

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 T_EX 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 T_EX macros, TABMAC, provides for tabular material. Another set of PLAIN T_EX macros, EDSTANZA, assists in typesetting verse.

The eledmac package makes the EDMAC, TABMAC and EDSTANZA facilities available to authors who would prefer to use L^AT_EX. 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 L^AT_EX 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 v1.16.0, last revised 2015/01/23.

[†]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 eledmac	9
1.2.3 List of works edited with (e)ledmac	9
2 The eledmac package	10
3 Text lines and paragraphs numbering	10
3.1 Text lines numbering	10
3.2 Paragraphs	10
3.3 Lineation commands	13
3.4 Changing the line numbers	14
4 The apparatus	15
4.1 Commands	15
4.2 Disambiguation of identical words in the apparatus	18
4.3 Alternate footnote formatting	19
4.4 Display options	19
4.4.1 Control line number printing	19
4.4.2 Separator between the lemma and the note content	21
4.4.3 Font style	21
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	22
4.4.7 Options for notes in columns	22
4.4.8 Options for paragraphed footnotes	23
4.4.9 Options for block of notes	23
4.5 Page layout	24
4.6 Fonts	24
4.7 Create a new series	25
5 Verse	26
5.1 Repeating stanza indents	26
5.2 Manual stanza indent	27
5.3 Stanza breaking	27
5.4 Hanging symbol	28
5.5 Long verse and page break	28
5.6 Various tools	28
5.7 Hanging symbol	29
5.8 Text before/after verses	29
6 Grouping	29

<i>Contents</i>	3
7 Crop marks	30
8 Endnotes	30
9 Cross referencing	30
10 Side notes	32
11 Familiar footnotes	33
11.1 Position of the familiar footnotes	33
12 Indexing	34
13 Tabular material	35
14 Sectioning commands	38
15 Quotation environments	39
16 Page breaks	39
17 Miscellaneous	40
17.1 Known and suspected limitations	41
17.2 Use with other packages	42
17.3 Parallel typesetting	43
17.4 Notes for EDMAC users	43
18 Implementation overview	45
19 Preliminaries	45
19.1 Package options	46
19.2 Loading packages	47
19.3 Boolean flags	47
19.4 Messages	47
19.5 Gobbling	51
19.6 Miscellaneous commands	51
20 Sectioning commands	52
21 Line counting	55
21.1 Choosing the system of lineation	55
21.2 List macros	60
21.3 Line-number counters and lists	61
21.4 Reading the line-list file	65
21.5 Commands within the line-list file	67
21.6 Writing to the line-list file	73

22 Marking text for notes	77
22.1 <code>\edtext</code> and <code>\critext</code> themselves	78
22.2 Substitute lemma	83
22.3 Substitute line numbers	84
22.4 Lemma disambiguation	84
23 Paragraph decomposition and reassembly	88
23.1 Boxes, counters, <code>\pstart</code> and <code>\pend</code>	88
23.2 Processing one line	92
23.3 Line and page number computation	94
23.4 Line number printing	97
23.5 Pstart number printing in side	101
23.6 Add insertions to the vertical list	102
23.7 Penalties	103
23.8 Printing leftover notes	104
24 Critical footnotes	104
24.1 Fonts	104
24.2 Outer-level footnote commands	105
24.3 Normal footnote formatting	106
24.4 Standard footnote definitions	112
24.5 Paragraphed footnotes	114
24.6 Insertion of the footnotes separator	120
24.7 Columnar footnotes	120
25 Familiar footnotes	125
25.1 Generality	125
25.2 Footnote formats	127
25.3 Two columns footnotes	131
25.4 Three columns footnotes	132
25.5 Paragraphed footnotes	134
26 Footnotes' width for two columns	137
27 Footnotes' order	138
28 Footnotes' rule	138
29 Footnotes' output	139
30 Endnotes	139
31 Generate series	142
31.1 Test if series is still existing	142
31.2 Create all commands to memorize display options	142
31.3 Create inserts, needed to add notes in foot	143
31.4 Create commands for critical apparatus, <code>\Xfootnote</code>	143

