eledmac

A presumptuous attempt to port
EDMAC, TABMAC and EDSTANZA to LaTeX∗

Peter Wilson
Herries Press†
Maïeul 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 eledmac 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 documented 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 eledmac mail list in:

http://geekographie.maieul.net/146

∗This file (eledmac.dtx) has version number v1.19.0, last revised 2015/03/06.
†herries dot press at earthlink dot net
‡maieul at maieul dot net
## Contents

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

2 The (e)ledmac package 10

3 Text lines and paragraphs numbering 10
   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 16
   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 beginning 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
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 Sectioning commands</td>
<td>52</td>
</tr>
<tr>
<td>21 Line counting</td>
<td>56</td>
</tr>
<tr>
<td>21.1 Choosing the system of lineation</td>
<td>56</td>
</tr>
<tr>
<td>21.2 List macros</td>
<td>60</td>
</tr>
<tr>
<td>21.3 Line-number counters and lists</td>
<td>61</td>
</tr>
<tr>
<td>21.4 Reading the line-list file</td>
<td>65</td>
</tr>
<tr>
<td>21.5 Commands within the line-list file</td>
<td>67</td>
</tr>
<tr>
<td>21.6 Writing to the line-list file</td>
<td>73</td>
</tr>
<tr>
<td>22 Marking text for notes</td>
<td>77</td>
</tr>
<tr>
<td>22.1 \edtext (and \critext) itself</td>
<td>78</td>
</tr>
<tr>
<td>22.2 Substitute lemma</td>
<td>83</td>
</tr>
<tr>
<td>22.3 Substitute line numbers</td>
<td>84</td>
</tr>
<tr>
<td>22.4 Lemma disambiguation</td>
<td>84</td>
</tr>
<tr>
<td>23 Paragraph decomposition and reassembly</td>
<td>88</td>
</tr>
<tr>
<td>23.1 Boxes, counters, \pstart and \pend</td>
<td>88</td>
</tr>
<tr>
<td>23.2 Processing one line</td>
<td>92</td>
</tr>
<tr>
<td>23.3 Line and page number computation</td>
<td>94</td>
</tr>
<tr>
<td>23.4 Line number printing</td>
<td>97</td>
</tr>
<tr>
<td>23.5 Pstart number printing in side</td>
<td>101</td>
</tr>
<tr>
<td>23.6 Add insertions to the vertical list</td>
<td>102</td>
</tr>
<tr>
<td>23.7 Penalties</td>
<td>103</td>
</tr>
<tr>
<td>23.8 Printing leftover notes</td>
<td>104</td>
</tr>
<tr>
<td>24 Critical footnotes</td>
<td>105</td>
</tr>
<tr>
<td>24.1 Fonts</td>
<td>105</td>
</tr>
<tr>
<td>24.2 Outer-level footnote commands</td>
<td>105</td>
</tr>
<tr>
<td>24.3 Normal footnote formatting</td>
<td>107</td>
</tr>
<tr>
<td>24.4 Standard footnote definitions</td>
<td>113</td>
</tr>
<tr>
<td>24.5 Paragraphed footnotes</td>
<td>115</td>
</tr>
<tr>
<td>24.6 Insertion of the footnotes separator</td>
<td>121</td>
</tr>
<tr>
<td>24.7 Columnar footnotes</td>
<td>121</td>
</tr>
<tr>
<td>25 Familiar footnotes</td>
<td>126</td>
</tr>
<tr>
<td>25.1 Generality</td>
<td>126</td>
</tr>
<tr>
<td>25.2 Footnote formats</td>
<td>128</td>
</tr>
<tr>
<td>25.3 Two columns footnotes</td>
<td>132</td>
</tr>
<tr>
<td>25.4 Three columns footnotes</td>
<td>133</td>
</tr>
<tr>
<td>25.5 Paragraphed footnotes</td>
<td>135</td>
</tr>
<tr>
<td>26 Footnotes’ width for two columns</td>
<td>138</td>
</tr>
<tr>
<td>27 Footnotes’ order</td>
<td>139</td>
</tr>
<tr>
<td>28 Footnotes’ rule</td>
<td>139</td>
</tr>
</tbody>
</table>
1 Introduction

The EDMAC macros [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 [Bre96], by permission of its author, Herbert Breger. The verse-related code is by courtesy of Wayne Sullivan, the author of EDSTANZA [Sul92], who has kindly supplied more than his original macros.

Since 2011’s Mâieul 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.
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 elledmac 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](http://www.homepages.ucl.ac.uk/~ucgadkw/edmac/index.html).

Apart from elledmac 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 [Lüc03], by Uwe Lück and Christian Tapp, is another \LaTeX{} package being developed for critical editions. Unlike elledmac 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](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 elledmac will work together.

Critical authors may find it useful to look at EDMAC, EDNOTES, elledmac, 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 elledmac, and other programs.

- Jerónimo Leal pointed me to [http://www.guit.sssup.it/latex/critical.html](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](http://www.maurolico.unipi.it/mtex/mtex.htm)). These sites are both in Italian.

- Dirk-Jan Dekker maintains [http://www.djdekker.net/ledmac](http://www.djdekker.net/ledmac) which is a FAQ for typesetting critical editions and elledmac.

This manual contains a general description of how to use the \LaTeX{} version of EDMAC, namely elledmac (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

documentation in sections 2, unless you are particularly interested in the innards of ededmac.

1.2 History

1.2.1 EDMAC

The original version of EDMAC was TEXED.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-

\footnote{This version of the macros was used to format the Sanskrit text in volume I of Metarules of Pāṇini Grammar by Dominik Wujastyk (Groningen: Forsten, 1993).}

\footnote{Gerhard Brey used EDMAC in the production of Hubert L. L. Busard and Menso Folkerts, Robert of Chester’s (?) Redaction of Euclid’s Elements, the so-called Adelard II Version, 2 vols., (Basel, Boston, Berlin: Birkhäuser, 1992).}
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 Shujā’ 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 Compilati in Studio Generali Quinqueclesiensi in Regno Ungarie*, the collected letters and papers of Leibniz, Theodosius’s *Sphercos*, the German *Algorismus* of Sacrobosco, the Sanskrit text of the *Kāśīkāvṛtti* 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.225).

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/critical_editions_typeset_with_edmac_ledmac_and_eledmac/](https://www.zotero.org/groups/critical_editions_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.
3 Text lines and paragraphs numbering

3.1 Text lines numbering

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

\begin{numbering}
(text)
\end{numbering}

The \begin{numbering} 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 \begin{numbering} also opens a file called ⟨jobname⟩.end to receive the text of the endnotes. \end{numbering} 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 \begin{numbering} and \end{numbering} 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 \begin{numbering} 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

\pstart \pend  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

The lines of this paragraph are not numbered.

\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 numbered.
6 The lines of this paragraph are not numbered.
7 And here the numbering 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 Producing automatically \pstart...\pend

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, \noindent, or \leavevmode, or using \pstart itself.\footnote{13}

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, so 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-
3.3 Lineation commands

\resumenumbering
bers and footnotes in memory as it goes through a numbered section. But at the end 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 \LTÉX may reach its memory limit. There are two solutions to this. The first is to get a larger \LTÉX 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.
\pend
\pausenumbering
\pstart
Another paragraph.
\pend
\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
Line numbering can be disabled with \numberlinefalse. It can be enabled again with \numberlinetrue.

3.3.2 Setting lineation start and step

\firstlinenum
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} 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 permissable 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.
\rightlinenum You can redefine \leftlinenum and \rightlinenum to change the way marginal
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.

\starts
You insert the \starts and \ends 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 numberered 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.

\setlock
The \setlock 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.

\setdisp
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 \setdisp{⟨arg⟩}; its argument is a word, either first, last, or all. The package initially sets this as \setdisp{first}.

\advanceline
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 \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. \advanceline 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.

\setlinenum
The \setlinenum 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.

\linenumberstyle
Line numbers are normally printed as arabic numbers. You can use \linenumberstyle{⟨style⟩} 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.
4  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{\textlangle commands\textrangle}, may contain a series of subsidiary commands that generate various kinds of notes.

- \texttt{\textbackslash Afootnote} (commands)\texttt{\textrangle} in a layer just below the main text, followed by the next in turn, down to the 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{\textbackslash Afootnote[nonum]{(text)}}.

The package also maintains five separate series of endnotes. Like footnotes each macro takes a single argument like \texttt{\textbackslash 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{\textbackslash usepackage[parapparatus]{eledmac}}

If you want to change the lemma that gets passed to the notes, you can do this by using \texttt{\textbackslash 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{\textbackslash edtext(I saw my friend}\n\texttt{\textbackslash edtext{Smith}\{Afootnote{Jones C, D.}} on Tuesday.}\n\texttt{\{}\texttt{\textbackslash lemma{\textbackslash \textbackslash dota\ Tuesday.}}\n\texttt{\textbackslash Bfootnote{The date was}\n\texttt{\textbackslash Afootnote{Jones C, D.}} on Tuesday.}\n\texttt{\{}\texttt{\textbackslash lemma{\textbackslash \textbackslash dota\ Tuesday.}}\n\texttt{\texttt{\textbackslash Bfootnote{The date was}\n\texttt{July 16, 1954.}}\n\texttt{\}}%}

You can use \texttt{\textbackslash linenum\{\textlangle arg\textrangle\}} 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 simply 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 ⟨lemma⟩ 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:

```
\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 times) in the same line. `eledmac` provides tools to disambiguate references in the critical notes. The lemma will be followed by a reference number if a given word occurs more than once in the same line.

To use this tool, you have to mark every occurrence of the potentially ambiguous

\sameword

\footnote{We use `\newcommand` and `\newcommandx` instead of classical `\let` command because the `edtable`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`.}
4.3 Alternate footnote formatting

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}}

\showwordrank 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.

\footparagraph \foottwocol \footthreecol

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 plus b minus c. The final length ⟨m⟩ is calculated by TeX to have: a − c ≤ ⟨m⟩ ≤ a + b. If you use relative unity\textsuperscript{15}, it will be relative to fontsize of the footnote.

#### 4.4.1 Control line number printing

\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 \numberonlyfirstinline[⟨s⟩]. Use \numberonlyfirstinline[⟨s⟩][false] to cancel it (⟨s⟩ can be empty if you want to disable it for every series).

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

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

\nonumberinfootnote You can use \nonumberinfootnote[⟨s⟩] if you don’t want to have the line number in a footnote. To cancel it, use \nonumberinfootnote[⟨s⟩][false].

\pstartinfootnote You can use \pstartinfootnote[⟨s⟩] if you want to print the pstart number in the footnote, before the line and subline number. Use \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:

- If you use lineation by pstart, the option is enabled.
- If you use lineation by section or by page, the option is disabled.

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

\textsuperscript{15}Like \em which is the width of a M.
4.4 Display options

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.

onlypstartinfootnote

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).

beforenumberinfootnote

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.

afternumberinfootnote

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.

\nonbreakableafternumber

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).

beforeymulinenum

With \beforeymulinenum[(s)]{(l)} you can add some space before the line
symbol in a footnote. The default value is value set by \beforenumberinfootnote.

afterymulinenum

With \afterymulinenum[(s)]{(l)} you can add some space after the line sym-
bol in a footnote. The default value is value set by \afternumberinfootnote.

inplaceofnumber

If no number or symbolic line number is printed, you can add a space, with
\inplaceofnumber[(s)]{(l)}. The default value is 1 em.

\boxlinenum

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
an other column:

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

\boxymulinenum

\boxymulinenum[(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
\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 cm.
Using \afterlemmaseparator\{(s)\}\{\ell\} 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.
\nYou can suppress the lemma separator, using \nolemmaseparator\{(s)\}, which is simply a alias of \lemmaseparator\{(s)\}\{\ell\}.
\nWith \inplaceoflemmaseparator\{(s)\}\{\ell\} you can add a space if no lemma separator is printed. The default value is 1 em.

4.4.3 Font style
\nXnotenumfont\{(s)\}\{\langle command\rangle\} is used to change the font style for line numbers in critical footnotes; \langle command\rangle must be one (or more) switching command, like \bfseries.
\nXendnotenumfont\{(s)\}\{\langle command\rangle\} is used to change the font style for line numbers in critical footnotes. \langle command\rangle must be one (or more) switching command, like \bfseries.
\notenumfont\{(s)\}\{\langle command\rangle\} is used to change the font style for note numbers in familiar footnotes. \langle command\rangle must be one (or more) switching command, like \bfseries.
\notefontsize\{(s)\}\{\langle command\rangle\} is used to define the font size of critical footnotes of the series. The default value is \footnotesize. The \langle command\rangle must not be a size in pt, but a standard \LaTeX size, like \small.
\notefontsizeX\{(s)\}\{\langle command\rangle\} is used to define the font size of critical footnotes of the series. The default value is \footnotesize. The \langle command\rangle must not be a size in pt, but a standard \LaTeX size, like \small.
\Xendnotefontsize\{(s)\}\{\langle command\rangle\} is used to define the font size of end critical footnotes of the series. The default value is \footnotesize. The \langle command\rangle must not be a size in pt, but a standard \LaTeX size, like \small.

4.4.4 Font of the lemma
\lemmadesablefontselection 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 \lemmadesablefontselection\{(s)\} command allows to disable it for a specific series.
\endlemmadesablefontselection 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 \endlemmadesablefontselection\{(s)\} 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 \xhangindent\{(s)\}\{\ell\}, 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 \xhangindent\{(s)\}\{\ell\},
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 a note.

### 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:

\bhookXnote\{\renewcommand\{\thepstart\}\{\arabic{pstart}.\}\}

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

\bhookXendnote\{\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.

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

\hsizethreecol\{\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

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

\parafootsep\{\s\}\{\l\}\ 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\{\$\}\{\$\}.

\Xragged\{\s\}\{\l\}\ Text in paragraphed critical notes is justified, but you can use \Xragged\{\s\}\{\l\} if you want it to be ragged left, or \Xragged\{\s\}\{\r\} if you want it to be ragged right.

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

\textbeforeXnotes You can add some text before critical notes with \textbeforeXnotes\{⟨text⟩\}. 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 \LaTeXX 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 \LaTeXX 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 \LaTeXX footnote rule, which is used by eledmac, adds 2.6pt. These 2.6pt 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 \LaTeXX 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 0 pt.

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 the current font. For example, if you want the note to take, at most, 33 of the text height, do \maxhXnotes\{.33\text{\textwidth}\}.

\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.\(^{16}\)

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.

\(^{16}\)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 \footnote{118} explains why this restriction is necessary.
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 polyglossia, 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.

\stanza

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.

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

\stanzaindentbase Lines within a stanza may be indented. The indents are integer multiples of the length \stanzaindentbase, whose default value is 20pt.

\setstanzaindents In order to use the stanza macros, one must set the indentation values. First the value of \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 \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 \hanginsymbol: see p. 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 indents, and say they are repeated, defining the value of the \stanzaindentstanzainsertionsrepetition counter at \( n \). For example:

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

is like

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

Be careful: the feature change in eledmac 1.5.1. See Appendix A.3 p. 226.

If you don’t use the \stanzaindentstanzainsertionsrepetition counter, make sure you have at least one more numerical entry in \setstanzaindents than the number of lines in the stanza.

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

\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 You can set the indent of some specific verse by calling \stanzaindent{(value)} 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)\} 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

\setstanzapenalties 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 \endstanzaeextra 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

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

\renewcommand{\hangingsymbol}{\{\,}

You can also use it to force hanging verse to be flush right:
5.5 Long verse and page break

If you want to prevent page breaks inside long verses, use the option \nopbinverse when loading package, or use \lednopbinversetrue. Read [16 p. 40] for further details.

5.6 Various tools

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

\endstanzaextra The macro \endstanzaextra, 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 memoir class, it provides a length \stanzaskip which may come in handy.

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

\flagstanza Putting \flagstanza[\text] at the start of a line in a stanza (or elsewhere) will typeset \text before the line. The default \text is \stanzaindentbase.
For example, to put a verse number before the first line of a stanza you could proceed along the lines:
\newcounter{stanzanum}
\setcounter{stanzanum}{0}
\newcommand*{\startstanzahook}{\refstepcounter{stanzanum}}
\newcommand{\numberit}{\flagstanza{\thestanzanum}}
...
\stanza
\numberit First line...&
rest of stanza\&

\stanza
\numberit First line, second stanza...

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 eledmac, redefine macro \hangingsymbol with this code:
\renewcommand{\hangingsymbol}{[\textbackslash,\textbackslash]}

\renewcommand{\hangingsymbol}{\protect\hfill}
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 11) 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 \ledgroupsized environment is similar to \ledgroup except that you must specify a width for the environment, as with a minipage.

\begin{ledgroupsized}{\textwidth}

The required \width 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 \eledmac package does not provide crop marks. These are available with either the \memoir class [Wil02] or the \crop package.
8 Endnotes

\doendnotes{⟨letter⟩} closes the .end file that contains the text of the endnotes, if it’s open, and prints one series of endnotes, as specified by a series-letter argument, e.g., \doendnotes{A}. \endprint is the macro that’s called to print each note. It uses \select@lemmfont to select fonts, just as the footnote macros do (see p.105 above). 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.

9 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.

\edlabel{⟨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’.

\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.}

Elsewhere in the text, either before or after the \edlabel, you can refer to its location via \edpageref{⟨lab⟩}, or \edlineref{⟨lab⟩}, or \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 \TeX.\aux file. You will need to process your document through \TeX 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 \texttt{\ldots} command, the \texttt{\label} should be defined inside the first argument, e.g.,

\begin{verbatim}
The \texttt{\textit{creature}\label{elephant} was quite unafraid}\footnote{Of the mouse, that is.}
\end{verbatim}

However, there are situations in which you’ll want \texttt{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 \LaTeX{} 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 \texttt{\linenum}, for example. For this situation, three variants of the reference commands, with the \texttt{x} prefix, are supplied: \texttt{xpageref}, \texttt{mlineref}, \texttt{xsublineref} and \texttt{xpstartref}. They have these limitations:

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

The macros \texttt{\xref} and \texttt{\xmakelabel} let you manipulate numbers and labels in ways which you may find helpful in tricky situations.

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

Sometimes the \texttt{\ldlabel} 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 \texttt{\xmakelabel{(lab)}{(numbers)}} macro so that you can “roll your own” label. For example, if you say \texttt{\xmakelabel{elephant}{(10|25|0)}} you will create a new label, and a later call to \texttt{\xpageref{elephant}} would print ‘10’ and \texttt{\mlineref{elephant}} would print ‘25’. The sub-line number here is zero. It is usually best to collect your \texttt{\xmakelabel} statements near the top of your document, so that you can see them at a glance.

The normal \texttt{\label}, \texttt{\ref} and \texttt{\pageref} macros may be used within numbered text, and operate in the familiar fashion.

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

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

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

\ledleftnote \\ledsidenote\{\text\} will put \text into the margin specified by the current setting of \text. The permissible value for \text is one out of the list left, right, inner, or outer, for example \sidenotemargin{left}. The package’s default setting is \sidenotemargin{right}.

If two, say, \ledleftnote, commands are called in the same line the second \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 \ledlsnotewidth and the right text into a box of width \ledrsnotewidth. These are initially set to the value of \linenumsep.

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

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

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 \sidenotesep.

11 Familiar footnotes

The footmisc package \footmisc by Robin Fairbairns has an option whereby sequential footnote marks in the text can be separated by commas, like so. As a convenience eledmac provides this automatically.

\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 \footnote{} macros takes one argument which is the series letter (e.g., B). \footnormal{} is the typical footnote format. With \footparagraph{} the series is typeset a one paragraph, with \foottwocol{} the notes are in two columns, and are in three columns with \foothreecol{}.

As well as using the \footnote{} 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 \thefootnote{} macro; the default is:

\newcommand*{\thefootnote}{\arabic{footnote}}

The appearance of the mark in the text is controlled by \bodyfootmark{} which is defined as:

\newcommand*{\bodyfootmark}{\hbox{\textsuperscript{\@nameuse{\@thefnmark}}}}

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

\newcommand*{\footfootmark}{\textsuperscript{\@nameuse{\@thefnmark}}}.

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 \ref{newseries} p.26).

### 11.1 Position of the familiar footnotes

\fnpos{} There is a historical incoherence in (e)\LaTeX. 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 \textit{item} and
the current page number should be added to the raw index (idx) file. The \texttt{\textbackslash edindex\{item\}} macro can be used in numbered text to specify that \texttt{\langle item\rangle} 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 \texttt{\edindex[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 \texttt{\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 \texttt{-} is the default separator used by the \texttt{MAKEINDEX} program).

\texttt{\edindexlab} The \texttt{\edindex} process uses a \texttt{\label/\ref} mechanism to get the correct line number. It automatically generates labels of the form \texttt{\label\{\edindexlab N\}}, where \texttt{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\{\$&\text{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.

\texttt{edarray*} environments are for math and \texttt{edarray} for text entries. The final \texttt{l}, \texttt{c}, or \texttt{r} in the environment names indicate that the entries will be flushleft (\texttt{l}), centered (\texttt{c}) or flushright (\texttt{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.
\begin{edtabularc}
1 & 2 & 3 \\
a & bb & ccc \\
AAA & BB & C
\end{edtabularc}

Entries in the environments are the same as for the normal array and tabular environments but there must be no ending `\\' at the end of the last row. There must be the same number of column designators (the &) in each row. There is no equivalent to any line drawing commands (such as `\hline'). However, unlike the normal environments, the ed... environments can cross page breaks.

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

For example:

\begin{numbering}
\pstart
\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 \edtext{round}{\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} & & \\
& But it keeps them on the knife.
\end{edtabularr}
\pend
\end{numbering}

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.

The distance between the columns is controlled by the length \edtabcolsep.

\begin{edarrayl}
1 & 2 & 3 & 4 \\
& \spreadmath{F+G+C} & & \\
a & bb & ccc & dddd
\end{edarrayl}

The macro \edrowfill{\langle start\rangle}{\langle end\rangle}{\langle fill\rangle} fills columns number \langle start\rangle to \langle end\rangle inclusive with \langle fill\rangle. The \langle fill\rangle argument can be any horizontal ‘fill’. For example \hrulefill or \upbracefill.

\begin{edtext}
\begin{array}{l}
\sp\{m\} \quad \sp\{t\}
\end{array}
\end{edtext}
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 & & & & & & ghweropjklnmbvcxys \\
1 & 2 & 3 & & \texttt{\textbackslash edrowfill\{4\}{}\{5\}{}\{\textbackslash hrulefill\} & \\
\end{tabularr}
\end{verbatim}

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

\texttt{\textbackslash newcommand\{\textbackslash upbracketfill\}{{%}
\texttt{\textbackslash vrule height 4pt depth 0pt\textbackslash hrulefill\textbackslash vrule height 4pt depth 0pt}}}

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 & \texttt{\textbackslash upbracketfill} & & \\
k & 1 & co & ghweropjklnmbvcxys \\
1 & 2 & 3 & \texttt{\textbackslash hrulefill} & \\
\end{edarrayc}
\end{verbatim}

\texttt{\textbackslash edatleft} \texttt{\{\textbackslash math\}\{\textbackslash symbol\}\{\textbackslash halfheight\}} typesets the math \texttt{\{symbol\}} as \texttt{\{math\}} with the optional \texttt{\{math\}} centered before it. The \texttt{\{symbol\}} is twice \texttt{\{halfheight\}} tall. The \texttt{\textbackslash edatright} macro is similar and it typesets \texttt{\{symbol\}} with \texttt{\{math\}} centered after it.
\begin{edarrayc}
& 1 & 2 & 3 & \\
& 4 & 5 & 6 & \\
& 7 & 8 & 9 & \\
\end{edarrayc}

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

\edbeforetab \edbeforetab{(text)}{(entry)}, where \langle entry \rangle is an entry in the leftmost column, typesets \langle text \rangle left justified before the \langle entry \rangle. Similarly \edaftertab{(entry)}{(text)}, where \langle entry \rangle is an entry in the rightmost column, typesets \langle text \rangle right justified after the \langle entry \rangle.

For example:
\begin{edarrayl}
A & 1 & 2 & 3 \\
\edbeforetab{Before}{B} & 1 & 3 & 6 \\
C & 1 & 4 & \edaftertab{After}{8} \\
D & 1 & 5 & 0
\end{edarrayl}

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

\edvertdot The macro \edvertdot{\langle height \rangle} draws a vertical line \langle height \rangle high (contrast this with \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 & L & cvb & \edvertdot{4pc}
\end{edarrayr}

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

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

The standard sectioning command ($\texttt{\chapter}, \texttt{\section}$ 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:

- $\texttt{\ledchapter}\{\texttt{text}\}\{\texttt{critical\ text}\}$
- $\texttt{\ledchapter}$
- $\texttt{\ledsection}\{\texttt{text}\}\{\texttt{critical\ text}\}$
- $\texttt{\ledsection*}$
- $\texttt{\ledsubsection}\{\texttt{text}\}\{\texttt{critical\ text}\}$
- $\texttt{\ledsubsection*}$
- $\texttt{\ledsubsubsection}\{\texttt{text}\}\{\texttt{critical\ text}\}$
- $\texttt{\ledsubsubsection*}$

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:

- $\texttt{\eledchapter}\{\texttt{text}\}\{\texttt{critical\ text}\}$
- $\texttt{\eledchapter*}$
- $\texttt{\eledsection}\{\texttt{text}\}\{\texttt{critical\ text}\}$
- $\texttt{\eledsection*}$
- $\texttt{\eledsubsection}\{\texttt{text}\}\{\texttt{critical\ text}\}$
- $\texttt{\eledsubsection*}$
- $\texttt{\eledsubsubsection}\{\texttt{text}\}\{\texttt{critical\ text}\}$
- $\texttt{\eledsubsubsection*}$

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

\pstart
\eledsection*{xxxx\ledsidenote{section}}
\pend
\pstart
\eledsubsection*{xxxx\ledsidenote{sub}}
\pend

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 \eledxxx 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 eledpar 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 eledpar 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

$\texttt{\extensionchars}$  When the package assembles the name of the auxiliary file for a section, it prefixes $\texttt{\extensionchars}$ 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 $\texttt{\renewcommand{\extensionchars}{!}}$, then you would get temporary files called $\texttt{jobname.1}$, $\texttt{jobname.2}$, etc.

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

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

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

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

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

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:

$\texttt{\ifledfinal\else}$

$\texttt{\renewcommand{\showlemma}[1]{\textbf{#1}}}% or simply ...[1]{#1}$

$\texttt{\fi}$

17.1 Known and suspected limitations

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

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

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

$\texttt{\ballast}$  $\LaTeX$ 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 $\LaTeX$ never settle down. At each successive run, $\texttt{eledmac}$ may oscillate between two
different sets of page decisions. To stop this happening, should it arise, Wayne Sul-
vivan suggested the inclusion of the quantity \texttt{ballast}. The amount of \texttt{ballast}
will be subtracted from the penalties which apply to the page breaks calculated
on the \textit{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 \textsuperscript{16} p. 25 and described in more detail on p. 118 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.

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
68) to increase the estimate. You have to use \texttt{renewcommand} for this, like:
\texttt{renewcommand{\texttt{footfudgefiddle}}{68}}

Help, suggestions and corrections will be gratefully received.

\section{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{edtext} and
\*footnote macros (this is discussed in more detail in section \texttt{22} and in particular
the discussion about \texttt{no@expands} and \texttt{morenoexpands}). 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 *bottom* of the *footmisc* package, you must load this package *before* the *eledmac* package.

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

\[ \cdots \edtext{\colorbox{mycolor}{lemma}}\{Afootnote{\cdots\colorbox{\cdots}}\} \]

If you actually try this\(^{20}\) you will find \LaTeX whinging '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}

(\@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 \textcolor instead, like

\[ \cdots \edtext{\textcolor{mycolor}{lemma}}\{Afootnote{\cdots\textcolor{\cdots}}\} \]

there is no need to fiddle with \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 *eledmac* specifically for parallel typesetting of critical texts. This also cooperates with the *babel*\footnote{Reported by Dirk-Jan Dekker in the CTT thread ‘Incompatibility of “color” package’ on 2003/08/28.}
/ polyglossia packages for typesetting in multiple languages. The package has been called eldpar since September 2012.

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

We present the `eledmac` code in roughly the order in which it’s used during a run of `TeX`. The order is *exactly* that in which it’s read when you load The `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 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 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 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 23). The footnote commands (Section 24) and output routine (Section 32) finish the main part of the processing; cross-referencing (Section 33) and endnotes (Section 30) complete the story.

In what follows, macros with an @ in their name are more internal to the workings of `eledmac` than those made up just of ordinary letters, just as in Plain `TeX` (see *The TeXbook*, 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 `l@d` in macro names to help avoid name clashes, but this is not a hard and fast rule. For example, if an original `EDMAC` macro includes `edmac` We will simply change that to `eledmac`.

Announce the name and version of the package, which is targetted for `LaTeX2e`.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{eledmac}[2015/03/06 v1.19.0 LaTeX port of EDMAC]

Generally, these are the modifications to the original. `EDMAC` code:

• Replace as many \texttt{\textbackslash def}'s by \texttt{\textbackslash newcommand}'s as possible to avoid overwriting `\LaTeX` macros.

• Replace user-level `\LaTeX` counts by `\LaTeX` counters.

• Use the `\LaTeX` font handling mechanisms.

• Use `\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.

\ifledfinal
\ifparapparatus@
\ifndefquotation@
\ifndeflednopbinverse
\ifndefparledgroup
\ifndefwidthliketwocolumns
\ifndefledsecnolinenumber
\newif\ifledfinal
\newif\ifparapparatus@
\newif\ifndefquotation@
\newif\ifndeflednopbinverse
\newif\ifndefparledgroup
\newif\ifndefwidthliketwocolumns
\newif\ifndefledsecnolinenumber
\DeclareOption{noquotation}{\ifndefquotation@true}
\DeclareOption{final}{\ifledfinaltrue}
\DeclareOption{draft}{\ifledfinalfalse}
\DeclareOption{parapparatus}{\parapparatus@true}
\DeclareOption{nopbinverse}{\ifndeflednopbinversetrue}
\DeclareOption{ledsecnolinenumber}{\ifndefledsecnolinenumbertrue}
\DeclareOption{widthliketwocolumns}{\ifndefwidthliketwocolumnstrue}\
\ExecuteOptions{final}

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.

\ProcessOptions*\relax

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.

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

\if@RTL
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.

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

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

\ifledfinal
19.2 Loading packages

Loading package \texttt{xargs} to declare commands with optional arguments. \textit{Etoolbox} is also used to make code clearer - for example, in dynamic command names (which can replace \texttt{\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.

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

19.3 Boolean flags

\texttt{\ifl@dmemoir} Define a flag for if the \texttt{memoir} class has been used.
\begin{verbatim}
\newif\ifl@dmemoir
\@ifclassloaded{memoir}{\l@dmemoirtrue}{\l@dmemoirfalse}
\end{verbatim}

\texttt{\ifl@imakeidx} Define a flag for if the \texttt{imakeidx} package has been used.
\begin{verbatim}
\newif\ifl@imakeidx
\@ifpackageloaded{imakeidx}{\l@imakeidxtrue}{\l@imakeidxfalse}
\end{verbatim}

\texttt{\ifl@indextools} Define a flag for if the \texttt{indextools} package has been used.
\begin{verbatim}
\newif\ifl@indextools
\@ifpackageloaded{indextools}{\l@indextoolstrue}{\l@indextoolsfalse}
\end{verbatim}

\texttt{\if@RTL} The \texttt{\if@RTL} is defined by the \texttt{bidi} package, which is sometimes loaded by \texttt{polyglossia}. But we define it as well if the \texttt{bidi} package is not loaded.
\begin{verbatim}
\ifdef{\if@RTL}{}{\newif\if@RTL}
\end{verbatim}

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.
Write a warning message.
55 \newcommand{\eledmac@warning}[1]{\PackageWarning{eledmac}{#1}}

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

\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{edtext outside numbered paragraph (\pstart...\pend)}{\@ehc}}

\led@mess@NotesChanged
65 \newcommand*{\led@mess@NotesChanged}{\typeout{eledmac reminder: The number of the footnotes in this section has changed since the last run.}}
66 \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}}

\newcommand*{\led@err@PstartInPstart}{%}
\eledmac@error{string\pstart\space encountered while another \string\pstart\space was in effect}}

\newcommand*{\led@err@PendNotNumbered}{%}
\eledmac@error{string\pend\space must be used within a numbered section}}

\newcommand*{\led@err@PendNoPstart}{%}
\eledmac@error{string\pend\space must follow a \string\pstart}}

\newcommand*{\led@err@AutoparNotNumbered}{%}
\eledmac@error{string\autopar\space must be used within a numbered section}}

\newcommand*{\led@warn@BadAction}{%}
\eledmac@warning{Bad action code, value \next@action.}}

\newcommand*{\led@warn@DuplicateLabel}{%}
\eledmac@warning{Duplicate definition of label '1' on page \the\pageno.}}

\newcommand*{\led@warn@RefUndefined}{%}
\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}}
\texttt{\@warn@NoIndexFile}

\newcommand*{\@warn@NoIndexFile}[1]{
\eledmac@warning{Undefined index file #1}}

\texttt{\@warn@Addfootins\texttt{X}\texttt{Obsolete}}

\newcommand{\@warn@Addfootins\texttt{X}\texttt{Obsolete}}{%
\eledmac@warning{Addfootins\texttt{X} is obsolete in eledmac 1.0. Use newseries instead.}}%

\texttt{\@warn@Addfootins\texttt{Obsolete}}

\newcommand{\@warn@Addfootins\texttt{Obsolete}}{%
\eledmac@warning{Addfootins is obsolete in eledmac 1.0. Use newseries instead.}}%

\texttt{\@warn@SeriesStillExist}

\newcommand{\@warn@SeriesStillExist}[1]{
\eledmac@warning{Series #1 is still existing !}}%

\texttt{\@err@ManySidenotes}

\newcommand{\@err@ManySidenotes}{%
\ifledRcol%
\eledmac@warning{itemcount@\space sidenotes on line \the\line@numR\space p. \the\page@numR}\\
\else%
\eledmac@warning{itemcount@\space sidenotes on line \the\line@num\space p. \the\page@num}\\
\fi%
}%

\newcommand{\@err@ManyLeftnotes}{%
\ifledRcol%
\eledmac@warning{itemcount@\space leftnotes on line \the\line@numR\space p. \the\page@numR}\\
\else%
\eledmac@warning{itemcount@\space leftnotes on line \the\line@num\space p. \the\page@num}\\
\fi%
}%

\newcommand{\@err@ManyRightnotes}{%
\ifledRcol%
\eledmac@warning{itemcount@\space rightnotes on line \the\line@numR\space p. \the\page@numR}\\
\else%
\eledmac@warning{itemcount@\space rightnotes on line \the\line@num\space p. \the\page@num}\\
\fi%
}%

\texttt{\@warn@Falseverse\texttt{Deprecated}}

\newcommand{\@warn@Falseverse\texttt{Deprecated}}{%
\eledmac@warning{falseverse\space deprecated. Look at newverse\space instead.}}%

\newcommand{\@warn@led\texttt{\#1}\texttt{Deprecated}}{%
\eledmac@warning{\texttt{\string\\le#1} \space deprecated. Look at \texttt{\string\\e#1} instead.}}%

\texttt{\@err@TooManyColumns}

\texttt{\@err@UnequalColumns}

\texttt{\@err@LowStartColumn}

\texttt{\@err@HighEndColumn}

\texttt{\@err@ReverseColumns}
19.5 Gobbling

Here, we define some commands which gobble their arguments.

19.6 Miscellaneous commands

The code for the \texttt{linenumberlist} mechanism was given to Peter Wilson by Wayne Sullivan on 2004/02/11. Initialize it as \texttt{empty}

\begin{verbatim}
\let\linenumberlist=\empty
\end{verbatim}

In imitation of \LaTeX, we create a couple of scratch counters.

% \texttt{dtempcnta} and \texttt{dtempcntb} but Peter Wilson found in the past that it can be dangerous to use these (for example one of the AMS packages did something nasty to the \texttt{ccaption} package's use of one of these).

\begin{verbatim}
\newcount\dtempcnta \newcount\dtempcntb
\end{verbatim}
20 Sectioning commands

\section@num You use $\begin{numbering}$ and $\end{numbering}$ 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 $\section@num$ that counts how many $\begin{numbering}$ and $\resumenumbering$ 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 $⟨\text{jobname}.nn⟩$, where $nn$ is the section number. However, you may direct that an extra string be added before the $nn$ in that filename, in order to distinguish these temporary files from others: that string is called $\extensionchars$. Initially it’s empty, since different operating systems have greatly varying ideas about what characters are permitted in file names. So $\renewcommand{\extensionchars}{-}$ gives temporary files called $\text{jobname}$.–1, $\text{jobname}$.–2, etc.

\begin{align}
\newcount\section@num \\
\section@num=0 \\
\let\extensionchars=\empty
\end{align}

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

\begin{align}
\newif\ifnumbering \\
\numberingtrue \\
\numberingfalse
\end{align}

In preparation for the \eledpar{} package, these are related to the ‘left’ text of parallel texts (when $\ifl@dpairing$ is TRUE). They are explained in the \eledpar{} manual.

\begin{align}
\newif\ifl@dpairing \\
\l@dpairingfalse \\
\newif\ifl@dpaging \\
\l@dprintingpagesfalse \\
\l@dprintingcolumnstrue \\
\l@dprintingcolumnsfalse \\
\l@dpairingtrue \\
\l@dpairingfalse \\
\newif\ifps@rtedL \\
\pst@rtedLtrue \\
\pst@rtedLfalse \\
\newcount\l@dnumstartsL \\
\ifledRcol \\
\ifledRcol@
\end{align}

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

\begin{align}
\newif\ifledRcol \\
\newif\ifledRcol@
\end{align}
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.

\beginnumbering
203 \newcommand*{\beginnumbering}{%
204 \ifnumbering
205 \led@err@NumberingStarted
206 \endnumbering
207 \fi
208 \global\numberingtrue
209 \global\advance\section@num \@ne
210 \initnumbering@reg
211 \message{Section \the\section@num }%
212 \line@list@stuff{\jobname.\extensionchars\the\section@num}%
213 \l@dend@stuff
214 \setcounter{pstart}{1}
215 \ifl@dpairing
216 \global\l@dnumpstartsL \z@
217 \global\pst@rtedLfalse
218 \else
219 \begingroup
220 \begingroup
221 \initnumbering@sectcmd
222 \ifwidthliketwocolumns%
\csuse{setwidthliketwocolumns@\columns@position}\
\csuse{setpositionliketwocolumns@\columns@position}\
\fi\
\fi\
\gdef\eled@sections@@{}\
\if@noeled@sec\else\
\makeatletter\InputIfFileExists{\jobname.eledsec\the\section@num}{}{}\makeatother\
\immediate\openout\eled@sectioning@out=\jobname.eledsec\the\section@num\relax\
\fi\
\endnumbering
\newcommand*{\initnumbering@reg}{%
\global\pst@rtedLfalse
\global\l@dnumpstartsL \z@
\global\absline@num \z@
\gdef\normal@page@break{}
\gdef\l@prev@opb{}
\gdef\l@prev@nopb{}
\global\line@num \z@
\global\subline@num \z@
\global\@lock \z@
\global\sub@lock \z@
\global\sublines@false
\global\let\next@page@num=\relax
\global\let\sub@change=\relax
\resetprevline@
\resetprevpage@num
}\
\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.
\def\endnumbering{%
\ifnumbering
\global\numberingfalse
\normal@pars
\ifl@dpairing
\global\pst@rtedLfalse
\else
\ifx\insertlines@list\empty\else
\global\noteschanged@true
\fi
\fi\line@list\empty\else
\global\noteschanged@true
\fi
\else
\fi
\fi\noteschanged@
\led@mess@NotesChanged
\fi
\else
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.}

Our thanks to Wayne Sullivan, who suggested the idea behind these macros.
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. edmacro 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.

\ifbypager@ \bypager@true \bypager@false \ifbypstartr@ \bypstartr@true \bypstartr@false \ifbypagem@ \bypagem@true \bypagem@false \lineation \lineation{⟨word⟩} is the macro you use to select the lineation system. Its argument is a string: either page or section or pstart.
21.1 Choosing the system of lineation

You call \texttt{\linenummargin\{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 \let{\warn@BadLinenummargin}\@l@dtempcntb \m@ne \fi \fi \fi \fi \fi \fi \fi
\c@firstlinenum \c@linenumincrement
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.
\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 \lockdisp{⟨word⟩} macro specifies whether a line number—if 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 first, last, or all. Initially, it is set to 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.

