace{3.5ex \plus\baselineskip \minus\baselineskip}#%%2\leavevmode\vspace{2.3ex \plus\baselineskip \minus\baselineskip}#%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\baselineskip}%%2\leavevmode\vspace{-.2\parskip}%%2\leavevmode\vspace{-.2\par
\@patchforledchapter\chapter*{##1}\pend%
\pstart}
def\@patchforledchapter{
\patchcmd{\makeschapterhead}{1\par}{1}{}{}
\pretocmd{\makeschapterhead}{\par}{}{}
\apptocmd{\makeschapterhead}{\par}{}{}
\patchcmd{\makeschapterhead}{\vskip 40\p@}{}{}{}
\patchcmd{\makechapterhead}{1\par}{1}{}{}
\pretocmd{\makechapterhead}{\par}{}{}
\apptocmd{\makechapterhead}{\par}{}{}
\patchcmd{\makechapterhead}{\vskip 40\p@}{}{}{}
\apptocmd{\chapter}{\par\leavevmode\vspace{40 \p@}\skipnumbering}{\pstart}
\apptocmd{\schapter}{\par\leavevmode\vspace{40 \p@}\skipnumbering}{\pstart}
\newcommand{\beforeledchapter}{\pstart\cleardoublepage\pstart}
\patchcmd{\chapter}{\cleardoublepage}{\relax}{}{}
\patchcmd{\chapter}{\clearpage}{\relax}{}{}
}
\ifnoquotation@else
\renewcommand{\quotation}{\par\leavevmode%
\parindent=1.5em%
\skipnumbering%
\ifautopar%
\vskip-\parskip%
\else%
\vskip\topsep%
\fi%
\global\leftskip=\leftmargin%
\global\rightskip=\leftmargin%
}
\renewcommand{\endquotation}{\par%
\global\leftskip=0pt%
\global\rightskip=0pt%
\leavevmode%
\ifautopar%
\vskip-\parskip%
\else%
\vskip\topsep%
\fi%
}
\renewcommand{\quote}{\par\leavevmode%
\parindent=0pt%
\skipnumbering%
\ifautopar%
\vskip-\parskip%
\else%
\vskip\topsep%
\fi%
\global\leftskip=\leftmargin%
\global\rightskip=\leftmargin%}
40.2 New commands: \eledxxx

The new system of \eledxxxx commands to section text work like this:

1. When one of these commands is called, \eledmac writes to an auxiliary files:
   - The section level.
   - The section title.
   - The side (when \eledpar is used).
   - The pstart where the command is called.
   - If we have starred version or not.

2. \eledmac adds the title of the section to pstart, as normal content. This is to enable critical notes.

3. When \LaTeX is run a other time, this file is read. That:
   - Adds the pstart number to a list of pstarts where a sectioning command is used.
   - Defines a command, the name of which contains the pstart number, and which calls the normal \LaTeX sectioning command.
4. **This last command is called when the pstart is effectively printed.**

\beforeeledchapter  For technical reasons, not yet solved, page-breaking before chapters can’t be made automatically by eledmac. Users have to use \beforeeledchapter.

\if@eled@sectioning  The boolean \if@eled@sectioning is set to true when a sectioning command is called by a \eledxxx command, and set to false after. It is used to enable/disable line number printing.

\print@leftmargin@eledsection and \print@rightmargin@eledsection are added by eledmac inside the code of sectioning command, in order to affix line numbers. They include tests for RTL languages.
We have to patch \LaTeX{}, book and memoir sectioning commands in order to:

- Disable \texttt{edtext} inside.
- Disable page breaking (for \texttt{chapter}).
- Add line numbers and sidenotes.

Unfortunately, Maëul Rouquette was not able to try if memoir is loaded. That is why \texttt{eledmac} tries to define for both standard class and memoir class.

\begin{verbatim}
\catcode`\#=12 % Space NEEDS by \catcode
\AtBeginDocument{\patchcmd\chapter{\clearforchapter}{\if@eled@sectioning\else\clearforchapter\fi}{}{}}
\pretocmd\M@sect{\let\old@edtext=\edtext\let\edtext=\dummy@edtext@showlemma}{}
\apptocmd\M@sect{\let\edtext=\old@edtext}{}
\patchcmd\M@sect{#9}{#9\print@rightmargin@eledsection}{}
\patchcmd\M@sect{\hskip #3\relax}{\hskip #3\relax\print@leftmargin@eledsection}{}
\end{verbatim}
40.2 New commands: \eledxxx

\patchcmd{\@mem@old@ssect}
{#5}
{#5}
{\print@leftmargin@eledsection%}
\
{#5}
{#5}
{#5}
{\print@rightmargin@eledsection%}
\
{#5}
{#5}
{\patchcmd{\chapter}{\if@openright\cleardoublepage\else\clearpage\fi}%
{#5}
\
{#5}
{#5}
{\patchcmd{\@makechapterhead}
{#1}
{\print@leftmargin@eledsection%
#1%
{\print@rightmargin@eledsection%}
\
{#1}
{#1}
{\patchcmd{\@makechapterhead}}% For BIDI
{\if@RTL\raggedleft\else\raggedright\fi}%
{\if@eled@sectioning\else%
{\if@RTL\raggedleft\else\raggedright\fi%}
{\if@eled@sectioning\else%
{\if@RTL\raggedleft\else\raggedright\fi%}
{\if@openright\cleardoublepage\else\clearpage\fi%No clearpage inside a \eledsection: will keep critical notes from printing on the title page.
{\if@openright\cleardoublepage\else\clearpage\fi%}
{#5}
{#5}
{#5}
{#5}
{#5}
\pretocmd{\@sect}{\let\edtext=\dummy@edtext\showlemma}{%}
\apptocmd{\@sect}{\let\edtext=\old@edtext}{%}
\pretocmd{\@ssect}{\let\edtext=\dummy@edtext\showlemma}{%}
\apptocmd{\@ssect}{\let\edtext=\old@edtext}{%}

\texttt{hyperref} also redefines \@sect. That’s why, when manipulating arguments, we
patch \@sect and the same only if \texttt{hyperref} is not used. If it is, we patch the \NR
commands.
\@ifpackageloaded{nameref}{
\patchcmd{\NR@section}{#8}{#8\print@rightmargin@eledsection}{%}{%}
\patchcmd{\NR@ssection}{\hskip #3\relax}{\hskip #3\relax\print@leftmargin@eledsection}{%}{%}
\patchcmd{\NR@ssect}{#5}{#5\hskip #3\relax}{%}{%}
\patchcmd{\NR@sect}{#6}{#6\hskip #3\relax}{%}{%}
\patchcmd{\NR@ssect}{#5}{#5\hskip #3\relax}{%}{%}
\patchcmd{\NR@section}{#6}{#6\hskip #3\relax}{%}{%}
\patchcmd{\NR@ssection}{#5}{#5\hskip #3\relax}{%}{%}
40.2 New commands: \eledxxx

\print@rightmargin@eledsection%
\}
\}
\}
\patchcmd{\NR@ssect}
{\hskip #1}
{\hskip #1%
\print@leftmargin@eledsection%
}
\}
\}
\}
\patchcmd{\sect}
{#8}
{#8%
\print@rightmargin@eledsection%
}
\}
\}
\}
\patchcmd{\sect}
{\hskip #3\relax}
{\hskip #3\relax%
\print@leftmargin@eledsection%
}
\}
\}
\}
\patchcmd{\ssect}
{\hskip #1}
{\hskip #1%
\print@leftmargin@eledsection%
}
\}
\}
\}
\patchcmd{\ssect}
{\hskip #5}
{\hskip #5%
\print@rightmargin@eledsection%
}
\}
\}
\}
\patchcmd{\ssect}
{\hskip #1}
{\hskip #1%
\print@leftmargin@eledsection%
}
\}
\}
\}
\patchcmd{\ssect}
{\hskip #1}
{\hskip #1%
\print@leftmargin@eledsection%
}
\}
\}
\}
\patchcmd{\ssect}
{\hskip #5}
{\hskip #5%
\print@rightmargin@eledsection%
}
\}
\}
\}
\patchcmd{\ssect}
{\hskip #1}
{\hskip #1%
\print@leftmargin@eledsection%
}
\}
\}
\}
\patchcmd{\ssect}
{\hskip #5}
{\hspace NEEDS by \catcode \eled@sectioning@out
\eled@sectioning@out}
\eled@sectioning@out is the output file, to dump the pstarts where a sectioning
command is used.
\newwrite\eled@sectioning@out