31.5 Create tools for familiar footnotes (<code>\footnoteX</code>)	144
31.6 The endnotes	145
31.7 Init standards series (A,B,C,D,E,Z)	146
31.8 Some tools	146
31.9 Old commands, kept for backward compatibility	150
31.10 Hooks for a particular footnote	150
31.11 Alias	150
31.12 Line number printing	150
32 Output routine	152
33 Cross referencing	158
34 Side notes	163
35 Minipages and such	169
36 Indexing	173
36.1 Memoir compatibility	174
36.2 Normal setting	176
36.3 Choose the right variant	177
36.4 Hyperref compatibility	178
37 Macro as environment	179
38 Verse	182
39 Arrays and tables	186
40 Section's title commands	206
40.1 Deprecated commands	206
40.2 New commands : <code>\eledxxx</code>	209
41 Page breaking or no page breaking depending of specific lines	219
42 Long verse: prevents being separated by a page break	220
43 The End	221
Appendix A Some things to do when changing version	222
Appendix A.1 Migration from ledmac to eledmac	222
Appendix A.2 Migration to eledmac 1.5.1	222
Appendix A.3 Migration to eledmac 1.12.0	223
References	224
Index	224

Change History

245

List of Figures

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 Maïeul Rouquette begun to maintain and extend eledmac. As plain T_EX is used by little people, and L^AT_EX 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. L^AT_EX and eledmac will take care of the formatting and visual correlation of all the disparate types of information.

The original EDMAC can be used as a 'stand alone' processor or as part of a process. One example is its use as the formatting engine or 'back end' for

the output of an automatic manuscript collation program. `COLLATE`, written by Peter Robinson, runs on the Apple Macintosh, can collate simultaneously up to a hundred manuscripts of any length, and provides facilities for the scholar to tailor the collation interactively. For further details of this and other related work, visit the EDMAC home page at <http://www.homepages.ucl.ac.uk/~ucgadkw/edmac/index.html>.

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

`EDNOTES` [Lüc03], by Uwe Lück and Christian Tapp, is another \LaTeX package being developed for critical editions. Unlike `eledmac` which is based on `EDMAC`, `EDNOTES` takes a different (internal) approach and provides a different set of features. For example it provides additional facilities for overlapping lemmas and for handling tables. For more information there is a web site at <http://ednotes.sty.de.vu> or email to ednotes.sty@web.de.

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

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

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

- Jerónimo Leal pointed me to <http://www.guit.sssup.it/latex/critical.html>. This also mentions another package for critical editions called `MauroTeX` (<http://www.maurolico.unipi.it/mtex/mtex.htm>). These sites are both in Italian.
- Dirk-Jan Dekker maintains <http://www.djdekker.net/ledmac> which is a FAQ for typesetting critical editions and `eledmac`.

This manual contains a general description of how to use the \LaTeX version of `EDMAC`, namely `eledmac` (in sections 2 through 17.4); 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 documentation in sections 2, unless you are particularly interested in the innards of `eledmac`.

1.2 History

1.2.1 EDMAC

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

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

The present version was developed in the summer of 1990, with the intent of adding necessary features, streamlining and documenting the code, and further generalizing it to make it easily adaptable to the needs of editors in different disciplines. John did most of the general reworking and documentation, with the financial assistance of the Division of the Humanities and Social Sciences, California Institute of Technology. Dominik adapted the code to the conventions of Frank Mittelbach’s `doc` option, and added some documentation, multiple-column footnotes, cross-references, and crop marks.¹ A description by John and Dominik of this version of EDMAC was published as ‘An overview of EDMAC: a PLAIN T_EX 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 T_EX 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*,² an edition of the letters of Nicolaus Copernicus,³ Simon Bredon’s *Arithmetica*,⁴ a Latin translation by Plato of Tivoli of an

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

²Gerhard 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).