```latex
\newcount\sublock@disp
\newcommand{\sublockdisp}[1]{%
  \l@dgetlock@disp{#1}
  \ifnum\@l@dtempcntb>\m@ne
    \global\sublock@disp=\@l@dtempcntb
  \else
    \@warn@BadSublockdisp
  \fi}
\linenumberstyle
\linenumrep
\linenumr@p
\sublinenumberstyle
\sublinenumrep
\sublinenumr@p

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

```

\linenumberstyle
\linenumrep
\linenumr@p
\sublinenumberstyle
\sublinenumrep
\sublinenumr@p

```

Initialise the number styles to arabic.

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

```

\leftlinenum \rightlinenum are the macros that are called to print marginal line numbers on a page, for left- and right-hand margins respectively.

\linenumsep
\numlabfont
\ledlinenum

```

\linenumsep
\numlabfont
\ledlinenum

```

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 TeXbook, 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.

```
\newlength{\linenumsep}
```
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.
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.

\long\def\xleft@appenditem\#1\to\#2{\%\global\led@toksb=\expandafter{\#2}\%\xdef\#2{\the\led@toksa\expandafter{\#1}\the\led@toksb}\%\global\led@toksb={}\%}

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\lempty to test for an empty \l. The control sequences created by \gl@p are all global.

\long\def\gl@p\#1\to\#2{\expandafter\gl@poff\#1\gl@poff\#1\#2}
\long\def\gl@poff\\#1\#2\gl@poff\#3\#4{\gdef\#4{\#1}\gdef\#3{\#2}}

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.

\newcount\line@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.

\newcount\subline@num

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

\ifsublines@true \ifsublines@false

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.

\newif\ifsline
\absline@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.

\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.

\newcount\@lock \newcount\sub@lock

\line@list \insertlines@list \actionlines@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 \sub@line@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 \do@lockon and \do@lockoff macros, as called to implement the
\startlock and \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 \set@line@action macro,
as called to implement the \advanceline and \setline macros: this action
only occurs when the user has specified some change to the line numbers
using those macros. Normally \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:

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

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.

\newcount\page@num
\newcount\endpage@num
\newcount\endline@num
\newcount\endsubline@num

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 \TexteX, 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 \TexteX two more
times; there’s no fix needed to the document. The \ifnoteschanged@ flag is set
if such a change in the number of notes is discovered at any point.
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 \numberonlyfirstineline. 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@ command resets this macro for every series.

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

\read@linelist
\read@linelist{⟨file⟩} is the control sequence that’s called by \beginnumbering (via \line@list@stuff) to open and process a line-list file; its argument is the name of the file.

\get@linelistfile{#1}%;

\read@linelist
\read@linelist{⟨file⟩} is the control sequence that’s called by \beginnumbering (via \line@list@stuff) to open and process a line-list file; its argument is the name of the file.

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.
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 \texttt{\textbackslash next@actionline} and \texttt{\textbackslash next@action} macros, which specify where and what the next action to be taken is.

\begin{verbatim}
\global\page@num=\m@ne
\ifx\actionlines@list\empty
  \gdef\next@actionline{1000000}%
\else
  \gl@p\actionlines@list\to\next@actionline
  \gl@p\actions@list\to\next@action
\fi
\end{verbatim}

\texttt{\textbackslash list@clearing@reg} clears the lists for \texttt{\textbackslash read@linelist}.

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

\texttt{\textbackslash get@linelistfile} \texttt{eledmac} can take advantage of the \LaTeX{} ‘safe file input’ macros to get the line-list file.

\begin{verbatim}
\newcommand*{\get@linelistfile}[1]{% 
  \InputIfFileExists{#1}{% 
    \global\noteschanged@false
    \begingroup
    \catcode'\[=1 \catcode'\]=2
    \makeatletter \catcode'\^^M=9}{% 
    \led@warn@NoLineFile{#1}%
    \global\noteschanged@true
    \begingroup}%
  \end{verbatim}

This version of \texttt{\textbackslash 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 \texttt{\textbackslash pausenumbering} and \texttt{\textbackslash 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 \texttt{\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 \texttt{\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 a unfortunate tendency to change the number of the last line of the preceding 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}
511 \newcommand{\@nl}[2]{%
512 \fix@page(#1)\%
513 \@nl@reg}
514 \newcommand*{\@nl@reg}{%
515 \ifx\l@dchset@num\relax \else
516 \advance\absline@num \@ne
517 \set@line@action
518 \let\l@dchset@num=\relax
519 \advance\absline@num \m@ne
520 \advance\line@num \m@ne
521 \fi
522 \advance\absline@num \@ne
523 \ifx\next@page@num\relax \else
524 \page@action
525 \let\next@page@num=\relax
526 \fi
527 \ifx\sub@change\relax \else
\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 \@ne
  \fi
\else
  \ifnum\@lock<\tw@
    \advance\line@num \@ne \subline@num \z@
  \fi
\fi

\last@page@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.

\begin{verbatim}
\newcount\last@page@num
\last@page@num=-10000
\newcommand*{\fix@page}[1]{{% #ifnum #1=\last@page@num
\else
  \ifbypage@
    \line@num\z@ \subline@num\z@
  \fi
  \page@num=#1\relax
  \last@page@num=#1\relax
  \def\next@page@num{#1}\%
  \listxadd{\normal@page@break}{\the\absline@num}
\end{verbatim}
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.

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.

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

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

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

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

\page@action \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 \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@ 
  \@l@dtempcnta=-\subline@num 
  \else 
  \@l@dtempcnta=-\line@num 
  \fi 
  \advance\@l@dtempcnta by -5000 
  \xright@appenditem{\the\@l@dtempcnta}\to\actions@list}

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

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

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

\do@lockon \do@lockonL The current setting of the sub-lineation flag tells us whether this applies to line numbers or sub-line numbers.
Commands within the line-list file

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
\ifs sublines@
\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
\fi}

\lock@off \lock@off adds an entry to the action-code list to turn line number locking off.
\do@lockoff
\do@lockoffL
\skip@lockoff
\newcommand*{\do@lockoffL}{%
\xright@appenditem{\the\absline@num}\to\actionlines@list
\ifs sublines@
\xright@appenditem{-1006}\to\actions@list
\ifnum\sub@lock=\tw@
\sub@lock \thr@@
\else
\sub@lock \z@
\fi
\fi
\else
\ifnum\@lock=\tw@
\@lock \thr@@
\else
\fi
\fi
\fi
\fi
\@lock \z@ 
\fi 
\fi}
\newcommand*{\do@lockoff}{\do@lockoffL}
\newcommand*{\skip@lockoff}{\global\let\lock@off=\do@lockoff}
\global\let\lock@off=\do@lockoff

\n@num \n@num@reg
This macro implements the \skipnumbering command. It uses a new action code, namely 1007.
\newcommand*{\n@num}{\n@num@reg}
\newcommand*{\n@num@reg}{% 
\xright@appenditem{\the\absline@num}\to\actionlines@list 
\xright@appenditem{-1007}\to\actions@list}

\@ref \@ref marks the start of a passage, for creation of a footnote reference. It takes two arguments:
- #1, the number of entries to add to \insertlines@list for this reference. This value, here and within \edtext, which computes it and writes it to the line-list file, will be stored in the count \insert@count.
- #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.)

\dummy@ref 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.
\newcommand*{\dummy@ref}{[2]{#2}}

\@ref@reg The first thing \@ref (i.e. \@ref@reg) itself does is to add the specified number of items to the \insertlines@list list.
\newcommand*{\@ref}{[2]{% 
\@ref@reg[#1]{#2}}}
\newcommand*{\@ref@reg}{[2]{% 
\global\insert@count=#1\relax 
\loop\ifnum\insert@count\zer@ 
\xright@appenditem{\the\absline@num}\to\insertlines@list 
\global\advance\insert@count \m@ne 
\repeat

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


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 \texttt{true} before any \linenum@out file is opened. When such a file is opened for the first time, it's done using \texttt{\immediate}, so that it will at once be safe for the output routine to write to it; we then set this flag to \texttt{false}.

\iffirst@linenum@out@
\first@linenum@out@true
\first@linenum@out@false

\newwrite\linenum@out

\first@linenum@out@true
\line@list@stuff

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.

\newcommand*{\line@list@stuff}[1]{%
First, use the commands of the previous section to interpret the line-list file from the last run.
\read@linelist{#1}%%%%

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.
\iffirst@linenum@out@%
\immediate\closeout\linenum@out%
\global\first@linenum@out@false%
\immediate\openout\linenum@out=#1\relax%
\else
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.
\if@minipage%
\leavevmode%
\fi%
\closeout\linenum@out%
\openout\linenum@out=#1\relax%
\fi}

\new@line

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.
\newcommand*{\new@line}{%
\IfStrEq{\led@pb@setting}{after}{
{xifinlist{⟨the\absline@num⟩}{\l@prev@nopb}}%
{\xifinlist{⟨the\absline@num⟩}{\normal@page@break}}%
{\numgdef{\@next@page}{\thepage+1}%
\write\linenum@out{⟨string\@nl\@next@page⟩[\@next@page]}%
}}%
{\write\linenum@out{⟨string\@nl[\the@c@page]⟨the@page⟩⟩}}%
}
\IfStrEq{\led@pb@setting}{before}{
{\numdef{\next@absline}{\the\absline@num+1}%
{xifinlist{⟨the\absline@num⟩}{\l@prev@nopb}}%
{xifinlist{⟨the\absline@num⟩}{\normal@page@break}}%
{\numgdef{\nc@page}{\c@page+1}%
\write\linenum@out{⟨string\@nl[\nc@page]⟨\nc@page⟩⟩}}%
}}%
21.6 Writing to the line-list file

\if@noneed@Footnote \if@noneed@Footnote is a boolean to check if we have to print a error message when a \text{edtext} is called without any footnotes.
\fi
\fi
\text{flag@start} We enclose a lemma marked by \text{edtext} in \text{flag@start} and \text{flag@end}: these send the \text{@ref} command to the line-list file. \text{edtext} is responsible for setting the value of \text{insert@count} appropriately; it actually gets done by the various footnote macros.
\newif\if@noneed@Footnote
\newcommand*{\flag@start}{\ifledRcol\edef\next{\write\linenum@outR{\string\@ref[the\insert@countR]}}\next\ifnum\insert@countR<1\else\led@err@EdtextWithoutFootnote\fi\else\edef\next{\write\linenum@out{\string\@ref[the\insert@count]}}\next\ifnum\insert@count<1\else\led@err@EdtextWithoutFootnote\fi\fi\fi}
\page@start Originally the commentary was: \page@start 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 \text{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 \text{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 \startsub 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.

\newcommand*{\startlock}{\write\linenum@out{\string\lock@on}}
\def{\endlock}{\write\linenum@out{\string\lock@off}}

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

\newif{\l@dskipnumber}
\l@dskipnumberfalse
\newcommand*{\skipnumbering}{\skipnumbering@reg}
\newcommand*{\skipnumbering@reg}{\write\linenum@out{\string\n@num} \advanceline{-1}}

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 \texttt{\*text} macro may be used (somewhat) recursively; that is, \texttt{\*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 \texttt{\lemma} and \texttt{\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 \texttt{\*text} will fail if you try to use a copy that is called something other than \texttt{\*text}. In order to handle recursion, \texttt{\*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 \texttt{\*text}. There’s no problem as long as \texttt{\*text} is not invoked in the first argument. If you want to call \texttt{\*text} something else, it is best to create instead a macro that expands to an invocation of \texttt{\*text}, rather than copying \texttt{\*text} and giving it a new name; otherwise you will need to add an appropriate definition for your new macro to \texttt{\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 \texttt{\do@line}, p.\texttt{[2]}. Instead, the appropriate note-generating command is appended to the list macro \texttt{\inserts@list}, and when \texttt{\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 \texttt{\*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.

\section{22.1 \texttt{\edtext} (and \texttt{\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.

\texttt{\end@lemmas} To accommodate 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 \texttt{\end@lemmas} by using \texttt{\xleft@appenditem}. (Anything that needs to be done at the start of the lemma may be handled using \texttt{\aftergroup}, since the commands
specified within \texttt{\edtext}'s second argument are executed within a group that ends just before the lemma is added to the main text.)

\texttt{\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 \texttt{\end@lemmas} or of the \texttt{\aftergroup} trick. The general approach would be to define a macro to be used within the second argument of \texttt{\edtext} that would add the appropriate command to \texttt{\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.

\begin{verbatim}
\list@create{\end@lemmas}
\end{verbatim}

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

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

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

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

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

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

\begin{verbatim}
\newcommand{\dummy@edtext@showlemma}[2]{\showlemma{#1}}
\end{verbatim}

We’re going to need another macro that takes one argument and ignores it entirely. This is supplied by the \LaTeX{} \texttt{\gobble{⟨arg⟩}}.

\texttt{\no@expands} \texttt{\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 TeXbook}, 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 \texttt{\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 eledmac macros like \edlabel{} and \setline{} that write things to auxiliary files: that writing should be done only once. And we make \edtext{} itself, if it appears within its own argument, do nothing but copy its first argument.

Finally, we execute \morenoexpands{}. The version of \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 eledmac code. If you define your own \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 \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 \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 always active; within languages that make no special use of them, their associated control sequences should simply return the proper character.)

\newcommand*{\no@expands}{\let\select@@lemmafont=0\protect\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\@tag

Now, we define an empty \@tag command. It will be redefine by \edtext: its value is the first args. It will be used by the \Xfootnote commands.

\newcommand{\@tag}{\end{macrocode}\end{macro}{\end{macro}{\end{macro}{\end{macro}{\end{macro}}}
% \changes{v1.15.0}{2015/01/12}{New boolean \ct{if@edtext@}.}
% 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@
\long\def\critext#1#2/{\edtext{#1}{#2}}%
\edtext When executed, \edtext first ensures that we’re in horizontal mode.
\newcommand{\edtext}[2]{\leavevmode%
Then, check if we are in a numbered paragraph (\cs{pstart} \ldots \cs{pend}).
\ifnumberedpar@
We switch the \@edtext@ flag, to let know to other commands that we are in an \edtext.
\@edtext@true%
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.
\begingroup%
\global\renewcommand{\@tag}{\no@expands #1}%
Prepare more data for the benefit of note-generating macros: the line references and font specifier for this lemma go to \l@d@nums.
\set@line%
\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.
\ifledRcol \global\insert@countR \z@
\else \global\insert@count \z@ \fi%
Now process the note-generating macros in argument #2 (i.e., \footnote, \@lem, 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.

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.

We switch to false some flag.

• The one that checks having footnotes inside a \edtext.

• The one that says we are inside a \edtext.

• The one that says we are inside à @lemma.

If we are outside of a numbered paragraph, we send error message and print the first argument.
The `\ifnumberline` option can be set to FALSE to disable line numbering.

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

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

If no more lines are listed in `\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 `\edfont@info` returns coded information about the current font.

The `\lemma` macro allows you to change the lemma that’s passed on to the notes.

This boolean is set to TRUE inside a `\edtext` (or `\critext`) when a `\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

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

As argument \texttt{\linenum} takes a set of seven parameters separated by vertical bars, in the format used internally for \texttt{\l@d@nums} (see p. 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. 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 \texttt{\linenum{18|4|0|18|7|1|0}} is an invocation that changes all the parameters, but \texttt{\linenum{13}} only changes the starting line number, and leaves the rest unaltered.

We use \texttt{\textbackslash} as an internal separator for the macro parameters.

\begin{verbatim}
\newcommand*{\linenum}[1]{\xdef\@tempa{#1\\l@d@nums}\\global\let\l@d@nums=\empty\line@set\@tempa}
\end{verbatim}

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

\begin{verbatim}
def\line@set#1|#2\#3#4|\{\gdef\l@d@add{#1#2\#3\#4}{\gdef\l@d@add{#1}{\gdef\@tempb{#4}{\ifx\@tempb\empty\l@d@add{#3}{\else\l@d@add{#1}{\l@d@add{}\line@set#2\#4\{}{\fi}}}}}}\end{verbatim}

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

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

### 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 \texttt{\pstart} and a \texttt{\pend}, it does not know yet which are the line numbers.

The general mechanism is the following:

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

- Then this counter, associated with the argument of \texttt{\sameword} is stored (\@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%}
926 \texttt{\unless ifledRco1%}
927 \texttt{\csnumgdef{\sw@#2}{\csuse{\sw@#2}+1}}\%
928 \texttt{\unless if\texttt{\sw@list}\empty}%
929 \texttt{\protected@write\linenum@out{}{\string\sw@#2}{\csuse{\sw@#2}}}\%
930 \texttt{\unless if\texttt{\sw@list}\empty%
931 \texttt{\csuse{\sw@#2}+1}}\%
932 \texttt{\unless if\texttt{\sw@list}\empty%
933 \texttt{\unless if\texttt{\sw@list}\empty%
The command printed in the auxiliary files.

First, define a counter which store the second argument as value for a each paired absolute line number/first argument

If such argument was not defined for the preceding line, define it.

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.
The command called when \sameword is called in a edtext.

First, calculate the number of occurrences of the word in the current line

\ifsdef{sw@#1@\the\absline@numR \the\section@numR R}{}%\fi%
\numdef{\prev@line}{\the\absline@numR-1}%
\ifcsundef{sw@#1@\prev@line \the\section@numR R}{}%\fi%
\csnumgdef{sw@#1@\prev@line \the\section@numR R}{#2-1}%
\numdef{\the@sw}{#2-\csuse{sw@#1@\prev@line \the\section@numR R}}%
\xright@appenditem{\the@sw}\to\sw@listR%
\fi%
\enddef{\sameword@inedtext}

Just a precaution.
\ifx\sw@list@inedtext\empty%
\def\the@sw{999}%
\else%
\gl@p\sw@list@inedtext\to\the@sw%
\fi%
\fi%
\enddef{\sameword@inedtext}%

But in many cases, at this step, we should have some content in the list \sw@list@inedtext, which contains the reference for edtext.
\ifnumgreater{\sw@atthisline}{1}%
{\showwordrank{#1}{\the@sw}}%
\else%
\ifcsdef{sw@#1@\the\absline@numR \the\section@numR R}{}%\fi%
\numdef{\prev@line}{\the\absline@numR-1}%
\numdef{\sw@atthisline}{\csuse{sw@#1@\the\absline@numR \the\section@numR R}-\csuse{sw@#1@\prev@line \the\section@numR R}}%
\numdef{\sw@atthisline}{0}%%
\ifx\sw@list@inedtextR\empty%
\def\the@sw{999}%
\else%
\gl@p\sw@list@inedtextR\to\the@sw%
\fi%
\fi%
\ifnumgreater{\sw@atthisline}{1}%
{\showwordrank{#1}{\the@sw}}%
\else%
\enddef{\showwordrank{#1}{\the@sw}}%
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.
\section{Boxes, counters, \texttt{\pstart} and \texttt{\pend}}

\newcounter{pstart}
\renewcommand{\the\pstart}{\bfseries\@arabic\c@pstart}.}
\newif\numberpstart
\numberpstartfalse
\newif\labelpstart
\labelpstartfalse
\newcommandx*{\pstart}[1][1]{% #1 is a label
\ifstrempty{#1}{\atevery\pstart}{\noindent#1}\
\ifluatex\edef\l@luatextextdir@L{\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\list@clear{\inserts@list}\
\global\let
ext@insert=\empty\begingroup\normal@pars\global\advance \l@dnumpstartsL\@ne\global\setbox\raw@text=\vbox{\ifautopar\else\ifnumberpstart\ifinstanza\else\ifsip\else\thepstart\fi\fi\fi\numberedpar@true\ifnumberedpar@false\ifnumberpstart\ifinstanza\else\ifsip\else\thepstart\fi\fi\fi\ifnumberedpar@true\fi\fi\fi\fi\fi\fi\fi}}

\pend \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 \texttt{endgraf} to end the paragraph; this ensures that there'll be no large interline penalties to prevent us from slicing the paragraph into pieces. These penalties revert to the values that you set when the group for the \texttt{vbox} ends. Then we call \texttt{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.

\texttt{\l@dzeropenalties}
\texttt{\endgraf\global\num@lines=\prevgraf\egroup}
\texttt{\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 \texttt{pstart \setline} is parsed at the end of previous \texttt{pend}, and so, we must do it at the end of first line of pstart.

\texttt{\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 \texttt{pstart}.

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

\texttt{\AtEveryPend}\texttt{\at@every@pend}
\texttt{\newcommand{\AtEveryPend}[1]{\xdef\at@every@pend{\noindent\unexpanded{#1}}}\xdef\at@every@pend{}}

\texttt{\l@dzeropenalties}  A macro to zero penalties for \texttt{pend}.
\texttt{\newcommand*{\l@dzeropenalties}{}}
23.1 *Boxes, counters, \pstart and \pend*  

\autopar 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}

\ifautopar@pause
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

\do@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@\}
\do@linehook\}
\l@demptyd@ta\}
\global\setbox\one@line=\vsplit\raw@text to\baselineskip}%
\getline@num
\IfStrEq{\led@pb@setting}{before}{\led@check@pb\led@check@nopb}{%}
\ifnum\@lock>\@ne\}
inserthangingsymboltrue\}
\else\}
inserthangingsymbolfalse\}
\fi\}
\check@pb@in@verse\}
affixline@num

Depending whether a sectioning command is called at this pstart or not we print sectioning command or normal line,

\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 is for normal line, i.e. line without sectioning command.
\def\print@line{%}
Insert the pstart number in side, if we are in the first line of a pstart.
\affixpstart@num%
The line will be boxed, to have the good width.
23.2 Processing one line

User hook.
\do@insidelinehook\

Left line number
\l@dld@ta\

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

Print left notes.
\l@dlsn@te\

Boxes the line, writes information about new line in the numbered file.

\\l@dllfill\hb@xt@ \wd\one@line{\new@line\%}
If we use \LaTeX\TEX then restore the direction.
\ifluatex\%
\luatextextdir\l@luatextextdir@L\%
\fi\%

Insert, if needed, the hanging symbol.
\ins@rthangingsymbol \%Space keep for backward compatibility
And so, print the line.
\l@dunhbox@line{\one@line}\%\

Right line number
\\\l@drlfill\l@drd@ta\

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

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

\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.
\def\print@eledsection{%\add@inserts\affixside@note%\numdef{\tempo}{\l@dnumpstartsL-1}%\xifinlist{\tempo}{\eled@sections@@}{\@nobreaktrue}{\@nobreakfalse}%\eled@sectioningtrue\csuse{eled@sectioning@\the\l@dnumpstartsL}%\eled@sectioningfalse\global\csundef{eled@sectioning@\the\l@dnumpstartsL}%\if\@RTL%\hspace{-3}\paperwidth}%{\hbox{\l@dunhbox@line{\one@line}} \new@line}%
These high level commands just redefine the low level commands. They have to be used by user, without \makeatletter.

\do@linehook 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.

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

\lsnote 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 (\ledrfill) of each numbered line. The initial definitions correspond to the original code for \do@line.

\getline@num The \getline@num macro determines the page and line numbers for the line we’re about to send to the vertical list.
\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.

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

\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
process for this line, \texttt{\do@actions@next} is set equal to \texttt{\do@actions}; otherwise it’s just \texttt{\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.
  \fi
}

Next, we handle commands that change the line-number values. (We subtract \texttt{5001} rather than \texttt{5000} here because the line number is going to be incremented automatically in \texttt{getline@num}.)

\newcolumntype{C}[1]{>{\texttt{\@l@dtempcnta}=-\next@action\def\@l@dtempcnta{\@l@dtempcnta-5001}}\ifsublines@\global\subline@num=\@l@dtempcnta\else\global\line@num=\@l@dtempcnta\fi}

Now we get information about the next action off the list, and then set \texttt{\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 \texttt{\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.

\do@actions@next

\do@actions@fixedcode  This macro handles the fixed codes for \do@actions. It is one big case statement.

\newcommand*{\do@actions@fixedcode}{% \ifcase\@l@dtempcnta % 1001 \or% \global\sublines@true % 1002 \or% \global\sublines@false % 1003 \or% \global\@lock=\@ne % 1004 \or% % 1005 \or% \global\sub@lock=\@ne % 1006 \or% % 1007 \or% \global\sub@lock=\@ne % 1008 \global\@lock=\@ne \or% \global\sub@lock=\@ne \else \global\@lock=\z@ \global\sub@lock=\z@ \fi \fi \fi \fi \global\l@dskipnumbertrue \else \led@warn@BadAction \fi}

23.4 Line number printing

\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@ld@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:

\[ n = \text{int}((\text{linenum} - \text{firstlinenum})/\text{linenumincrement}) \]

\[ m = \text{firstlinenum} + (n \times \text{linenumincrement}) \]
Paragraph decomposition and reassembly

(where \textit{int} truncates a real number to an integer). \textit{m} will be equal to \textit{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 \line@num \leq \firstlinenum, we compare the two directly instead of making these calculations.

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

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

\newcommand*{\affixline@num}{%
No number is attached if \ifl@dskipnumber is TRUE (and then it is set to its normal FALSE value). No number is attached if \ifnumberline is FALSE (the normal value is TRUE).
\ifledgroupnotesL\else\ifnumberline
\ifl@dskipnumber
\global\l@dskipnumberfalse
\else
\ifsublines@
\@l@dtempcntb=\subline@num
\@ifnum\subline@num>c@firstsublinenum
\@l@dtempcnta=\subline@num
\divide\@l@dtempcnta by c@sublinenumincrement
\multiply\@l@dtempcnta by c@sublinenumincrement
\advance\@l@dtempcnta by c@firstsublinenum
\else
\@l@dtempcnta=c@firstsublinenum
\fi
\fi

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.

\ch@cksub@l@ck

Now the line number case, which works the same way.
\else
\@l@dtempcntb=\line@num

Check on the \linenumberlist If it’s \empty use the standard algorithm.
\ifx\linenumberlist\empty
\@ifnum\line@num>c@firstlinenum
\@l@dtempcnta=\line@num
\advance\@l@dtempcnta by c@firstlinenum
\divide\@l@dtempcnta by c@linenumincrement
\multiply\@l@dtempcnta by c@linenumincrement
\advance\@l@dtempcnta by c@firstlinenum
\else
\@l@dtempcnta=c@firstlinenum
\fi
The `\linenumberlist` wasn’t `\empty`, so here’s Wayne’s numbering mechanism. This takes place in TeX’s mouth.

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

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

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

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

```latex
\ifnum\@ldtempcnta=\@ldtempcntb
```

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\if@twocolumn
\l@drd@ta\else
```

Continuing the original code ...

```latex
\noexpand\rem@inder{\number\@ldtempcnta,####2}%%
\ifnum\@ldtempcntb>\@ne
\advance\@ldtempcntb\@ne
\fi
```

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

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

```latex
\gdef\l@dld@ta{\llap{{\leftlinenum}}}%
\else
```

```latex
\advance\@ldtempcnta\@ne
\fi
```

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

```latex
\l@dld@ta
```
Paragraph decomposition and reassembly

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 `\affixline`.