\if\noeled@sec
\if\noeled@sec
The boolean \if\noeled@sec is set to true when \noeled@sec is called. It is used
to disable external file creation.
\newif\if\noeled@sec
\newcommand{\noeledsec}{\global\noeled@sectrue}

\eledchapter
\eledsection
\eledsubsection
\eledsubsubsection
\eledchapter*
\eledsection*
\eledsubsection*
\eledsubsubsection*

And now, the user sectioning commands, which write to the file, and also add
content as a "normal" line.
\newcommand{\eledchapter}[2][]{% #2
\eledchapter*%\if\eledRcol%\immediate\write\eled@sectioningR@out{%\string\eled@chapter{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR}{\{}{R}%\}
\else%\immediate\write\eled@sectioning@out{%\string\eled@section{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL}{}{\}
}\fi%
\newcommand{\eledsection}[2][]{% #2
\eledsection*%\if\eledRcol%\immediate\write\eled@sectioningR@out{%\string\eled@section{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR}{\{}{R}%\}
\else%\immediate\write\eled@sectioning@out{%\string\eled@section{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL}{}{\}
}\fi%
\newcommand{\eledsubsection}[2][]{%
40.2 New commands: \eledxxx

5133 \eledsection*{#2}
5134 \eledchapter*{#2}
5135 \eledsubsection*{#2}
The sectioning macros, called in the auxiliary file. They have five arguments:

1. Optional arguments of \LaTeX sectioning command.
2. Mandatory arguments of \LaTeX sectioning command.
3. Pstart number.
4. Side: R if right, nothing if left.
5. Starred or not.
40.2 New commands: \eledxxx

\global\csdef{eled@sectioning@#3#5}{\section*{#2}}% Need for \pairs
\global\csdef{eled@sectioning@#3#5}{\section{#2}}% Need for \pairs
\global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext{}\sectionmark{#2}}% Need for \pairs, because of using parbox.
\global\csdef{eled@sectioning@#3#5}{\section{[#1]{#2}}}% Bug in LaTeX!
\global\csdef{eled@sectioning@#3#5}{\section*[#1]{#2}}% Bug in LaTeX!
\listcsgadd{eled@sections#5@@}{#3}%

\def\eled@subsection#1#2#3#4#5{%
  \ifstrempty{#4}{}
  \ifstrempty{#1}{}% \global\csdef{eled@sectioning@#3#5}{\subsection{#2}}%
  \global\csdef{eled@sectioning@#3#5}{\subsection{[#1]{#2}}}% Need for \pairs, because of using parbox. \csuse in case of \subsectionmark is not defined (book)
  \global\csdef{eled@sectioning@#3#5}{\subsection*{#2}}%
  \listcsgadd{eled@sections#5@@}{#3}%
}

\def\eled@subsubsection#1#2#3#4#5{%
  \ifstrempty{#4}{}
  \ifstrempty{#1}{}\global\csdef{eled@sectioning@#3#5}{\subsubsection{#2}}%
  \global\csdef{eled@sectioning@#3#5}{\subsubsection{[#1]{#2}}}% Need for \pairs, because of using parbox. \csuse in case of \subsubsectionmark is not defined (book)
  \global\csdef{eled@sectioning@#3#5}{\subsubsection*{#2}}%
  \listcsgadd{eled@sections#5@@}{#3}%
}
41 Page breaking or no page breaking depending of specific lines

By default, page breaks are automatic. However, the user can define lines which will force page breaks, or prevent page breaks around one specific line. On the first run, the line-list file records the line number of where the page break is being changed (either forced, or prevented). On the next run, page breaks occur either before or after this line, depending on how the user sets the command. The default setting is after the line.

```
\normalpagebreak
```

\normalpagebreak is an etoolbox list which contains the absolute line number of the last line, for each page.

5266 \def\normalpagebreak{}

\prevpb \prevnopb

\prevpb macro is a etoolbox list, which contains the lines in which page breaks occur (before or after). The \prevnopb macro is a etoolbox list, which contains the lines with NO page break before or after.

5267 \def\l@prevpb{}
5268 \def\l@prevnopb{}

\l@prevpb \l@prevnopb

\l@prevpb list. The \l@prevpb macro writes the call to \l@prevpb in line-list file. The \l@prevnopb macro writes the call to \l@prevnopb in line-list file. The \l@prevnopb macro writes the call to \l@prevnopb in line-list file.

5269 \newcommand{\l@prevpb}{\write\linenum@out{\string\l@prevpb}}
5270 \newcommand{\l@prevnopb}[1]{\write\linenum@out{\string\l@prevnopb[#1]}}
5271 \newcommand{\l@nopb}{\write\linenum@out{\string\l@nopb}}
5272 \newcommand{\l@nopbnum}[1]{\write\linenum@out{\string\l@nopbnum[#1]}}

\l@nopb \l@nopbnum \l@nopbnum

\l@nopb list. The \l@nopb adds the absolute line number in the \prevpb list. The \l@nopbnum adds the argument in the \prevpb list. The \l@nopb adds the absolute line number in the \prevnopb list. The \l@nopbnum adds the argument in the \prevnopb list.

5273 \newcommand{\l@nopb}{\listxadd{\l@prevpb}{\the\absline@num}}
5274 \newcommand{\l@nopbnum}[1]{\listxadd{\l@prevpb#1}{\the\absline@num}}
5275 \newcommand{\l@nopbnum}[1]{\listxadd{\l@prevnopb}{\the\absline@num}}
5276 \newcommand{\l@nopbnum}[1]{\listxadd{\l@prevnopbnum}{\the\absline@num}}

\l@nopbsetting \l@nopbsetting

\l@nopbsetting macro only changes the value of \l@nopbmacro, for which the default value is before.

5277 \def\l@nopbsetting{before}
5278 \newcommand{\l@nopbsetting}[1]{\gdef\l@nopbsetting{#1}}

\l@checkpb \l@checknopb

\l@checkpb and \l@checknopb are called before or after each line. They check if a page break must occur, depending on the current line and on the content of \@pb.

5279 \newcommand{\l@checkpb}{\xifinlist{\the\absline@num}{\l@prevpb}{\pagebreak[4]}}
\newcommand{\led@check@nopb}{%
  \IfStrEq{\led@pb@setting}{before}{%
    \xifinlist{\the\absline@num}{\l@prev@nopb}{%
      \numdef{\absprevline}{\the\absline@num-1}%
      \xifinlist{\absprevline}{\normal@page@break}{%
        \nopagebreak[4]\enlargethispage{\baselineskip}}%
    }{}}%
  }%
  \IfStrEq{\led@pb@setting}{after}{%
    \xifinlist{\the\absline@num}{\l@prev@nopb}{%
      \xifinlist{\the\absline@num}{\normal@page@break}{%
        \nopagebreak[4]\enlargethispage{\baselineskip}}%
    }{}}%
}%
\IfStrEq{\led@pb@setting}{after}{%
  \xifinlist{\the\absline@num}{\l@prev@nopb}{%
    \xifinlist{\the\absline@num}{\normal@page@break}{%
      \nopagebreak[4]\enlargethispage{\baselineskip}}%
  }{}}%
}

42 Long verse: prevents being separated by a page break

\iflednopbinverse The \texttt{\lednopbinverse} boolean is set to false by default. If set to true, \eledmac will automatically prevent page breaks inside verse. The declaration is made at the beginning of the file, because it is used as a package option.

\check@pb@in@verse The \texttt{\check@pb@in@verse} checks if a verse is broken in two page. If true, it adds:

- The absolute line number of the first line of the verse -1 in the \texttt{\led@pb} list, if the page break must occur before the verse.
- The absolute line number of the first line of the verse -1 in the \texttt{\led@nopb} list, if the page break must occur after the verse.