³Being prepared at the German Copernicus Research Institute, Munich.

⁴Being prepared by Menso Folkerts *et al.*, at the Institut für Geschichte der Naturwissenschaften in Munich.

Arabic astrolabe text,⁵ a Latin translation of part II of the Arabic *Algebra* by Abū Kāmil Shujā' b. Aslam,⁶ the Latin *Rithmachia* of Werinher von Tegernsee,⁷ a middle-Dutch romance epic on the Crusades,⁸ a seventeenth-century Hungarian politico-philosophical tract,⁹ an anonymous Latin compilation from Hungary entitled *Sermones Compilati in Studio Generali Quinqueecclesiensi in Regno Ungarie*,¹⁰ the collected letters and papers of Leibniz,¹¹ Theodosius's *Spherics*, the German *Algorismus* of Sacrobosco, the Sanskrit text of the *Kāśīkāvṛtti* of Vāmana and Jayāditya,¹² 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.19). 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 A.1 (p.222).

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/items. Please add your own edition made with (e)ledmac.

⁵Richard Lorch, Gerhard Brey *et al.*, at the same Institute.

⁶Richard Lorch, 'Abū Kāmil on the Pentagon and Decagon' in *Vestigia Mathematica*, ed. M. Folkerts and J. P. Hogendijk (Amsterdam, Atlanta: Rodopi, 1993).

⁷Menso Folkerts, 'Die *Rithmachia* des Werinher von Tegernsee', *ibid.*

⁸Geert H. M. Claassens, *De Middelnederlandse Kruisvaartromans*, (Amsterdam: Schiphoewer en Brinkman, 1993).

⁹Emil Hargittay, *Csáky István: Politica philosophiai Okoskodás-szerint való rendes életnek példája (1664–1674)* (Budapest: Argumentum Kiadó, 1992).

¹⁰Being produced, as was the previous book, by Gyula Mayer in Budapest.

¹¹Leibniz, *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>)

¹²Being prepared at Poona and Lausanne Universities.

2 The eledmac package

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

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

3 Text lines and paragraphs numbering

3.1 Text lines numbering

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

```
\beginnumbering
<text>
\endnumbering
```

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

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

3.2 Paragraphs

`\pstart` Within a numbered section, each paragraph of numbered text must be marked
`\pend`

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:

<code>\beginnumbering</code>	
<code>\pstart</code>	
This is a sample paragraph, with	
lines numbered automatically.	
<code>\pend</code>	1 This is a sample paragraph
	2 with lines numbered
<code>\pstart</code>	3 automatically.
This paragraph too has its	4 This paragraph too
lines automatically numbered.	5 has its lines automatically
<code>\pend</code>	6 numbered.
The lines of this paragraph are	The lines of this paragraph
not numbered.	are not numbered.
<code>\pstart</code>	7 And here the numbering
And here the numbering begins	8 begins again.
again.	
<code>\pend</code>	
<code>\endnumbering</code>	

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

Both `\pstart` and `\pend` can take a 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.

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

```

\begingroup
  \beginnumbering
  \autopar

  A paragraph of numbered text.      1 A paragraph of numbered
                                     2 text.

  Another paragraph of numbered      3 Another paragraph of
  text.                               4 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.¹³

By default, `eledmac` 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 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}

\firstsublinenum There are similar commands, \firstsublinenum{<num>} and \sublinenumincrement{<num>}
\sublinenumincrement for controlling sub-line numbering.
\numberpstarttrue You can use the command \numberpstarttrue to insert a number on every
                    \pstart. To stop the numbering, you must use \numberpstartfalse. To reset
                    the numbering of \pstarts, insert

\setcounter{pstart}{0}

```

`\pausenumbering` and `\resumenumbering` `eledmac` stores a lot of information about line numbers 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 `LATEX` may reach its memory limit. There are two solutions to this. The first is to get a larger `LATEX` 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:

¹³For a detailed study of the reasons for this restriction, see Barbara Beeton, 'Initiation rites', *TUGboat* **12** (1991), pp. 257–258.

```

\beginnumbering
\pstart
Paragraph of text.
\pend
\pausenumbering
1 Paragraph of
2 text.

\resumenummering
\pstart
3 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\resumenummering}
```

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

```

\numberpstarttrue
\numberpstartfalse
\thepstart

```

It's 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. On each `\beginnumbering` the numbering restarts.