`\chcksublo` 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.
\pageparbreak

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}

23.5 \textit{Pstart} number printing in \textit{side}

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

- The \textit{pstarts} counter is upgrade in the \texttt{\pend} command. Consequently, the \texttt{\affixpstart@num} command has not to upgrade it, unlike the \texttt{\affixline@num} which upgrades the lines counter.

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

\texttt{\inserts@list} is the list macro that contains the inserts that we save up for one paragraph.

\texttt{\add@inserts\add@inserts@next} is the penultimate macro used by \texttt{\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 \LaTeX{}’s optimization for tail recursion), we define a control-sequence called \texttt{\add@inserts@next} that is always the last thing that \texttt{\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.

\newcommand*{\add@inserts}{%
  \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.

\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.

  \ifx\next@insert\empty
    \ifx\insertlines@list\empty
      \global\noteschanged@true
      \edef\next@insert{100000}%
    \else
      \gl@p\insertlines@list\to\next@insert
    \fi
  \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.

  \ifnum\next@insert=\absline@num
    \gl@p\inserts@list\to\@insert
  \fi

  Make the recursive call, if necessary.
\fi

\add@inserts@next}

23.7 Penalties

\add@penalties is the last macro used by \doc@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 \vsplit 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 \ballast@count, which has been worked out in \do@ballast above (p. 95). Finally, the penalty is checked to see that it doesn't go below $-10000$.\newcommand*{\add@penalties}{\@l@dtempcnta=\ballast@count 1474 \ifnum\num@lines>\@ne 1475 \global\advance\par@line \@ne 1476 \ifnum\par@line=\@ne 1477 \advance\@l@dtempcnta \clubpenalty 1478 \fi 1479 \@l@dtempcntb=\par@line \advance\@l@dtempcntb \@ne 1480 \ifnum\@l@dtempcntb=\num@lines 1481 \advance\@l@dtempcntb \num@lines 1482 \ifnum\par@line<\num@lines 1483 \fi 1484 \ifnum\par@line<\num@lines 1485 \ifnum\par@line<\num@lines 1486 \fi 1487 \fi 1488 \@l@dtempcnta=\z@ 1489 \relax 1490 \else 1491 \@l@dtempcnta=-10000 1492 \penalty\@l@dtempcnta 1493 \else 1494 \penalty-10000 1495 \fi 1496 \fi} 1497 23.8 Printing leftover notes\flush@notes The \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.\newcommand*{\flush@notes}{% 1498 \@xloop 1499 \@repeat\@xloop\@xloop is a variant of the Plain \TeX \loop macro, useful when it’s hard to construct a positive test using the \TeX if commands—as in \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.

```latex
\def\@xloop#1\repeat{%
  \def\body{#1\expandafter\body\fi}%
  \body}
```

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.

```latex
\select@lemmafont
\select@@lemmafont
\select@lemmafont
\select@lemmafont
```

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.

```latex
\def\select@lemmafont#1|#2|#3|#4|#5|#6|#7|\{
  \select@@lemmafont#7|\}
\def\select@@lemmafont#1/#2/#3/#4|\{
  \fontencoding{#1}\fontfamily{#2}\fontseries{#3}\fontshape{#4}\selectfont}
```

24.2 Outer-level footnote commands

```latex
\footnoteoptions@
```

The `$\footnoteoption@(⟨side⟩){⟨options⟩}{⟨value⟩}$` change the value of on options of Xfootnote, to switch between true and false.

```latex
\newcommandx*{\footnoteoptions@}[3][1=L, usedefault]{%
  \def\do##1{\ifstrequal{#1}{L}{\xright@appenditem{\global\noexpand\settoggle{##1@}{#3}}\to\inserts@list% Switch toggle, in all case\global\advance\insert@count \@ne% Increment the left insert counter.}}
  \def\do#1{%}
```

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

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

\footspitskips Some setup code that is common for a variety of the footnotes. The setup is for:

- \interlinepenalty.
- \splittopskip (skip before last part of notes that flow from one page to another).
- \splitmaxdepth.
- \floatingpenalty, that is penalty values being added when a long note flows from one page to another. Here, we let it to 0 when we are processing
parallel pages in eledpar, in order to allow notes to flow from left to right pages and vice-versa. Otherwise, we let it to \@MM, which is the standard \LaTeX\ \texttt{floatingpenalty}.

\begin{verbatim}
\newcommand*{\footsplitskips}{%
  \interlinepenalty=\interfootnotelinepenalty
  \unless\if@oddpages%
    \floatingpenalty=\@MM%
  \fi%
  \splitallskip=\ht\strutbox \splitmaxdepth=\dp\strutbox
  \leftskip=\z@skip \rightskip=\z@skip}%
\end{verbatim}

\texttt{\mpnormalvfootnote} And a somewhat different version for minipages.

\begin{verbatim}
\notbool{parapparatus@}{\newcommand*}{\newcommand}{\mpnormalvfootnote}{2}{%
  \global\setbox\@nameuse{mp#1footins}\vbox{%
    \unvbox\@nameuse{mp#1footins}
    \csuse{bhookXnote@#1}
    \csuse{Xnotefontsize@#1}
    \hsize\columnwidth
    \@parboxrestore
    \color@begingroup
    \csname #1footfmt\endcsname #2[#1]\color@endgroup}%
\end{verbatim}

\texttt{\ledsetnormalparstuff} \texttt{\normalfootfmt} is a ‘normal’ macro to take the footnote line and page number information (see p. 62), and the desired text, and output what’s to be printed. Argument #1 contains the line and page number information and lemma font specifier; #2 is the lemma; #3 is the note’s text. This version is very rudimentary—it uses \texttt{\printlines} to print just the range of line numbers, followed by a square bracket, the lemma, and the note text; it’s intended to be copied and modified as necessary.

\texttt{\par} should always be redefined to \texttt{\endgraf} within the format macro (this is what \texttt{\normal@pars} does), to override tricky material in the main text to get the lines numbered automatically (as set up by \texttt{\autopar}, for example).

\begin{verbatim}
\newcommand*{\ledsetnormalparstuff}{%
  \ifluatex%
    \luatextextdir\footnote@luatextextdir%
  \fi%
  \luatexpardir\footnote@luatexpardir%
  \ifx%
  \csuse{\csuse{footnote@dir}}%
  \normal@pars%
  \noindent \parfillskip \z@ \@plus 1fil}
\end{verbatim}

\texttt{\notbool{parapparatus@}{\newcommand*}{\newcommandx}{\normalfootfmt}{4}[4=Z]{% 4th arg is optional, for backward compatibility
  \ledsetnormalparstuff%
  \hangindent=\csuse{Xhangindent@#4}
  \strut\texttt{\printlinefootnote(#1)#4}}%
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; 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.

The \fullstop 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 \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 \@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.

\TeX\ 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 has reverted to
traditional booleans.

\if\@d@pnum
\if\@d@ssub \newif\if\@d@pnum \@d@pnumfalse
\if\@d@elin \newif\if\@d@ssub \@d@ssubfalse
\if\@d@esl \newif\if\@d@elin \@d@elinfalse
\if\@dash \newif\if\@d@esl \@d@eslfalse
\newif\if\@d@dash \@d@dashfalse
\l@dparsedstartpage
\l@dparsedstartline
\l@dparsedstartsub
\l@dparsedendpage
\l@dparsedendline
\l@dparsedendsub
\l@dparsedfootspec \l@dparsedfootspec{⟨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 \@d@nums style format. The real work
is done by \l@dparsedfootspec which defines macros holding the numeric values.

\l@dparsedstartpage \newcommand*{\l@dparsedstartpage}[3]{\def\l@dparsedendpage{\@d@pnumfalse}{\@d@ssubfalse}{\@d@elinfalse}{\@d@eslfalse}{\@dashfalse}{\l@dparsedstartpage{#1}{#2}{#3}}{\l@dparsedendpage}}
\l@dparsedstartline \def\l@dparsedstartline{\l@dparsedendpage\{\l@dparsedstartpage\{\l@dparsedstartsub\{#1\}#2\{#3\}#4\{#5\}#6\{#7\}}{\l@dparsedendline}}
\l@dparsedendsub \l@dparsedendsub \def\l@dparsedendsub{\l@dparsedendline\{#1\}#2\{#3\}#4\{#5\}#6\{#7\}}
\l@dparsedendline \l@dparsedendline \l@dparsedendline
\l@dparsedstartpage
\l@dparsedstartline
\l@dparsedendsub
\l@dparsedendline

Initialise the several number value macros.
\def\l@dparsedstartpage{0}
\def\l@dparsedstartline{0}
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.

\setprintlines

Just a reminder of the arguments:
\printlines #1 | #2 | #3 | #4 | #5 | #6 | #7
\printlines 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.

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.

\printlines 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 \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 \skip\footins 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 \skip\footins 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.
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 \texttt{\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 \texttt{\abaselineskip}, since this is one of the quantities set in \texttt{\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 \texttt{eledmac} code.)

\begin{verbatim}
\skip\Afootins=12pt plus5pt minus5pt
\count\Afootins=1000
\dimen\Afootins=0.8\vsize
\let\vAfootnote=\normalvfootnote \let\Afootfmt=\normalfootfmt
\let\Afootstart=\normalfootstart \let\Afootgroup=\normalfootgroup
\let\Afootnoterule=\normalfootnoterule
\end{verbatim}

Instead of repeating ourselves, we define a \texttt{\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 \texttt{normal} setting.

\begin{verbatim}
\ledfootinsdim \newcommand*{\ledfootinsdim}{0.8\vsize} % kept for backward compatibility, should`nt be used anymore.
\newtoggle{preXnotes@}
\toggletrue{preXnotes@}
\newcommand{\preXnotes@}{0pt}
\newcommand*{\preXnotes}{[1]{\renewcommand{\preXnotes@}{#1}}
\end{verbatim}

The same, but for familiar footnotes.

\begin{verbatim}
\newtoggle{prenotesX@}
\toggletrue{prenotesX@}
\newcommand{\prenotesX@}{0pt}
\newcommand*{\prenotesX}{[1]{\renewcommand{\prenotesX@}{#1}}
\end{verbatim}

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

\begin{verbatim}
\newcommand*{\footnormal}{[1]{\csgdef{series@display#1}{normal}
\expandafter\let\csname #1footstart\endcsname=\normalfootstart
\expandafter\let\csname v#1footnote\endcsname=\normalvfootnote
\expandafter\let\csname #1footfmt\endcsname=\normalfootfmt
\expandafter\let\csname #1footgroup\endcsname=\normalfootgroup
\expandafter\let\csname #1footnoterule\endcsname=%\normalfootnoterule
\expandafter\let\csname #1footnoterule\endcsname=\normalfootnoterule
\expandafter\let\csname #1footstart\endcsname=\normalfootstart
\expandafter\let\csname v#1footnote\endcsname=\normalvfootnote
\expandafter\let\csname #1footfmt\endcsname=\normalfootfmt
\expandafter\let\csname #1footgroup\endcsname=\normalfootgroup
\expandafter\let\csname #1footnoterule\endcsname=%\normalfootnoterule
\expandafter\let\csname #1footstart\endcsname=\normalfootstart
\expandafter\let\csname v#1footnote\endcsname=\normalvfootnote
\expandafter\let\csname #1footfmt\endcsname=\normalfootfmt
\expandafter\let\csname #1footgroup\endcsname=\normalfootgroup
\expandafter\let\csname #1footnoterule\endcsname=\normalfootnoterule
\expandafter\let\csname #1footstart\endcsname=\normalfootstart
\expandafter\let\csname v#1footnote\endcsname=\normalvfootnote
\expandafter\let\csname #1footfmt\endcsname=\normalfootfmt
\expandafter\let\csname #1footgroup\endcsname=\normalfootgroup
\expandafter\let\csname #1footnoterule\endcsname=\normalfootnoterule
\expandafter\let\csname #1footstart\endcsname=\normalfootstart
\expandafter\let\csname v#1footnote\endcsname=\normalvfootnote
\expandafter\let\csname #1footfmt\endcsname=\normalfootfmt
\expandafter\let\csname #1footgroup\endcsname=\normalfootgroup
\expandafter\let\csname #1footnoterule\endcsname=\normalfootnoterule
\expandafter\let\csname #1footstart\endcsname=\normalfootstart
\expandafter\let\csname v#1footnote\endcsname=\normalvfootnote
\expandafter\let\csname #1footfmt\endcsname=\normalfootfmt
\expandafter\let\csname #1footgroup\endcsname=\normalfootgroup
\expandafter\let\csname #1footnoterule\endcsname=\normalfootnoterule
\expandafter\let\csname #1footstart\endcsname=\normalfootstart
\expandafter\let\csname v#1footnote\endcsname=\normalvfootnote
\expandafter\let\csname #1footfmt\endcsname=\normalfootfmt
\expandafter\let\csname #1footgroup\endcsname=\normalfootgroup
\expandafter\let\csname #1footnoterule\endcsname=\normalfootnoterule
\expandafter\let\csname #1footstart\endcsname=\normalfootstart
\expandafter\let\csname v#1footnote\endcsname=\normalvfootnote
\expandafter\let\csname #1footfmt\endcsname=\normalfootfmt
\expandafter\let\csname #1footgroup\endcsname=\normalfootgroup
\expandafter\let\csname #1footnoterule\endcsname=\normalfootnoterule
\expandafter\let\csname #1footstart\endcsname=\normalfootstart
\expandafter\let\csname v#1footnote\endcsname=\normalvfootnote
\expandafter\let\csname #1footfmt\endcsname=\normalfootfmt
\expandafter\let\csname #1footgroup\endcsname=\normalfootgroup
\expandafter\let\csname #1footnoterule\endcsname=\normalfootnoterule
\end{verbatim}
24.5 Paragraphed footnotes

Now do the setup for minipage footnotes. We use as much as possible of the normal setup as we can (so the notes will have a similar layout).

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.

\footparagraph

The \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 \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 \footparagraph only after the 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 hsize within the document, call \footparagraph again afterwards to take account of the new value. The argument of \footparagraph is the letter (A–E) denoting the series of notes to be paragraphed.

\newcommand*{\footparagraph}{\%}

\csxdef{series@display#1}{paragraph}
\expandafter\newcount\csname prevpage#1@num\endcsname
\expandafter\let\csname #1footstart\endcsname=\parafootstart
\expandafter\let\csname v#1footnote\endcsname=\para@vfootnote
\expandafter\let\csname #1footfmt\endcsname=\parafootfmt
\expandafter\let\csname #1footgroup\endcsname=\para@footgroup
\count\csname #1footins\endcsname=1000
\csxdef{default@#1footins}{1000}%Use this to confine the notes to one side only
And the extra setup for minipages.
\expandafter\let\csname mp\#1footnote\endcsname=\mppara@vfootnote
\expandafter\let\csname mp\#1footgroup\endcsname=\mppara@footgroup
\count\csname mp\#1footins\endcsname=1000
\dimen\csname mp\#1footins\endcsname=\csuse{maxhXnotes@#1}
\skip\csname mp\#1footins\endcsname=\csuse{beforeXnotes@#1}
}\}

\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. \footfudgefiddle can be increased from its default 64 (say to 70) to increase the estimate.
\providecommand{\footfudgefiddle}{64}
\para@footsetup \footparagraph calls the \para@footsetup macro to calculate a special fudge factor, which is the ratio of the \baselineskip to the \hsize. We assume that the proper value of \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.
\newcommand*{\para@footsetup}@[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 }}}

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.
\newcommand*{\parafootstart}@[1]{%}
\rightskip=0pt \leftskip=0pt \parindent=0pt
\ifdimequal{0pt}{\preXnotes@}{}{%}
\parafooterstart is the same as \normalfootstart, but we give it again to ensure that \rightskip and \leftskip are zeroed (this needs to be done before \para@footgroup 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.
\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 TeXbook, 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 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 \discretionary. 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{Wayne Sullivan pointed out to us another subtle problem that arises from the same cause: \TeX{} also leaves the \language whatsit nodes out of the horizontal list.\footnote{See 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{Michael’s unboxing macro is called \unvxh: unvbox, extract the last line, and unhbox it.}

\footnote{Wayne supplied his own macros to do this, but since they were almost identical to Michael’s, we have used the latter’s \unvxh macro since it is publicly documented.}
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. In other words, be very careful not to say `\break`, or `\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 `\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 eledmac is quite baroque enough already. If you think you are having this problem, look up Michael’s solutions.

One more thing; we set `\leftskip` and `\rightskip` to zero. This has the effect of neutralizing any such skips which may apply to the main text (cf. p. 112 above). We need to do this, since `footfudgefactor` is calculated on the assumption that the notes are `\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.

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`).

\footnote{Line Breaking', p. 610.}
This version is for minipages.

\begin{verbatim}
\newcommand*{\mppara@vfootnote}[2]{\global\setbox\@nameuse{mp#1footins}\vbox{\unvbox\@nameuse{mp#1footins}\csuse{bhookXnote@#1}\csuse{Xnotefontsize@#1}\footsplitskips\setbox0=\vbox{\hsize=\maxdimen\noindent\color@begingroup\csname #1footfmt\endcsname #2[#1]\color@endgroup}\setbox0=\hbox{\unvxh0[#1]}\dp0=\z@\ht0=\csname #1footfudgefactor\endcsname\wd0\box0\penalty0}}
\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 \texttt{\filbreak} of 10000, a \texttt{\filbreak} and a \texttt{\filbreak} (\textit{The \TeX{}book}, pp. 99–100). \unvxh cancels these unwanted paragraph-final items using \texttt{\unskip} and \texttt{\unpenalty}.

\newcommandx*{\unvxh}[2][2=Z]{\setbox0=\vbox{\unvbox#1\global\setbox1=\lastbox}\unhbox1\unskip % remove \filbreak, \unskip % remove \filbreak, \unpenalty % remove \filbreak of 10000, \hskip\csuse{afternote@#2}} % but add the glue to go between the notes
\parafootfmt
\parafootfmt is \texttt{\normalfootfmt} adapted to do the special stuff needed for paragraped notes—leaving out the \texttt{\endgraf} at the end, sticking in special penalties and kern, and leaving out the \texttt{\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=Z]{\insertparafootsep{#4}\ledsetnormalparstuff\printlinefootnote{#1}{#4}{\nottoggle{Xlemmadisablefontselection@#4}{\select@lemmafont#1|#2}{#2}}\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\penalty-10}
\end{verbatim}
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 \texttt{\insertparafootsep} command is used to insert the \texttt{\parafootsepseries} between each note in the same page.

\texttt{\para@footgroup} This footgroup code is modelled on the macros in \textit{The TeXbook}, p. 399. The only difference is the \texttt{\unpenalty} in \texttt{\makeboxofhboxes}, which is there to remove the penalty of 0 which was added to the end of each footnote by \texttt{\para@vfootnote}.

The call to \texttt{\notefontsetup} is to ensure that the correct \texttt{\baselineskip} for the footnotes is used. The argument is the note series letter.

\texttt{\mppara@footgroup} The minipage version.
24.6 Insertion of the footnotes separator

The command \insertparafootsep{⟨series⟩} must be called at the beginning of \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
\footthreecol 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.
\threecolfootsetup The \threecolfootsetup macro calculates and sets some numbers for three-column footnotes.

The \footstart and \footnoterule macros for these notes assume the normal values (p. 112 above).

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.
24.7 Columnar footnotes

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.

\newcommand*{\threecolfootsetup}[1]{% 
  \count\csname #1footins\endcsname 333
  \csxdef{default@#1footins}{333}%Use this to confine the notes to one side only
  \multiply\dimen\csname #1footins\endcsname \thr@@}

\mpthreecolfootsetup The setup for minipages.

\newcommand*{\mpthreecolfootsetup}[1]{% 
  \count\csname mp#1footins\endcsname 333
  \multiply\dimen\csname mp#1footins\endcsname \thr@@}

\threecolvfootnote \threecolvfootnote is the \vfootnote command for three-column notes. 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., .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).

\notbool{parapparatus@}{\newcommand*}{\newcommand}{\threecolvfootnote}[2]{% 
  \insert\csname #1footins\endcsname \bgroup
  \csuse{Xnotefontsize@#1}
  \footsplitskips
  \csname #1footfmt\endcsname #2[#1]\egroup}

\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.

\notbool{parapparatus@}{\newcommandx*}{\newcommandx}{\threecolfootfmt}[4][4=Z]{% 
  \normal@pars
  \hsize \csuse{hsizethreecol@#4}
  \parindent=0pt
  \tolerance=5000
  \raggedright
  \hangindent=\csuse{Xhangindent@#4}
  \leavevmode
  \strut\printlinefootnote{#1}{#4}
  \{\nottoggle{Xlemmadisablefontselection@#4}{\select@lemmafont#1|#2}{#2}\}%
  \iftoggle{nosep@}{\hskip\csuse{inplaceoflemmaseparator@#4}}{
    \ifcempty{lemmaseparator@#4}{\hskip\csuse{inplaceoflemmaseparator@#4}}{}}%
  \iftoggle{nobreak}{\hskip\csuse{beforelemmaseparator@#4}\csuse{lemmaseparator@#4}\hskip\csuse{afterlemmaseparator@#4}}{}}%
  #3\strut\par\allowbreak}
\threecolfootgroup

And here is the \texttt{footgroup} macro that’s called within the output routine to regroup the notes into three columns. Once again, the call to \texttt{notefontsetup} is there to ensure that it is the right \texttt{splittopskip}—the one used in footnotes—which is used to provide the third argument for \texttt{rigidbalance}. This third argument (\texttt{@h}) is the \texttt{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 \textit{The \TeX\book}, p. 398, Donald Knuth suggests retrieving the output of \texttt{rigidbalance}, putting it back into the insertion box, and then printing the box. Here, we just print the \texttt{line} which comes out of \texttt{rigidbalance} directly, without any re-boxing.

\begin{verbatim}
1983 \newcommand*{\threecolfootgroup}[1]{{\csuse{Xnotefontsize@#1}\% \\
1984 \noindent\csuse{txtbeforeXnotes@#1}\par\\
1985 \splittopskip=\ht\strutbox\\
1986 \expandafter\\
1987 \rigidbalance\csname #1footins\endcsname \thr@@ \splittopskip}}
\end{verbatim}

\\texttt{mpthreecolfootgroup} The setup for minipages.

\begin{verbatim}
1988 \newcommand*{\mpthreecolfootgroup}[1]{{\% \\
1989 \vskip\skip\@nameuse{mp#1footins}\% \\
1990 \ifl@dpairing\ifparledgroup\% \\
1991 \leavevmode\marks\parledgroup@{begin}\% \\
1992 \marks\parledgroup@series{#1}\% \\
1993 \marks\parledgroup@type{Xfootnote}\% \\
1994 /fi\fi\normalcolor\\
1995 \ifparledgroup\% \\
1996 \ifl@dpairing\% \\
1997 \else\% \\
1998 \setXnoteswidthliketwocolumns@{#1}\% \\
1999 \setXnotespositionliketwocolumns@{#1}\% \\
2000 \print@Xfootnoterule@{#1}\% \\
2001 \vskip\csuse{afterXrule@#1}\% \\
2002 \fi\% \\
2003 \else\% \\
2004 \setXnoteswidthliketwocolumns@{#1}\% \\
2005 \setXnotespositionliketwocolumns@{#1}\% \\
2006 \print@Xfootnoterule@{#1}\% \\
2007 \vskip\csuse{afterXrule@#1}\% \\
2008 \fi\% \\
2009 {\csuse{Xnotefontsize@#1}\noindent\csuse{txtbeforeXnotes@#1}}\par \\
2010 \splittopskip=\ht\strutbox\\
2011 \expandafter\\
2012 \rigidbalance\csname mp#1footins\endcsname \thr@@ \splittopskip}}
\end{verbatim}

Two columns

\foottwocol

You say \texttt{foottwocol\{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.
24.7 Columnar footnotes

The additional setup for minipages.

\twocolfootsetup  Here is a series of macros which are very similar to their three-column counterparts.
\twocolvfootnote In this case, each note is assumed to contribute only a half a line of text. And the
\twocolfootfmt notes are set in columns giving a gap between them of one tenth of the \hsize.
\twocolfootgroup

\newcommand*{\twocolfootsetup}[1]{% 2014 \newcommand*{\foottwocol}[1]{% 2015 \csgdef{series@display#1}{twocol} 2016 } 2017 \expandafter\let\csname v#1footnote\endcsname=twocolvfootnote 2018 \expandafter\let\csname #1footfmt\endcsname=twocolfootfmt 2019 \dimen\csname #1footins\endcsname=\csuse{maxhXnotes@#1}% 2020 } 2021 \twocolfootsetup{#1} 2022 2023 \twocolfootsetup  \twocolvfootnote  \twocolfootfmt  \twocolfootgroup

\newcommand*{\twocolfootsetup}{% 2024 \count\csname #1footins\endcsname 500 2025 \csxdef{default@#1footins}{500}%Use this to confine the notes to one side only 2026 } 2027 \notbool{parapparatus@}{\newcommand}{\newcommand}{\twocolvfootnote}[2]{\insert\csname #1footins\endcsname\bgroup 2028 } 2029 \footsplitskips 2030 \csname #1footfmt\endcsname #2[#1]\egroup

\notbool{parapparatus@}{\newcommandx*}{\newcommand}{\twocolfootfmt}[4]=[Z]{% 2031 4th arg is optional, for backward compatibility 2032 \normal@pars 2033 \hsize \csuse{hsizetwocol@#4} 2034 \parindent=0pt 2035 \tolerance=5000 2036 \raggedright 2037 } 2038 \hangindent=\csuse{Xhangindent@#4} 2039 \leavevmode 2040 \strut{\printlinefootnote{#1}{#4}}% 2041 \nottoggle{Xlemmadisablefontselection@#4}{\select@lemmafont#1|#2}{#2} 2042 \iftoggle{nosep@}{\hskip\csuse{inplaceoflemmaseparator@#4}}{\ifcsempty{lemmaseparator@#4} 2043 {\hskip\csuse{inplaceoflemmaseparator@#4}}% 2044 {\nobreak\hskip\csuse{beforelemmaseparator@#4}\csuse{lemmaseparator@#4}\hskip\csuse{afterlemmaseparator@#4} 2045 }}% 2046 }\par

\newcommand*{\twocolfootgroup}{% 2047 \newcommand*{\foottwocolgroup}{{\csuse{Xnotefontsize@#1} 2048 \noindent\csuse{txtbeforeXnotes@#1}\par% 2049 \splittopskip=\ht\strutbox 2050 \expandafter\let\csname #1footnotes\endcsname=\tw@ \splittopskip}}}} 2051 \notbool{parapparatus@}{\newcommand}{\newcommand}{\twocolfootsetup}[1]{% 2052 \expandafter\let\csname v#1footnote\endcsname=mpnormalvfootnote 2053 \expandafter\let\csname mp#1footgroup\endcsname=mptwocolfootgroup 2054 } 2055

\mptwocolfootsetup  The versions for minipages.
\mptwocolfootgroup
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 medmac 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 footnotes have their marks separated by commas. This seems such a useful ability that it is provided automatically by eledmac.

\multifootnotemarker These macros may have been defined by the memoir class, are provided by the footmisc package and perhaps by other footnote packages.

\multifootnotemarker\multifootsep

25.2 Causes of Unification

When a user types a \footnote, say \footnote{a footnote}, the \LaTeX processor does not respond. Instead, it inserts a blank line and then the line containing the \footnote. The reason is that these commands are not defined in the \LaTeX kernel. Multiple footnotes are sometimes separated by punctuation, e.g., a comma

\footnote{a comma}.
\m@mmf@prepare A pair of self-cancelling kerns. This may have been defined in the \texttt{memoir} class.
\providecommand*{\m@mmf@prepare}{\kern-\multiplefootnotemarker\kern\multiplefootnotemarker\relax}

\m@mmf@check This may have been defined in the \texttt{memoir} class. If it recognises the last kern as \texttt{\multiplefootnotemarker} it typesets \texttt{\multfootsep}.
\providecommand*{\m@mmf@check}{\ifdim\lastkern=\multiplefootnotemarker\relax
\edef\@x@sf{\the\spacefactor}
\unkern\multfootsep\spacefactor\@x@sf\relax\fi}

We have to modify \texttt{\@footnotetext} and \texttt{\@footnotemark}. However, if \texttt{memoir} is used the modifications have already been made.
\@ifclassloaded{memoir}{}{\@footnotetext Add \texttt{\m@mmf@prepare} at the end of \texttt{\@footnotetext}.
\apptocmd{\@footnotetext}{\m@mmf@prepare}{}{}}

\@footnotemark Modify \texttt{\@footnotemark} to cater for adjacent \texttt{footnotes}.
\renewcommand*{\@footnotemark}{\leavevmode\ifhmode\edef\@x@sf{\the\spacefactor}\m@mmf@check\nobreak\@makefnmark\m@mmf@prepare\ifhmode\spacefactor\@x@sf\fi\relax}

Finished the modifications for the non-memoir case.
\}
\l@doldold@footnotetext
\@footnotetext In order to enable the regular \texttt{footnotes} in numbered text we have to play around with its \texttt{\@footnotetext}, using different forms for when in numbered or regular text.
\pretocmd{\@footnotetext}{\ifnumberedpar@\edtext{}{\l@dbfnote{#1}}\else\fi\m@mmf@prepare\ifhmode\spacefactor\@x@sf\fi\relax}
\apptocmd{\@footnotetext}{\fi}{}{}}}
\l@dbfnote \l@dbfnote adds the footnote to the insert list, and \vl@dbfnote calls the original
\vl@dbfnote \@footnotetext.