\newcommand{\check@pb@in@verse}{%
43  The End

"i/code;i"
Appendix A  Some things to do when changing version

Appendix A.1  Migrating from edmac

If you have never used \textsc{EDMAC}, ignore this section. If you have used \textsc{EDMAC} and are starting on a completely new document, ignore this section. Only read this section if you are converting an original \textsc{EDMAC} document to use eledmac.

The package still provides the original \texttt{text} command, but it is (a) deprecated, and (b) its name has been changed\textsuperscript{30} to \texttt{critext}; use the \texttt{edtext} macro instead. However, if you do use \texttt{critext} (the new name for \texttt{text}), the following is a reminder.

\begin{verbatim}
\textbackslash critext{⟨lemma⟩}⟨commands⟩/
\end{verbatim}

The \texttt{⟨lemma⟩} argument is the lemma in the main text: \texttt{critext} both prints this as part of the text, and makes it available to the \texttt{⟨commands⟩} you specify to generate notes. The / at the end terminates the command; it is part of the macro’s definition so that spaces after the macro will be treated as significant.

For example:

\begin{verbatim}
I saw my friend \texttt{critext}{Smith} \textbackslash Afootnote{Jones C, D.}/ on Tuesday.
\end{verbatim}

The lemma \texttt{Smith} is printed as part of this sentence in the text, and is also made available to the footnote that specifies a variant, \texttt{Jones C, D}. The footnote macro is supplied with the line number at which the lemma appears in the main text.

The \texttt{⟨lemma⟩} may contain further \texttt{critext} commands. Nesting makes it possible to print an explanatory note on a long passage together with notes on variants for individual words within the passage. For example:

\begin{verbatim}
\texttt{critext}{I saw my friend} \texttt{critext}{Smith}{\textbackslash Afootnote{Jones C, D.}/ on Tuesday.} \texttt{Bfootnote{The date was July 16, 1954.}/}
\end{verbatim}

However, \texttt{critext} cannot handle overlapping but unnested notes—for example, one note covering lines 10–15, and another covering 12–18; a \texttt{critext} that starts in the \texttt{⟨lemma⟩} argument of another \texttt{critext} must end there, too. (The \texttt{lemma} and \texttt{linenum} commands may be used to generate overlapping notes if necessary.)

\textsuperscript{30} A name like \texttt{text} is likely to be defined by other \LaTeX{} packages (it certainly is by the AMS packages) and it seems sensible to try and avoid clashes with other definitions.
Appendix A  Some things to do when changing version

The second argument of the \critext macro, \textit{⟨commands⟩}, is the same as the second argument to the \edtext macro.

It is possible to define aliases for \critext, which can be easier to type. You can make a single character substitute for \critext by saying this:

\catcode\langle=\active
\let<=\critext

Then you might say \textit{⟨{Smith}\textit{variant}{Jones}⟩}. This of course destroys the ability to use \textit{⟨} in any new macro definitions, so long as it remains in effect; hence it should be used with care.

Changing the character at the end of the command requires more work:

\catcode\langle=\active
\def\xtext#1#2>{\critext{#1}{#2}/}
\let<=\xtext

This allows you to say \textit{⟨{Smith}\textit{Afootnote}{Jones}⟩}.

Aliases for \critext of the first kind shown here also can’t be nested—that is, you can’t use the alias in the text that forms the first argument to \critext. (See section \textsection{22} to find out why.) Aliases of the second kind may be nested without any problem.

If you really have to use \critext in any of the tabular or array environments, then \edtext must not be used in the same environment. If you use \critext in one of these environments then you have to issue the declaration \texttt{\usingcritext} beforehand. The declaration \texttt{\usingedtext} must be issued to revert to the default assumption that \edtext will be used.

Appendix A.2  Migration from ledmac to eledmac

In eledmac, some changes were made in the code to allow for easy customization. This can cause problems for people who have made their own customizations. The next sections explain how to correct this.

If you created your own series using \texttt{\addfootins} and \texttt{\addfootinsX}, you should instead use the \texttt{\newseries} command (see 4.7 p.26). You must delete your \texttt{\Xfootnote} command.

If you customized the \texttt{\XXXXXfmt} command, you should see if commands for display options (4.4 p.20) and options in \texttt{\Xfootnote} (4.1 p.17) can’t do the same things. If not, you can add a new ticket in Github to request a new function it\textsuperscript{31}.

If for some reason you don’t want to make the modifications to use eledmac new functions, you can continue to use your own \texttt{\XXXXXfmt} command, but you must replace:

\renewcommand\{\XXXXXfmt\}[3]

\textsuperscript{31}https://github.com/maieul/ledmac/issues
Appendix A.3 Migration to eledmac 1.5.1

If you don’t do that, you will see a spurious \[X\], where X is series letter.
If you used a \protect command inside a \footnote command inside a numbered section, you must change the \protect to \noexpand. If you don’t, the command after the \protect won’t be displayed.

Appendix A.3 Migration to eledmac 1.5.1

The version 1.5.1 corrects a bug with \texttt{stanzaindentsrepetition} (cf. p. 27). This bug had two consequences:

1. \texttt{stanzaindentsrepetition} didn’t work when its value was greater than 2.
2. \texttt{stanzaindentsrepetition} worked wrong when its value was equal to 2.

So, if you used \texttt{stanzaindentsrepetition} with value equal to 2, you must change your \texttt{setstanzaindents}. Explanation:

\begin{verbatim}
\setcounter{stanzaindentsrepetition}{2}
\setstanzaindents{5,1,0}
\end{verbatim}

This code, in a version older than 1.5.1, made that the first verse had an indent of 0, the secund verse of 1, the third verse of 0, the fourth verse of 1 etc.

But instead the code should have assigned the reverse: the first verse had an indent of 1, the secund verse of 0, the third verse of 1, the fourth verse of 0 etc.

So version 1.5.1 corrected this bug. If you want to keep the older presentation, you must change:

\begin{verbatim}
\setcounter{stanzaindentsrepetition}{2}
\setstanzaindents{5,1,0}
\end{verbatim}

by:

\begin{verbatim}
\setcounter{stanzaindentsrepetition}{2}
\setstanzaindents{5,0,1}
\end{verbatim}

Appendix A.4 Migration to eledmac 1.12.0

The migration to eledmac 1.12.0 is easy:

- You must delete all the auxiliary files, and so one, make the normal three runs.
- If you have modified \texttt{l@reg}, which is not advisable, you must rename it to \texttt{\@nl@reg}.
Anyway, there is another problem. If you have text in brackets just after `\pstart` or `\pend`, the text will be considered an optional argument of `\pstart` or `\pend` (see 3.2.2, p. 11). In this case, just add a `\relax` between `\pstart/\pend` and the brackets.

The version 1.12.0 adds new best way to manage section title inside numbered text. Please read §14 (p. 39).

**Appendix A.5  Migration to eledmac 17.1**

The version change the default behavior of `\pstartinfootnote`. Henceforth, the psstart will be printed if footnote only for the section of text where you have called `\numberpstarttrue`.

We don’t see any reason to print it in other section. However, if you want to print the pstart number in all footnote, with or without `\numberpstarttrue`, you can use `\pstartinfootnoteeverytime`. 
References


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

\@ledsidenote .................................. 4112, 4120
\# ........................................... 4907, 5090 \@online .................................... 1928
\& ........................................... 22, 3966, \@wrindexm@ ... 3793, 3795, 3798,
3970, 3971, 4033, 4048, 4069, 4071 3805, 3807, 3810, 3815, 3817, 3820
\@ledleftnote ................. 4111, 4119 \@EDROWFILL@ ................. 4303, 4518
\@ledrightnote ............... 4110, 4118 \@M ..................................... 1928, 4026, 4038
Abu Kamil Shuja’ b. Aslam 4