With the `\sidepstartnumtrue` command, the number of `\pstart` will be printed in side. 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`.

3.3 Lineation commands

```

\numberlinefalse
\numberlinetrue
\lineation

```

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

```
\linenummargin
```

The command `\linenummargin<location>` specifies the margin where the line (or `pstart`) numbers will be printed. The permissible value for `<location>` is one out of the list `left`, `right`, `inner`, or `outer`, for example `\linenummargin{inner}`. The package's default setting is

```
\linenummargin{left}
```

to typeset the numbers in the left hand margin. You can change this whenever you're not in the middle of making a paragraph.

More precisely, the value of `\linenummargin` used is that in effect at the `\pend` of a numbered paragraph. Apart from an initial setting for `\linenummargin`,

only change it after a `\pend`, whereupon it will apply to all following numbered paragraphs, until changed again (changing it between a `\pstart` and `\pend` pair will apply the change to all the current paragraph).

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

In most cases, you will not want a number printed for every single line of the text. Four L^AT_EX counters control the printing of marginal numbers and they can be set by the macros `\firstlinenum{<num>}`, etc. `\firstlinenum` specifies the number of the first line in a section to number, and `\linenumincrement` is the increment between numbered lines. `\firstsublinenum` and `\sublinenumincrement` do the same for sub-lines. Initially, all these are set to 5 (e.g., `\firstlinenum{5}`).

`\linenumberlist`

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.

`\leftlinenum`
`\rightlinenum`
`\linenumsep`

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

3.4 Changing the line numbers

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

`\startsub`
`\endsub`

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

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

`\startlock`
`\endlock`

The `\startlock` command, used in running text, locks the line number at its current value, until you say `\endlock`. It can tell for itself whether you are in a patch of line or sub-line numbering. One use for line-number locking is in printing

poetry: there the line numbers should be those of verse lines rather than of printed lines, even when a verse line requires several printed lines.

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

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

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

\linenumberstyle Line numbers are normally printed as arabic numbers. You can use **\linenumberstyle{<style>}**
\sublinenumberstyle to change the numbering style. **<style>** must be one of:

Alph Uppercase letters (A...Z).

alph Lowercase letters (a...z).

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

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

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

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

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

\skipnumbering 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:

<code>I saw my friend \edtext{Smith}{</code>	1 I saw my friend
<code>\Afootnote{Jones C, D.}}</code>	2 Smith on Tuesday.
<code>on Tuesday.</code>	<u>2</u> Smith] Jones C, D.

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:

<code>\edtext{I saw my friend</code>	1 I saw my friend
<code>\edtext{Smith}{\Afootnote{Jones</code>	2 Smith on Tuesday.
<code>C, D.}} on Tuesday.}{</code>	<u>2</u> Smith] Jones C, D.
<code>\Bfootnote{The date was</code>	<u>1–2</u> I saw my friend
<code>July 16, 1954.}</code>	Smith on Tuesday.] The
<code>}</code>	date was July 16, 1954.

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

Commands used in `\edtext`'s second argument The second argument of the `\edtext` macro, `⟨commands⟩`, may contain a series of subsidiary commands that generate various kinds of notes.