2119 \newcommand{\l@dbfnote}[1]{%
2120 \ifnumberedpar@
2121 \gdef\@tag{#1\relax}%
2122 \xright@appenditem{\noexpand\vl@dbfnote{\expandonce\@tag}{\@thefnmark}}%
2123 \global\advance\insert@count @one
2124 \fi
2125 \ignorespaces}
2126 \newcommand{\vl@dbfnote}[2]{%
2127 \def\@thefnmark{#2}%
2128 \@footnotetext{#1}%
2129 \}%

2130 \newcommand*{\normal@footnotemarkX}{%
2131 \leavevmode
2132 \ifhmode
2133 \edef\@x@sf{\the\spacefactor}%
2134 \m@mmf@check
2135 \nobreak
2136 \fi
2137 \newcommand{\postbodyfootmark}{%
2138 \m@mmf@prepare
2139 \ifhmode\spacefactor\@x@sf\fi \relax}
2140 \newcommand*{\normalbodyfootmarkX}{%
2141 \newcommand*{\normalbodyfootmarkX}{\if\series\}
2142 \leavevmode
2143 \ifhmode
2144 \edef\@x@sf{\the\spacefactor}%
2145 \m@mmf@check
2146 \nobreak
2147 \fi}
2148 \newcommand*{\normalvfootnoteX}{%
2149 \insert{\langle\text\rangle}}

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.
\prebodyfootmark Two convenience macros for use by \...@footnotemark... macros.
\postbodyfootmark

\newcommand*{\normal@footnotemarkX}{%
\leavevmode
\ifhmode
\edef\@x@sf{\the\spacefactor}%
\m@mmf@check
\nobreak
\fi}
\newcommand{\postbodyfootmark}{%
\m@mmf@prepare
\ifhmode\spacefactor\@x@sf\fi \relax}
\newcommand*{\normalbodyfootmarkX}{%
\leavevmode
\ifhmode
\edef\@x@sf{\the\spacefactor}%
\m@mmf@check
\nobreak
\fi}
\newcommand*{\normalvfootnoteX}{%
\insert{\langle\text\rangle}}

\normalbodyfootmarkX The \normalbodyfootmarkX{\langle series\rangle} really typesets the in-text marker. The
style is the normal superscript.
\normalvfootnoteX \normalvfootnoteX{\langle series\rangle}{\langle text\rangle} does the \insert for the \langle series\rangle and calls
the series’ \footfmt... to format the \langle text\rangle.
25.2 Footnote formats

\newcommand*{\normalvfootnoteX}[2]{% 
  \insert\nameuse{footins#1}\bgroup 
  \csuse{bhooknoteX@#1} \csuse{notefontsizeX@#1} \footnoteskips 
  \ifl@dpairing\ifl@dpaging\else% 
  \setnotesXwidthliketwocolumns@{#1} \fi\fi% 
  \setnotesXpositionliketwocolumns@{#1} \footsplitskips% 
  \if@RTL\else\noindent\leavevmode\fi\@nameuse{footfmt#1}{#1}{#2}\egroup} 

The minipage version.
\newcommand*{\mpnormalvfootnoteX}[2]{% 
  \global\setbox\@nameuse{mpfootins#1}\vbox{% \unvbox\@nameuse{mpfootins#1} 
  \csuse{bhooknoteX@#1} \csuse{notefontsizeX@#1} \hsize\columnwidth \@parboxrestore \color@begingroup \@nameuse{footfmt#1}{#1}{#2}\color@endgroup}} 

\normalfootfmtX \normalfootfmtX{⟨series⟩}{⟨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\par\strut\par}} 

\normalfootfootmarkX \normalfootfootmarkX{⟨series⟩} is called by \normalfootfmtX to typeset the footnote marker in the footer before the footnote text.
\newcommand*{\normalfootfootmarkX}[1]{% 
  \ifdimequal{0pt}{\prenotesX@}{}% 
  {\iftoggle{prenotesX@}{}{% 
    \textsuperscript{\@nameuse{@thefnmark#1}}}} 

\normalfootstartX \normalfootstartX{⟨series⟩} is the ⟨series⟩ footnote starting macro used in the output routine.
\newcommand*{\normalfootstartX}[1]{% 
  \ifdimequal{0pt}{\prenotesX@}{}% 
  \iftoggle{prenotesX@}{}{%
\normalfootnoteruleX \ The rule drawn before the footnote series group.
\let\normalfootnoteruleX=\footnoterule
\normalfootgroupX \normalfootgroupX\{'\text{series}'\} sends the contents of the \text{\textasciitilde{}series} insert box to the output page without alteration.
\newcommand*{\normalfootgroupX}[1]{\unvbox\@nameuse{footins#1}}
\hsize=\old@hsize%
\normalfootgroupX The minipage version.
\newcommand*{\mpnormalfootgroupX}[1]{\vskip\skip\@nameuse{mpfootins#1}}
\ifl@dpairing\ifparledgroup\leavevmode\marks\parledgroup@{begin}\marks\parledgroup@series{#1}\marks\parledgroup@type{footnoteX}\fi\fi\normalcolor
\ifparledgroup\ifl@dpairing\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
\else\setnotesXwidthliketwocolumns@{#1}\setnotesXpositionliketwocolumns@{#1}\print@footnoteXrule{#1}\vskip\csuse{afterruleX@#1}\fi
Footnote formats

\normalbfnoteX
\newcommand{\normalbfnoteX}[2][]{%
  ifnumberedpar@%
  \protected@edef{thisfootnote}{\csuse{thefootnote#1}}%
  \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\expandonce{\thisfootnote}}}%
  \to\inserts@list
  \global\advance\insert@count \@ne%
  \fi\ignorespaces}
\vbfnoteX
\newcommand{\vbfnoteX}[3][]{%
  \@namedef{@thefnmark#1}{#3}%
  \@nameuse{regvfootnote#1}{#1}{#2}}
\vnumfootnoteX
\newcommand{\vnumfootnoteX}[2][]{%
  ifnumberedpar@
  edtext{}{\normalbfnoteX{#1}{#2}}%
  else
  \@nameuse{regvfootnote#1}{#1}{#2}%
  \fi}
\footnormalX\footnormalX{⟨series⟩} initialises the settings for the ⟨series⟩ footnotes. This should always be called for each series.
\newcommand*{\footnormalX}[1]{%
  \csgdef{series@displayX#1}{normalX}
  \expandafter\let\csname footstart#1\endcsname=\normalfootstartX
  \@namedef{@footnotemark#1}{\normal@footnotemarkX{#1}}
  \@namedef{bodyfootmark#1}{\normalbodyfootmarkX{#1}}
  \expandafter\let\csname regvfootnote#1\endcsname=\normalvfootnoteX
  \expandafter\let\csname vfootnote#1\endcsname=\vnumfootnoteX
  \expandafter\let\csname footfmt#1\endcsname=\normalfootfmtX
  \expandafter\let\csname footfootmark#1\endcsname=\normalfootfootmarkX
  \expandafter\let\csname footgroup#1\endcsname=\normalfootgroupX
  \expandafter\let\csname footnoterule#1\endcsname=\normalfootnoteruleX
  \count\csname footins#1\endcsname=1000
  \csxdef{default@footins#1}{1000}%Use to have note only for one side
  \dimen\csname footins#1\endcsname=\csuse{maxhnotesX@#1}
  \skip\csname footins#1\endcsname=\csuse{beforenotesX@#1}

  Addition for minipages.
  \expandafter\let\csname mpvfootnote#1\endcsname=\mpnormalvfootnoteX
  \expandafter\let\csname mpfootgroup#1\endcsname=\mpnormalfootgroupX
  \count\csname mpfootins#1\endcsname=1000
25.3 Two columns footnotes

The following macros set footnotes in two columns. It is assumed that the length of each footnote is less than the column width.

\foottwocolX \foottwocolX{(series)}
\twocolfootsetupX \twocolfootsetupX{(series)}
\mptwocolfootsetupX \mptwocolfootsetupX{(series)}
\twocolvfootnoteX \twocolvfootnoteX{(series)}
\twocolfootfmtX \twocolfootfmtX{(series)}

\newcommand*{\foottwocolX}[1]{%
\csgdef{series@displayX#1}{twocol}
\expandafter\let\csname regvfootnote#1\endcsname=\twocolvfootnoteX
\expandafter\let\csname footfmt#1\endcsname=\twocolfootfmtX
\expandafter\let\csname footgroup#1\endcsname=\twocolfootgroupX
\dimen\csname #1footins\endcsname=\csuse{maxhnotesX@#1}%
\twocolfootsetupX{#1}
\expandafter\let\csname mpvfootnote#1\endcsname=\mpnormalvfootnoteX
\expandafter\let\csname mpfootgroup#1\endcsname=\mptwocolfootgroupX
\mptwocolfootsetupX{#1}
}

\newcommand*{\twocolfootsetupX}[1]{%
\count\csname footins#1\endcsname 500
\csxdef{default@footins#1}{500}%Use this to confine the notes to one side only
\multiply\dimen\csname footins#1\endcsname by \tw@
}

\newcommand*{\mptwocolfootsetupX}[1]{%
\count\csname mpfootins#1\endcsname 500
\multiply\dimen\csname mpfootins#1\endcsname by \tw@
}

\newcommand*{\twocolvfootnoteX}[2]{%
\insert\csname footins#1\endcsname\bgroup
\csuse{notefontsizeX@#1}
\footsplitskips
\spaceskip=z@skip \xspaceskip=z@skip
\@nameuse{footfmt#1}{#1}{#2}\egroup
}

\newcommand*{\twocolfootfmtX}[2]{%
\protected@edef\@currentlabel{\@nameuse{@thefnmark#1}}%
\normal@pars
\hangindent=\csuse{hangindentX@#1}%
}
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{{\footthreecolX{{\footthreecolX{{\footthreecolX{{{\footthreecolX{}}}}}}}}}}}}
25.5 \textit{Paragraphed footnotes}

The following macros set footnotes as one paragraph.

\footparagraphX \footparagraphX{(\textit{series})}
\para@footsetupX(#1)
\para@footsetupX{(series)}
\newcommand*{\para@footsetupX}[1]{\csuse{notefontsizeX@#1}
\setnotesXwidthliketwocolumns@{#1}\
\dimen0=\baselineskip
\multiply\dimen0 by 1024
\divide\dimen0 by \columnwidth \multiply\dimen0 by \footfudgefiddle\relax%
\expandafter \expandafter \xdef\csname footfudgefactor#1\endcsname{\expandafter\strip@pt\dimen0 }}
\parafootstartX \parafootstartX{(series)}
\newcommand*{\parafootstartX}[1]{\ifdimequal{0pt}{\prenotesX@}{}\
{\iftoggle{prenotesX@}{{\togglefalse{prenotesX@}\skip\csname footins#1\endcsname=\csuse{prenotesX@}}}{}\vskip\skip\@nameuse{footins#1}\leftskip=\z@\rightskip=\z@\parindent=\z@\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}\footsplitskips
\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}
25.5 Paragraphe footnotes

\begin{verbatim}
\newcommand*{\mpara@vfootnoteX}[2]{%\global\setbox\@nameuse{mpfootins#1}\vbox{%
csuse{bhooknoteX@#1}
csuse{notefontsizeX@#1}
\footnoteskips
\setbox0=vbox\hspace=\maxdimen
\noindent\color\@nameuse{footfmt#1}{#1}{#2}\color
\setbox0=\hbox{\unvxh0[#1]}\dp0=\z@
\ht0=\csname footfudgefactor#1\endcsname\wd0
\box0
\penalty0}}
\parafootfmtX \parafootfmtX{(series)}
\newcommand*{\parafootfmtX}{%\protected@edef\@currentlabel{\@nameuse{@thefnmark#1}}%
\insertparafootsep{#1}{}\ledsetnormalparstuff{
\csuse{notenumfontX@#1}\csuse{notenumfontX@#1}\@nameuse{footfootmark#1}\strut\enspace#2\penalty-10}}
\para@footgroupX \para@footgroup{(series)}
\mpara@footgroupX \newcommand*{\mpara@footgroupX}{%\csuse{notefontsizeX@#1}\csuse{notefontsizeX@#1}\@nameuse{footfootmark#1}\strut\enspace#2\penalty-10}}
\begin{verbatim}
\newcommand*{\mpara@vfootnoteX}[2]{%\global\setbox\@nameuse{mpfootins#1}\vbox{%
csuse{bhooknoteX@#1}
csuse{notefontsizeX@#1}
\footnoteskips
\setbox0=vbox\hspace=\maxdimen
\noindent\color\@nameuse{footfmt#1}{#1}{#2}\color
\setbox0=\hbox{\unvxh0[#1]}\dp0=\z@
\ht0=\csname footfudgefactor#1\endcsname\wd0
\box0
\penalty0}}
\parafootfmtX \parafootfmtX{(series)}
\newcommand*{\parafootfmtX}{%\protected@edef\@currentlabel{\@nameuse{@thefnmark#1}}%
\insertparafootsep{#1}{}\ledsetnormalparstuff{
\csuse{notenumfontX@#1}\csuse{notenumfontX@#1}\@nameuse{footfootmark#1}\strut\enspace#2\penalty-10}}
\para@footgroupX \para@footgroup{(series)}
\mpara@footgroupX \newcommand*{\mpara@footgroupX}{%\csuse{notefontsizeX@#1}\csuse{notefontsizeX@#1}\@nameuse{footfootmark#1}\strut\enspace#2\penalty-10}}
\end{verbatim}
\end{verbatim}
\end{verbatim}
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.

\newdimen\old@hsize
\setXnoteswidthliketwocolumns@
\setnotesXwidthliketwocolumns@

\localhsize
\setXnoteswidthliketwocolumns@
\setnotesXwidthliketwocolumns@
These two commands set the position of the critical/familiar footnotes, depending on the hooks \texttt{Xnoteswidthliketwocolumns} and \texttt{notesXwidthliketwocolumns}. They call commands which are defined only in \texttt{eledpar}, because this feature has no sense without \texttt{eledpar}.

\begin{verbatim}
\newcommand{\setXnotespositionliketwocolumns@}[1]{% 
  \iftoggle{Xnoteswidthliketwocolumns@#1}{% 
    \csuse{setnotespositionliketwocolumns\columns@position}% 
  }{% 
  }% 
}
\newcommand{\setnotesXpositionliketwocolumns@}[1]{% 
  \iftoggle{notesXwidthliketwocolumns@#1}{% 
    \csuse{setnotespositionliketwocolumns\columns@position}% 
  }{% 
  }% 
}
\end{verbatim}

27 Footnotes’ order

The \texttt{\fnpos} and \texttt{\mpfnpos} simply place their arguments in \texttt{\@fnpos} and \texttt{\@mpfnpos}, which will be used later in the output routine.

\begin{verbatim}
\def\@fnpos{familiar-critical} 
\def\@mpfnpos{critical-familiar} 
\end{verbatim}

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 \texttt{vbox}.

\begin{verbatim}
\print@Xfootnoterule 
\print@footnoteXrule 
\newcommand{\print@Xfootnoterule}[1]{% 
  \nointerlineskip% 
  \moveleft-\leftskip\vbox{\csuse{#1footnoterule}}}% 
\end{verbatim}
We have to add all the new kinds of familiar footnotes to the output routine. These are the class 1 feet. The normal way to add one series. \texttt{\print@notesX} is replaced by \texttt{eledpar} when using \texttt{Pages}.

We print all the series of notes by looping on them. We check before printing them that there are not voided. We also call \texttt{\correct@footinsX@box} if \texttt{eledpar} is loaded. Read \texttt{eledpar} handbook about this macro.

\addfootinsX Juste for backward compatibility: print a warning message.
30 Endnotes

Endnotes of all varieties are saved up in a file, typically named \texttt{⟨jobname⟩.end}. \texttt{\textbackslash l@d@end} is the output stream number for this file, and \texttt{\textbackslash ifl@dend@} is a flag that’s \texttt{true} when the file is open.

\texttt{\textbackslash l@dend@include} and \texttt{\textbackslash l@dend@close} are the macros that are used to open and close the endnote file. Note that all our writing to this file is \texttt{\textbackslash immediate}: all page and line numbers for the endnotes are generated by the same mechanism we use for the footnotes, so that there’s no need to defer any writing to catch information from the output routine.

\texttt{\newcommand{\textbackslash l@dend@open}[1]{\global\l@dend@true\immediate\openout\l@d@end=#1\relax}}
\texttt{\newcommand{\textbackslash l@dend@close}{\global\l@dend@false\immediate\closeout\l@d@end}}

\texttt{\l@dend@stuff} is used by \texttt{\textbackslash beginnumbering} to do everything that’s necessary for the endnotes at the start of each section: it opens the \texttt{\textbackslash l@d@end} file, if necessary, and writes the section number to the endnote file.

\texttt{\newcommand{\textbackslash l@dend@stuff}{\%}
\texttt{\textbackslash ifl@dend@relax\textbackslash else\textbackslash \textbackslash l@dend@open\textbackslash jobname.end\textbackslash \textbackslash fi}
\texttt{\immediate\write\l@d@end{\string\l@d@section{\the\section@num}}}}

\texttt{\textbackslash endprint} The \texttt{\textbackslash endprint} here is nearly identical in its functioning to \texttt{\textbackslash normalfootfmt}.

The endnote file also contains \texttt{\textbackslash l@d@section} commands, which supply the section numbers from the main text; standard \texttt{eledmac} does nothing with this information, but it’s there if you want to write custom macros to do something with it.

\texttt{\global\notbool{parallelparatus\textbackslash long\def\endprint{\frac{\textbackslash cuse\textbackslash book\textbackslash endnote\textbackslash \#4}{\textbackslash select\textbackslash lemmafont\textbackslash \textbackslash \#2\textbackslash \#2}\textbackslash \enskip}\textbackslash par}}
\texttt{\let\l@d@section=\textbackslash gobble}
The `\setprintendlines` macro is similar to `\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; `\setprintendlines` provides this by always printing the page number. The coding is slightly simpler than `\setprintlines`.

First of all, we print the second page number only if the ending page number is different from the starting page number.

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.
\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@dend@close
\begingroup
\makeatletter
\expandafter\let\csname #1end\endcsname=\endprint
\input\jobname.end
\endgroup}
\noendnotes You can say \noendnotes before the first \beginnumbering in your file 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@dend@stuff=\relax
\global\chardef\l@d@end=16 }

31 Generate series

In this section, X means the name of the series (A, B etc.)
\series \series\series creates one more newseries. It’s the public command, which just
loops on the private command \newseries@.
\newcommand{\newseries}{\newseries@[1]{}%\def\do\#1{\newseries@[#1]}%\do\csvlist[#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.
\newseries@\newcommand{\newseries@}[1]{
31.1 Test if series is still existing
\ifinlist{#1}{\@series}{\led@warn@SeriesStillExist{#1}}%
%
31.2 Init specific to \texttt{eledpar}
When calling \texttt{\newseries@} after having loaded \texttt{eledpar}
\ifdefined\newseries@eledpar%
\newseries@eledpar{#1}%
\fi%

31.3 Create all commands to memorize display options
\newtoggle{Xlemmadisablefontselection@#1}
\newtoggle{Xendlemmadisablefontselection@#1}
\csgdef{Xhangindent@#1}{\opt}%
\csgdef{hangindentX@#1}{\opt}%
\csgdef{Xragged@#1}{}%
\csgdef{raggedX@#1}{}%
\csgdef{hsizetwocol@#1}{0.45 \hsize}%
\csgdef{hsizetwocolX@#1}{0.45 \hsize}%
\csgdef{hsizethreecol@#1}{.3 \hsize}%
\csgdef{hsizethreecolX@#1}{.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}%
\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}%
31.4 Create inserts, needed to add notes in foot

As regards inserts, see chapter 15 of the TeXBook by D. Knuth

31.5 Create commands for critical apparatus, \texttt{Xfootnote}

Note the double \# in command: it’s because command is made inside another command.
31.6 Create tools for familiar footnotes (\footnoteX)

First, create the \footnoteX command.

\begin{macrocode}
\global\expandafter\newcommand\csname footnote#1\endcsname[1]{%
  \begingroup
  \newcommand{\content}{##1}%
  \stepcounter{footnote#1}%
  \protected@csxdef{@thefnmark#1}{\csuse{thefootnote#1}}%
  \nottoggle{nomk@}%Nomk is set to true when using \footnoteXnomk with eledpar
  \csuse{@footnotemark#1}{}%
  \csuse{vfootnote#1}{#1}{\expandonce\content}\m@mmf@prepare%
  \endgroup%}
\end{macrocode}

The counters.

\begin{macrocode}
\newcounter{footnote#1} \global\expandafter\renewcommand\csname thefootnote#1\endcsname{\arabic{footnote#1}}
\end{macrocode}

\end{macrocode}

31.7 The endnotes

The \Xendnote macro functions to write one endnote to the .end file. We change \newlinechar so that in the file every space becomes the start of a new line; this
31.8 Init standards series (A,B,C,D,E,Z)

generally ensures that a long note doesn’t exceed restrictions on the length of lines in files.

\endnote commands called \Xend commands on to the endnote file; these are analogous to the various \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 end command for the series we want to \endprint, and leave the rest equated to \@gobblethree, which just skips over its three arguments.\footnote{Christophe Hebeisen (christophe.hebeisen@epfl.ch) emailed on 2003/11/05 to say he had found that \@gobblethree was also defined in the amsfonts package.}

31.9 Some tools

\firstseries \seriesatbegin\{(s)\} changes the order of series, to put the series \(s\) at the beginning of the list. The series can be the result of a command.
\seriesatend And \seriesatend moves the series to the end of the list.

\newcommand{\seriesatend}[1]{
\edef\series{#1}
\def\new{}
\def\do##1{\ifcsstring{series}{##1}{}{\listadd{\new}{##1}}}
\dolistloop{\@series}
\listeadd{\new}{\series}
\xdef@series{\new}
}
\end{macro}
\end{macro}
% \subsection{Display}
% \changes{v1.0}{2012/09/15}{New generic commands to customize footnote display.}
% \subsection{Options}
% \begin{macro}{\settoggle@series}
% \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.}
% \cs{settoggle@series}\marg{series}\marg{toggle}\marg{value} is a generic command to switch toggles for some series.
\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}}}
\setcommand@series{\settoggle@series}{\series}{\command}{\value} is a generic command to change commands for some series.
\newcommandx{\setcommand@series}[4][4]{\csgdef{#2@##1}{#3}\ifstrequal{#4}{reload}{\csuse{foot\csuse{series@display##1}}{##1}\csuse{foot\csuse{series@displayX##1}}{##1}}{}}
\ifstrempty{#1}{\dolistloop{\@series}}
\newhookcommand@series \command names is a generic command to add new commands for hooks, like \hsizetwocol.

\newhookcommand@series\hangindent
\hangindentX
\ragged
\raggedX
\hsizetwocol
\hsizethreecol
\hsizetwocolX
\hsizethreecolX
\notenumfont
\notenumfontX
\endnotenumfont
\bhooknoteX
\bhookXnote
\bhookXendnote
\notefontsize
\notefontsizeX
\endnotefontsize
\boxlinenum
\boxsymlinenum
\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}[1]{%
  \global\expandafter\newcommand\expandafter*\csname #1\endcsname[2][]{%
    \setcommand@series{##1}{#1}{##2}[reload]%
  }%
}
\newhookcommand@series@reload{beforeXnotes}
\newhookcommand@series@reload{beforenotesX}
\newhookcommand@series@reload{maxhnotesX}
\newhookcommand@series@reload{maxhxnotes}
\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 \newhookcommand@toggle@reload[1]{% 
\global\expandafter\newcommandx\expandafter\*\csname #1\endcsname[2]{% 
\settoggle@series{##1}{#1}{##2}[reload]{% 
}}%
\newhooktoggle@series{Xnoteswidthliketwocolumns}{% 
\newhooktoggle@series{notesXwidthliketwocolumns}{% 
\newhookcommand@toggle@reload{Xlemmadisablefontselection}{% 
\newhookcommand@toggle@reload{Xendlemmadisablefontselection}{% 

31.10 Old commands, kept for backward compatibility

The next commands are kept for ascendant compatibility, but should’nt be used anymore.

\notenumfont \notefontsetup \ifledplinenum \symplinenum \newcommand*{\notenumfont}\{\normalfont\} \newcommand*{\notefontsetup}\{\footnotesize\} \newcommand*{\ifledplinenum}\{\true\} \newcommand*{\symplinenum}\{} 

31.11 Hooks for a particular footnote

\nonum\nonum toggle is used to disable line number printing in a particular footnote.
\nosep\nonum toggle is used to disable the lemma separator in a particular footnote.
3013 \newtoggle{nosep@}

\nomk\nomk toggle is used by eledpar to remove the footnote mark in the text when using \footnoteXmk. Read eledpar handbook.
3014 \newtoggle{nomk@}

### 31.12 Alias
\nolemmaseparator\nolemmaseparator[\langle series\rangle] is just an alias for \lemmaseparator[\langle series\rangle]{\}. 3015 \newcommandx{\nolemmaseparator}{\lemmaseparator[\langle series\rangle]}{\}
\interparanote
\ipn\skip skip and \interparanote command are kept for backward compatibility, but should not be used anymore.
3016 \newskip\ipn@skip
3017 \newcommand*{\interparanote}[1]{\notefontsetup\global\ipn@skip=#1 \relax}
3019 \interparanote{1em plus.4em minus.4em}

\parafootftmsep The \parafootftmsep macro is kept for backward compatibility. It is default value of \parafootsep@series.
3020 \newcommand{\parafootftmsep}{\}

### 31.13 Line number printing
\printlinefootnote The \printlinefootnote macro is called in each \type<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.
3021 \newcommand{\printlinefootnote}[2]{\%}
3022 \edef\extractline@##1|##2|##3|##4|##5|##6|##7|{##2}\%
3023 \edef\extractsubline@##1|##2|##3|##4|##5|##6|##7|{##3}\%
3024 \edef\extractendline@##1|##2|##3|##4|##5|##6|##7|{##5}\%
3025 \edef\extractendsubline@##1|##2|##3|##4|##5|##6|##7|{##6}\%
3026 \iftoggle{numberonlyfirstintwolines@#2}{\%}
3027 \edef\lineinfo@{\extractline@ #1| - \extractsubline@ #1| - \extractendline@ #1| - \extractendsubline@ #1|}\
3028 \hspace{\csuse{inplaceofnumber@#2}}\%
3029 \%\%
3030 \edef\lineinfo@{\extractline@ #1| - \extractsubline@ #1}{\%
3031 \iftoggle{nonumberinfootnote@#2}{\%}
3032 \iftoggle{nonum@}{\%}
3033 \hspace{\csuse{inplaceofnumber@#2}}\%
3034 \%\%
3035 \%\%
3036 \iftoggle{nonumberinfootnote@#2}{\%}
3037 \iftoggle{nonum@}{\%}
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 \printlinefootnote depending on the options about repeating line numbers. The first argument is line information, the second is the notes series (A, B, C etc.)

\printlinefootnotearea

\boxfootnotenumbers Depending on the user settings, this macro will box line numbers (or not). The first argument is line information, the second is the notes series (A, B, C etc.)
Output routine

The previous \printlinefootnotearia calls it.

\newcommand{\boxfootnotenumbers}[2]{\ifdim equal{\csuse{boxlinenum@#2}}{0pt}{\ifboolexpr{\begingroup\boolpass{\togglyph{pstartinfootnote@#2}}{\booltrue{numberpstart}}\or\togglyph{pstartinfootnoteeverytime@#2}}{\printpstart}{}\iftoggle{onlypstartinfootnote@#2}{}{\printlines#1|}}\hbox to \csuse{boxlinenum@#2}{\printlinefootnotenumbers{#1}{#2}}\hfill}}

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 \boxlinefootnotecalls it.

\newcommand{\printlinefootnotenumbers}[2]{\ifboolexpr{\begingroup\boolpass{\togglyph{pstartinfootnote@#2}}{\booltrue{numberpstart}}\or\togglyph{pstartinfootnoteeverytime@#2}}{\printpstart}{}\iftoggle{onlypstartinfootnote@#2}{}{\printlines#1|}}

\printlinefootnotenumbers 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 \boxlinefootnotecalls it.

\countdef\pageno=0 \pageno=1
\newcommand*{\advancepageno}{\ifnum\pageno<\z@ \global\advance\pageno\m@ne \else\global\advance\pageno\@ne\fi}

\pageno is a page number, starting at 1, and \advancepageno increments the number.

\newcommand*{\advancepageno}{\ifnum\pageno<\z@ \global\advance\pageno\m@ne \else\global\advance\pageno\@ne\fi}

\pageno is a page number, starting at 1, and \advancepageno increments the number.

32 Output routine

Now we begin the output routine and associated things.

\newcommand{\pageno}{\countdef\pageno=0 \pageno=1 \newcommand*{\advancepageno}{\ifnum\pageno<\z@ \global\advance\pageno\m@ne \else\global\advance\pageno\@ne\fi}}
The next portion is probably the trickiest part of moving from TeX to \LaTeX. The original code is below, but we need something very different. This is a new output routine, with changes to handle printing all our footnotes. Those changes have not been added directly, but are in macros that get called here: that should make it easier to see what would need to be taken over to a different output routine. We continue to use the \pagebody, \makeheadline, \makefootline, and \dosupereject macros of Plain TeX; for those macros, and the original version of \output, see The TeXbook, p. 364.

\output{$\texttt{edmac@output}$}
\def\edmac@output{$\texttt{shipout}\texttt{vbox}{\texttt{normal@pars}
  \vbox{\texttt{makeheadline}\texttt{pagebody}\texttt{makefootline}}}%
  \advancepageno
  \ifnum\outputpenalty>-\@MM\else\dosupereject\fi}$

\def\pagecontents{$\texttt{page@start}$
  \ifvoid\topins\else\unvbox\topins\fi
  \dimen@=\dp\@cclv \unvbox\@cclv \% open up \texttt{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 centipedal edition; straightforward modifications of EDMAC are all that’s required. However, the myriapedal edition is ruled out by eTeX 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 \% open up \texttt{box255}
      \unvbox \@cclv
      \vskip \skip\footins
      \color@begingroup
      \normalcolor
      \footnoterule
      \unvbox \footins
      \color@endgroup
    }%
  \fi
  \xdef\@freelist{\@freelist\@midlist}\%
  \global \let \@midlist \@empty
  \@combinefloats
Now we start actually changing things.

These macros are defined in the `memoir` class and form part of the definition of `@makecol`.

This is a partitioned version of the `standard` `@makecol`, with the initial code put into another macro.
\texttt{\textbackslash ifFN@bottom} The \texttt{\textbackslash ifFN@bottom} macro is defined by the \texttt{footmisc} package. If this package is not loaded, we define it. 

\texttt{\textbackslash AtBeginDocument{\textbackslash@ifpackageloaded{footmisc}{\textbackslash newif\textbackslash ifFN@bottom}}} 

\texttt{\textbackslash l@ddofootinsert} This macro essentially holds the initial portion of the kernel \texttt{\textbackslash @makecol} code.

\begin{verbatim}
\newcommand*{\l@ddofootinsert}{% 
%%% \page@start 
\ifvoid\footins 
\setbox\@outputbox \box\@cclv 
\else 
\setbox\@outputbox \vbox {% 
\boxmaxdepth \@maxdepth 
\@tempdima\dp\@cclv 
\unvbox \@cclv 
\ifFN@bottom\vfill\fi\vskip \skip\footins%%% If the option bottom of loadmisc package is loaded 
\color@begingroup 
\normalcolor 
\footnoterule 
\unvbox \footins 
\color@endgroup 
} 
\fi }% 

That's the end of the copy of the kernel code. We finally call a macro to handle all the additional ELMAC feet. 
\l@ddoxtrafeet 
\end{verbatim}

\texttt{\textbackslash doxtrafeet} \texttt{\textbackslash doxtrafeet} is the code extending \texttt{\textbackslash @makecol} to cater for the extra ELMAC 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).

\begin{verbatim}
\newcommand*{\l@ddoxtrafeet}{% 
\IfStrEq{familiar-critical}{\@fnpos} 
{\doxtrafeeti\doxtrafeetii}% 
{\doxtrafeetii\doxtrafeeti}% 
{\doxtrafeetii\doxtrafeeti}% 
} 

\doxtrafeetii is the code extending \texttt{\textbackslash @makecol} to cater for the extra critical feet (class 2 feet). \texttt{\textbackslash doxtrafeetii} is the code extending \texttt{\textbackslash @makecol} to cater for the extra critical feet (class 2 feet). \texttt{\textbackslash doxtrafeetii} is the code extending \texttt{\textbackslash @makecol} to cater for the extra critical feet (class 2 feet). \texttt{\textbackslash doxtrafeetii} is the code extending \texttt{\textbackslash @makecol} to cater for the extra critical feet (class 2 feet). \texttt{\textbackslash doxtrafeetii} is the code extending \texttt{\textbackslash @makecol} to cater for the extra critical feet (class 2 feet). \texttt{\textbackslash doxtrafeetii} is the code extending \texttt{\textbackslash @makecol} to cater for the extra critical feet (class 2 feet). \texttt{\textbackslash doxtrafeetii} is the code extending \texttt{\textbackslash @makecol} to cater for the extra critical feet (class 2 feet).
\@opxtrafeetii \unvbox\@outputbox
\@opxtrafeetii\) \unvbox\@outputbox \@opxtrafeetii

The extra critical feet to be added to the output. The normal way to add one
series. \print@Xnotes is replaced by \eledpar when using \Pages.

\newcommand\print@Xnotes[1]{\csuse{#1footstart}{#1}\csuse{#1footgroup}{#1}}\%\)

We print all series of notes by looping on them. We check before printing them that
they are not voided. We also call \correct@Xfootins@box if \eledpar is loaded.
Read \eledpar handbook about this macro.

\newcommand*{\@opxtrafeetii}{\def\do##1{\ifvoid\csuse{##1footins}\else\fi\csuse{##1footgroup}{##1}\insert\csuse{##1footins}{\unvbox\csuse{##1footins}}\%\fi}\dolistloop{\@series}}\%\)

\l@d@reinserts \l@d@reinserts 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 \l@d@doextrafeet.

\newcommand*{\l@d@doextrafeet}{\ifvoid\footins\else\insert\footins{\unvbox\footins}\fi\l@d@reinserts\ifvbox\@kludgeins\insert\@kludgeins{\unvbox\@kludgeins}\fi}\%

\doreinxtrafeetii \doreinxtrafeetii is the code for catering for the class 2 extra critical footnotes
within \reinserts. The implementation may well have to change.

\newcommand*{\doreinxtrafeetii}{\def\do##1{\ifvoid\csuse{##1footins}\else\fi\insert\csuse{##1footins}{\unvbox\csuse{##1footins}}\%\fi}\dolistloop{\@series}}\%

\l@d@reinserts \l@d@reinserts is the modified version of \reinserts.