B

ballast 37
ballast@count 1225, 1228, 1233, 1474
Beeton, Barbara Ann Neuhaus Friend 8
beforeeledchapter 4870
beforeeledchapter 4813
beforelemmaseparator 17
beforenotesX 19
beforenumberinfootnote 16
beforesymlinenum 17
beforeXnotes 19
beginnumbering 6, 204, 302, 1045, 1119
beginnumberingR 1144
Bendnote 13
Bfootnote 13
bseries 1027
bbooknoteX 19
bbookXendnote 19
bbookXnote 19
body 1507, 1508, 3968, 4070
bodyfootmarkA 30
box 1201, 1203, 1824, 1839, 1908, 1927, 2453, 2467, 3118, 3212, 3240
boxfootnotenumbers 3046, 3051, 3056
boxlinenum 17
boxmaxdepth 3121
boxylinenum 17
Breder, Simon 4
Breger, Herbert 2, 5, 182
Brey, Gerhard 4
brokenpenalty 1104
Burt, John 3
Busard, Hubert L. L. 4
bypage@false 306, 320, 326
bypage@true 306, 314
bypstart@false 306, 315, 327
bypstart@true 306, 321
C

c@addcolcount 4487
c@ballast 1225, 1233
c@firstlinenum 365, 1320, 1322, 1325, 1327
c@firstsublinenum 365, 1320, 1322, 1325, 1327

c@labidx 3735
c@linenumincrement 365, 1323, 1324
c@MPfootnote 3650, 3686, 3713
c@page 744, 746, 752, 755, 757, 760, 3432, 3440
Index 231

\c@pstart 1027, 3279
\c@pstartR 3266
\c@sublinenumincrement 369, 1310, 1311
\Cendnote 4423, 4428, 4434, 4439, 4445, 4450, 4672, 4674
\chaptermark 4866, 5203, 5207
\char 1588, 1795, 2157, 2419, 2532, 2545, 3648, 3701
\content 2730, 2742, 2755, 2770, 2774, 2787, 2790, 3453, 3456, 3460, 3468, 3471, 3475, 3483, 3486, 3490
Copernicus, Nicolaus 4
\count 1763, 1768, 1780, 1786, 1951, 1954, 2020, 2048, 2252, 2257, 2273, 2276, 2341, 2344, 2405, 2410
\countdef 3087
\cr 1929, 1932
\create@edindex@for@memoir 3769, 3873
\create@edindex@notfor@memoir 3827, 3869, 3872, 3876
\CRITEXT 218, 832, 848, 4196, 4278, 4307
\cs 843, 844, 2798, 2827–2829, 2951, 3054, 3071
\csdef 4284, 5202, 5203, 5206, 5207, 5212, 5213, 5221, 5222, 5225, 5226, 5230, 5231, 5239, 5240, 5243, 5244, 5248, 5249, 5256, 5257, 5260, 5261
\csgdef 1756, 1774, 1940, 2009, 2242, 2263, 2331, 2398, 2675–2693, 2699, 2701, 2702, 2704–2706, 2708–2719, 2765, 2782, 2851
\clet 2707, 2793
\cletcs 2797, 3609, 3635, 4094, 4104, 4279, 4290, 4295, 4310
\csnumdef 1080, 1082
\csnumgdef 927, 944, 968, 976
\csundef 472, 1178
Index

Leal, Jerónimo@Leal, Jerónimo 3
\leads 159
\leads 1140, 1151, 5270
\leads 1140, 1151, 5270
\leads 98, 1113, 1118
\leads 63, 877
\leads 172, 772, 781
\leads 161, 4510
\leads 75, 310
\leads 161, 4506
\leads 134, 3585
\leads 134, 3592
\leads 134, 3565
\leads 57, 269
\leads 57, 300
\leads 57, 206
\leads 98, 1075
\leads 98, 1072
\leads 98, 1048
\leads 98, 1044
\leads 161, 4514
\leads 161, 4186
\leads 161
\leads 175, 3729
\leads 172
\leads 178
\leads 65, 266
\leads 73, 288
\leads 5271, 5273
\leads 5272, 5273
\leads 5269, 5273
\leads 738, 748, 760, 1140, 1151, 5277, 5281, 5290, 5303, 5307
\leads 5270, 5273
\leads 442, 450
\leads 442, 449–451
\leads 155, 4042
\leads 128, 3176
\leads 125
\leads 125, 2589
\leads 125
\leads 112, 1296
\leads 88, 597
\leads 88, 591
\leads 78, 330
\leads 78, 358
\leads 78, 385
\leads 94, 800
\leads 94, 807
\leads 121, 3410
\leads 78, 411
\leads 114, 3302
\leads 123, 3777
\leads 86, 506
\leads 119, 3377
\leads 114, 3337
\leads 131, 2671
\leads 4738
\leads 4738
\leads 14
\leads 13
\leads 175, 3729
\leads 28, 3428
\leads 3767
\leads 3767
\leads 28, 3428, 4111, 4114, 4119
\leads 423
\leads 1160, 1205, 3702, 3706
\leads 29, 3421, 3535
\leads 29, 1201, 3421
\leads 29, 3421, 3535
\leads 36, 5269
\leads 16, 36
\leads 5269, 5309
\leads 28, 3439
\leads 3428
\leads 36, 5269
\leads 5269, 5305
\leads 36, 5277
\leads 2979
\leads 28, 3428, 4110, 4113, 4118
\leads 1166, 1205, 3703, 3710
\leads 29, 3421, 3543
\leads 29, 1203, 3421
\leads 29, 3421, 3543
Index


\mathchardef 3982

\maxdepth 3112

\maxdimen 1819, 1834, 2448, 2462

\maxhnotesX 20

\maxhXnotes 20

Mayer, Gyula 5

\measurembody 4581, 4587, 4617, 4635

\measuremcell 4222, 4248

\measuremrow 4246, 4592

\measuretbody 4597, 4603, 4625, 4643

\measuretcell 4234, 4253

\measuretrow 4251, 4609

\message 74, 212

Middleton, Thomas 5, 57

\minipage (environment) 26

Mittelbach, Frank 4

\morenoexpands 38, 826

\moveleft 2566, 2572, 4526, 4531, 4539

\moveright 4552, 4565, 4574

\mpfnpos 30, 2560

\mpnormalfootgroup 1721, 1767

\mpnormalfootgroupX 4199, 2256

\mpnormalvfootnote 1583, 1766, 1945, 2014

\mpnormalvfootnoteX 2152, 2255, 2268, 2336

\mppara@footgroup 1785, 1873

\mppara@footgroupX 2409, 2479

\mppara@vfootnote 1784, 1828

\mppara@vfootnotex 2408, 2442

\mpthreecolfootgroup 1946, 1982

\mpthreecolfootgroupX 2337, 2368

\mpthreecolfootsetup 1947, 1953

\mpthreecolfootsetupX 2338, 2340

\mptwocolfootgroup 2015, 2047

\mptwocolfootgroupX 2269, 2301

\mptwocolfootsetup 2016, 2047

\mptwocolfootsetupX 2270, 2272

\multfootsep 29, 2076, 2086

\multiplefootnotemarker 2076, 2080, 2081, 2083

\NeedsTeXFormat 2

\new 2809, 2811, 2815, 2816, 2818, 2819

\newline 737, 1160, 1181, 1184

\newbox 1017, 1020, 3418, 3419, 4135, 4137, 4718, 4719

\newcommand 925, 1032, 1071, 1226, 1413, 1438, 1635, 1637, 1639, 1641, 1643, 2606, 2978, 3114, 3189, 3448, 3550, 3725, 3726, 3747, 3956, 3958, 4501, 4875, 5092

\newcounter 365, 367, 369, 371, 1026, 1226, 2777, 3284, 3285, 3737, 3990, 4487

\newhookcommand@series 2863

\newhookcommand@series@reload 2936

\newhooktoggle@series 250, 253, 256–2560, 2967

\newhooktoggle@series@reload 2967, 2973, 2974


\newinsert 2723–2726

\newlength 423, 3974

\newlinechar 2785

\newread 479

\newrobustcmd 3315

\newseries 2665, 2803, 2804

\newseries@ 2666, 2670

\newtoggle 1747, 1751, 2673, 2674, 2694–2698, 2700, 2703, 2720, 2721, 2981, 2982

\newverse 156, 4028

\newwrite 720, 2605, 5091

\next 4218, 4223, 4226, 4231, 4232, 4235, 4238, 4243, 4244, 4247, 4249, 4250, 4252, 4254, 4255, 4260, 4419, 4422, 4424, 4425, 4427, 4429, 4430, 4433, 4435, 4436, 4438, 4440, 4441, 4444, 4446, 4447, 4449, 4451, 4452, 4457, 4458, 4460
Index