<code>\Afootnote</code>	Five separate series of the footnotes are maintained; each macro taking one
<code>\Bfootnote</code>	argument like <code>\Afootnote{⟨text⟩}</code> . When all five are used, the A notes appear
<code>\Cfootnote</code>	in a layer just below the main text, followed by the rest in turn, down to the E
<code>\Dfootnote</code>	notes at the bottom. These are the main macros that you will use to construct
<code>\Efootnote</code>	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:

- `nonum` to disable line numbering for this note.
- `nosep` to disable the lemma separator for this note.

Example: `\Afootnote[nonum]{text}`.

`\Aendnote` The package also maintains five separate series of endnotes. Like footnotes
`\Bendnote` each macro takes a single argument like `\Aendnote{text}`. Normally, none of
`\Cendnote` them are printed: you must use the `\doendnotes` macro described below (p.30)
`\Dendnote` to call for their output at the appropriate point in your document.

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

`\usepackage[parapparatus]{eledmac}`

`\lemma` If you want to change the lemma that gets passed to the notes, you can do this
 by using `\lemma{alternative}` within the second argument to `\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:

<code>\edtext{I saw my friend</code>	
<code>\edtext{Smith}{\Afootnote{Jones</code>	1 I saw my friend
<code>C, D.}} on Tuesday.</code>	2 Smith on Tuesday.
<code>{\lemma{I \dots\ Tuesday.}}</code>	<u>2 Smith</u>] Jones C, D.
<code>\Bfootnote{The date was</code>	<u>1-2 I ... Tuesday.]</u>
<code>July 16, 1954.}</code>	The date was July 16, 1954.
<code>}</code>	

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

This command doesn't change the marginal line numbers in any way; it just
 changes the numbers passed to the footnotes. Its use comes in situations that
`\edtext` has trouble dealing with for whatever reason. If you need notes for
 overlapping passages that aren't nested, for instance, you can use `\lemma` and
`\linenum` to generate such notes despite the limitations of `\edtext`. If the
`<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.30) to compute them
 automatically.

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

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

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

4.2 Disambiguation of identical words in the apparatus

Sometimes, the same word occurs twice (or more time) in the same line. `eledmac` provides tools to 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.

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

¹⁴We use `\newcommand` and `\newcommandx` instead of classical `\let` command because the edtabular 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`.

```
\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` 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 $\langle l \rangle$, is used, it can be stretchable: **a plus b minus c**. The final length m is calculated by \LaTeX to have: $a - c \leq m \leq a + b$. If you use relative unity¹⁵, 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

¹⁵Like `em` which is the width of a M.

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⟩]` 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.

`\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⟩]` to cancel it (`<s>` can be empty if you want to disable it for every series).

`\beforesymlinenum`

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

`\aftersymlinenum`

With `\aftersymlinenum[⟨s⟩]{⟨l⟩}` you can add some space after the line symbol 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}
```

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

4.4.2 Separator between the lemma and the note content

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

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

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

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

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

4.4.3 Font style

`\Xnotenumfont` `\Xnotenumfont[⟨s⟩]{⟨command⟩}` is used to change the font style for line numbers in critical footnotes ; `⟨command⟩` must be one (or more) switching command, like `\bfseries`.

`\Xendnotenumfont` `\Xendnotenumfont[⟨s⟩]{⟨command⟩}` is used to change the font style for line numbers in critical footnotes. `⟨command⟩` must be one (or more) switching command, like `\bfseries`.

`\notenumfontX` `\notenumfontX[⟨s⟩]{⟨command⟩}` is used to change the font style for note numbers in familiar footnotes. `⟨command⟩` must be one (or more) switching command, like `\bfseries`.

`\Xnotefontsize` `\Xnotefontsize[⟨s⟩]{⟨command⟩}` is used to define the font size of critical footnotes of the series. The default value is `\footnotesize`. The `⟨command⟩` must not be a size in pt, but a standard L^AT_EX size, like `\small`.

`\notefontsizeX` `\notefontsizeX[⟨s⟩]{⟨command⟩}` is used to define the font size of critical footnotes of the series. The default value is `\footnotesize`. The `⟨command⟩` must not be a size in pt, but a standard L^AT_EX size, like `\small`.