\gdef \l@d@reinserts{\ifvoid\footins\else\insert\footins{\unvbox\footins}\fi\l@d@doextrafeet\ifvbox\@kludgeins\insert\@kludgeins{\unvbox\@kludgeins}\fi}
The \texttt{memoir} class does not use the \texttt{@makecol} and \texttt{@reinserts}, due to its sidebar insert. We had better add that code if \texttt{memoir} is used. (It can be awkward dealing with \texttt{if} code within \texttt{if} code, so don't use \texttt{if\@d@memoir} here.)

\begin{verbatim}
\@ifclassloaded{memoir}{%
  \texttt{memoir} is loaded so we use \texttt{memoir}'s built in hooks.
  \g@addto@macro{\m@mdoextrafeet}{\l@ddoxtrafeet}%
  \g@addto@macro{\m@mdodoreinextrafeet}{\l@ddodoreinxtrafeet}%
}{%
  \texttt{memoir} has not been loaded, so redefine \texttt{@makecol} and \texttt{@reinserts}.
  \gdef{\@makecol}{\l@d@makecol}%
  \gdef{\@reinserts}{\l@d@reinserts}%
}
\end{verbatim}

\texttt{\addfootins} is for backward compatibility, but shouldn't be used anymore.

\begin{verbatim}
\newcommand*{\addfootins}{[1]{% \\
  \led@warn@AddfootinsObsolete% \\
  \footnormal{#1} \\
  \g@addto@macro{\@opxtrafeetii}{% \\
    \ifvoid{\@nameuse{#1footins}}\else \\
    \@nameuse{#1footstart{#1}}\@nameuse{#1footgroup}{#1}\fi} \\
  \g@addto@macro{\doreinxtrafeetii}{% \\
    \ifvoid{\@nameuse{#1footins}}\else \\
    \insert\@nameuse{#1footins}{\unvbox\@nameuse{#1footins}}\fi} \\
  \g@addto@macro{\l@dedbeginmini}{% \\
    \expandafter\let\csname #1footnote\endcsname = \@nameuse{mp#1footnote}} \\
  \g@addto@macro{\l@dedendmini}{% \\
    \ifvoid{\@nameuse{mp#1footins}}\else\@nameuse{mpfootgroup#1{#1}}\fi} \\
}}
\end{verbatim}

It turns out that \texttt{@doclearpage} also needs modifying.

\begin{verbatim}
\@ifled@nofoot \@led@extranofeet we have to check if there are any leftover feet. \texttt{@led@extranofeet} is a hook for handling further footnotes.
\newif{\@led@extranofeet}
\newcommand*{\@led@extranofeet}{}\end{verbatim}

\begin{verbatim}
\@ifclassloaded{memoir}{%
  If the \texttt{memoir} class is loaded we hook into its modified \texttt{@doclearpage}.
}{%
\gaddto@macro{\@mem@extranofeet}{% \\
  \ifvoid{\csuse{#1footins}}\else\@mem@nofootfalse\fi%
\end{verbatim}
As `memoir` is not loaded we have to do it all here.

\ifvoid\csuse{footins#1}\else\@mernofootfalse\fi%
\dolistloop{\@series}%
\@led@extranofeet}%
\@led@testifnofoot
\@doclearpage
\renewcommand{\@doclearpage}{%}
\if@led@nofoottrue
\ifvoid\footins\else\@led@nofootfalse\fi
\def\do##1{%\ifvoid\csuse{##1footins}\else\@led@nofootfalse\fi%
\ifvoid\csuse{footins##1}\else\@led@nofootfalse\fi}%
\dolistloop{\@series}
\@led@extranofeet%
\renewcommand{\@doclearpage}{%}
\if@led@testifnofoot
\@doclearpage
\if@led@testifnofoot
\if@led@nofoot
\setbox\@tempboxa\vsplit\@cclv to\z@ \unvbox\@tempboxa
\setbox\@tempboxa\box\@cclv
\xdef\@deferlist{\@toplist\@botlist\@deferlist}%
\global \let \@toplist \@empty
\global \let \@botlist \@empty
\global \@colroom \@colht
\ifx \@currlist\@empty
\else
\latexerr{Float(s) lost}@ehb
\global \let \@currlist \@empty
\fi
\fi
\@makefcolumn\@deferlist
\if@twocolumn
\if@firstcolumn
\global \let \@currlist \@empty
\fi
\@makefcolumn\@deferlist
\@whilesw\if@fcolmade \fi{
\global \let \@currlist \@empty
\global \let \@currlist \@empty
\global \@colht \textheight
\begingroup
\@dblfloatplacement
\@makefcolumn\@dbldeferlist
\@whilesw\if@fcolmade \fi{
\@outputpage
\@makefcolumn\@dbldeferlist
\endgroup
\else
\vbox{}
clearpage
\fi
\else
\setbox\@cclv\vbox{\box\@cclv\vfil}%
33 Cross referencing

Peter Wilson has 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 \edlineref 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} % 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 \textbackslash@bsphack and \textbackslash@esphack to eliminate extra space problems and also the LaTeX write methods for the .aux file.

Jesse Billett\footnote{jdb43@cam.ac.uk} via the \texttt{ctt} thread ‘ledmac cross referencing’, 25 August 2003.
3283 \newcommand*{\edlabel}[1]{{}
3284 \ifl@dpairing\ifautopar
3285 \strut
3286 \fi\fi
3287 \@bsphack
3288 \ifledRcol
3289 \write\linenum@outR{\string\@lab}%
3290 \ifx\labelref@listR\empty%
3291 \xdef\label@refs{\zz@@@}%
3292 \else%
3293 \gl@p\labelref@listR\to\label@refs%
3294 \fi%
3295 \ifvmode%
3296 \addvspace\label@refs%
3297 \fi%
3298 \protected@write\@auxout{}
3299 \ifdef{\hypertarget}{\hypertarget{#1}{}}%
3300 \else%
3301 \write\linenum@out{\string\@lab}%
3302 \ifdef{\hypertarget}{\hypertarget{#1}{}}%
3303 \else%
3304 \g@p\labelref@list\to\label@refs%
3305 \fi%
3306 \ifvmode%
3307 \addvspace\label@refs%
3308 \fi%
3309 \protected@write\@auxout{}
3310 \ifdef{\hypertarget}{\hypertarget{#1}{}}%
3311 \else%
3312 \ifdef{\hypertarget}{\hypertarget{#1}{}}%
3313 \else%
3314 \fi%
3315 \@esphack%
3316

Use code from the kernel \label command to write the correct page number (it seems possible that the original EDMAC’s \pagetext scheme might also have had problems in this area). Also define an hypertarget if hyperref package is loaded.

3317 \newcounter{line}%
3318 \newcounter{subline}%
3319 \newcommand{\advancelabel@refs}{%
3320 \setcounter{line}{\expandafter\labelrefsparseline\label@refs}%
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`.

```latex
\newcommand*{\@lab}{\xright@appenditem
  \linenumrep{\line@num}|%
  \ifsublines@ \sublinenumrep{\subline@num}\else 0\fi}
\to\labelref@list
```

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.

```latex
\AtBeginDocument{%
  \def\@lab@list@#1|#2|#3|#4|#5{}%
}%
```

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.

```latex
\AtBeginDocument{%
  \def\@lab@list@#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`.

```latex
\newcommand*{\@lab}{\xright@appenditem
  \linenumrep{\line@num}|%
  \ifsublines@ \sublinenumrep{\subline@num}\else 0\fi}
```
\wrap@edcrossref  \wrap@edcrossref is called around all eledmac crossref commands, except those which start with x. It adds the hyperlink.

\edef\hyperlink{#1}{#2}\%}
3349 \ifdef{\hyperlink}
3350 \{\hyperlink[#1]{#2}\%
3351 \{#2\%
3352 }

\edpageref  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.

\newcommand*{\edpageref}[1]{\l@dref@undefined{#1}\wrap@edcrossref{#1}{\l@dgetref@num{1}{#1}}}
3354 \newcommand*{\xpageref}[1]{\l@dgetref@num{1}{#1}}
3355

\edlineref  If the specified label exists, \elineref gives its line number.

\lineref \newcommand*{\edlineref}[1]{\l@dref@undefined{#1}\wrap@edcrossref{#1}{\l@dgetref@num{2}{#1}}}
\xlineref \AtBeginDocument{\ifdef{\elineref}{\let{\elineref}{\edlineref}}}
3356 \newcommand*{\xlineref}[1]{\l@dgetref@num{2}{#1}}

\sublineref  If the specified label exists, \sublineref gives its sub-line number.

\sublineref \newcommand*{\sublineref}[1]{\l@dref@undefined{#1}\wrap@edcrossref{#1}{\l@dgetref@num{3}{#1}}}
3363 \newcommand*{\xsublineref}[1]{\l@dgetref@num{3}{#1}}

\pstartref  If the specified label exists, \pstartref gives its pstart number.

\pstartref \newcommand*{\pstartref}[1]{\l@dref@undefined{#1}\wrap@edcrossref{#1}{\l@dgetref@num{4}{#1}}}
3366 \newcommand*{\xpstartref}[1]{\l@dgetref@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@dref@undefined  The \l@dref@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@dref@undefined}[1]{\%}
3369 \expandafter\ifx\csname the@label\endcsname\relax
3370 \led@warn@RefUndefined{#1}\%
Next, \l@getref@num fetches the number we want. It has two arguments: the first is simply a digit, specifying whether to fetch a page (1), line (2) or sub-line (3) number. (This switching is done by calling \l@label@parse.) The second argument is the label-macro, which because of the \@lab macro above is defined to be a string of the type 123|456|789.

\newcommand*{\l@getref@num}[2]{% 
\expandafter\ifx\csname the@label#2\endcsname \relax% 
000% 
\else% 
\expandafter\expandafter\expandafter\l@label@parse\csname the@label#2\endcsname|#1% 
\fi}

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@label@parse, which extracts the appropriate 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@label@parse}{% 
def\l@label@parse\csname the@label\#1\endcsname\csname the@label\#2\endcsname\csname the@label\#3\endcsname\csname the@label\#4\endcsname\csname the@label\#5\endcsname\relax% \ifcase \#5% 
\or \#1% 
\or \#2% 
\or \#3% 
\or \#4% 
\or \#5% 
\fi}

The \xxref command takes two arguments, both of which are labels, e.g., \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 \linenum and sets the beginning page, line, and sub-line numbers to those of the place where \label{mouse} was placed, and the ending numbers to those at \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 \critext for one reason or another. Using \xxref in the second argument of \critext lets you set things up at least semi-automatically.

\newcommand*{\xxref}[2]{% 
\expandafter\ifx\csname the@label#1\endcsname \relax% 
\expandafter\let\csname the@label#1\endcsname\zz0000% 
\else%
Sometimes the \edlabel{} command cannot be used to specify exactly the page and line desired; you can use the \edmakenewlabel{} macro to make your own label.

For example, if you say \edmakenewlabel{}{elephant}{10|25|0} you will have created 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. \edmakenewlabel{} takes a label, followed by a page and a line number(s) as arguments. LaTeX defines a \makelabel{} macro which is used in lists. I’ve changed the name to \edmakenewlabel{}.

\newcommand*{\edmakenewlabel}{[]}{\expandafter\xdef\csname the@label#1\endcsname{#2}}

(If you are only going to refer to such a label using \xxref{}, then you can omit entries in the same way as with \linenum{} (see pp. 84 and 62), since \xxref{} makes a call to \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 \xympars{} 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 \linenum@margin{} 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

\newcount\sidenote@margin
\newcommand*{\sidenotemargin}{%\l@dgetsidenote@margin{#1}%%}
\ifnum\@l@dtempcntb>\m@ne
\global\sidenote@marginR=\@l@dtempcntb
\else
\global\sidenote@margin=\@l@dtempcntb
\fi}
\newcommand*{\l@dgetsidenote@margin}{%\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@BadSidenotemargin
\@l@dtempcntb \m@ne
\fi
\fi
\fi
\fi}
\sidenotemargin{right}

\l@dlp@rbox \l@drp@rbox We need two boxes to store sidenote texts.
\ledlnotewidth These specify the width of the left/right boxes (initialised to \marginparwidth, their distance from the text (initialised to \linenumsep, and the fonts used.

\ledlnotesep\newdimen\ledlnotesep \ledlnotesep=\marginparwidth
\ledrsnotesep\newdimen\ledrsnotesep \ledrsnotesep=\marginparwidth
\ledlnotefontsetup\newdimen\ledlnotefontsetup \ledlnotefontsetup=\linenumsep
\ledrsnotefontsetup\newdimen\ledrsnotefontsetup \ledrsnotefontsetup=\linenumsep

These specify the width of the left/right boxes (initialised to \marginparwidth, their distance from the text (initialised to \linenumsep, and the fonts used.

\ledlnotesep\newdimen\ledlnotesep \ledlnotesep=\marginparwidth
\ledrsnotesep\newdimen\ledrsnotesep \ledrsnotesep=\marginparwidth
\ledlnotefontsetup\newdimen\ledlnotefontsetup \ledlnotefontsetup=\linenumsep
\ledrsnotefontsetup\newdimen\ledrsnotefontsetup \ledrsnotefontsetup=\linenumsep

\newcommand*{\ledlnotefontsetup}{\raggedleft\footnotesize}
\newcommand*{\ledrsnotefontsetup}{\raggedright\footnotesize}

\ledleftnote, \ledrightnote, \ledinnernote, \ledouternote 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.

\newcommand*{\ledsidenote}[1]{\edtext{}{\l@dcsnote{#1}}}
\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%
\l@dlsnote{#1}\
\else%
\l@drsnote{#1}\
\fi%}
\newcommand*{\ledouternote}[1]{\ifodd\c@page% Do not use \page@num, because it is not yet calculated when command is called%
\l@drsnote{#1}\
\else%
\l@dlsnote{#1}\
\fi%}

\l@dlsnote . The ‘footnotes’ for left, right, and moveable sidenotes. The whole scheme is
reminiscent of the critical footnotes code.
\l@drsnote
\newif\ifrightnoteup
\newcommand*{\l@dlsnote}[1]{%\ifodd\c@page% Do not use \page@num, because it is not yet calculated when command is called%
\begingroup%
\newcommand{\content}{#1}\
\ifnumberedpar@
\ifledRcol%
\xright@appenditem{\noexpand\vl@dlsnote{\expandonce\content}}%\to\inserts@listR
\global\advance\insert@countR 1\else%
\l@dlsnote{#1}\
\fi%}
\newcommand*{\l@drsnote}[1]{%\ifodd\c@page% Do not use \page@num, because it is not yet calculated when command is called%
\begingroup%
\newcommand{\content}{#1}\
\ifnumberedpar@
\l@dlsnote{#1}\
\else%
\l@drsnote{#1}\
\fi%
\vl@dlsnote\vl@drsnote\vl@dcsnote

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:
  
  – _\@dcsmnotetext_ if _\@sidenote_ is to be put on left.
  – _\@dcsmnotetext@l_ if _\@sidenote_ is to be put on right.

\newcommand*{\vl@dlsnote}[1]{%
  \def\l@dcsnotetext{\l@dcsnotetext}
  \ifledRcol@
    \edef\@l@dtempcntb{\sidenote@marginR}
    \ifnum\@l@dtempcntb>\@ne
      \advance\@l@dtempcntb by\page@numR
    \fi
    \else
      \edef\@l@dtempcntb{\sidenote@margin}
      \ifnum\@l@dtempcntb>\@ne
        \advance\@l@dtempcntb by\page@num
      \fi
    \fi
    \ifodd\@l@dtempcntb
      \listgadd{\l@dcsnotetext@l}{#1}
    \else
      \listgadd{\l@dcsnotetext}{#1}
    \fi
  \fi%
}
\newcommand*{\vl@drsnote}[1]{%
  \def\l@dcsnotetext{\l@dcsnotetext}
  \ifledRcol@
    \edef\@l@dtempcntb{\sidenote@marginR}
    \ifnum\@l@dtempcntb>\@ne
      \advance\@l@dtempcntb by\page@numR
    \fi
    \else
      \edef\@l@dtempcntb{\sidenote@margin}
      \ifnum\@l@dtempcntb>\@ne
        \advance\@l@dtempcntb by\page@num
      \fi
    \fi
    \ifodd\@l@dtempcntb
      \listgadd{\l@dcsnotetext@r}{#1}
    \else
      \listgadd{\l@dcsnotetext}{#1}
    \fi
  \fi%
}
\newcommand*{\vl@dcsnote}[1]{\listgadd{\l@dcsnotetext}{#1}}
\setl@dlp@rbox
\setl@drpr@box
\setl@dlprbox{⟨lednums⟩}{⟨tag⟩}{⟨text⟩} puts ⟨text⟩ into the \l@dlp@rbox box.
\setl@drprbox And similarly for the right side box. It is these boxes that finally get displayed in the margins.
\ifleftnoteup
  =\vbox to\z@{\vss #1}\
\else
  =\vbox to 0.70\baselineskip{\strut#1\vss}\
\fi}
\newcommand*{\setl@drp@rbox}[1]{\parindent\z@\hsize=\ledrsnotewidth\ledrsnotefontsetup\global\setbox\l@drp@rbox\ifrightnoteup
  =\vbox to\z@{\vss#1}\
\else
  =\vbox to0.7\baselineskip{\strut#1\vss}\
\fi}
\newif\ifleftnoteup
\leftnoteuptrue
\sidenotesep
This macro is used to separate sidenotes of the same line.
\newcommand{\sidenotesep}{, }
\affixside@note
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.
\newcommand*{\affixside@note}{\def\sidenotecontent@{}\numgdef{\itemcount@}{0}\def\do##1{\ifnumequal{\itemcount@}{0}{{\appto\sidenotecontent@{##1}}}% Not print not separator before the 1st note{\appto\sidenotecontent@{\sidenotesep ##1}}%}{\numgdef{\itemcount@}{\itemcount@+1}]]}}\dolistloop{\l@dcsnotetext}\
\ifnumgreater{\itemcount@}{1}{\led@err@ManySidenotes}{}% And we do the same for left and right notes (not movable).
\gdef@templ@d{}\gdef@templ@n{\l@dcsnotetext\l@dcsnotetext@l\l@dcsnotetext@r}\ifx@templ@d@templ@n \else\if@twocolumn\if@firstcolumn\setl@dlp@rbox{##1}{\sidenotecontent@}%\else\setl@drp@rbox{\sidenotecontent@}%\fi\else\fi\fi\else\fi\fi\if0\tempcntb=\sidenote@margin
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.

These will be the hooks in \@iiiminipage and \endminipage They can be extended to handle other things if necessary.

These handle the initiation and closure of critical footnotes in a minipage environment.
\l@dedendmini\newcommand*{\l@dedendmini}{%\ifl@dpairing\ifledRcol\flush@notesR\else\flush@notes\fi\fi\def\do##1{\ifvoid\csuse{mp##1footins}\else\ifl@dpairing\ifparledgroup\ifledRcol\dimgdef{\parledgroup@beforenotesR}{\parledgroup@beforenotesR+\skip\@nameuse{mp##1footins}}\else\dimgdef{\parledgroup@beforenotesL}{\parledgroup@beforenotesL+\skip\@nameuse{mp##1footins}}\fi\fi\fi%\def\do##1{\csletcs{vfootnote##1}{mpvfootnote##1}}%\dolistloop{\@series}}\l@dfambeginmini\l@dfamendmini

These handle the initiation and closure of familiar footnotes in a minipage environment.

\l@dfambeginmini\newcommand*{\l@dfambeginmini}{%\def\do##1{\csletcs{vfootnote##1}{mpvfootnote##1}}%\dolistloop{\@series}}\newcommand*{\l@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 \texttt{ltboxes.dtx}.

\l@dfabeginmini\l@dfeetbeginmini% added

\l@dfabeginmini\l@dfeetbeginmini% added

\l@dfabeginmini\l@dfeetbeginmini% added

The next line is our addition to the original.
\endminipage  This is our extended form of the kernel \endminipage defined in \texttt{ltboxes.dtx}.

\begin{verbatim}
\def\endminipage{% 
  \par
  \unskip
  \ifvoid\@mpfootins\else
    \l@dunboxmpfoot
  \fi
  \l@dfeetendmini% added
  \@minipagefalse
  \color@endgroup
  \egroup
  \expandafter\@iiiparbox\@mpargs{\unvbox\@tempboxa}
  \l@dunboxmpfoot
\end{verbatim}

\l@dunboxmpfoot

\texttt{\newcommand*{\l@dunboxmpfoot}{\% 
  \vskip\skip\@mpfootins 
  \normalcolor
  \footnoterule
  \ifparledgroup
    \ifl@dpairing
      \ifledRcol
        \dimgdef{\parledgroup@beforenotesR}{\parledgroup@beforenotesR+\skip\@mpfootins}
      \else
        \dimgdef{\parledgroup@beforenotesL}{\parledgroup@beforenotesL+\skip\@mpfootins}
      \fi
    \fi
  \fi
  \fi
  \unvbox\@mpfootins}}}

ledgroup  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{verbatim}
\newenvironment{ledgroup}{% 
  \resetprevpage@num
  \def\@mpfn{mpfootnote}\def\thempfn{\thempfootnote}\c@mpfootnote\z@
  \let\@footnotetext\@mpfootnotetext
  \l@dfeetbeginmini%
}{{% 
  \par
  \unskip
  \ifvoid\@mpfootins\else

\end{verbatim}
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\{"width\}\end{ledgroupsized}

These boolean tests check if we are in the notes of a ledgroup. If we are, we don’t number the lines.
36 Indexing

Here's some code for indexing using page & line numbers.

First, ensure that imakeidx or indextools is loaded before eledmac.