\printpstart \hfill 1624, 3077 \hfill \reserveinserts \hfill 24, 38
\process@tendbody \hfill \resetprevline@ \hfill 60, 247, 471, 1085, 1244
\hfill \resetprevpage@ \hfill 475
\ProcessOptions \hfill 20 \hfill \resetprevpage@num \hfill 60, 248, 475, 3685
\protected@csxdef \hfill 2772, 4096 \hfill \restore@familiarnotes \hfill 4092, 4128
\protected@edef \hfill \restore@notes \hfill 4112, 4329,
\hfill 4348, 4364, 4380, 4395, 4412, 4610
\protected@write \hfill \restore@idnotes \hfill 4109, 4127
\hfill \resumenumbering \hfill 8, 280
\right \hfill 4465, 4468, 4473, 4476
\rightctab \hfill 4542, 4634
\rightlinenum \hfill \rightlinenumR

\ProvidesPackage \hfill 3 \hfill 10, 423, 1344, 1352, 4881, 4897
\ps@rtedLfalse \hfill 190, 218, 234, 256 \hfill \rightnoteupfalse \hfill 4882, 4898
\ps@rtedLtrue \hfill 190, 286 \hfill \righttab \hfill 4555, 4616
\pstart \hfill 6, 64, 99, 102, 103, 108,
\hfill 1022, 1125, 1410, 4020, 4742,
\hfill 4745, 4750, 4753, 4758, 4761,
\hfill 4766, 4769, 4774, 4777, 4782,
\hfill 4875, 4793, 4795, 4799, 4801, 4813
\pstartref \hfill 27, 3332
\pstartinfootnote \hfill 16, 316, 322, 328
\pstartinfootnoteeverytime \hfill 16
\pstartline \hfill 1082, 1085
\pstartnum \hfill 1412
\pstartnumfalse \hfill 1443, 1450
\pstartnumtrue \hfill 1092, 1439
\pstartref \hfill 27, 3332
\pstartinfootnote \hfill 16, 316, 322, 328
\pstartinfootnoteeverytime \hfill 16
\pstartline \hfill 1082, 1085
\pstartnum \hfill 1412
\pstartnumfalse \hfill 1443, 1450
\pstartnumtrue \hfill 1092, 1439
\pstartref \hfill 27, 3332
\Q \hfill 4738
\quote \hfill 4738
\{ discourage \hfill 1572, 2379
\} discourage \hfill 1572, 2379
\R \hfill 1598
\set@line@action \hfill 517, 601, 608, 619
\reset@line@action \hfill 517, 601, 608, 619
\sectionmark \hfill 2665
\section@num \hfill 186, 210,
\hfill 212, 213, 229, 230, 287–289,
\hfill 965, 967, 968, 970, 984, 986, 2614
\section@numR \hfill 973, 975, 976, 978, 998, 1000
\sectionmark \hfill 4867, 5222, 5226
\select@lemmafont \hfill 827, 1510
\select@lemmafont \hfill 21
\select@lemmafont \hfill 21
\sch"opf, Rainer \hfill 4
\shorttitle \hfill 2849
\shorttitletext \hfill 4078, 4595, 4727
\rtab \hfill 4078, 4595, 4727
\rtabtext \hfill 4078, 4595, 4727

\S \hfill 5
\sacrobosco \hfill 14, 833, 925
\sameword \hfill 833, 925
\sameword@linedtext \hfill 833, 925
\s@on@list \hfill 1332, 1334
\sch"opf, Rainer \hfill 4
\section@num \hfill 186, 210,
\hfill 212, 213, 229, 230, 287–289,
\hfill 965, 967, 968, 970, 984, 986, 2614
\section@numR \hfill 973, 975, 976, 978, 998, 1000
\sectionmark \hfill 4867, 5222, 5226
\select@lemmafont \hfill 827, 1510
\select@lemmafont \hfill 21
\select@lemmafont \hfill 21
\sch"opf, Rainer \hfill 4
\section@num \hfill 186, 210,
\hfill 212, 213, 229, 230, 287–289,
\hfill 965, 967, 968, 970, 984, 986, 2614
\section@numR \hfill 973, 975, 976, 978, 998, 1000
\sectionmark \hfill 4867, 5222, 5226
\select@lemmafont \hfill 827, 1510
\select@lemmafont \hfill 21
\select@lemmafont \hfill 21
\sch"opf, Rainer \hfill 4
\section@num \hfill 186, 210,
\hfill 212, 213, 229, 230, 287–289,
\hfill 965, 967, 968, 970, 984, 986, 2614
\section@numR \hfill 973, 975, 976, 978, 998, 1000
\sectionmark \hfill 4867, 5222, 5226
\select@lemmafont \hfill 827, 1510
\select@lemmafont \hfill 21
\select@lemmafont \hfill 21
\sch"opf, Rainer \hfill 4
Index

\sub@change ................. 246, 527, 528, 534, 578, 580, 583, 585
\sub@lock ................. 243, 459, 542, 544, 546, 549, 646, 647, 649, 650, 668, 669, 671, 1214, 1286, 1288, 1289, 1291, 1367, 1403, 1405, 1407
\sub@off ................. 577, 794
\sub@on ................. 577, 790
\sublinenumber ............ 241, 429, 430, 456, 550, 554, 564, 589, 590, 592, 604, 622, 711, 715, 1215, 1220, 1243, 1251, 1306–1308, 3313
\sublinenumberstyle ....... 11, 414
\sublinenumincrement ..... 9, 374
\sublinenum@p ............ 414
\sublinenum@p ............ 414, 430, 1691, 1695, 2648, 2652, 3313
\sublineref ............... 27, 3329
\sublines@false ........... 244, 457, 531, 1276
\sublines@true ........... 457, 529, 1274
\subblock@disp ........... 405, 1369, 1373, 1377
\subblock@disp ........... 85, 405
\subsection ............... 2823, 4759, 4767, 5239, 5243, 5248, 5249
\subsection@actionmark .... 4868, 5240, 5244
\subsubsection ........... 2825, 4775, 4783, 5256, 5257, 5260, 5261
Sullivan, Wayne .......... 4, 5, 22, 37, 46, 50, 112, 113, 155, 178
\sw@atthisline .......... ... 986, 988, 994, 1000, 1002, 1008
\sw@list ................. 497, 923, 929, 930, 971
\sw@list@inedtext ....... 498, 923, 933, 935, 989, 992
\sw@list@inedtextR ..... 950, 952, 1003, 1006
\sw@list@R .............. 946, 947, 979
\synlinenum .............. 16
\synlinenum .............. 2699, 2976
\sz@apenalty ............ 4081, 4040, 4062

T
\tab@llzwischen ........... 4654, 4662
\tabeskip ................. 4666, 4706–4708, 4714–4716
\tab@Hilfbox ............. 4705, 4707, 4709, 4713, 4715, 4717, 4718
\tab@Hilfbox ............. 4704, 4706, 4708, 4712, 4714, 4716, 4718
Tapp, Christian .......... 3
\temp ............... 1173, 1174, 3893, 3898
\texbf ................. 877
\texheight .............. 3228
\textindent ............. 1612–1614
\textsc ................. 877
\texsuperscript ......... 1015, 2077, 2139, 2172
\textwidth .............. 3648, 3700
\the@sw ................. 970, 971, 978, 979, 990, 992, 995, 1004, 1006, 1009
\thead@colcount ........ 4487, 4494, 4497
\theendpageline ........ 3745, 3795, 3807, 3817, 3835, 3847
\thefootnote ............ 30
\thelabidx ............... 3741, 3744, 3903, 3907, 3912, 3914
\theline ................. 3292, 3294
\temp@fn ................. 3650, 3686, 3713
\temp@footnote ........... 3650, 3686, 3713
Theodosius ............... 5
\thepage ................. 741, 744, 746, 755, 757, 760, 3266, 3279, 3744
\thepageline ............. 3743, 3798, 3810, 3820, 3838, 3851
\thepart ................. 8, 1022, 1125, 1441, 1448, 1632
\thepart@L ............... 1629
\thepart@R ............... 1627
\thestartpageline ........ 3745, 3793, 3805, 3815, 3834, 3844
\thesubline .............. 3292
\thinspace ............... 1617, 1619
\thisfootnote ........... 2224, 2225
\threecol@footfmt ........ 1942, 1962
\threecol@footfmtX ....... 2333, 2353
\threecol@group ........ 1943, 1977
\threecol@groupX ......... 2334, 2368
\threecol@setup ........... 1944, 1950
\threecol@setupX ........ 2335, 2340
\threecol@footnote ........ 1941, 1957
\threecol@footnoteX ........ 2332, 2347
\togglefalse ............. 1701, 1804, 2178, 2428
\toggletrue .............. 1748, 1752
\tolerance ............... 1966, 2030, 2295, 2362
\twocol@footfmt ........ 2011, 2019
\twocol@footfmtX ........ 2265, 2286
\twocol@group ........... 2012, 2019
\twocol@groupX .......... 2266, 2301
\twocol@setup ........... 2013, 2019
\twocol@setupX ........ 2267, 2272
### Change History