`\Xendnotefontsize` `\Xendnotefontsize[⟨s⟩]{⟨l⟩}` is used to define the font size of end critical footnotes of the series. The default value is `\footnotesize`. The `⟨command⟩` must not be a size in pt, but a standard L^AT_EX size, like `\small`.

4.4.4 Font of the lemma

`\Xlemmadisablefontselection` 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 `\Xlemmadisablefontselection[⟨s⟩]` command allows to disable it for a specific series.

`\Xendlemmadisablefontselection` 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 `\Xendlemmadisablefontselection[⟨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⟩]{⟨l⟩}`, which will be applied in the second line of notes. It can help to make distinction between a new note and a break in a note. The default value is 0 pt.

`\hangindentX` For familiar notes NOT paragraphed you can define an indent with `\Xhangindent[⟨s⟩]{⟨l⟩}`, which will be applied in the second line of notes. It can help to make a distinction between a new note and a break in 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}}.}}
```

`\bhookXnote` `\bhookXnote[⟨series⟩]{⟨code⟩}` is to be used at the beginning of the critical footnotes.

`\bhooknoteX` `\bhooknoteX[⟨series⟩]{⟨code⟩}` is to be used at the beginning of the familiar footnotes.

`\bhookXendnote` `\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` `\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` `\hsizethreecol[⟨s⟩]{⟨l⟩}` is used to change width of a column when critical notes are displaying in three columns. Default value is `.3 \hsize`.

`\hsizetwocolX` `\hsizetwocolX[⟨s⟩]{⟨l⟩}` is used to change width of a column when familiar notes are displaying in two columns. Default value is `.45 \hsize`.

`\hsizethreecolX` `\hsizethreecolX[⟨s⟩]{⟨l⟩}` is used to change width of a column when familiar notes are displaying in three columns. Default value is `.3 \hsize`.

4.4.8 Options for paragraphed footnotes

<code>\afternote</code>	You can add some space after a note by using <code>\afternote[⟨s⟩]{⟨l⟩}</code> . The default value is <code>1em plus .4em minus .4em</code> .
<code>\parafootsep</code>	For paragraphed footnotes (see below), you can choose the separator between each note by <code>\parafootsep[⟨s⟩]{⟨l⟩}</code> . A common separator is a double pipe (<code>\$ \$</code>), which you can set by <code>\parafootsep{\$ \$}</code> .
<code>\Xragged</code>	Text in paragraphed critical notes is justified, but you can use <code>\Xragged[⟨s⟩]+L+</code> if you want it to be ragged left, or <code>\Xragged[⟨s⟩]+R</code> if you want it to be ragged right.
<code>\raggedX</code>	Text in paragraphed footnotes is justified, but you can use <code>\raggedX[⟨s⟩]+L+</code> if you want it to be ragged left, or <code>\raggedX[⟨s⟩]+R</code> if you want it to be ragged right.

4.4.9 Options for block of notes

<code>\txtbeforeXnotes</code>	You can add some text before critical notes with <code>\txtbeforeXnotes[⟨s⟩]{⟨text⟩}</code> .
<code>\beforeXnotes</code>	<p>You can change the vertical space printed before the rule of the critical notes with <code>\beforeXnotes[⟨s⟩]{⟨l⟩}</code>. The default value is <code>1.2em plus .6em minus .6em</code>.</p> <p>Be careful, the standard L^AT_EX footnote rule, which is used by <code>eledmac</code>, decreases by 3pt. This 3pt decrease is not changed by this command..</p>
<code>\beforenotesX</code>	<p>You can change the vertical space printed before the rule of the familiar notes with <code>\beforenotesX[⟨s⟩]{⟨l⟩}</code>. The default value is <code>1.2em plus .6em minus .6em</code>.</p> <p>Be careful, the standard L^AT_EX footnote rule, which is used by <code>eledmac</code>, decreases 3pt. These 3pt are not changed by this command.</p>
<code>\afterXrule</code>	<p>You can change the vertical space printed after the rule of the critical notes with <code>\afterXrule[⟨s⟩]{⟨l⟩}</code>. The default value is <code>0pt</code>.</p> <p>Be careful, the standard L^AT_EX footnote rule, which is used by <code>eledmac</code>, adds 2.6pt. These 2.6pt are not changed by this command.</p> <p>Be careful with this setting: it can place notes by the page number, at the bottom of the page.</p>