```latex
\newif\ifledgroupnotesL@
\newif\ifledgroupnotesR@
\AtBeginDocument{\unless\ifl@imakeidx\@ifpackageloaded{imakeidx}{\led@error@ImakeidxAfterEledmac}{}\fi\unless\ifl@indextools\@ifpackageloaded{indextools}{\led@error@indextoolsAfterEledmac}{}\fi}
\newcommand{\pagelinesep}{-}
\newcommand{\edindexlab}{$&}
\newcounter{labidx}
\setcounter{labidx}{0}
\newcommand{\doedindexlabel}{\stepcounter{labidx}\edlabel{\edindexlab\thelabidx}}
\newcommand{\thepageline}{\thepage\pagelinesep\xlineref{\edindexlab\thelabidx}}
\newcommand{\thestartpageline}{\l@dparsedstartpage\pagelinesep\l@dparsedstartline}
\newcommand{\theendpageline}{\l@dparsedendpage\pagelinesep\l@dparsedendline}
\if@edindex@fornote@true
\prepare@edindex@fornote
\newif\if@edindex@fornote@
\begin@edindex@fornote
\newcommand{\prepare@edindex@fornote}{\l@dp@rsefootspec#1\edindexlab\l@dparsedstartpage\pagelinesep\l@dparsedstartline\edindexlab\l@dparsedendpage\pagelinesep\l@dparsedendline\edindexlab\l@dparsedstartline\pagelinesep\l@dparsedstartline\edindexlab\l@dparsedendline\pagelinesep\l@dparsedendline}
\if@edindex@fornote@true
\prepare@edindex@fornote
\begin@edindex@fornote
\newcommand{\prepare@edindex@fornote}{\l@dp@rsefootspec#1}
\end@edindex@fornote
```

In order to get a correct line number we have to use the label/ref mechanism. These macros are for that.

- \pagelinesep
- \edindexlab
- \setcounter{labidx}{0}
- \doedindexlabel
- \thepageline
- \thestartpageline
- \theendpageline

This macro sets an \edlabel.

This macro makes up the page/line number combo from the label/ref.

These macros make up the page/line start/end number when the \edindex command is called in critical notes.

This boolean test is switching at the beginning of each critical note, to allow indexing in this note.

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.
36.1 Memoir compatibility

This macro is used to analyse if a text to be indexed has a command after a |.

3787 \def\get@index@command#1|#2+{%  
3788 \gdef\@index@txt{#1}  
3789 \gdef\@index@command{#2}  
3790 \edef\index@parenthesis{}  
3791 \IfBeginWith{\@index@command}{(}{}%  
3792 \StrGobbleLeft{\@index@command}{1}[\@index@command]\%  
3793 \global\let\@index@command\@index@command@%  
3794 \xdef\@index@parenthesis{}%  
3795 \}{}%  
3796 \IfBeginWith{\@index@command}{)}{}%  
3797 \StrGobbleLeft{\@index@command}{1}[\@index@command]\%  
3798 \global\let\@index@command\@index@command@%  
3799 \xdef\@index@parenthesis{}%  
3800 \}{}%  
3801 \}

These macros are used to specify that an index reference points to a note.

3802 \newcommand{\ledinnote}[2]{\csuse{#1}{#2\textit{n}}}  
3803 \newcommand{\ledinnotehyperpage}[2]{\csuse{#1}{\hyperpage{#2}\textit{n}}}  

The memoir class provides more flexible indexing than the standard classes. We need different code if the memoir class is being used, except if \imakeidx or \indextools is used.

36.1 Memoir compatibility

\create@edindex@for@memoir define the \edindex command and related tool when:

1. Memoir class is used.
2. AND \imakeidx is not used.
3. AND \indextools is not used.

Need to add the definition of \edindex to \makeindex, and initialise \edindex to do nothing. In this case \edindex has an optional argument. We use the hook provided in memoir v1.61.

3804 \def\create@edindex@for@memoir{  
3805 \g@addto@macro{\makememindexhook}{%  
3806 \def\edindex{\@bsphack%  
3807 \l@d@index{[\jobname]}\%  
3808 \edef\edindex{\@bsphack\@esphack}

\l@d@index is the first stage of \edindex, handling the idx file. This a virtually a verbatim copy of memoir's \index, the change being calling \l@d@wrindex@m instead of \@wrindex@m.
\l@d@indexm{item} writes the idx file name and the indexed item to the \l@d@wrindexhyp aux file. These are almost verbatim copies of memoir's \windexm{item} and \windexhyp.

\newcommand{\l@d@wrindexm}{\l@d@wrindexm{item}}{\l@d@wrindexhyp}

% \l@d@wrindexm{item} writes the idx file name and the indexed item to the \l@d@wrindexhyp aux file. These are almost verbatim copies of memoir's \windexm{item} and \windexhyp.

\newcommand{\l@d@wrindexm}[1]{\l@d@wrindexhyp##1||\}
\def\l@d@wrindexhyp##1|##2|##3\{|%
36.2 Normal setting

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.

\def\create@edindex@notfor@memoir{
\newcommandx{\@wredindex}[2][1=\expandonce{\jobname},usedefault]{% #1 = the index name, #2 = the text
\global\let\old@Rlineflag@Rlineflag% \gdef\Rlineflag{}% \if\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|+}% \expandafter\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% \else% \IfSubStr[1]{##2}{1}{\{\get@index@command##2+}{\get@index@command##2|+}% \expandafter\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%
36.3 Choose the right variant

Then call \createindexformemoir or \createindexnotformemoir depending on the use of memoir and imakeidx.

36.4 Hyperref compatibility

\hyperlinkformat command is to be used to have both a internal hyperlink and a format, when indexing.

\hyperlinkR command is to be used to create a internal hyperlink and \indexflag, when indexing.
\hyperlinkformatR command is to be used to create an internal hyperlink, a format and a Rlineflag, when indexing.

newcommand{\hyperlinkformatR}{% \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
      #1
      \catcode\' =10 % space need for catcode
    }%
    \IfSubStr{\temp@}{|} {%
      \get@index@command#1+% 
      \ifledRcol% 
        \gdef@edindex@hyperref{|\index@parenthesis %space kept \hyperlinkformatR{\index@command}% \edindexlab\thelabidx}% 
      \else% 
        \gdef@edindex@hyperref{|\hyperlinkR{\edindexlab\thelabidx}}% 
      \fi% 
    }% 
    \else% 
      \gdef\edindex@hyperref{\index@command}% 
      \edindexlab\thelabidx%
    \fi%
  }% 
  \get@index@command\index@command#1+% 
  \ifledRcol% 
    \gdef@edindex@hyperref{\hyperlinkR{\edindexlab\thelabidx}}% 
  \else% 
    \gdef@edindex@hyperref{\hyperlink{\edindexlab\thelabidx}}% 
  \fi%
  \fi% 

\ifdef{\hyperlink} {
  \edef\temp@ {%
    \catcode\' =9 %space need for catcode
    #1
    \catcode\' =10 % space need for catcode
  }%
  \IfSubStr{\temp@}{|} {%
    \get@index@command#1+% 
    \ifledRcol% 
      \gdef@edindex@hyperref{|\index@parenthesis %space kept \hyperlinkformatR{\index@command}% \edindexlab\thelabidx}% 
    \else% 
      \gdef@edindex@hyperref{|\hyperlinkR{\edindexlab\thelabidx}}% 
    \fi% 
  }% 
  \else% 
    \gdef\edindex@hyperref{\index@command}% 
    \edindexlab\thelabidx%
  \fi%

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 \texttt{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 \texttt{amsgen} package.

\l@denvbody A token register to contain the body.

\addtol@denvbody \ \texttt{\addtol@denvbody\{arg\}} adds \texttt{arg} to the token register \texttt{\l@denvbody}.

\l@dcollect@body The macro \texttt{\l@dcollect@body} starts the scan for the \texttt{\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 \texttt{#1}, then the environment form, \texttt{\begin{env}} would call \texttt{\l@dcollect@body\{cenv\}}.

\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.
the \end and its argument in the material we are adding to the environment body accumulator.

There was a question on CTT about how to use \collect@body for a macro taking an argument. The following is part of that thread.

From: Heiko Oberdiek <oberdiek@uni-freiburg.de>
Newsgroups: comp.text.tex
Subject: Re: Using \collect@body with commands that take >1 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}[1]{\colorbox{red}{#1}}
> > \makeatletter
> > \newenvironment{redbox}{\collect@body{\redbox}}{}
>
You will get an error message: Command \redbox already defined.
Thus you must rename either the command \redbox or the environment name.

> \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}[1]{\colorbox{red}{#1}}
\makeatletter
\newenvironment{coloredbox}[1]{\def\next\colorbox{#1}\ignorespaces\unskip}{\end\collect@body\next\ignorespaces\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

\end{document}
\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 [Sul92].

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:

\begin{verbatim}
\renewcommand{\hangingsymbol}{[,}
\end{verbatim}

The \ifinstanza boolean is used to be sure that we are in a stanza part.

\begin{verbatim}
\hangingsymbol
\ifinstanza
\newcommand*{\hangingsymbol}{}
\end{verbatim}

\begin{verbatim}
\newcommand*{\hangingsymbol}{[,}
\ifinstanza
\newif\instanza
\instanzafalse
\newcommand{\inserthangingsymbol}{%}
\fi%
\fi%
\end{verbatim}

The boolean \ifinserthangingsymbol is set to TRUE when \texttt{@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 \texttt{do@line} before the printing of line but after the line number calculation.

\begin{verbatim}
\newcommand*{\inserthangingsymbol}{%}
\newcommand{\inserthangingsymbol}{%}
\fi%
\fi%
\end{verbatim}

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.

\begin{verbatim}
\newcommand*{\ampersand}{\char‘&}
\end{verbatim}

Before we can define the main macros we need to save and reset some category codes. To save the current values we use \texttt{next} and \texttt{body} from the \texttt{loop} macro.

\begin{verbatim}
\newcommand*{\ampersand}{\char‘&}
\end{verbatim}
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 \texttt{\stanzaindentbase} is 20pt.

\begin{verbatim}
\newcount\stanza@count
\newlength{\stanzaindentbase}
\setlength{\stanzaindentbase}{20pt}
\strip@szacnt
\setstanzavalues
\end{verbatim}

The indentations of stanza lines are non-negative integer multiples of the unit called \texttt{\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 \texttt{\mathchardef}. Though this does limit the range from 0 to 32767, it should suffice for most applications, including \texttt{penalties}, which will be discussed below.

\begin{verbatim}
\def\strip@szacnt#1,#2|{\def\@tempb{#1}\def\@tempa{#2|}}
\newcommand*{\setstanzavalues}[2]{\def\@tempa{#2,,|}\stanza@count\z@
ext\strip@szacnt\@tempa\@tempb
\ifx\@tempb\empty\let\next\relax\else\mathchardef\csname #1@\number\stanza@count\endcsname\@tempb\relax\advance\stanza@count\@ne\fi\next}\
\next}
\end{verbatim}

\begin{verbatim}
\setstanzaindents
\setstanzapenalties
\managestanza@modulo
\end{verbatim}

In the original \texttt{\setstanzavalues{sza}{...}} had to be called to set the indents, and similarly \texttt{\setstanzavalues{szp}{...}} to set the penalties. These two macros are a convenience to give the user one less thing to worry about (mis-spelling the first argument). Since version 0.13, the \texttt{stanzaindentsrepetition} counter can be used when the indentation is repeated every n verses. The \texttt{\managestanza@modulo} is a command which modifies the counter \texttt{stanza@modulo}. The command adds 1 to \texttt{stanza@modulo}, but if \texttt{stanza@modulo} is equal to the \texttt{stanzaindentsrepetition} counter, the command restarts it.

\begin{verbatim}
\newcommand*{\setstanzaindents}[1]{\setstanzavalues{sza}{#1}}
\newcommand*{\setstanzapenalties}[1]{\setstanzavalues{szp}{#1}}
\newcommand*{\managestanza@modulo}{\let\next\relax\strip@szacnt#1,#2|\@tempb\def\@tempa{#2,,|}\stanza@count\z@
ext\strip@szacnt\@tempa\@tempb\ifx\@tempb\empty\let\next\relax\else\mathchardef\csname #1@\number\stanza@count\endcsname\@tempb\relax\advance\stanza@count\@ne\fi\next}
\newcommand*{\setstanzaindentsrepetition}[1]{\let\next\relax\strip@szacnt#1,#2|\@tempb\def\@tempa{#2,,|}\stanza@count\z@
ext\strip@szacnt\@tempa\@tempb\ifx\@tempb\empty\let\next\relax\else\mathchardef\csname #1@\number\stanza@count\endcsname\@tempb\relax\advance\stanza@count\@ne\fi\next}
\end{verbatim}
The macro \texttt{\stanzaindent}, when called at the beginning of a verse, changes the indentation normally defined for this verse by \texttt{\setstanzaindent}. The starred version skips the current verse for the repetition of stanza indent.

\begin{verbatim}
\newcommand{\stanzaindent}[1]{\hspace{\dimexpr#1\stanzaindentbase-\parindent\relax}\ignorespaces}
\WithSuffix\newcommand\stanzaindent*[1]{\stanzaindent{#1}\global\advance\stanza@modulo-\@ne\ifnum\stanza@modulo=0\global\stanza@modulo=\value{stanzaindentsrepetition}\fi\ignorespaces}
\end{verbatim}

Now we arrive at the main works. \texttt{\stanza@line} sets the indentation for the line and starts a numbered paragraph—each line is treated as a paragraph. \texttt{\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. \texttt{\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.

\begin{verbatim}
\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\managestanza@modulo\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@}
\end{verbatim}

Now we have the components of the \texttt{\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 \endstanzaextra 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 begin-
nning 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 \texttt{\stanzaindentbase}.

\begin{Verbatim}
\newcommand*{\flagstanza}[2][\stanzaindentbase]{% 
  \hskip -#1\llap{#2}\hskip #1\ignorespaces}
\end{Verbatim}

The ampersand & is used to mark the end of each stanza line, except the last, which is marked with \&. This means that \texttt{\halign} may not be used directly within a stanza line. This does not affect macros involving alignments defined outside \texttt{\stanza \&}. Since these macros usurp the control sequence \&, the replacement \texttt{\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.

\begin{Verbatim}
\catcode'&=\next
\catcode'@=\body
\% % let\ampersand=\&
\setstanzavalues{szp}{0}
\end{Verbatim}

\section{39 Arrays and tables}

This is based on the work by Herbert Breger in developing \texttt{tabmac.tex}.

\begin{verbatim}
% This is file tabmac.tex 1.0.
% You find here macros for tabular structures compatible with
% Edmac (authored by Lavagnino/Wujastyk). The use of the macros is
% explained in German language in file tabanlei.dvi. The macros were
% developed for Edmac 2.3, but this file has been adjusted to Edmac 3.16.
% ATTENTION: This file uses some Edmac control sequences (like
% \text, \Afootnote etc.) and redefines \morenoexpands. If you yourself
% redefined some Edmac control sequences, be careful: some adjustments
% might be necessary.
% October 1996
%
% My kind thanks to Nora G"{o}deke for valuable support. Any hints and
% comments are welcome, please contact Herbert Breger,
% Leibniz-Archiv, Waterloostr. 8, D -- 30169 Hannover, Germany
% Tel.: 511 - 1267 327

The original \texttt{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@dtabnoexpands An extended and modified version of the original additional no expansions..
\newcommand*{% 
let rtab=0
let ctab=0
let ltab=0
let rtabtext=0
let ltabtext=0
let ctabtext=0
let edbeforetab=0
let edaftertab=0
let edatab=0
let edatabell=0
let edatleft=0
let edatright=0
let edvertline=0
let edvertdots=0
let edrowfill=0
}

\disable@familiarnotes
\restore@familiarnotes
Macros to disable and restore familiar notes, to prevent them from printing multiple times in edtabularx and edarrayx environments.

\newcommand{\disable@familiarnotes}{% 
def\do##1{% 
\csletcs{footnote@@##1}{footnote##1}% \expandafter\renewcommand \csname footnote##1\endcsname[1]{% \protected@csxdef{@thefnmark##1}{\csuse{thefootnote##1}}\csuse{@footnotemark##1} % }% \dolistloop{\@series}% }% 
\newcommand{\restore@familiarnotes}{% \def\do##1{% \csletcs{footnote##1}{footnote@@##1}% }% \dolistloop{\@series}% }

\disable@sidenotes\restore@sidenotes
The same, for side notes.

\newcommand{\disable@sidenotes}{% \let\@@ledrightnote\ledrightnote\let\@@ledleftnote\ledleftnote\let\@@ledsidenote\ledsidenote\let\ledrightnote\@gobble\let\ledleftnote\@gobble\let\ledsidenote\@gobble\}\dolistloop{\@series}% 
\newcommand{\restore@sidenotes}{% \let\ledrightnote\@@ledrightnote\let\ledleftnote\@@ledleftnote\let\ledsidenote\@@ledsidenote\}\dolistloop{\@series}%

\let\ledleftnote\@\ledleftnote\%
\let\ledsidenote\@\ledsidenote\%
\disable@notes \ Disable/restore side and familiar notes.
\restore@notes  \restore@notes\% 
\newcommand{\disable@notes}{\disable@sidenotes\% \disable@familiarnotes\%}
\newcommand{\restore@notes}{\restore@sidenotes\% \restore@familiarnotes\%}
\l@dampcount    \l@dampcount is a counter for the & column dividers and \l@dcolcount is a
\l@dcolcount    \l@dcolcount counter for the columns. These were \ Undcount and \stellencount respectively.
\newcount\l@dampcount\l@dampcount=1\relax
\newcount\l@dcolcount\l@dcolcount=0\relax
\hilfsbox Some (temporary) helper items.
\hilfsskip \newbox\hilfsbox
\hilfsbox \newskip\hilfsskip
\hilfscount \newbox\hilfsbox
\hilfscount \newcount\hilfscount\hilfscount
\hilfscount
\hilfscount

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).
\newdimen\dcoli \newdimen\dcoli
\newdimen\dcolii \newdimen\dcolii
\newdimen\dcoliii \newdimen\dcoliii
\newdimen\dcoliv \newdimen\dcoliv
\newdimen\dcolv \newdimen\dcolv
\newdimen\dcolvi \newdimen\dcolvi
\newdimen\dcolvii \newdimen\dcolvii
\newdimen\dcolviii \newdimen\dcolviii
\newdimen\dcolix \newdimen\dcolix
\newdimen\dcolx \newdimen\dcolx
\newdimen\dcolxi \newdimen\dcolxi
\newdimen\dcolxii \newdimen\dcolxii
\newdimen\dcolxiii \newdimen\dcolxiii
\newdimen\dcolxiv \newdimen\dcolxiv
\newdimen\dcolxv \newdimen\dcolxv
\newdimen\dcolxvi \newdimen\dcolxvi
\newdimen\dcolxvii \newdimen\dcolxvii
This is a cunning way of storing the columnwidths indexed by the column number, like an array. (was \herumordnung)

\set@dcolcount
This increments the column counter, and issues an error message if it is too large.

\set@dsetmaxcolwidth
Sets the column width to the maximum value seen so far. (was \herumordnung)

\EDTEXT
We need to be able to modify the \edtext and \critext macros and also restore their original definitions.
We need to be able to modify and restore the \texttt{edlabel} macro.

Macros supporting modification and restoration of \texttt{edindex}.

Macro supporting restoration of \texttt{linenum}.

\texttt{l@dgobbledarg} replaces its delineated argument by \texttt{relax} (was \texttt{verschwinden}).
\texttt{l@dgobbleoptarg} \texttt{⟨arg⟩\{⟨arg⟩\}} replaces these two arguments (first is optional) by \texttt{relax}.

Measure (recursively) the width required for a math cell. (was \texttt{messen})
\let\NEXT\measuremcell% 
\fi\NEXT}

\measuretcell Measure (recursively) the width required for a text cell. (was \messentext)
\def\measuretcell #1&{% 
  \ifx #1\relax 
    \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)
\def\measuremrow #1\relax{% 
  \ifx #1\relax 
    \else\measuremcell #1&\relax%
    \stepl@dcolcount%
    \l@dsetmaxcolwidth%
    \let\NEXT\measuremcell%
  \fi\NEXT}

\measuretrow Measure (recursively) the width required for a text row. (was \Messentext)
\def\measuretrow #1\relax{% 
  \ifx #1\relax 
    \else\measuretcell #1&\relax%
    \stepl@dcolcount%
    \let\NEXT\measuretcell%
  \fi\NEXT}

\edtabcolsep The length \edtabcolsep controls the distance between columns. (was \abstand)
\newskip\edtabcolsep
\global\edtabcolsep=10pt

\NEXT
\Next \let\NEXT\relax
\let\Next=\next

\variab
\newcommand{\variab}{\relax}

\l@dcheckcols Check that the number of columns is consistent. (was \tabfehlermeldung)
\newcommand*{\l@dcheckcols}%%%%
\ifnum\l@dcolcount=1 \relax

\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\nulledindex\%
  \let\linenum\@gobble}
\newcommand{\l@drestoreforcritext}{\%
  \def\do##1{\global\csletcs{##1footnote}{##1@@footnote}}
  \dolistloop{\@series}\%
  \let\edindex\xedindex\%
}%

\l@dmodforedtext  Modify and restore various macros for when \edtext is used.
\l@drestoreforedtext
\newcommand{\l@dmodforedtext}{\%
  \let\edtext\relax
  \def\do##1{\global\csletcs{##1footnote}{l@dgobbleoptarg}}
  \dolistloop{\@series}\%
  \let\edindex\nulledindex\%
  \let\linenum\@gobble}
\newcommand{\l@drestoreforedtext}{\%
  \def\do##1{\global\csletcs{##1footnote}{##1@@footnote}}
  \dolistloop{\@series}\%
  \let\edindex\xedindex\%
}%

\l@dnullfills Nullify and restore some column fillers, etc.
\l@drestorefills
\newcommand{\l@dnullfills}{\%
  \def\edlabel##1{}}
\newcommand{\l@drestorefills}{\%
  \def\edrowfill##1##2##3{\@EDROWFILL@{##1}{##2}{##3}}}
%

The original definition of \verteilen and friends (‘verteilen’ is approximately ‘distribute’) was along the lines:

\def\do##1{\csuse{##1@@footnote}}
\dolistloop{\@series}
\let\edindex\xindex
where the lines

\let\critext=xcritext
\let\edtext=xedtext
\let\edindex=xedindex
\def\do##1{\global\csletcs{##1footnote}{##1@@footnote}}
dolistloop{\@series}
\let\linenum=\@line@@num

were common across the several \texttt{*verteilen*} macros, and also

\def\footnoteverschw{%
  \let\critext=relax
  \let\Afootnote=verschwinden
  \let\Bfootnote=verschwinden
  \let\Cfootnote=verschwinden
  \let\Dfootnote=verschwinden
  \let\linenum=\@gobble}

\texttt{\let\critext=relax\let\edtext=\edtext\let\edindex=\edindex\def\do##1{\global\csletcs{##1footnote}{##1@@footnote}}\dolistloop{\@series}\let\linenum=\@line@@num}

\texttt{\let\critext=\relax\let\Afootnote=\verschwinden\let\Bfootnote=\verschwinden\let\Cfootnote=\verschwinden\let\Dfootnote=\verschwinden\let\linenum=\@gobble}
\setmcellright Typeset (recursively) cells of display math right justified. (was \rverteilen)
\def\setmcellright #1&\def\edlabel##1\{
edlabel{##1}\}
\let\edindex\nulledindex
\ifx #1\%\ifnum\l@dcolcount=0\%
\removelastskip
\let\l@Next\relax%
\else\l@dcolcount=0%
\let\l@Next=\setmcellright%
\fi%
\else%
\disablel@dtabfeet%
\step\l@dcolcount%
\disable@notes%
\setbox\hilfsbox=\hbox{$\displaystyle{#1}$%}
\restore@notes%
\letsforverteilen%
\hskip\hilfsskip$\displaystyle{#1}$%
\hskip\edtabcolsep%
\let\l@Next=\setmcellright%
\fi\l@Next}

\settcellright Typeset (recursively) cells of text right justified. (was \rverteilentext)
\def\settcellright #1&\def\edlabel##1\{
edlabel{##1}\}
\let\edindex\nulledindex
\ifx #1\%\ifnum\l@dcolcount=0\%
\removelastskip
\let\l@Next\relax%
\else\l@dcolcount=0%
\let\l@Next=\settcellright%
\fi%
\else%
\disablel@dtabfeet%
\step\l@dcolcount%
\disable@notes%
\setbox\hilfsbox=\hbox{#1}%
\restore@notes%
\letsforverteilen%
\hskip\hilfsskip#1%
\hskip\edtabcolsep%
\let\l@Next=\settcellright%
\fi\l@Next}

\setmcellleft Typeset (recursively) cells of display math left justified. (was \lverteilen)
\def\setmcellleft #1\&\def\edlabel#1\{\edlabel{#1}\}
\let\edindex\nulledindex
\settcelleft Typeset (recursively) cells of text left justified. (was 1vertei\text{lltext})
\def\settcelleft #1& {\let\edlabel##1{}\let\edindex\nulledindex\ifx #1\ifnum\l@dcolcount=0 \let\Next\relax\else\l@dcolcount=0\let\Next=\settcelleft\fi\else\disablel@dtabfeet\stepl@dcolcount\disable@notes\setbox\hilfsbox=\hbox{#1}\restore@notes\letsforverteilen\hskip 0.5\hilfsskip\hskip\edtabcolsep\let\Next=\settcelleft\fi\Next}

\setmcellcenter Typeset (recursively) cells of display math centered. (was zvertei\text{lltext})
\def\setmcellcenter #1& {\let\edlabel##1{}\let\edindex\nulledindex\ifx #1\ifnum\l@dcolcount=0 \let\Next\relax\else\l@dcolcount=0\let\Next=\setmcellcenter\fi\else\disablel@dtabfeet\stepl@dcolcount\disable@notes\setbox\hilfsbox=\hbox{$\displaystyle{#1}$}\restore@notes\letsforverteilen\hskip 0.5\hilfsskip\hskip\edtabcolsep\hskip 0.5\hilfsskip\hskip\edtabcolsep\let\Next=\setmcellcenter\fi\Next}
\settcellcenter Typeset (recursively) cells of text centered. (new)
\settrowright Typeset (recursively) rows of right justified math. (was \rsetzen)
\settrowleft Typeset (recursively) rows of left justified math. (was \lsetzen)
\let\NEXT=\settrowleft
\fi\NEXT}

\settrowcenter Typeset (recursively) rows of centered math. (was \zsetzen)
\def\settrowcenter #1\{% 
\ifx #1& \let\NEXT\relax%
\else \centerline{\setmcellcenter #1&\&\&}
\let\NEXT=\settrowcenter
\fi\NEXT}

\settrowcenter Typeset (recursively) rows of centered text. (new)
\def\settrowcenter #1\{% 
\ifx #1& \let\NEXT\relax
\else \centerline{\settcellcenter #1&\&\&}
\let\NEXT=\settrowcenter
\fi\NEXT}

\nullsetzen (was \nullsetzen)
\newcommand{\nullsetzen}{% 
\step1@dcolcount%
\l@dcolwidth=0pt%
\ifnum\l@dcolcount=30\let\NEXT\relax%
\l@dcolcount=0\relax
\else\let\NEXT\nullsetzen%
\fi\NEXT}

\edatleft \edatleft[\(math\)]{\langle symbol\rangle}{\langle len\rangle} (combination and generalisation of original \Seklam and \Seklamgl). 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{$\left#2\vrule width0pt height #3$
\right.$\hss\vfil}
\else
\vbox to 4pt{\vss\hbox{$#1\left#2\vrule width0pt height #3$
\right.$\vfil}
\depth 0pt \right. $}$\vfil}
\fi}

\edatright \edatright[\(math\)]{\langle symbol\rangle}{\langle len\rangle} (combination and generalisation of original \Seklam and \Seklamgl). 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
\right\hbox{$#1\left#2\vrule width0pt height #3$
\right.$\vfil}
\depth 0pt \right. $}$\vfil}
\else
\vbox to 4pt{\vss\hbox{$#1\left#2\vrule width0pt height #3$
\right.$\vfil}
\depth 0pt \right. $}$\vfil}
\fi}
\edvertline \edvertline{⟨len⟩} vertical line ⟨len⟩ high. (was \sestrich)
\newcommand{\edvertline}[1]{\vbox to 8pt{\vss\hbox{\vrule height #1}}\vfil}}
\edvertdots \edvertdots{⟨len⟩} vertical dotted line ⟨len⟩ high. (was \sepunkte)
\newcommand{\edvertdots}[1]{\vbox to 1pt{\vss\vbox to #1%\cleaders\hbox{$\m@th\hbox{.}$}\vfil}}

I don’t know if this is relevant here, and I haven’t tried it, but the following appeared on CTT.

From: mdw@nsict.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]

\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....

4522 \newcounter{addcolcount}
4523 \renewcommand{\theaddcolcount}{\roman{addcolcount}}

\@dtabaddcols \@dtabaddcols{⟨startcol⟩}{⟨endcol⟩} adds the widths of the columns ⟨startcol⟩
through ⟨endcol⟩ to \edfilldimen. It is a LaTeX style reimplementation of the
original \@add@.

4524 \newcommand{\@dtabaddcols}[2]{%
4525 \@dtcheckstartend(#1)(#2)%
4526 \if\@dtstartendok
4527 \setcounter{addcolcount}{#1}%
4528 \@whilenum \value{addcolcount}¬#2\relax \do
4529 \advance\edfilldimen by \the\csname dcol\theaddcolcount\endcsname
4530 \advance\edfilldimen by \edtabcolsep
4531 \stepcounter{addcolcount}%
4532 \advance\edfilldimen by \the\csname dcol\theaddcolcount\endcsname
4533 \fi
4534 %}

\@dtcheckstartend\@dtcheckstartend{⟨startcol⟩}{⟨endcol⟩} checks that the values of ⟨startcol⟩
and ⟨endcol⟩ are sensible. If they are then \if\@dtstartendok is set TRUE, otherwise
it is set FALSE.

4536 \newif\@dtstartendok
4537 \newcommand{\@dtcheckstartend}[2]{%
4538 \@dtstartendoktrue
4539 \@ifnum #1<\@ne
4540 \@dtstartendokfalse
4541 \led@err@LowStartColumn
4542 \fi
4543 \@ifnum #2>30\relax
4544 \@dtstartendokfalse
4545 \led@err@HighEndColumn
4546 \fi
4547 \@ifnum #1>\#2\relax
4548 \@dtstartendokfalse
4549 \led@err@ReverseColumns
4550 \fi
4551 %}

\edrowfill \edrowfill{⟨startcol⟩}{⟨endcol⟩}fill fills columns ⟨startcol⟩ to ⟨endcol⟩ inclusive
\@edrowfill\@edrowfill{⟨startcol⟩}{⟨endcol⟩} with {fill} (e.g. \hrulefill, \upbracefill). This is a LaTeX style reimplementation
and generalization of the original \waklam, \Waklam, \waklamec, \wastricht
and \wapunktel macros.

4553 \newcommand*{\edrowfill}[3]{%
4554 \@dtabaddcols(#1)(#2)%
The macro \texttt{\edbeforetab\{text\}\{math\}} puts \textit{text} at the left margin before array cell entry \textit{math}. Conversely, the macro \texttt{\edaftertab\{math\}\{text\}} puts \textit{text} at the right margin after array cell entry \textit{math}. \texttt{\edbeforetab} should be in the first column and \texttt{\edaftertab} in the last column. The following macros support these.

\texttt{\leftltab}\{text\} for \texttt{\edbeforetab} in \texttt{\ltab}. (was \texttt{\linksltab})

\texttt{\leftrtab}\{text\}\{math\} for \texttt{\edbeforetab} in \texttt{\rtab}. (was \texttt{\linksrtab})

\texttt{\leftctab}\{text\}\{math\} for \texttt{\edbeforetab} in \texttt{\ctab}. (was \texttt{\linksztab})

\texttt{\rightctab}\{math\}\{text\} for \texttt{\edaftertab} in \texttt{\ctab}. (was \texttt{\rechtsztab})
\rightltab \rightltab\{\textit{math}\}\{\textit{text}\} for \texttt{edaftertab} in \texttt{ltab}. (was \texttt{rechtsltab})

\newcommand{\rightltab}{\rightltab}[2]{%  
\setbox\hilfsbox=\hbox{\def\edlabel##1{}%  
\disablel@dtabfeet#2}\l@dampcount=\l@dcolcount%  
#1\bb@xt\z\z@{\vbox{\edm@t@b{\l@dcolcount=\l@dampcount%  
\advance\Hilfsskip by\l@dcolwidth%  
\advance\Hilfsskip by-\wd\hilfsbox%  
\setbox\hilfsbox=\hbox{\def\edlabel##1{}%  
\disablel@dtabfeet$\displaystyle{#1}$}%  
\advance\Hilfsskip by-\wd\hilfsbox%  
\advance\Hilfsskip by\edm@t@bcolsep%  
\moveright\Hilfsskip\hbox{ #2}}\hss}  
}

\rightrtab \rightrtab\{\textit{math}\}\{\textit{text}\} for \texttt{edaftertab} in \texttt{rtab}. (was \texttt{rechtsrtab})

\newcommand{\rightrtab}{\rightrtab}[2]{%  
\setbox\hilfsbox=\hbox{\def\edlabel##1{}%  
\disablel@dtabfeet#2}%  
#1\bb@xt\z\z@{\vbox{\edm@t@b}\edm@t@b{\l@dcolcount=\l@dampcount%  
\advance\Hilfsskip by-\wd\hilfsbox%  
\advance\Hilfsskip by\edm@t@bcolsep%  
\moveright\Hilfsskip\hbox{ #2}}\hss}%

\rtab \rtab\{\textit{body}\} typesets \textit{body} as an array with the entries right justified. (was \texttt{rtab})
\edbeforetab \rtab \edaftertab \texttt{davor} and \texttt{danach} The original \texttt{rtab} and friends included a fair bit of common code which I have extracted into macros.

The process is first to measure the \textit{body} to get the column widths, and then in a second pass to typeset the body.

\measurembody \measurembody\{\textit{body}\} measures the array \textit{body}.

\newcommand{\measurembody}{\measurembody}[1]{%  
\disablel@dtabfeet%  
\l@dnullfills%  
\def\edbeforetab##1##2{\leftrtab{##1}{##2}}%  
\def\edataftertab##1##2{\rightrtab{##1}{##2}}%  
\measurembody{#1}%  
\l@drestorefills%  
\variab%  
\setmrowright #1\\&\%  
\enablel@dtabfeet}
\texttt{\textbackslash rtabtext} \texttt{(body)} typesets \texttt{(body)} as a tabular with the entries right justified. (was \texttt{\textbackslash rtabtext})

\begin{verbatim}
\newcommand{\rtabtext}[1]{
  \l@dnullfills
  \measuretbody{#1}
  \l@drestorefills
  \settrowright #1\&\%
  \enablel@dtabfeet}
\end{verbatim}

\texttt{\textbackslash measuretbody} \texttt{(body)} measures the tabular \texttt{(body)}.

\begin{verbatim}
\newcommand{\measuretbody}[1]{
  \disable@notes
  \disablel@dtabfeet
  \l@dcolcount=0
  \nullsetzen
  \measuretbody{#1}
  \restore@notes
  \global\l@dampcount=1}
\end{verbatim}

\texttt{\textbackslash ltab} Array with entries left justified. (was \texttt{\textbackslash ltab})

\begin{verbatim}
\newcommand{\ltab}[1]{
  \l@dnullfills
  \ltabtext{#1}
  \l@drestorefills
  \setmrowleft #1\&\%
  \enablel@dtabfeet}
\end{verbatim}

\texttt{\textbackslash ltabtext} Tabular with entries left justified. (was \texttt{\textbackslash ltabtext})

\begin{verbatim}
\newcommand{\ltabtext}[1]{
  \l@dnullfills
  \measuretbody{#1}
  \l@drestorefills
  \settrowleft #1\&\%
  \enablel@dtabfeet}
\end{verbatim}
\ctab  Array with centered entries. (was \tab)
\edbeforetab \newcommand{\ctab}[1]{%
\@dnullfills
\def\edbeforetab##1##2{\leftctab{##1}{##2}}%
\def\edaftertab##1##2{\rightctab{##1}{##2}}%
\measurembody{#1}%
\@drestorefills
\variab
\setmrowcenter #1\&\%
\enablel@dtabfeet}%
\ctabtext  Tabular with entries centered. (new)
\newcommand{\ctabtext}[1]{%
\@dnullfills
\measuretbody{#1}%
\@drestorefills
\variab
\settrowcenter #1\&\%
\enablel@dtabfeet}%

\spreadtext  (was \breitertext)
\newcommand{\spreadtext}[1]{%\l@dcolcount=\l@dampcount%
\hb@xt@ \the\l@dcolwidth{\hbox{#1}\hss}}
\spreadmath  (was \breiter, ‘breiter’ = ‘broadly’)
\newcommand{\spreadmath}[1]{%\l@dcolcount=\l@dampcount%
\hb@xt@ \the\l@dcolwidth{\hbox{$\displaystyle{#1}$}\hss}}

I have left the remaining \tabs 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.

\tabellzwischen  (was \tabellzwischen)
\def\tabellzwischen #1&{%\l@dcolcount=\l@dampcount%
\ifx #1\\let\NEXT\relax \l@dcolcount=0
\else \step1\l@dcolcount%
\l@dcolwidth = #1 mm
\let\NEXT=\tabellzwischen
\fi \NEXT }

\edatabell  For example \edatabell 4 & 19 & 8 \ specifies 3 columns with widths of 4, 19, and 8mm. (was \atabell)