**v0.1.**

- General: First public release  .......... 1

**v0.2.**

- General: Added tabmac code, and extended indexing  .......... 1
- \eledmac@error: Added \eledmac@error and replaced error messages  .......... 43
- \ifl@dmemoir: Added \ifl@dmemoir for memoir class having been used  .......... 42
- \morenoexpands: Added \@dtabnoexpands to \no@expands  .......... 75

**v0.2.1.**

- \@lab: Removed page setting from \@lab  .......... 157
- General: Added text about normal labeling  .......... 28
- Bug fixes and match with mempatch v1.8  .......... 1
- Major changes to insert code when memoir is loaded  .......... 152
- \doxtrafeet: Renamed \doxtrafeet to \@dodxtrafeet  .......... 151
- \edlabel: Tweaked \edlabel to get correct page numbers  .......... 155
- \@d@dmakecol: Rewrote \@dmakecol, calling it \@d@dmakecol  .......... 150
- \@d@dododoreinxtrafeet: Renamed \@d@dododoreinxtrafeet to \@d@dododoreinxtrafeet  .......... 152
- \@d@dodofootinsert: Renamed \@d@dodofootinsert as \@d@dodofootinsert  .......... 150
- \m@makecolintro: Added \m@makecolfloats, \m@makecoltext and \m@makecolintro  .......... 150
- \morenoexpands: Removed some \@lets from \no@expands. These were in EDMAC but I feel that they should not have been as

**v0.3.**

- \@lab: Replaced \the\line@num by \linenumr@p\line@num in \@lab, and similar for sub-lines  .......... 1
- \@nl@reg: Added a bunch of code to \@nl for handling \setlinenum  .......... 62
- General: Includes edstanza and more  .......... 1
- \ledlinenum: Added \linenumr@p and \sublinenumr@p to \leftlinenum and \rightlinenum  .......... 54
- \linenumberlist: Added \linenumr@p and \sublinenumr@p to \printendlines  .......... 46
- \printendlines: Added \linenumr@p and \sublinenumr@p to

**v0.2.2.**

- General: Improved paragraph footnotes  .......... 1
- New Dekker example  .......... 1
- Used \providecommand for \gobblethree to avoid clash with the amssfonts package  .......... 46
- \footfudgefiddle: Added \footfudgefiddle  .......... 110
- \line@list@stuff: Added initial write of page number in \line@list@stuff  .......... 69
- \para@footsetup: Added \footfudgefiddle to \para@footsetup  .......... 111
- \para@footsetupX: Added \footfudgefiddle to \para@footsetupX  .......... 130
- \doxtrafeet: Renamed \doxtrafeet to \@dodxtrafeet  .......... 151
- \edlabel: Tweaked \edlabel to get correct page numbers  .......... 155
- \@d@dmakecol: Rewrote \@dmakecol, calling it \@d@dmakecol  .......... 150
- \@d@dododoreinxtrafeet: Renamed \@d@dododoreinxtrafeet to \@d@dododoreinxtrafeet  .......... 152
- \@d@dodofootinsert: Renamed \@d@dodofootinsert as \@d@dodofootinsert  .......... 150
- \m@makecolintro: Added \m@makecolfloats, \m@makecoltext and \m@makecolintro  .......... 150
- \morenoexpands: Removed some \@lets from \no@expands. These were in EDMAC but I feel that they should not have been as

**Z**

- \z@skip  .......... 1575, 1581, 2149, 2283
Change History

\printlines .................. 106
\sublinenum: Added \linenumberstyle
and \sublinenumstyle ... 54

v0.3.1.
General: Not released. Added remarks about the parallel package .................. 1

v0.4.
\@iiiminipage: Modified kernel \@iiiminipage and \endminipage to cater for critical footnotes ................. 167
General: Added minipage, etc., support .................. 1
\ledgroup: Added ledgroup environment .................. 168
\ledgroupsized: Added ledgroupsized environment ............ 168
\edtext: Added \showlemma to \edtext (and \critext) .... 76
\footnormal: Added minipage footnote setup to \footnormal . 109
\ifledsecnolinenumber: Added final/draft options ........... 41
\l@dfeetendmini: Added \l@dfeetbeginmini, \l@dfeetendmini and all their supporting code 166
\mpnormalfootgroup: Added \mpnormalfootgroup ...... 107
\mpnormalvfootnote: Added \mpnormalvfootnote ...... 102
\showlemma: Added \showlemma . 41

v0.4.1.
\@opxtrafeetii: Added \@opxtrafeetii .................. 151
General: Added code for changing \doclearpage ........... 153
Not released. Minor editorial improvements and code tweaks . 1
Only change \footnotetext and \footnotemark if memoir not used .......... 121
\doxtrafeetii: Changed \doxtrafeetii code for easier
extensions .............. 151
\edindex: Let ledmac take advantage of memoir's indexing . 171

v0.5.
\@footnotetext: Enabled regular \footnote in numbered text 122
\@sympar: Eliminated \marginpar disturbance ............... 160
General: Added left and right side notes .................. 160
Added sidenotes, familiar footnotes in numbered text .... 1

v0.5.1.
General: Added moveable side note 160
Fixed right line numbers killed in v0.5 .................. 1
\affixline0num: Changed \affixline0num to cater for sidenotes .................. 93
\ledgroupsized: Only change \hsize in ledgroupsized environment otherwise page number can be in wrong place .... 168
\l@dgetsidenote@margin: Added \sidenote and \sidenote@margin .... 160

v0.6.
\@lopR: Added \@pend, \@pendR, \@lopL and \@lopR in anticipation of parallel processing .... 64
\onl@reg: Added \fixpage to \onl ................. 62
Extended \onl to include the page number ................. 62
General: Fixed long paragraphs looping .................. 1
Fixed minor typos .................. 1
Prepared for eledpar package . 1
\fixpage: Added \lastpage and \fixpage .............. 63
\new@line: Extended \new@line to output page numbers .... 69
\page@start: Made \page@start a no-op ............... 70
\vl@dbfnote: Changed \l@dbfnote and \vl@dbfnote as originals could give incorrect markers in the footnotes .......... 122