<code>\afterruleX</code>	<p>You can change the vertical space printed after the rule of the familiar notes with <code>\beforenotesX[⟨s⟩]{⟨l⟩}</code>. The default value is <code>0pt</code>.</p> <p>Be careful, the standard L^AT_EX footnote rule, which is used by <code>eledmac</code>, adds 2.6pt. These 2.6pt are not changed by this command.</p> <p>Be careful with this setting: it can place notes by the page number, at the bottom of the page.</p>
<code>\preXnotes</code>	<p>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 <code>\preXnotes{⟨l⟩}</code>. Default value is <code>0pt</code>. You can disable this feature by setting the length to <code>0pt</code>.</p> <p>Be careful with this setting: it can place notes by the page number, at the bottom of the page.</p>
<code>\prenotesX</code>	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 <code>0pt</code> . You can

do it with `\prenotesX{<l>}`. You can disable this feature by setting the length to 0 pt.

`\maxhXnotes` 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\textheight}`.

`\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

`Xnoteswidthliketwocolumns`
`notesXwidthliketwocolumns`

If you use `eledpar` `\columns` macro, you can call :

- `\Xnoteswidthliketwocolumns[<s>]` to create critical notes with a two-column size width. Use `\Xnoteswidthliketwocolumns[<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`.

`\numlabfont`

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

¹⁶There is one tiny proviso about using paragraphed notes: you shouldn't force any explicit line-breaks inside such notes: do not use `\par`, `\break`, or `\penalty=-10000`. If you must have a line-break for some obscure reason, just suggest the break very strongly: `\penalty=-9999` will do the trick. Page 117 explains why this restriction is necessary.

`\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.

`\endashchar`
`\fullstop`
`\rbracket`

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 T_EX they are taken from a math font which does not have the en-dash or full stop in the same places as a text font. If you (or your macros) just typed `$\oldstyle 12--34$` or `$\oldstyle 55.6$` you would get ‘12”34’ and ‘55>6’. So we define `\endashchar` and `\fullstop`, which produce an en-dash and a full stop respectively from the normal document font, whatever font you are using for the numbers. These two macros are used in the macros which format the line numbers in the margins and footnotes, instead of explicit punctuation. We also define an `\rbracket` macro for the right square bracket printed at the end of the lemma in many styles of textual notes (including `eledmac`’s standard style). For polyglossia, when the lemma is RTL, the bracket automatically switches to a left bracket.

`\select@lemmafont`

We will briefly discuss `\select@lemmafont` 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@lemmafont` 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@lemmafont` 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@lemmafont` selects the appropriate font for the note using that font specifier.

`eledmac` uses `\select@lemmafont` 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 Sullivan¹⁷ 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^AT_EX `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.

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

`\setstanzaindent` 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
`\setstanzaindent{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; T_EX 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 indentations, and say they are repeated, defining the value of the `stanzaindentrepetition` counter at n . For example:

```
\setstanzaindent{5,1,0}
\setcounter{stanzaindentrepetition}{2}
```

is like

¹⁷Department of Mathematics, University College, Dublin 4, Ireland

```
\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.2 p. 222.

If you don't use the `stanzaindentsrepetition` counter, make sure you have at least one more numerical entry in `\setstanzavalues` 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 T_EX's grouping conventions, so if one stanza among several has a different structure, its indentations (penalties) may be set within a group; the prior values will be restored when the group ends.