\def\edatabell #1\&{% \tabellzwischen #1&\&}
\Setzen (was \Setzen, 'setzen' = 'set')
\def\Setzen #1&{\ifx #1\relax \let\NEXT=\relax \let\TABLES\width\let\EDTAB #1| \let\NEXT=\Setzen \fi\NEXT}
\EDATAB (was \ATAB)
\def\EDATAB #1\{\ifx #1\Relax \centerline{\Setzen #1\relax&} \let\Next=\relax \else \centerline{\Setzen #1&\relax&} \let\Next=\EDATAB \fi\Next}
\edatab (was \atab)
\newcommand{\edatab}[1]{\variab\EDATAB #1\Relax\}\}
\HILFSkip More helpers.
\Hilfsskip \newskip\HILFSkip \newskip\Hilfsskip
\EDTABINDENT (was \TABINDENT)
\newcommand{\EDTABINDENT}{\ifnum\l@dcolcount=30\let\NEXT=\relax\l@dcolcount=0% \else\stepl@dcolcount% \let\TABLES\width\let\EDTAB #1| \let\NEXT=\Setzen \fi\NEXT}%
\edtabindent (was \tabindent)
\newcommand{\edtabindent}{\l@dcolcount=0\relax \Hilfsskip=0pt \hilfscount=1\relax \EDTABINDENT% \hilfsskip=\hsize \advance\hilfsskip-\Hilfsskip% \Hilfsskip=0.5\hilfsskip%}
EDTAB \ (was \ TAB)\n\def\EDTAB #1|#2|{%\n\setbox\tabhilfbox=hbox{$\displaystyle{#1}$}%\n\setbox\tabHilfbox=hbox{$\displaystyle{#2}$}%\n\advance\tabelskip -\wd\tabhilfbox%\n\advance\tabelskip -\wd\tabHilfbox%\n\unhbox\tabhilfbox\hskip\tabelskip%\n\unhbox\tabHilfbox}%\n\EDTABtext \ (was \ TABtext)\n\def\EDTABtext #1|#2|{%\n\setbox\tabhilfbox=hbox{#1}%\n\setbox\tabHilfbox=hbox{#2}%\n\advance\tabelskip -\wd\tabhilfbox%\n\advance\tabelskip -\wd\tabHilfbox%\n\unhbox\tabhilfbox\hskip\tabelskip%\n\unhbox\tabHilfbox}%
\tabhilfbox \ Further helpers.\ndarrayl The ‘environment’ forms for \ltab, \ctab and \rtab.\nedarrayc \newenvironment{edarrayl}{\l@dcollect@body\ltab}{}\nedarrayr \newenvironment{edarrayc}{\l@dcollect@body\ctab}{}\nedarrayr \newenvironment{edarrayr}{\l@dcollect@body\rtab}{}\nedarrayl \newenvironment{edarrayl}{\l@dcollect@body\ltabtext}{}\nedarrayc \newenvironment{edarrayc}{\l@dcollect@body\ctabtext}{}\nedarrayr \newenvironment{edarrayr}{\l@dcollect@body\rtabtext}{}\nedarrayl The ‘environment’ forms for \ltabtext, \ctabtext and \rtabtext.\nedarrayc \newenvironment{edarrayl}{\l@dcollect@body\ltabtext}{}\nedarrayc \newenvironment{edarrayc}{\l@dcollect@body\ctabtext}{}\nedarrayr \newenvironment{edarrayr}{\l@dcollect@body\rtabtext}{}\nedarrayl \newcommand{\usingcritext}{\disablel@dtabfeet\enablel@dtabfeet}\nedarrayl \newcommand{\usingedtext}{\disablel@dtabfeet\enablel@dtabfeet}\nedarrayl The default at this point is for \edtext.\nedarrayl \usingcritext \Disclaimer text for using \critext{}.../ or using \edtext{}{} inside tabulars.\nedarrayl \disablel@dtabfeet \The default at this point is for \edtext.
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.

\newcommand\initnumbering@sectcmd{\newcommand\ledsection[2]{%\led@war@ledxxxDeprecated\section}{%\leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\ifl@dpairing\else\skipnumbering\fi%\pstart%\leavevmode\ifledsecnolinenumber\skipnumbering\fi\section[##1]{##2}\leavevmode\vspace{2.3ex \@plus.2ex}\skipnumbering\pend%\vspace{-2\parskip}\vspace{-2\baselineskip}%%\ifautopar\else\pstart\fi}{%\led@war@ledxxxDeprecated\subsection}{%\leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\ifl@dpairing\else\skipnumbering\fi%\pstart%\leavevmode\ifledsecnolinenumber\skipnumbering\fi\subsection[##1]{##2}\leavevmode\vspace{1.5ex \@plus .2ex}\skipnumbering\pend%\vspace{-2\parskip}\vspace{-2\baselineskip}%%\ifautopar\else\pstart\fi}{%\led@war@ledxxxDeprecated\subsection*}{%\leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\ifl@dpairing\else\skipnumbering\fi%\pstart%\leavevmode\ifledsecnolinenumber\skipnumbering\fi\subsection*{##1}\leavevmode\vspace{1.5ex \@plus .2ex}\skipnumbering\pend%\vspace{-2\parskip}\vspace{-2\baselineskip}%%\ifautopar\else\pstart\fi}{%\led@war@ledxxxDeprecated\subsubsection}{%\leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\ifl@dpairing\else\skipnumbering\fi%\pstart%\leavevmode\ifledsecnolinenumber\skipnumbering\fi\subsubsection[##1]{##2}\leavevmode\vspace{1.5ex \@plus .2ex}\skipnumbering\pend%\vspace{-2\parskip}\vspace{-2\baselineskip}%%\ifautopar\else\pstart\fi}{%\led@war@ledxxxDeprecated\subsubsection*}{%\leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\ifl@dpairing\else\skipnumbering\fi%\pstart%\leavevmode\ifledsecnolinenumber\skipnumbering\fi\subsubsection*{##1}\leavevmode\vspace{1.5ex \@plus .2ex}\skipnumbering\pend%\vspace{-2\parskip}\vspace{-2\baselineskip}%%\ifautopar\else\pstart\fi}{%\led@war@ledxxxDeprecated\chapter}{%\leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\ifl@dpairing\else\skipnumbering\fi%\pstart%\leavevmode\ifledsecnolinenumber\skipnumbering\fi\chapter\leavevmode\vspace{2.3ex \@plus .2ex}\skipnumbering\pend%\vspace{-2\parskip}\vspace{-2\baselineskip}%%\ifautopar\else\pstart\fi}{%\led@war@ledxxxDeprecated\chapter*}{%\leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\ifl@dpairing\else\skipnumbering\fi%\pstart%\leavevmode\ifledsecnolinenumber\skipnumbering\fi\chapter*{##1}\leavevmode\vspace{2.3ex \@plus .2ex}\skipnumbering\pend%\vspace{-2\parskip}\vspace{-2\baselineskip}%%\ifautopar\else\pstart\fi}{%\led@war@ledxxxDeprecated\quote}{%\leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\ifl@dpairing\else\skipnumbering\fi%\pstart%\leavevmode\ifledsecnolinenumber\skipnumbering\fi\quote\leavevmode\vspace{2.3ex \@plus .2ex}\skipnumbering\pend%\vspace{-2\parskip}\vspace{-2\baselineskip}%%\ifautopar\else\pstart\fi}{%\led@war@ledxxxDeprecated\endquote}{%\leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\ifl@dpairing\else\skipnumbering\fi%\pstart%\leavevmode\ifledsecnolinenumber\skipnumbering\fi\endquote\leavevmode\vspace{2.3ex \@plus .2ex}\skipnumbering\pend%\vspace{-2\parskip}\vspace{-2\baselineskip}%%\ifautopar\else\pstart\fi}{%\led@war@ledxxxDeprecated\quotation}{%\leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\ifl@dpairing\else\skipnumbering\fi%\pstart%\leavevmode\ifledsecnolinenumber\skipnumbering\fi\quotation\leavevmode\vspace{2.3ex \@plus .2ex}\skipnumbering\pend%\vspace{-2\parskip}\vspace{-2\baselineskip}%%\ifautopar\else\pstart\fi}{%\led@war@ledxxxDeprecated\endquotation}{%\leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\ifl@dpairing\else\skipnumbering\fi%\pstart%\leavevmode\ifledsecnolinenumber\skipnumbering\fi\endquotation\leavevmode\vspace{2.3ex \@plus .2ex}\skipnumbering\pend%\vspace{-2\parskip}\vspace{-2\baselineskip}%%\ifautopar\else\pstart\fi}{%\led@war@ledxxxDeprecated\quote}{%\leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\ifl@dpairing\else\skipnumbering\fi%\pstart%\leavevmode\ifledsecnolinenumber\skipnumbering\fi\quote\leavevmode\vspace{2.3ex \@plus .2ex}\skipnumbering\pend%\vspace{-2\parskip}\vspace{-2\baselineskip}%%\ifautopar\else\pstart\fi}{
Section's title commands

\newcommand{\ledsubsubsection}[2][]{%
    \ifdim0pt=\dimexpr\min#1 .\exminus .2\exext@plus#1\exfill@
        \leavevmode\pend\vspace{3.5ex \plus 1ex \minus .2ex}\else\skipnumbering\fi
    \pstart%
    \leavevmode\ifl@dpairing\else\skipnumbering\fi
    \pstart%
    \leavevmode\ifledsecnolinenumber\skipnumbering\fi\subsubsection[#1]{#2}\leavevmode\vspace{1.5ex \plus .2ex}\skipnumbering\pend%
    \vspace{-2\parskip}\vspace{-2\baselineskip}\
    \ifautopar\else\pstart\fi
} \WithSuffix\newcommand{\ledsubsubsection*}[1][]{%
    \ifdim0pt=\dimexpr#1 .\exminus .2\exext@plus#1\exfill@
        \leavevmode\pend\vspace{3.5ex \plus 1ex \minus .2ex}\else\skipnumbering\fi
    \pstart%
    \leavevmode\ifledsecnolinenumber\skipnumbering\fi\subsubsection*{#1}\leavevmode\vspace{1.5ex \plus .2ex}\skipnumbering\pend%
    \vspace{-2\parskip}\vspace{-2\baselineskip}\
    \ifautopar\else\pstart\fi
} \newcommand{\ledchapter}[2][]{%
    \ifdim0pt=\dimexpr#1 .\exminus .2\exext@plus#1\exfill@
        \leavevmode\pend\vspace{3.5ex \plus 1ex \minus .2ex}\else\skipnumbering\fi
    \pstart%
    \leavevmode\ifdmemoir\gdef\ch@pt@c{#1}\fi
    \skipnumbering
    \pstart%
    \@patchforledchapter\chapter[#1]{#2}\
    \pstart}
\WithSuffix\newcommand{\ledchapter*}[1][]{%
    \ifdim0pt=\dimexpr#1 .\exminus .2\exext@plus#1\exfill@
        \leavevmode\pend\vspace{3.5ex \plus 1ex \minus .2ex}\else\skipnumbering\fi
    \pstart%
    \leavevmode\ifdmemoir\gdef\ch@pt@c{#1}\fi
    \skipnumbering
    \pstart%
    \@patchforledchapter\chapter*{#1}\
    \pstart}
\def\@patchforledchapter{
    \patchcmd{\@makeschapterhead}{1\par}{1}{}{0}
    \pretocmd{\@makeschapterhead}{\par}{}{0}
    \apptocmd{\@makechapterhead}{\par}{}{0}
    \patchcmd{\@makechapterhead}{\vskip 40\p@}{}{0}
    \pretocmd{\@makechapterhead}{1\par}{}{0}
    \apptocmd{\@makechapterhead}{1\par}{}{0}
    \patchcmd{\@makechapterhead}{\vskip 40\p@}{}{0}
    \pretocmd{\@chapter}{\par\leavevmode\vspace{40 \p@}\skipnumbering}{}{0}
    \apptocmd{\@chapter}{\par\leavevmode\vspace{40 \p@}\skipnumbering}{}{0}
    \newcommand{\beforeledchapter}{\pend\cleardoublepage\pstart}
    \patchcmd{\chapter}{\cleardoublepage}{\relax}{}{0}
    \patchcmd{\chapter}{\clearpage}{\relax}{}{0}
}\ifnoquotation@else\renewcommand{\quotation}{\par\leavevmode% 
    \parindent=1.5em}%
40.1 Deprecated commands

\skipnumbering
\ifautopar
  \vskip-\parskip
\else
  \vskip\topsep
\fi
\global\leftskip=\leftmargin
\global\rightskip=\leftmargin
}
\renewcommand{\endquotation}{\par\leavevmode
\global\leftskip=0pt
\global\rightskip=0pt
\skipnumbering
\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
}
\renewcommand{\endquote}{\par\leavevmode
\global\leftskip=0pt
\global\rightskip=0pt
\skipnumbering
\ifautopar
  \vskip-\parskip
\else
  \vskip\topsep
\fi
\renewcommand{\endquote}{\par
\global\leftskip=0pt
\global\rightskip=0pt
\skipnumbering
\ifautopar
  \vskip-\parskip
\else
  \vskip\topsep
\fi
}
\newcommand{\ledsectnotoc}{\let\addcontentsline@gobblethree}

\ledsectnotoc The \ledsectnotoc only disables the \addcontentsline macro.
\newcommand{\ledsectnotoc}{\let\addcontentsline@gobblethree}

\ledsectnomark The \ledsectnomark only disables the \chaptermark, \sectionmark and \subsectionmark macros.
40.2 New commands: \eledxxx

The new system of \eledxxx 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.

\begin{eledchapter}

For technical reasons, not yet solved, page-breaking before chapters can’t be made automatically by eledmac. Users have to use \eledchapter.

\if\if@memoir
\newcommand{\eledchapter}{\clearforchapter}
\else
\newcommand{\eledchapter}{\if@openright\cleardoublepage\else\clearpage\fi}
\fi

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.

\def{\print@rightmargin@eledsection}

\print@leftmargin@eledsection and \print@rightmargin@eledsection are added by eledmac inside the code of sectioning command, in order to affix lines 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, Maiéul 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.

\catcode'\#=12 % Space NEEDS by \catcode
\AtBeginDocument{%
\patchcmd{\chapter}{\clearforchapter}{%\if@eled@sectioning\else\clearforchapter\fi}{}{}}
\catcode'\#12
\if@eled@sectioning%\begingroup%
\if@RTL%
\let\llap\rlap%
\let\leftlinenum\rightlinenum%
\let\leftlinenumR\rightlinenumR%
\let\l@drd@ta\l@dld@ta%
\let\l@drsn@te\l@dlsn@te%
\fi
\hfill\l@drd@ta \csuse{LR}{\l@drsn@te}%
\endgroup%
\fi
\def\print@leftmargin@eledsection{%
\if@eled@sectioning%
\leavevmode%
\begingroup%
\if@RTL%
\let\rlap\llap%
\let\rightlinenum\leftlinenum%
\let\rightlinenumR\leftlinenumR%
\let\l@dld@ta\l@drd@ta%
\let\l@dlsn@te\l@drsn@te%
\fi
\l@dld@ta\csuse{LR}{\l@dlsn@te}%
\endgroup%
\fi
}%
\fi
%
\chapter
\M@sect
\@makechapterhead
\@makeschapterhead
\section
\\section

40.2 New commands : \eledxxx
Section's title commands

\pretocmd{\M@sect}{\let\old@edtext=\edtext\let\edtext=\dummy@edtext\showlemma}
\apptocmd{\M@sect}{\let\edtext=\old@edtext}
\patchcmd{\M@sect}{\hskip #3\relax}{\hskip #3\relax\print@leftmargin@eledsection}
\patchcmd{\@mem@old@ssect}{#5}{#5\print@leftmargin@eledsection}
\patchcmd{\chapter}{\if@openright\cleardoublepage\else\clearpage\fi}{\if@eled@sectioning\else\if@openright\cleardoublepage\else\clearpage\fi%No clearpage inside a \eledsection : will keep critical notes from printing on the title page.
40.2 New commands: \eledxxx

\fi%
\}%
{\}%
{\}%
\patchcmd{\makechapterhead}{#1}{\print@leftmargin@eledsection#1\print@rightmargin@eledsection}{}{}
\patchcmd{\makechapterhead}{\if@RTL\raggedleft\else\raggedright\fi}{\if@eled@sectioning\else\if@RTL\raggedleft\else\raggedright\fi}{\fi}{}\patchcmd{\makeschapterhead}{#1}{\print@leftmargin@eledsection#1\print@rightmargin@eledsection}{}{}
\pretocmd{\@sect}{\let\old@edtext=\edtext\let\edtext=\dummy@edtext@showlemma}{}
\apptocmd{\@sect}{\let\edtext=\old@edtext}{}
\pretocmd{\@ssect}{\let\old@edtext=\edtext\let\edtext=\dummy@edtext@showlemma}{}
\apptocmd{\@ssect}{\let\edtext=\old@edtext}{}
\apptocmd{\@ssect}{\let\edtext=old@edtext}{\{}{\} \par
\hyperref{\@sect} also redefines \@sect. That’s why, when manipulating arguments, we patch \@sect and the same only if \hyperref{} is not used. If it is, we patch the \NR commands.
\@ifpackageloaded{nameref}{\par
\patchcmd{\NR@sect}{#8}{#8\print@rightmargin@eledsection}{\}{} \par
\patchcmd{\NR@sect}{\hskip #3\relax}{\hskip #3\relax\print@leftmargin@eledsection}{\}{} \par
\patchcmd{\NR@ssect}{#5}{#5\print@rightmargin@eledsection}{\}{} \par
\patchcmd{\NR@ssect}{\hskip #1}{\hskip #1\print@leftmargin@eledsection}{\}{} \par
\patchcmd{\@sect}{#8}{#8\print@rightmargin@eledsection}{\}{} \par
40.2 New commands: \eledxxx

\patchcmd{\@sect}
{\hspace #3\relax}
{\hspace #3\relax%
\print@leftmargin@eledsection%
}
{}
{}
{}
\patchcmd{\@ssect}
{#5}
{#5%
\print@rightmargin@eledsection%
}
{}
{}
{}
\patchcmd{\@ssect}
{\hspace #1}
{\hspace #1%
\print@leftmargin@eledsection%
}
{}
{}
{}
\catcode'#=6 %Space NEEDS by \catcode

\eled@sectioning@out \eled@sectioning@out is the output file, to dump the pstarts where a sectioning command is used.
\newwrite\eled@sectioning@out
\noeledsec The boolean \if@noeled@sec is set to true when \noeledsec is called. It is used to disable external file creation.
\newif\if@noeled@sec
\newcommand{\noeledsec}{\global\@noeled@sectrue}
\eledchapter \eledsection
\eledchapter* \eledsection*
\eledsubsection
\eledsubsection* \eledsubsection*

And now, the user sectioning commands, which write to the file, and also add content as a “normal” line.
40  Section's title commands

\string\eled@chapter{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL}{}{}
\fi
\fim
newcommand{\eledsection}[2][]{}% #2%
\ifledRcol%
  \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][]{}% #2%
\ifledRcol%
  \immediate\write\eled@sectioningR@out{%
    \string\eled@subsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR}{}{R}
  }%
\else%
  \immediate\write\eled@sectioning@out{%
    \string\eled@subsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL}{}{}
  }%
\fi%

newcommand{\eledsubsubsection}[2][]{}% #2%
\ifledRcol%
  \immediate\write\eled@sectioningR@out{%
    \string\eled@subsubsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR}{}{R}
  }%
\else%
  \immediate\write\eled@sectioning@out{%
    \string\eled@subsubsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL}{}{}
  }%
\fi%

\WithSuffix newcommand{\eledchapter*[2][]}{}% #2%
\ifledRcol%
  \immediate\write\eled@sectioningR@out{%
    \string\eled@chapter{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR}{}{R}
  }%
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.

\def\eled@chapter#1#2#3#4#5{% 
  \ifstrempty{#4}% 
    \ifstrempty{#1}% 
      \global\csdef{eled@sectioning@#3#5}{\section{#2}}% 
      \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext\csuse{subsectionmark}{#1}}% 
    \else{}% 
      \global\csdef{eled@sectioning@#3#5}{\chapter{#2}}% 
      \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext\chaptermark{#2}}% 
    \fi{}% 
  \else{}% 
    \ifstrempty{#1}% 
      \global\csdef{eled@sectioning@#3#5}{\section*{#2}}% 
      \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext\chaptermark{#2}}% 
    \else{}% 
      \global\csdef{eled@sectioning@#3#5}{\chapter{#2}}% 
      \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext\chaptermark{#2}}% 
    \fi{}% 
  \fi{}% 
  \listcsgadd{eled@sections#5@@}{#3}% 
}% 
\def\eled@section#1#2#3#4#5{% 
  \ifstrempty{#4}% 
    \ifstrempty{#1}% 
      \global\csdef{eled@sectioning@#3#5}{\section{#2}}% 
      \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext\csuse{subsectionmark}{#1}}% 
    \else{}% 
      \global\csdef{eled@sectioning@#3#5}{\chapter{#2}}% 
      \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext\chaptermark{#2}}% 
    \fi{}% 
  \else{}% 
    \ifstrempty{#1}% 
      \global\csdef{eled@sectioning@#3#5}{\section*{#2}}% 
      \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext\chaptermark{#2}}% 
    \else{}% 
      \global\csdef{eled@sectioning@#3#5}{\section{#2}}% 
      \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext\csuse{subsectionmark}{#1}}% 
    \fi{}% 
  \fi{}% 
  \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@sectmark@#3#5}{\let\edtext=\dummy@edtext\csuse{subsectionmark}{#1}}% 
    \else{}% 
      \global\csdef{eled@sectioning@#3#5}{\section{#2}}% 
      \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext\csuse{subsectionmark}{#1}}% 
    \fi{}% 
  \else{}% 
    \ifstrempty{#1}% 
      \global\csdef{eled@sectioning@#3#5}{\subsection*{#2}}% 
      \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext\chaptermark{#2}}% 
    \else{}% 
      \global\csdef{eled@sectioning@#3#5}{\subsection{#2}}% 
      \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext\chaptermark{#2}}% 
    \fi{}% 
  \fi{}% 
  \listcsgadd{eled@subsections#5@@}{#3}% 
}% 

40 Section's title commands
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.

\normal@page@break is an etoolbox list which contains the absolute line number of the last line, for each page.

\def\normal@page@break{}

The \l@prev@pb macro is a etoolbox list, which contains the lines in which page breaks occur (before or after). The \l@prev@nopb macro is a etoolbox list, which contains the lines with NO page break before or after.

\def\l@prev@pb{}
\def\l@prev@nopb{}

The \ledpb macro writes the call to \led@pb in line-list file. The \ledpbnum macro writes the call to \led@pbnum in line-list file. The \lednopb macro writes
the call to \texttt{\textbackslash led@nopb} in line-list file. The \texttt{\textbackslash lednopbnum} macro writes the call to \texttt{\textbackslash led@nopbnum} in line-list file.

\begin{Verbatim}
\newcommand{\ledpb}{\write\linenum@out{\string\led@pb}}
\newcommand{\ledpbnum}[1]{\write\linenum@out{\string\led@pbnum{#1}}}
\newcommand{\lednopb}{\write\linenum@out{\string\led@nopb}}
\newcommand{\lednopbnum}[1]{\write\linenum@out{\string\led@nopbnum{#1}}}
\end{Verbatim}

\texttt{\textbackslash led@pb} adds the absolute line number in the \texttt{prev@pb} list. The \texttt{\textbackslash led@pbnum} adds the argument in the \texttt{prev@pb} list. The \texttt{\textbackslash led@nopb} adds the absolute line number in the \texttt{prev@nopb} list. The \texttt{\textbackslash led@nopbnum} adds the argument in the \texttt{prev@nopb} list.

\begin{Verbatim}
\newcommand{\ledpb}{\listxadd{\l@prev@pb}{\the\absline@num}}
\newcommand{\ledpbnum}[1]{\listxadd{\l@prev@pb}{#1}}
\newcommand{\lednopb}{\listxadd{\l@prev@nopb}{\the\absline@num}}
\newcommand{\lednopbnum}[1]{\listxadd{\l@prev@nopb}{#1}}
\end{Verbatim}

\texttt{\textbackslash ledpbsetting} macro only changes the value of \texttt{\textbackslash led@pb@macro}, for which the default value is \texttt{before}.

\begin{Verbatim}
\def\ledpbsetting{before}
\newcommand{\ledpbsetting}[1]{\gdef\ledpbsetting{#1}}
\end{Verbatim}

\texttt{\textbackslash led@check@pb} and \texttt{\textbackslash led@check@nopb} 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 \texttt{\textbackslash pb}.

\begin{Verbatim}
\newcommand{\led@check@pb}{\xifinlist{\the\absline@num}{\l@prev@pb}{\pagebreak[4]}}
\newcommand{\led@check@nopb}{\IfStrEq{\led@pbsetting}{before}{\xifinlist{\the\absline@num}{\l@prev@nopb}{\normal@page@break}{\enlargethispage{\baselineskip}}}}
\IfStrEq{\led@pbsetting}{after}{\xifinlist{\the\absline@num}{\l@prev@nopb}{\pagebreak[4]}}
\end{Verbatim}
42 Long verse: prevents being separated by a page break

\ilednopbinverse  The \ilednopbinverse 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 \check@pb@in@verse checks if a verse is broken in two pages. If true, it adds:

- The absolute line number of the first line of the verse -1 in the \iled@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 \iled@nopb list, if the page break must occur after the verse.

\newcommand{\check@pb@in@verse}{%  
  \ifinstanza\ilednopbinverse\ifinserthangingssymbol\% Using stanzas and enabling page breaks in verse control, while on a hanging verse.
  \ifnum\page@num=\last@page@num\else\%If we have change page
  \IfStrEq{\iled@pb@setting}{before}{%          \numgdef{\abs@line@verse}{\the\absline@num-1}\%
  \iledpbnum{\abs@line@verse}\%
  }{\%}
  \IfStrEq{\iled@pb@setting}{after}{%              \numgdef{\abs@line@verse}{\the\absline@num-1}\%
  \ilednopbnum{\abs@line@verse}\%
  }{\%}
  \fi\%
  \fi\fi\fi\%
}\%
}

43 The End

\end{code}

Appendix A  Some things to do when changing version

Appendix A.1  Migrating from edmac

If you have never used EDMAC, ignore this section. If you have used EDMAC and are starting on a completely new document, ignore this section. Only read this section if you are converting an original EDMAC document to use eledmac.

The package still provides the original \text command, but it is (a) deprecated, and (b) its name has been changed\(^{30}\) to \critext; use the \edtext macro instead. However, if you do use \critext (the new name for \text), the following is a reminder.

\critext

Within numbered paragraphs, footnotes and endnotes are generated by forms of the \critext macro:

\critext{⟨lemma⟩}{⟨commands⟩}/

The ⟨lemma⟩ argument is the lemma in the main text: \critext both prints this as part of the text, and makes it available to the ⟨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:

I saw my friend \critext{Smith}/
\Afootnote{Jones C, D.}/
on Tuesday.

1 I saw my friend
2 Smith 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 \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:

\critext{I saw my friend}
\critext{Smith}{⟨Afootnote{Jones C, D.}⟨commands⟩⟨commands⟩/}
on Tuesday.}
\Bfootnote{The date was July 16, 1954.}/

1–2 I saw my friend
Smith on Tuesday. The date was July 16, 1954.

However, \critext cannot handle overlapping but unnested notes—for example, one note covering lines 10–15, and another covering 12–18; a \critext that starts in the ⟨lemma⟩ argument of another \critext must end there, too. (The ⟨lemma⟩ and \linenum commands may be used to generate overlapping notes if necessary.)

\(^{30}\)A name like \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.
The second argument of the `\critext` macro, \textit{\langle commands \rangle}, 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:

\begin{verbatim}
\catcode'\langle=\active
\let<=\critext
\end{verbatim}

Then you might say `<\{Smith\}variant{Jones}`). This of course destroys the ability to use `< 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:

\begin{verbatim}
\catcode'\langle=\active
\def\xtext#1#2>{\critext{#1}{#2}/}
\let<=\xtext
\end{verbatim}

This allows you to say `<\{Smith\}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` \hspace{1em} (See section \ref{sec:Aliasing} 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 `\usingcritext` beforehand. The declaration `\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 `\addfootins` and `\addfootinsX`, you should instead use the `\newseries` command (see \ref{sec:CustomizingSeries} p.26). You must delete your `\Xfootnote` command.

If you customized the `\XXXXXXfmt` command, you should see if commands for display options \begin{verbatim}[4.4 p.20]\end{verbatim} and options in `\Xfootnote` \begin{verbatim}[4.1 p.17]\end{verbatim} can’t do the same things. If not, you can add a new ticket in Github to request a new function it\cite{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 `\XXXXXXfmt` command, but you must replace:

\begin{verbatim}
\renewcommand*{XXXXXfmt}[3]
\end{verbatim}

\hspace{1em}\begin{verbatim}[31]https://github.com/maieul/ledmac/issues\end{verbatim}
with

\renewcommandx*{XXXXfmt}[4][4=Z]

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 stanzaindentsrepetition (cf. p. 27). This bug had two consequences:

1. stanzaindentsrepetition didn’t work when its value was greater than 2.
2. stanzaindentsrepetition worked wrong when its value was equal to 2.

So, if you used stanzaindentsrepetition with value equal to 2, you must change your \setstanzaindents. Explanation:

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

This code, in a version older than 1.5.1, made that the first verse had an indent of 0, the second 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 second 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:

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

by:

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

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 \l@reg, which is not advisable, you must rename it to \@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 pstart 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


[Fai03] Robin Fairbairns. footmisc — a portmanteau package for customising footnotes in \LaTeX. February 2003. (Available from CTAN in macros/latex/ contrib/footmisc)


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

\# .................................. 4942, 5125 \@edline ......................... 1932
\& .................................. 22, 4001, \@edrowindexm@m ... 3828, 3830, 3833, 4005, 4006, 4068, 4083, 4104, 4106 3840, 3842, 3845, 3850, 3852, 3855
\@edleftnote .............. 4146, 4154 \@edrowfill@ .................. 4338, 4553
\@edrightnote ............ 4145, 4153 \@m ...................... 1932, 4061, 4073
Index
\@makecol ........................................ 3204
\@makefcolumn .................................. 3255, 3256, 3264, 3266
\@makeshapteread ................................ 4838-4841, 4942
\@makeparagraph ................................ 3115
\@makeownfoot ................................... 2305, 2374, 2741, 4776,
\@makeownfootfalse ............................... 3227, 3228
\@makeownfoottrue ................................
\@makeownreffalse ................................
\@makeownreftrue ................................
\@makeownsect ....................................
\@maketoc ......................................... 4942
\@makewindex ..................................... 3205
\@maxdepth ....................................... 3130, 3139
\@memextranofeet ................................ 3226
\@memnofootfalse ................................ 3227, 3228
\@memnofoottrue .................................. 4847
\@memoryfalse ....................................
\@memorytrue ......................................
\@maxdepth ....................................... 3130, 3139
\@maxdepth ....................................... 4800, 4802, 4808, 4810, 4816
\@maxdepth ....................................... 4838-4841, 4942
\@maxdepth ....................................... 4778, 4784, 4786, 4792, 4794,
\@maxdepth ....................................... 4800, 4802, 4808, 4810, 4816,
\@maxdepth ....................................... 4838-4841, 4942
\@maxdepth ....................................... 4778, 4784, 4786, 4792, 4794,
\@maxdepth ....................................... 4800, 4802, 4808, 4810, 4816,
\@maxdepth ....................................... 4838-4841, 4942
\@maxdepth ....................................... 4778, 4784, 4786, 4792, 4794,
\@maxdepth ....................................... 4800, 4802, 4808, 4810, 4816,
\@maxdepth ....................................... 4838-4841, 4942
\@maxdepth ....................................... 4778, 4784, 4786, 4792, 4794,
\@maxdepth ....................................... 4800, 4802, 4808, 4810, 4816,
\@maxdepth ....................................... 4838-4841, 4942
\@maxdepth ....................................... 4778, 4784, 4786, 4792, 4794,
\@maxdepth ....................................... 4800, 4802, 4808, 4810, 4816,
\@maxdepth ....................................... 4838-4841, 4942
\@maxdepth ....................................... 4778, 4784, 4786, 4792, 4794,
\@maxdepth ....................................... 4800, 4802, 4808, 4810, 4816,
\@maxdepth ....................................... 4838-4841, 4942
\@maxdepth ....................................... 4778, 4784, 4786, 4792, 4794,
\@maxdepth ....................................... 4800, 4802, 4808, 4810, 4816,
\@maxdepth ....................................... 4838-4841, 4942
\@maxdepth ....................................... 4778, 4784, 4786, 4792, 4794,
\@maxdepth ....................................... 4800, 4802, 4808, 4810, 4816,
\@maxdepth ....................................... 4838-4841, 4942
\@maxdepth ....................................... 4778, 4784, 4786, 4792, 4794,
\@maxdepth ....................................... 4800, 4802, 4808, 4810, 4816,
\@maxdepth ....................................... 4838-4841, 4942
\@maxdepth ....................................... 4778, 4784, 4786, 4792, 4794,
\@maxdepth ....................................... 4800, 4802, 4808, 4810, 4816,
\c@pstart .................................. 1027, 3312
\c@pstartR .................................. 3299
\c@sublinenumincrement 369, 1310, 1311
\Cendnote .................................. 13
\center .................................... 4458, 4463, 4469, 4474, 4480, 4485, 4707, 4709
\Cfootnote .................................. 13
\check@pb@in@verse 1146, 5335
Chester, Robert of 4
Claassen, Geert H. M. 4
class 1 feet 135, 152
class 2 feet 152, 153
\cleaders .................................. 4517
\cleardoublepage 4848, 4849
\cleardoublepage 4848, 4849, 4908, 4998, 5000
\clearforchapter 4906, 4944, 4946
\closeout .................................. 1146, 5335
Chester, Robert of 4
Claassen, Geert H. M. 4
class 1 feet 135, 152
class 2 feet 152, 153
\cs .......................................... 843, 844, 2828, 2857–2859, 2982
\csdef .................................... 4319, 5237
\c@sublinenumincrement 4458, 4463,
4469, 4474, 4480, 4485, 4707, 4709
\chaptermark .................................. 4829, 4835, 4849,
4850, 4942, 5237, 5241, 5247, 5248
\chaptermark .................................. 4901, 5238, 5242
\char ........................................ 1592, 1839, 2167, 2477, 3143, 3681
\char ........................................ 1592, 1839, 2168, 2477, 3147, 3700
\color@begingroup .................................. 2758, 2770, 2783, 2798, 2804,
2817, 2820, 3488, 3491, 3495,
3503, 3506, 3510, 3518, 3521, 3525
Copernicus, Nicolaus 4
\correct@ footinsX@box 2599, 2600
\correct@Xfootins@box 3174, 3175
\count 1765, 1771, 1783, 1790, 1956,
1960, 2027, 2056, 2260, 2266,
2283, 2287, 2353, 2357, 2418, 2424
\countdef .................................. 3105
\cr .......................................... 1933, 1936
\create@edindex@for@memoir 3804, 3908
\create@edindex@notfor@memoir .................................. 3862, 3904, 3907, 3911
\critext 219, 832, 848, 4231, 4313, 4342
\cs .......................................... 4319, 5237
\csdef ... 1758, 1777, 1944, 2015,
2250, 2272, 2342, 2411, 2703–
2721, 2727, 2729, 2730, 2732–
2734, 2736–2747, 2793, 2812, 2881
\cslet .................................... 2758, 2770, 2783, 2798,
2804, 2817, 2820, 3488, 3491, 3495,
3503, 3506, 3510, 3518, 3521, 3525
\csletcs .................................... 4229
\csgdef 1758, 1777, 1944, 2015,
2250, 2272, 2342, 2411, 2703–
2721, 2727, 2729, 2730, 2732–
2734, 2736–2747, 2793, 2812, 2881
\csnumdef .................................. 4219, 4313, 4325, 4330, 4345
\csnumgdef .................................. 1080, 1082