v0.7.
\onl@reg: Added \onl@reg .... 62
\oref@reg: Added \oref@reg .... 67
General: edmac having been available for 2 years, deleted the commented out original edmac texts .................. 1
Maïeul Rouquette new maintainer ........................................... 1
Made macros of all messages ........................................... 42
Replaced all \interfootnotelinepenalty etc., by just \interfootnotelinepenalty ........................................... 1
Tidying up for eledpar and ledarab packages ................. 1
\affixline@num: Added skipnumbering to \affixline@num ... 93
\do@actions@fixedcode: Added \do@actions@fixedcode ... 92
\do@actions@next: Added number skipping to \do@actions ... 91
\do@insidelinehook: Added \do@linehook for use in \do@line ........................................... 89
\endnumbering: Changed \endnumbering for eledpar .... 49
\f@r@l@cks: Added \ch@cksub@l@ck, \ch@ck@l@ck and \f@r@l@cks ... 95
\footsplitskips: Added \footsplitskips for use in many footnote styles .......... 102
\get@linelistfile: Added \get@linelistfile ................. 61
\ifledRcol@: Added \l@dnumpstartsL, \ifl@dpairing and \ifpst@rted for/from eledpar .......... 47
\initnumbering@reg: Added \initnumbering@reg ............ 48
\l@dcsnotetext@r: Added \l@demptyd@ta ....................... 89
\l@ddofootinsert: Deleted \page@start from \l@ddofootinsert ........................................... 150
\l@dgtl@ine@margin: Added \l@dgtl@ine@margin .......... 52
\l@dgtl@ock@disp: Added \l@dgtl@ock@disp .................. 53
\l@dgetsidenote@margin: Added \l@dgetsidenote@margin .. 160
\l@drsn@te: Added \l@dlsn@te and \l@drsn@te for use in \do@line ........................................... 89
\l@dunboxmpfoot: Added \l@dunboxmpfoot containing some common code .......... 167
\l@dzeropenalties: Added \l@dzeropenalties ...................... 85
\ledlinenum: Added \ledlinenum for use by \leftlinenum and \rightlinenum .................. 54
\line@list@stuff: Deleted \page@start from \line@list@stuff ........................................... 69
\list@clearing@reg: Added \list@clearing@reg ............. 61
\n@num@reg: Added \n@num ............... 67
\normalbfnoteX: Removed extraneous space from \normalbfnoteX ....................... 125
\resumenumbering: Changed \resumenumbering for eledpar 50
\setprintendlines: Added \setprintendlines for use by \printendlines .................. 136
\setprintlines: Added \setprintlines for use by \printlines ............. 105
\skipnumbering@reg: Added \skipnumbering and supports 72
\sublinenumincrement: Added \firstlinenum, \linenumincrement, \firstsublinenum and \linenumincrement .......... 53
\sublinenum@p: Using \linenum@p instead of \linenum@p .... 54
\sublinenum@p: Using \sublinenum@p instead of \sublinenum@p .......... 54
\vnumfootnoteX: Removed extraneous space from \vnumfootnoteX ....................... 125
v0.8.
General: Bug on endnotes fixed: in a // text, all endnotes will print and be placed at the ends of columns () .................. 1
v0.8.1.
General: Bug on \text ; \critex ; \lemma fixed: we can now use non-switching commands .... 1
v0.9.
General: No more ledpatch. All patches are now in the main file. .................. 1
v0.9.1.
General: Fix some bugs linked to integrating ledpatch on the main file. .................. 1
v0.10.
General: Corrections to \section and other titles in numbered sections ................ 1

v0.11.
General: Makes it possible to add a symbol on each verse's hanging, as in French typography. Redefines the command \hangingsymbol to define the character. .................... 1

v0.12.
General: For compatibility with eledpar, possibility to use \autopar on the right side. 1
Possibility to number \pstart . 8
Possibility to number the pstart with the commands \numberpstarttrue. .............. 1
\ifledRcol@: Added \ifledRcol and \ifnumberingR for/from eledpar ..................... 47

v0.12.1.
General: Don't number \pstarts of stanza. ......................... 1
The numbering of \pstarts restarts on each \beginnumbering. ...................... 1

v0.13.
General: New stanzaindentsrepetition counter to repeat stanza indents every n verses. ....... 23
New stanzaindentsrepetition counter: to repeat stanza indents every n verses. .......... 1
\managestanza@modulo: New stanzaindentsrepetition counter to repeat stanza indents every n verses. ................................ 180

v0.13.1.
General: \thepstartL and \thepstartR use now \bfseries and not \bf, which is deprecated and makes conflicts with memoir class. .... 1

v0.14.
General: Tweaked \edlabel to get correct line number if the command is first element of a paragraph. .............. 1
\edlabel: Tweaked \edlabel to get correct line number if the command is first element of a paragraph. .............. 155

v0.15.
General: Line numbering can be reset at each pstart. .............. 51
Possibility to print \pstart number inside. ..................... 8
\affixline@num: Line numbering can be disabled. .............. 93
\ifinserthangingsymbol: New management of hangingsymbol insertion, preventing undesirable insertions. .............. 179
\printlines: Line numbering can be reset at each pstart. ....... 106

v0.16.
\ifinserthangingsymbol: New new management of hangingsymbol insertion, preventing undesirable insertions. ....... 179

v1.0.
General: \lemma can contain commands. .......................... 13
Debug in lineation command . 10
New generic commands to customize footnote display. ...... 16
Options nonum and nosep in \Xfootnote. ..................... 13
Options of \Xfootnotes. ....... 100
Possibility to have commands in sidenotes. .............. 28
Some compatibility break with eledmac. Change of name: eledmac. .................. 1
\morenoexpands: Change to be compatible with new features 75

v1.0.1.
General: Correction on \numberonlyfirstinline with lineation by pstart or by page. .................... 16

v1.1.
General: Add \labelpstarttrue. .... 8
Add \numberonlyfirstintwolines ......................... 16
Add \pstartinfootnote and \onlypstartinfootnote ....... 16
New hook to add arbitrary code at the beginning of the notes . 18
New options for block of notes. 19
New package option: paraparatus. .......................... 1
New tools to change order of series .......................... 141
\ledfootinsdim: Deprecated .......................... 109
\preXnotes: New skip \preXnotes@ .......................... 109
v1.1.0.
General: Sectioning commands. . . 34
v1.2.
\endquote: Compatibility of \ledchapter with the memoir class. .............. 202
\preXnotes: Debug in familiar footnotes (but introduced by v1.1). 109
v1.3.
\endquote: Quotation and quote environment inside the numbering sections. .............. 202
v1.4.
General: Compatibility with LuaTeX of RTL notes. .... 41, 42
\edtext: Compatibility of \edtext (and \critext) with the right-to-left direction (with Polyglossia). .............. 76
\newseries@: Remembers the language of the lemma, in order to create a correct direction for the footnote separator. .... 139
\normalfootfmt: Direction of footnotes with polyglossia. .... 103
\rbracket: Switch the right bracket to a left bracket when the lemma is RTL (needs polyglossia or LuaTeX). .............. 104
v1.4.1.
\affixside@note: Remove spurious spaces. .............. 164
\endquote: New option noquotation. .............. 202
\labelrefsparsesubline: Fix bug with \edlabel. .............. 156
v1.4.2.
General: Debug with some special classes. .............. 1
v1.4.3.
General: Add \nonbreakableafternumber. 
Spurious space after familiar footnotes. .............. 1
v1.4.4.
General: Label inside familiar footnotes. .............. 1
v1.4.5.
General: Bug with komasscript + edelpar + chapter. .............. 1
v1.4.6.
General: Bug with memoir class introduced by 1.4.5. .............. 1
v1.4.7.
\endquote: Compatibility of sectioning commands with \autopar. .............. 202
v1.4.8.
General: Corrects a bug with parallel texts introduced by 1.1. .... 1
v1.4.9.
\normalbfnoteX: Allow to redefine \thefootnoteX with alph when some packages are loaded. .... 125
v1.5.
General: Correct indexing when the call is made in critical notes. .... 169
\do@insidelinehook: Added \do@insidelinehook for use in \do@line .............. 89
\edindex: Compatibility with \imakeidx package, and possibility to use multiple index with \edindex. .............. 171
\ifFN@bottom: Use the bottom option of footmisc package. .... 150
v1.5.1.
\managestanza@modulo: Correct stanzaindentsrepetition counter .............. 180
\normalvfootnoteX: Fix bug with normal familiar footnotes when mixing RTL and LTR text. .... 123
v1.6.0.
\falseverse: Add \falseverse macro. .............. 181
v1.6.1.
General: Corrects a false hanging verse when a verse is exactly the
length of a line. 
\AtEveryPstart: Spurious space in \pstart. 
\iftinsethangingsymbol: Hang verse is now not automatically flush right.
\l@dunbbox@line: Move the call to \inserthangingsymbol to allow use \hfill inside.
\pend: Spurious space in \pend.

v1.7.0.
General: New features for managing page breaks.

v1.8.0.
General: Compatibility with parled-group option of eledpar package.
If \imakeidx and \hyperref are loaded, adds hyperref in the index.
\endquote: Correction of sectioning commands in parallel texts.
\get@index@command: Debug \get@index@command and compatibility with hyperref package.
\newhookcommand@series@reload: Debug \beforenotesX and \maxhnotesX which didn’t work.
\prevpage@num: Correct \parafootsep when using with ledgroup.

v1.8.1.
General: Debug endnotes when more than one series is used (change the position where tools for endnotes are defined).

v1.8.2.
General: Debug compatibility problem with hebrew option of babel package.

v1.8.3.
General: Fixes spurious spaces added by v1.7.0.

v1.8.5.
General: Debug indexing in right column, with eledpar.

v1.9.0.
\doxtrafeet: Add \fnpos to choose the order of footnotes.
\l@defeetendmini: Add \mpfnpos to choice the order of footnotes in minipage / ledgroup.

v1.10.0.
General: Add \pstartref and \xpstartref to refer to a pstart number (extension of \edlabel).
\endquote: Correction of sectioning commands in parallel texts.

v1.10.1.
General: Compatibility with cleveref.

v1.10.2.
General: Compatibility of stanza with v1.8a of babel-greek.

v1.10.3.
General: Debug of cross-referencing.

v1.10.4.
General: Debug of critical notes in edtabular environment.

v1.10.5.
General: Debug of \pausenumbering.
Debug of \xxref. 

v1.10.6.
General: Debug of interaction between \autopar and \pausenumbering.

v1.11.0.
General: Add hooks to disable the font selection for lemma in footnote.

v1.11.1.
General: Correct a bug when a critical note starts with plus or minus.

v1.12.0.
\@nl@reg: To ensure compatibility with musixtex, \@l becomes \@l. Consequently, \@l@reg becomes \@nl@reg.
General: Add \ledinnernote and \ledouternote commands.
Add hyperlink to crossref (needs hyperref package).
Compatibility with musixtex.
Debug \eledmac sectioning command after using \resumenumbering. 
Ensure that \imakeidx is loaded before \eledmac. 
New hooks: \afterXrule and \afterruleX. 
New options for ragged-paragraph notes. 
New sectioning commands. 
Optional arguments for \psstart and \pend. 
\AtEveryPstart: New optional argument for \psstart, to execute code before it. 
\edindex: Use correctly default index when \imakeidx is loaded. 
\endquote: \eledxxx sectioning commands are deprecated and replaced by \eleledxxx commands. 
\ifledRcol@: Add \ifledRcol@ for \eledpar. 
\initnumbering@reg: \beginnumbering is defined only on \eledmac, not on \eledpar. 
\l@dcsnote, \l@dlsnote, \l@drsnote and \l@dcsnote defined only one time, in \eledmac, including needs for \eledpar case. 
\l@dgetsidenote@margin: \sidenotemargin is now directly defined in \eledmac to be able to manage \eledpar. 
\l@dunhbox@line: \do@line is split in more little commands. 
\newhookcommand@series@reload: Debug \beforenotesX and \maxXnotesX which didn’t work when called after \footparagraphX. 
\pend: New optional argument for \pend, to execute code after it. 
\stanza: \& can have an optional argument: content to be printed after. 
\Stanza can have an optional argument: content to be printed before. 
Add \newverse macro, \falseverse deprecated. 

v1.12.1. 
\wrap@edcrossref: Fix spurious spaces. 

v1.12.2. 
\l@dunhbox@line: Fix a bug with critical notes at the tops of pages (added by v12.0.0). 

v1.12.3. 
General: Add macros for new messages since v0.7. 
Correct bug with side and familiar notes in tabular environments. 
Debug \eledxxx with some paper size. 
Debug \eledinnernote and \eledouternote commands in the top of pages. 
Underline lemma in \eledxxx when using draft mode. 
\eledmac@error: Replaced error messages. 
\flag@end: \flag@start and \flag@end are now defined only one time for \eledmac and \eledpar. 
\flag@start send a error message when a \edtext is done without insert (note). 

v1.12.4. 
General: Debug spurious page breaks before \chapter (bug added by 1.12.0). 

v1.12.5. 
\edindex@hyperref: Debug \edindex when \hyperref is not loaded. 
\ssect: Debug \eledchapter in parallel with \memoir. 
\doinsidelinehook: Added \do@linehook and \doinsidelinehook.
\endnumbering: Allow to mix parallel columns and normal text when using \pausenumbering 49
\l@dgobblearg: \l@dgobblearg becomes \l@dgobbeloptarg 187
\l@drestoreforedtext: Debug optional arguments of \Xfootnote in tabular context 189
\resumenumbering: Debug \resumenumbering 50
\noeledsec: Add \noeledsec macro. 211
\v1.12.6.
\\wrap@edcrossref: \wrap@edcrossref is now robust 157
\v1.12.7.
\flag@end: \flag@start don’t send an error message when a \edtext is done without insert (note) but have a endnote 70
\v1.13.0.
\General: Add \noteswidthliketwocolumns and \notesXwidthliketwocolumns. 20
\\ifledsecnolinenumber: Added widthliketwocolumns option 41
\\newhooktoggle@series: Add \newhookcommand@toggle@reload 145
\\para@footsetupX: In \para@footsetupX, use \columnwidth instead of \hsize 130
\v1.13.1.
\General: Coming back of page and line breaking penalties’s management, deleted by error in v0.17. 1
\Debug quotation environment inside of a \pstart preceded by a sectioning command. 1
\\thestart: Add \l@dzeropenalties in \pstart 83
\v1.13.2.
\General: Fix bug with normal footnotes, added by v1.13.0. 1
\\ifledRcol@: Add \ifl@dpaging for \eledpar 47
\v1.13.3.
\General: Fix extra spaces with paragraphed footnotes, added by v1.13.0. 1
\v1.13.4.
\General: Fix bug with index when memoir class is used without hyperref 1
\v1.14.0.
\General: Debug spurious characters before endnotes. 135
Delete previous override of \l@@wrindexhyp at the beginning of a document when hyperref is not loaded. 175
Moves gobbling command 46
Provide \@gobblefour 46
\\edindex: Let eledmac take advantage of imakeidx even when memoir class is used 171
\v1.14.1.
\\osssect: Debug sectioning commands when using both handout and hyperref package. 209
\v1.14.2.
\\osssect: Debug \edtext after starred sectioning commands when using memoir class. 207
\v1.15.0.
\General: Fix bug with footnotes layout when using some options of the geometry package (bug add by v1.13.0). 1
\New commands \AtEveryPstart and \AtEveryPend. 7
\New tools to prevent ambiguous references in lemma. 14
\\endsub: Restore subline feature (disabled by mistake in v1.8.0). 71
\\footparagraphX: Correct bug with paragraphed familiar footnotes setting. 129
\\iflemmacommand@: New boolean \iflemmacommand@ 78
\v1.15.1.
\\line@list@stuff: Revert modification of 1.5.2 which makes bug with numbering. Leave vertical mode to solve spurious space before minipage. 69
v1.16.0.
General: Compatibility of standard footnotes with some biblatex styles. ................. 1
New \stanzaindent command. . 1\critext: \critext and \edtext are now defined only in eledmac, not in eledpar. Debug wrong numbering when using \sameword + eledpar + \tag command. .................. 76
v1.16.1.
\lineref: \lineref is not defined if defined by some other package, like lineno. Eledmac provides \edlineref instead. . . 157
v1.17.0.
\critext: The historical \critext now just refers to \edtext (code refactoring). ............... 76\edtext: Error message when calling \edtext outside of a numbered paragraph. ............... 76
v1.18.0.
\@edindex@hyperref: Fix spurious space with \edindex when using \texttt{imakeidx/indextools + hyperref}. .................. 174
General: Add \pstartinfootnoteeverytime ................. 16
General: Add \numberonlyfirstinline
Compatiblity with LuaLaTeX RTL languages. ................. 1
Debug \onlypstartinfootnote when using \numberonlyfirstinline and the current line number differs from the previous. ..... 16
\edlabel: \edlabel is now defined only one time for both eledmac and eledpar ............... 155
\ifledRcol@: Add \ifledRprintingpages and \ifledRprintingcolumns for eledpar .................. 47
\l@Find@section: Option parapparatous works for endnotes. ....... 135
\print@line: Compatibility with LuaLaTeX RTL languages. ....... 87
\printlinefootnote: Code refactoring in \printlinefootnote: the printing of the numbers are factorized in \printlinefootnotearia . . 146
\printpstart: Debug \pstartinfootnote with parallel pages and columns (eledpar) ............... 104