\csnumdef .................................. 927, 944, 968, 976
\csundef .................................. 472, 1178
\csuse .................................... 223, 224, 229, 296, 927, 928,
944, 945, 970, 978, 986, 1000,
1176, 1568, 1569, 1588, 1589,
1600, 1606, 1609–1611, 1617,
1703, 1714, 1718, 1736, 1742,
1745, 1767, 1768, 1772, 1773,
1785, 1786, 1791, 1792, 1795,
1808, 1815, 1820, 1821, 1835,
1836, 1854, 1861–1863, 1871,
1872, 1891, 1897, 1903, 1904,
1919, 1921–1923, 1925, 1948,
2019, 2031, 2036, 2040, 2044–
2046, 2049, 2050, 2071, 2077,
2079, 2150, 2151, 2158, 2163,
2164, 2175, 2176, 2186, 2198,
2220, 2226, 2232, 2262, 2263,
2267, 2268, 2276, 2292, 2302,
2303, 2309, 2312, 2329, 2335,
2346, 2362, 2370, 2372, 2378,
2381, 2398, 2404, 2420, 2421,
2425, 2426, 2429, 2442, 2453,
2459, 2460, 2473, 2474, 2490,
2499, 2516, 2522, 2529, 2541,
2554, 2564, 2570, 2580, 2586,
2591, 2592, 2598, 2609, 2641,
2732, 2733, 2769, 2782, 2788,
2800, 2802, 2804, 2867, 2868,
2883, 2884, 3033, 3039, 3050,
3052–3056, 3077, 3082, 3086,
3099, 3103, 3168, 3169, 3173,
3188, 3189, 3227, 3228, 3236,
Index

\enable@dtabfeet .......... 4620, 4636, 4656, 4664, 4674, 4682, 4764
\end@lemmas .......... 822, 864, 865
\endashchar ... 21, 1614, 1694, 2674
\endgraf .......... 1078, 1124, 1128
\endline@num .......... 466, 710, 716
\endlock I, 812, 829, 4072, 4089, 4097
\endminipage .......... 3692
\endnumbering ... 6, 207, 251, 282, 301
\endpage@num .......... 466, 709, 716
\endprint .......... 26, 2641, 2685
\endquotation .......... 4773
\endquote .......... 4773
\endstanzaeextra .......... 25, 4063
\endsub .......... 11, 788, 828
\endsubline@num .......... 466, 711, 717
\enlarge@thispage .......... 5320, 5328
\enskip .......... 26, 2642
\enspace . 2176, 2309, 2378, 2490, 2642
environments:
  edarrayc .......... 31, 4756
  edarrayl .......... 31, 4756
  edarrayr .......... 31, 4756
  edtabularc .......... 31, 4760
  edtabularr .......... 31, 4760
  ledgroup .......... 26, 3719
  ledgroupsized .......... 26, 3733
  minipage .......... 26
Euclid .......... 4
\ExecuteOptions .......... 19
\expandonce . 1551, 1562, 2122, 2233, 2770, 2783, 2804, 2820, 3491, 3495, 3506, 3510, 3521, 3525, 3863
\extensionchars .. 37, 186, 213, 289
\extractdline@ .......... 3024, 3027
\extractendsubline@ .......... 3025, 3027
\extractline@ .......... 3022, 3027, 3030
\extractsubline@ .... .... 3023, 3027, 3030
\firstlinenum .......... 9, 374
\firstseries .......... 2835
\firstsublinenum .......... 9, 374
\fix@page .......... 512, 558
\flag@end .......... 763, 860, 870, 871, 881
\flag@start .......... 763, 859, 860, 871
\flagstanzan .......... 25, 4101
\floatingpenalty .......... 1580
\flush@notes .......... 1088, 1498, 3652
\flush@notesR .......... 3650
\fnpos .......... 30, 2574
Folkerts, Menso .......... 4
\fontencoding .......... 1512
\fontfamily .......... 1512
\footnote@luatexpardir .......... 1598
\footnote@luatextdir .......... 1597, 1619
\footnoteA .......... 29
\footnoteB .......... 29
\footnoteC .......... 29
\footnoteD .......... 29
\footnoteE .......... 29
\footnotelang@lua .......... 1528, 2762, 2775
\footnotelang@poly .......... 1542, 2766, 2779
\footnoteoptions@ ..... .... 1515, 2767, 2771, 2780, 2785
\footnoterule .......... 1716, 2200, 3145, 3707
\footnotesize .......... 3008, 3460, 3461
\footnotemark .......... 2801
\footparagraph .......... 15, 1776
\footparagraphX .......... 29, 2410
\footsplitskips .......... 1570, 1577, 1822, 1837, 1966, 2032, 2152, 2293, 2363, 2461, 2475
\footthreecol .......... 15, 1943
\footthreecolX .......... 29, 2341
\foottwocol .......... 15, 2014
\foottwocolX .......... 29, 2271
\fullstop .......... 21, 430, 1614, 1691, 1693, 1695, 1697, 2673, 2677
Index

\initnumbering@sectioncmd 221, 293, 4773
\inplaceoflemmaseparator ........... 17
\inplaceofnumber ........... 17
\InputIfFileExists ........... 229, 501
\insert ........... 1567, 1818, 1964,
2030, 2149, 2291, 2361, 2457,
2609, 2623, 3189, 3195, 3197, 3216
\insert@count ........... 689, 690,
777, 779, 856, 1519,
1531, 1533, 1546, 1549, 1552,
2124, 2235, 2784, 3497, 3512, 3527
\insert@countR ........... 768, 770,
855, 1523, 1537, 1539, 1557,
1560, 1563, 2772, 3493, 3508, 3523
\insert@copy ...... 1164, 3994
\insert@copyfalse ........... 1144
\insert@copytrue ........... 1142
\insert@copysymbol .......... 3993
\insert@list ........... 258, 461, 494, 697, 1457, 1461
\insert@parafootsep ........... 1857, 1918, 2488
\insert@listR ........... 1051, 1452,
1455, 1465, 1500, 1501, 1518,
1530, 1532, 1545, 1548, 1551,
2123, 2234, 2783, 3496, 3511, 3526
\insert@listR ........... 1522, 1536, 1538, 1556,
1559, 1560, 2770, 3492, 3507, 3522
\instanzafalse ........... 3992, 4092
\instanzatrue ........... 4066
\interfootnotelinepenalty ........... 1578
\interlinepenalty ....... 1105, 1485, 1578, 4072
\interparanoteglue ........... 3016
\ipn@skip ........ 136, 138,
143, 145, 150, 152, 3590, 3592,
3597, 3600, 3618, 3620, 3625, 3627
\J
Jayaditya ........... 5

\K
Kabelschacht, Alois ........... 100
Krukov, Alexej ........... 70

\L
\l@d@@wrindexhyp ........... 3822
\l@d@add ........... 913, 915, 919, 921
\l@d@dashfalse ........... 1646, 1664, 2647

\l@d@dashtrue ........... 1668, 1674, 1686, 2650, 2655, 2667
\l@d@elinfalse ........... 1642, 1671, 2652
\l@d@elintrue .......... 1671, 1673, 2652, 2654
\l@d@end ........... 2630, 2632, 2633, 2639, 2689, 2818
\l@d@UnequalColumns ....... 4306
\l@d@eslfalse ........... 1644, 1680, 1683, 2661, 2664
\l@d@eslt true ........... 1683, 1685, 2664, 2666
\l@d@index ........... 3807, 3809, 4238
\l@d@makecol ........... 3126, 3204, 3274
\l@d@nums ........... 854, 893, 896, 907, 908,
921, 2768, 2770, 2781, 2783, 2819
\l@d@pnumfalse ........... 1638, 1664, 2647
\l@d@pnumtrue ........... 1667, 2649
\l@d@reinserts ........... 3194, 3205
\l@d@section ........... 2639, 2641
\l@d@set ........... 610, 809
\l@d@subfalse ........... 1640, 1676, 2657
\l@d@subtrue ........... 1678, 2659
\l@d@windexcmd ........... 3821, 3822
\l@d@ampcount ........... 4165,
4302, 4304, 4309, 4569, 4579,
4580, 4592, 4593, 4628, 4646, 4684
\l@d@beginstack ........... 3967, 3978–3980
\l@d@dbfnote ........... 2115, 2119
\l@d@checkcols ........... 4259, 4271, 4299
\l@d@checkstartend ........... 4525, 4536
\l@d@csnotetext ........... 3482, 3483
\l@d@colcount ........... 4165, 4207,
4219, 4220, 4258, 4260, 4270,
4272, 4300, 4304, 4309, 4354,
4356, 4373, 4375, 4391, 4392,
4407, 4408, 4422, 4423, 4439,
4440, 4492, 4493, 4569, 4579,
4580, 4592, 4593, 4624, 4626,
4641, 4643, 4684, 4690, 4720, 4729
\l@d@collect@body ........... 3969, 3977
\l@d@collect@body ........... 4076, 4078
\l@d@collect@body ........... 3964, 4756–4758, 4760–4762
\l@d@colwidth ........... 4207, 4225, 4226,
4348, 4491, 4555, 4581, 4594,
4685, 4687, 4692, 4701, 4722, 4723
\l@d@csnotetext ........... 3482, 3483
\l@d@csnotetext ........... 1193, 3546, 3562, 3567, 3599, 3602
\l@d@csnotetext@1 ........... 1193, 3544, 3602, 3619
\l@d@csnotetext@r ........... 1193, 3564, 3602, 3626
\l@d@dodoreinxtrafeet 3182, 3196, 3202
423
### Index

| O | \oldedtext | 4953, 4960, 5034, 5041, 5046, 5053 |
|   | \oldedsize | 1709, 1720, 1874, 2193, 2204, 2532 |
|   | \oldedRlineflag | 3864, 3891 |
|   | \onedline | 1017, 1137, 1138, 1160, 1165, 1181, 1184 |
|   | \onlypstartinfootnote | 16 |
|   | \openout | 230, 728, 734, 2632 |
| P | \pstart | 1067 |
|   | \PackageError | 56 |
|   | \PackageWarning | 55 |
|   | \pagemove | 524, 616, 703 |
|   | \page@num | 138, 145, 152, 466, 484, 566, 709, 714, 1241, 1349, 1425, 1919, 1928, 3467, 3475, 3540, 3558, 3613, 5337 |
|   | \page@numR | 136, 143, 150, 3535, 3553 |
|   | \pagemovebreak | 786, 3134 |
|   | \pagemovebreak | 5314 |
|   | \pagelinesep | 31, 3770, 3779–3781 |
|   | \page@line | 315 |
|   | \pagemovebreak | 28 |
|   | \pairs | 5239, 5242, 5257, 5261, 5275, 5279 |
|   | \paperwidth | 1180, 1183 |
|   | \par@line | 1017, 1079, 1476, 1477, 1480, 1484 |
|   | \para@footgroup | 1782, 1866 |
|   | \para@footgroup | 2416, 2493 |
|   | \para@footsetup | 1787, 1795 |
|   | \para@footsetup | 2427, 2429 |
|   | \para@vfootnote | 1780, 1817 |
|   | \para@vfootnote | 2414, 2456 |
|   | \parafootfmt | 1781, 1856 |
|   | \parafootfmt | 2415, 2484 |
|   | \parafootfmtsep | 2740, 3020 |
|   | \parafootsep | 19 |
|   | \parafootstart | 1779, 1803 |
|   | \parafootstart | 2413, 2438 |
|   | \parapparatus@false | 11 |
|   | \parapparatus@true | 15 |
|   | \parledgroup@ | 1726, 1881, 1991, 2061, 2210, 2319, 2388, 2506 |
|   | \parledgroup@ | 3661, 3713 |
|   | \parledgroup@ | 3659, 3711 |
|   | \parledgroup@series | 1727, 1882, 1992, 2062, 2211, 2320, 2389, 2507 |
|   | \parledgroup@type | 1728, 1883, 1993, 2063, 2212, 2321, 2390, 2508 |
|   | \patchcmd | 4838, 4841, 4842, 4845, 4849, 4850, 4944, 4964, 4972, 4982, 4990, 4998, 5006, 5015, 5024, 5059, 5067, 5075, 5083, 5092, 5100, 5108, 5116 |
|   | \pausenumbering | 8, 280 |
|   | \pend | 6, 64, 105, 108, 1049, 1071, 1126, 1410, 4090, 4097, 4776, 4778, 4784, 4786, 4792, 4794, 4800, 4802, 4808, 4810, 4816, 4818, 4827, 4830, 4833, 4835, 4848 |
|   | \platoofTivoli | 4 |
|   | \postbodyfootmark | 2130, 2144 |
|   | \postdisplaypenalty | 1106 |
|   | \prebodyfootmark | 2130, 2142 |
|   | \predisplaypenalty | 1105 |
|   | \prenotesX | 20, 1756 |
|   | \prenotesX@ | 1755, 1756, 2183, 2439 |
|   | \prepare@edindex@fornote | 2768, 2781, 3783 |
|   | \pretocmd | 2113, 3939, 4839, 4843, 4952, 5033, 5045 |
|   | \prev@line | 966–968, 970, 974–976, 978, 985, 986, 999, 1000 |
|   | \prevnop | 5302 |
|   | \prev@line | 5302 |
|   | \prevgraf | 1078 |
|   | \prevline | 1920, 3045 |
|   | \prevpage@num | 1918 |
|   | \prevnotes | 20, 1749, 1753 |
|   | \prevnotes@ | 1700, 1749, 1753, 1805 |
|   | \print@eledsection | 1149, 1171 |
|   | \print@footnoterule | 2197, 2219, 2225, 2328, 2334, 2397, 2403, 2452, 2515, 2521, 2578 |
|   | \print@leftmargin@eledsection | 4911, 4975, 4985, 5008, 5026, 5070, 5086, 5103, 5119 |
|   | \print@line | 1150, 1153 |
|   | \print@notesX | 2590 |
|   | \print@rightmargin@eledsection | 4911, 4967, 4993, 5010, 5028, 5062, 5078, 5095, 5111 |
|   | \print@Xfootnoterule | 1713, 1735, 1741, 1814, 1890, 1896, 2000, 2006, 2070, 2076, 2578 |
Change History

\notefontsize ................. 18
\notenumfont ................. 18
\notewidthliketwocolumns ..... 21
\xpgaref ........................ 27, 3353
\xpxstartref .................... 27, 3365
\xrragged ....................... 19
\xrright@appenditem ........... 442, 617,
618, 620, 627, 629, 631, 633,
643, 645, 654, 665, 667, 674,
687, 688, 697, 713, 933, 935,
950, 952, 971, 979, 1518, 1522,
1530, 1532, 1536, 1538, 1545,
1548, 1551, 1556, 1559, 1562,
2122, 2233, 2768, 2781, 3344,
3491, 3495, 3506, 3510, 3521, 3525
\xspaceskip ........................ 1575, 2157, 2294
\xz@skip ........................ 1575, 1583, 2157, 2294
\zz@@@@ ................................ 3280, 3291, 3304, 3393, 3398

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 \@itabnoexpands
to \no@expands ............. 75

v0.2.1.
\xlab: Removed page setting from
\xlab ............................. 158
General: Added text about normal
labeling .......................... 28
Bug fixes and match with mem-
patch v1.8 ...................... 1
Major changes to insert code
when memoir is loaded ....... 154
\doxtrafeet: Renamed \doxtrafeet
to \@dodotexrafeet .......... 152
\xledlabel: Tweaked \xledlabel to
correct page numbers ..... 156
\xld@macol: Rewrote \xld@macol,
calling it \xld@macol ..... 151
\xld@odoreinxtrefet: Renamed
\xld@odoreinxtrefet to
\xld@odoreinxtrefet ...... 153
\xld@dofootinsert: Renamed
\dofootinsert as \xld@dofootinsert
\xz@skip ........................ 1575, 1583, 2157, 2294
\zz@@@@ ....... 3280, 3291, 3304, 3393, 3398

v0.2.2.
General: Improved paragraph foot-
notes ............................ 1
New Dekker example ......... 1
Used \providecommand for
\gobblethree to avoid clash
with the amsfonts package ... 46
\footfudgefiddle: Added
\footfudgefiddle ......... 111
\xline@list@stuff: Added ini-
tial write of page number in
\xline@list@stuff ........... 69
\xparaf@setupt: Added
\xparaf@setupt to
\xparaf@setupt ......... 111
\xparaf@setuptX: Added
\xparaf@setupt to
\xparaf@setuptX ....... 131
\@lab, and similar for sub-lines 158
\@nl@reg: Added a bunch of code to
\@nl for handling \setlinenum 62
General: Includes edstanza and more 1
\ledlinenum: Added \linenumr@p
and \sublinenum@p
to \leftlinenum and \rightlinenum 54
\linenumberlist: Added \linenumr@p
and \sublinenumr@p
to \printendlines 137
\printlines: Added \linenumr@p
and \sublinenumr@p
to \printlines 107
\sublinenumr@p: Added \linenumberstyle
and \sublinenumberstyle 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 168
General: Added minipage, etc., support 1
\ledgroup: Added ledgroup environment 169
\ledgroupsized: Added ledgroup-sized environment 170
\edtext: Added \showlemma to \edtext (and \critext) 76
\footnormal: Added minipage footnote setup to \footnormal 110
\ifledsecnolinenum: Added final/draft options 41
\@dfeetendmini: Added \@dfeetbeginmini, \@dfeetendmini
and all their supporting code 167
\mpnormalfootgroup: Added \mpnormalfootgroup 108
\mpnormalfootnote: Added \mpnormalfootnote 103
\showlemma: Added \showlemma 41
v0.4.1.
General: Added code for changing
\doclearpage 154
Not released. Minor editorial improvements and code tweaks 1
Only change \@footnotetext
and \@footnotemark if memoir not used 122
\doxtrafeetii: Changed
\doxtrafeetii code for easier extensions 152
\edindex: Let edemacs take advantage of memoir's indexing 172
\printXnotes: Added \@opxtrafeetii 153
v0.5.
\@footnotetext: Enabled regular \footnote in numbered text 122
\@xymar: Eliminated \marginpar disturbance 161
General: Added left and right side notes
Added sidenotes, familiar footnotes in numbered text 1
v0.5.1.
General: Added moveable side note 161
Fixed right line numbers killed in v0.5 1
\affixline@num: Changed \affixline@num to cater for sidenotes 93
\ledgroupsized: Only change \hsize in ledgroupsized environment otherwise page number can be in wrong place 170
\\getsidefootnote@margin: Added \sidenotemargin and \sidenote@margin 162
v0.6.
\@lopR: Added \@pend, \@pendR,
\@lopL and \@lopR in anticipation of parallel processing 64
\@nl@reg: Added \fix@page to \@nl 62
Extended \@nl to include the page number 62
General: Fixed long paragraphs looping 1
Fixed minor typos 1
Prepared for eledpar package 1
Change History

\fix@page: Added \last@page\@num and \fix@page 
\new@line: Extended \new@line to output page numbers 
\page@start: Made \page@start a no-op 
\vl@dbfnote: Changed \l@dbfnote and \vl@dbfnote as originals could give incorrect markers in the footnotes 
\do@actions@fixedcode: Added
\do@actions@next: Added number skipping to \do@actions
\do@insidelinehook: Added \do@linehook for use in \do@line 
\endnumbering: Changed \endnumbering for eledpar 
\f@x@l@cks: Added \ch@cksub@l@ck, \ch@ck@l@ck and \f@x@l@cks 
\footspli\textskips: Added \footspli\textskips for use in many footnote styles 
\get@linelist\textfile: Added \get@linelist\textfile 
\ifledRcol@: Added \l@dnumpstarts\L, \ifl@dpairing and \ifpst@rted for/from eledpar 
\initnumbering@reg: Added \initnumbering@reg 
\l@demptyd@ta: Added
\l@ddofootinsert: Deleted
\page@start from \l@ddofootinsert 
\l@dgetline@margin: Added \l@dgetline@margin 
\l@dlock@disp: Added \l@dlock@disp 
\l@dtsidenote\@margin: Added \l@dtsidenote\@margin 
\l@dran@te: Added \l@dlsn@te and \l@dran@te for use in \do@line 
\l@dunboxmpfoot: Added \l@dunboxmpfoot containing some common code 
\l@dzeropenalties: Added \l@dzeropenalties 
\ledlinenum: Added \ledlinenum for use by \l@f@xt@linenum and \r@ightlinenum 
\list@clearing@reg: Added \setprintendlines for use by \printendlines 
\n@num@reg: Added \setprintlines for use by \printlines 
\skipnumbering@reg: Added \sublinenum@r@p instead of \sublinenum@r@p 
\vnumfootnoteX: Removed extraneous space from \vnumfootnoteX 

v0.7.
\on@l@reg: Added \on@l@reg 
\@ref@reg: Added \@ref@reg 
General: eledmac having been available for 2 years, deleted the commented out edmac texts 
Mai\-el Rouquette new maintainer 
Made macros of all messages 
Replaced all \interfootnotelinepenalty etc., by just \interfootnotelinepenalty

Tidying up for eledpar and \readar packages 
\affixline@num: Added skipnumbering to \affixline@num 
\do@actions@\textfixedcode: Added \do@actions@\textfixedcode 
\do@actions@\textnext: Added number skipping to \do@actions 
\do@insidelinehook: Added \do@linehook for use in \do@line 
\endnumbering: Changed \endnumbering for eledpar 
\f@x@l@cks: Added \ch@cksub@l@ck, \ch@ck@l@ck and \f@x@l@cks 
\footspli\textskips: Added \footspli\textskips for use in many footnote styles 
\get@linelist\textfile: Added \get@linelist\textfile 
\ifledRcol@: Added \l@dnumpstarts\L, \ifl@dpairing and \ifpst@rted for/from eledpar 
\initnumbering@reg: Added \initnumbering@reg 
\l@demptyd@ta: Added
\l@ddofootinsert: Deleted \page@start from \l@ddofootinsert 
\l@dgetline@margin: Added \l@dgetline@margin 
\l@dlock@disp: Added \l@dlock@disp 
\l@dtsidenote\@margin: Added \l@dtsidenote\@margin 
\l@dran@te: Added \l@dlsn@te and \l@dran@te for use in \do@line 
\l@dunboxmpfoot: Added \l@dunboxmpfoot containing some common code 
\l@dzeropenalties: Added \l@dzeropenalties 
\ledlinenum: Added \ledlinenum for use by \l@f@xt@linenum and \r@ightlinenum 
\list@clearing@reg: Added \setprintendlines for use by \printendlines 
\n@num@reg: Added \setprintlines for use by \printlines 
\skipnumbering@reg: Added \sublinenum@r@p instead of \sublinenum@r@p 
\vnumfootnoteX: Removed extraneous space from \vnumfootnoteX
v0.8.
General: Bug on endnotes fixed: in a `//` text, all endnotes will print and be placed at the ends of columns.

v0.8.1.
General: Bug on `\edtext; \critex; `\lemma` fixed: we can now use non-switching commands.

v0.9.
General: No more ledpatch. All patches are now in the main file.

v0.9.1.
General: Fix some bugs linked to integrating ledpatch on the main file.

v0.10.
General: Corrections to `//section` and other titles in numbered sections.

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.

v0.12.
General: For compatibility with `eledpar`, possibility to use `\autopar` on the right side. Possibility to number `\pstart`. Possibility to number the `pstart` with the commands `\numberpstarttrue`. Added `\ifledRcol` and `\ifnumberingR` for/from `eledpar`.

v0.12.1.
General: Don’t number `\pstarts` of stanza.

v0.13.
General: New stanzaindentrepetition counter to repeat stanza indents every n verses.

v0.13.1.
General: Don’t number `\pstarts` of stanza.

v0.14.
General: Tweaked `\edlabel` to get correct line number if the command is first element of a paragraph.

v0.15.
General: Line numbering can be reset at each `pstart`. Possibility to print `\pstart` number inside.

v1.0.
General: `\lemma` can contain commands. Debug in lineation command. New generic commands to customize footnote display.

New stanzaindentrepetition counter: to repeat stanza indents every n verses.

`\managestanza@modulo`: New stanzaindentrepetition counter to repeat stanza indents every n verses.

v0.17.
`\ifinserthangingsymbol`: New management of hangingsymbol insertion, preventing undesirable insertions.

v1.0.1.
General: `\lemma` can contain commands. Debug in lineation command.

New generic commands to customize footnote display.

Options nonum and nosep in `\Xfootnote`.

Options of `\Xfootnotes`.
Change History

253

Possibility to have commands in sidenotes. .................. 28
Some compatibility break with eledmac. Change of name: eled-
mac. .............. 1
\morenoexpands: Change to be compatible with new features 75

v1.0.1.
General: Correction on \numberonlyfirstintwolines
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: parapparat-
us. .............. 1
New tools to change order of se-
ries .............. 142
\ledfootinsdim: Deprecated
\ledfootinsdim .......... 109
\preXnotes: New skip \preXnotes@
.............. 109

v1.1.0.
General: Sectioning commands. ... 34

v1.2.
endquote: Compatibility of
\ledchapter with the memoir
class. .................. 204
\preXnotes: Debug in familiar foot-
notes (but introduced by v1.1). 109

v1.3.
endquote: Quotation and quote
environment inside the number-
ing sections. .............. 204

v1.4.
General: Compatibility with Lua-
TeX of RTL notes. ....... 41, 42
\edtext: Compatibility of \edtext
(and \critext) with the right-
to-left direction (with Polyglos-
sia). .................. 76
\newseries@: Remembers the lan-
guage of the lemma, in order to
create a correct direction for the
footnote separator. .............. 140
\normalfootfmt: Direction of foot-
notes with polyglossia. ....... 103
rbracket: Switch the right bracket
to a left bracket when the lemma is RTL (needs polyglos-
sia or LuaTeX). .............. 104

v1.4.2.
\affixside@note: Remove spuri-
ous spaces. .............. 166
endquote: New option noquota-
tion. .................. 204
\labelrefsparsesubline: Fix bug
with \edlabel. .......... 157

v1.4.3.
\labelpstarttrue: New option noquota-
tion. .................. 17
\normalbfnoteX: Allow to redefine
\thefootnoteX with alph when
some packages are loaded. .... 126

v1.4.4.
General: Label inside familiar foot-
notes. .................. 1

v1.4.5.
\normalbfnoteX: Allow to redefine
\thefootnoteX with alph when
some packages are loaded. .... 126

v1.4.6.
General: Bug with memoir class in-
troduced by 1.4.5. .......... 1

v1.4.7.
endquote: Compatibility of
sectioning commands with
\autopar. .............. 204

v1.4.8.
\normalbfnoteX: Allow to redefine
\thefootnoteX with alph when
some packages are loaded. .... 126

v1.4.9.
endquote: Compatibility of
sectioning commands with
\autopar. .............. 204
\normalbfnoteX: Allow to redefine
\thefootnoteX with alph when
some packages are loaded. .... 126

v1.5.
General: Correct indexing when the
call is made in critical notes. .... 171
\do@insidelinehook: Added
\do@insidelinehook for use in
\do@line .............. 89
\texttt{\textbackslash edindex}: Compatibility with the \texttt{imakeidx} package, and possibility to use multiple index with \texttt{\textbackslash edindex}. .......................... 172
\texttt{\textbackslash ifFN@bottom}: Use the bottom option of footmisc package. .......................... 152

v1.5.1.
\texttt{\textbackslash managestanza@modulo}: Correct stanzaindent repetition counter .......................... 181
\texttt{\textbackslash normalvfootnoteX}: Fix bug with normal familiar footnotes when mixing RTL and LTR text. .......................... 123

v1.6.0.
\texttt{\textbackslash falseverse}: Add \texttt{\textbackslash falseverse} macro. .......................... 182

v1.6.1.
\texttt{General}: Corrects a false hanging verse when a verse is exactly the
length of a line. .......................... 1
\texttt{\textbackslash AtEveryPstart}: Spurious space in \texttt{\textbackslash pstart}. .......................... 83
\texttt{\textbackslash fninserthangingsymbol}: Hang verse is now not automatically
flush right. .......................... 180
\texttt{\textbackslash dunhbox@line}: Move the call to \texttt{\textbackslash fninserthangingsymbol} to allow
use \texttt{\textbackslash hfill} inside. .......................... 87
\texttt{\textbackslash pend}: Spurious space in \texttt{\textbackslash pend}. .......................... 84

v1.7.0.
\texttt{General}: New features for managing page breaks. .......................... 36

v1.8.0.
\texttt{General}: Compatibility with parledgroup option of ledpar package. .......................... 1
\texttt{\textbackslash get@index@command}: Debug \texttt{\textbackslash get@index@command} and compatibility with \texttt{hyperref} package. .......................... 172
\texttt{\textbackslash newhookcommand@series@reload}: Debug \texttt{\textbackslash beforefootnoteX} and \texttt{\textbackslash maxnotesX} which didn't work. .......................... 145
\texttt{\textbackslash prevpage@num}: Correct \texttt{\textbackslash parafootsep} when using with ledgroup. .......................... 116

v1.8.1.
\texttt{General}: Debug endnotes when more than one series is used
(change the position where tools for endnotes are defined). .......................... 136

v1.8.2.
\texttt{General}: Debug compatibility problem with hebrew option of babel package. .......................... 1

v1.8.3.
\texttt{General}: Fixes spurious spaces added by v1.7.0. .......................... 1

v1.8.5.
\texttt{General}: Debug indexing in right column, with ledpar. .......................... 171

v1.9.0.
\texttt{\textbackslash doxtrafeet}: Add \texttt{\textbackslash fnpos} to choice the order of footnotes. .......................... 152
\texttt{\textbackslash doxtrafeet@mini}: Add \texttt{\textbackslash mpfnpos} to choice the order of footnotes
in minipage / ledgroup. .......................... 167

v1.10.0.
\texttt{General}: Add \texttt{\textbackslash pstartref} and \texttt{\textbackslash xpstartref} to refer to a
pstart number (extension of \texttt{\textbackslash edlabel}). .......................... 1
\texttt{\textbackslash endquote}: Correction of sectioning commands in parallel texts. .......................... 204

v1.10.1.
\texttt{General}: Compatibility with clevref. .......................... 1

v1.10.2.
\texttt{General}: Compatibility of stanza with v1.8a of babel-greek. .......................... 1

v1.10.3.
\texttt{General}: Debug of cross-referencing. .......................... 1

v1.10.4.
\texttt{General}: Debug of critical notes in edtabular environment. .......................... 1

v1.10.5.
\texttt{General}: Debug of \texttt{\textbackslash pausenumbering}. .......................... 1
\texttt{\textbackslash pausenumbering}: Debug of \texttt{\textbackslash xxref}. .......................... 1

v1.10.6.
\texttt{General}: Debug of interaction between \texttt{\textbackslash autopar}
and \texttt{\textbackslash pausenumbering}. .......................... 1
v1.11.0.
General: Add hooks to disable the
font selection for lemma in foot-
note. .......................... 18

v1.11.1.
General: Correct a bug when a criti-
cal note starts with plus or mi-
nus. ............................ 1

v1.12.0.
\Onl@reg: To ensure compatibility
with \musixtex, \@l becomes
\@l. Consequently, \@l@reg be-
comes \@nl@reg. ................. 62
General: Add \ledinnernote and
\ledouternote commands. ... 28
Add hyperlink to crossref (needs
hyperref package). .............. 27
Compatibility with musixtex . . 1
Debug eledmac section-
ing command after using
\resumenumbering .......... 1
Ensure that imakeidx is loaded
before eledmac ............... 171
New hooks: \afterXrule and
\afterruleX .................. 20
New options for ragged-
paragraph notes ............... 19
New sectioning commands. ... 34
Optional arguments for \pstart
and \pend ..................... 7
\AtEveryPstart: New optional ar-
gument for \pstart, to execute
code before it. ................ 83
\edindex: Use correctly default in-
dex when imakeidx is loaded. 172
\endquote: \eledxxx sectioning
commands are deprecated and
replaced by \eledxxx com-
mands. .......................... 204
\ifledRcol@: Add \ifledRcol@ for
eledpar ........................ 47
\initnumbering@reg: \beginningnumber-
ing is defined only on eledmac, not
on eledpar. ..................... 48
\@ddc@note, \@dls@note,
\@dcs@note and \@dcsnote defined only
one time, in eledmac, including
needs for eledpar case. ...... 163
\@dgetsidenote@margin: \sidemargin@margin
is now directly defined in eled-
mac to be able to manage eled-
par. ............................ 162
\@dunhbox@line: \do@line is split
in more little commands. .... 87
\newhookcommand@series@reload:
Debug \beforenotesX and
\maxhnotesX which didn’t
work when called after
\footparagraphX ............ 145
Debug \beforenotes and
\maxhnotes which didn’t
work when called after
\footparagraph .............. 145
\pend: New optional argument for
\pend, to execute code after it. 84
\stanza: \can have an optional ar-
gument: content to be printed
after. ........................ 182
\Stanza can have an optional ar-
gument: content to be printed
before. ........................ 182
Add \newverse macro,
\falseverse deprecated. ... 182

v1.12.1.
\wrap@edcrossref: Fix spurious
spaces. ........................ 158

v1.12.2.
\@dunhbox@line: Fix a bug with
critical notes at the tops of
pages (added by v12.0.0) .... 87

v1.12.3.
General: Add macros for new mes-
sages since v0.7 .............. 42
Correct bug with side and fa-
miliar notes in tabular environ-
ments. ......................... 1
Debug \eledxxx with some pa-
paper size .................... 1
Debug \ledinnernote and
\ledouternote commands in
the top of pages. ............ 28
Debug left and right notes (bugs
added by 1.12.0) .......... 1
Underline lemma in \eledxxx
when using draft mode. ...... 1
\eledmac@error: Replaced error
messages ........................ 43
\flag@end: \flag@start and
\flag@end are now defined only
...
Change History

one time for eledmac and eledpar
\flag@start send a error message when a \edtext is done
without insert (note)
\flag@end
\flag@start don’t send a error message when a \edtext is done
without insert (note)
\flag@end

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
\dolinehook and \doinsidelinehook
\endnumbering: Allow to mix par-
allel columns and normal text
when using \pausenumbering
\@dgobblearg: \@dgobblearg be-
comes \@dgobbleoptarg
\@drestoreforedtext: Debug op-
tional arguments of \Xfootnote
in tabular context
\resumenumbering: Debug
\resumenumbering
\noeledsec: Add \noeledsec
macro
\wrap@edcrossref: \wrap@edcrossref
is now robust
\ifledRcol@: Add \ifl@dpaging
for eledpar
v1.12.6.
\para@footsetupX: In \para@footsetupX,
use \columnwidth instead of
\hsize
v1.13.1.
General: Coming back of page and
line breaking penalties’s man-
agement, deleted by error in
v0.17
Debug quotation environment in-
side of a \pstart preceded by a
sectioning command.
\the@footsetupX: Add \l@dzeropenalties
in \pstart

v1.13.2.
General: Fix bug with normal foot-
notes, added by v1.13.0
\ifledRcol@: Add \ifl@dpaging
for eledpar

v1.13.3.
General: Fix extra spaces with
paragraphed footnotes, added
by v1.13.0

v1.13.4.
General: Fix bug with index when
memoir class is used without
hyperref

v1.14.0.
General: Debug spurious characters
before endnotes
\edindex: Let eledmac take ad-
vantage of imakeidx even when
memoir class is used
\@ssect: Debug sectioning com-
mmands when using both \handout
and hyperref

v1.14.2.
\@ssect: Debug \edtext after
starred sectioning commands
when using memoir

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. ................. 130
\if@lemmacommand@: 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. .. 159
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 \imakeidx/indextools + hyperref. ............... 176
General: Add \pstartinfootnoteeverytime
................. 16
Compatibility with LuaL\LaTeX 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 ............... 156
\ifldef\rcol@: Add \ifldef\printingpages and \@d\printingcolumns for eledpar ............... 47
\l@d@section: Option parapparatus works for endnotes. ....... 136
\print@line: Compatibility with LuaL\LaTeX RTL languages. ... 87
\printlinefootnote: Code refactoring in \printlinefootnote: the printing of the numbers are factorized in \printlinefootnoteara ... 147
\printpstart: Debug \pstartinfootnote with parallel pages and columns (eledpar) ............... 104
v1.19.0.
General: \maxhXnotes and \maxhnotesX work now for both two-columns and three-columns setting. ............... 1
Compatibility with eledpar
v1.13.0. ............... 1
\footsplitskips: \footsplitskips doesn’t set \floatingpenalty to \@MM when processing parallel pages. ............... 103
\xxref: \xxref works also with right side numbers, when \Rlineflag is not empty. ... 160