5.2 Manual stanza indent

```
\stanzaindent
\stanzaindent*
```

You can set the indent of some specific verse by calling `\stanzaindent{⟨value⟩}` 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 word, using `\stanzaindent` in a verse does not shift the indent repetition.

However, if you want to shift the indent repetition, and so the next verse to have the indent which should have been 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 T_EX, 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 then example above could be omitted. The control sequence

`\endstanzaextra` can be defined to include a penalty. A penalty of 10000 will prevent a page break; such a penalty is included automatically where there is stanza hanging indentation. A penalty of -10000 (corresponding to the entry value 20000 in this context) forces a page break. Values in between act as suggestions as to the desirability of a page break at a given line. There is a subtle interaction between penalties and *glue*, so it may take some adjustment of skips and penalties to achieve the best results.

5.4 Hanging symbol

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

```
\renewcommand{\hangingsymbol}{[,]}
```

You can also use it to force hanging verse to be flush right:

```
\renewcommand{\hangingsymbol}{\protect\hfill}
```

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

<code>\ampersand</code>	If you need to print an <code>&</code> symbol in a stanza, use the <code>\ampersand</code> macro, not <code>\&</code> which will end the stanza.
<code>\endstanzaextra</code>	The macro <code>\endstanzaextra</code> , 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 <code>memoir</code> class, it provides a length <code>\stanzaskip</code> which may come in handy.
<code>\startstanzahook</code>	Similarly, if <code>\startstanzahook</code> is defined, it is called by <code>\stanza</code> at the start. This can be defined to do something.
<code>\flagstanza</code>	Putting <code>\flagstanza[⟨len⟩]{⟨text⟩}</code> at the start of a line in a stanza (or elsewhere) will typeset <code>⟨text⟩</code> at a distance <code>⟨len⟩</code> before the line. The default <code>⟨len⟩</code> is <code>\stanzaindentbase</code> .

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

`\hangingsymbol` 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}{[\,]}
```

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.

`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.

`ledgroup` 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.

`ledgroupsize` The `ledgroupsize` environment is similar to `ledgroup` except that you must specify a width for the environment, as with a minipage.
`\begin{ledgroupsize}[\langle pos \rangle]{\langle width \rangle}`.

The required $\langle width \rangle$ argument is the text width for the environment. The optional $\langle pos \rangle$ 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{ledgroupsize}{\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` `\doendnotes{ $\langle letter \rangle$ }` 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@lemmafnt` 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{ $\langle num \rangle$ }` is used to print these numbers. Its default definition is:

`\newcommand*{\printnpnum}[1]{p.#1}`

`\noendnotes` 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` First you place a label in the text using the command `\edlabel{ $\langle lab \rangle$ }`. $\langle lab \rangle$ can be almost anything you like, including letters, numbers, punctuation, or a combination—anything but spaces; you might say `\edlabel{toves-3}`, for example.¹⁸

`\edpageref` Elsewhere in the text, either before or after the `\edlabel`, you can refer to
`\lineref`
`\sublineref`
`\pstartref`

¹⁸More precisely, you should stick to characters in the T_EX categories of 'letter' and 'other'.

its location via `\edpageref{<lab>}`, or `\lineref{<lab>}`, `\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`, `\lineref`, `\sublineref`, `\pstartref` commands can also be used in the apparatus to refer to `\edlabels` in the text.

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

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

If you want to refer to a word inside an `\edtext{...}{...}` command, the `\edlabel` should be defined inside the first argument, e.g.,

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

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

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

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

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

the place where `\edlabel{mouse}` was placed, and the ending numbers to those where `\edlabel{elephant}` occurs.

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

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

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 `\sidenotemargin{<location>}`. The permissible value for `<location>` is one out of the list `left`, `right`, `inner`, or `outer`, for example `\sidenotemargin{outer}`. The package’s default setting is `\sidenotemargin{right}`

to typeset `\ledsidenotes` in the right hand margin. This is the opposite to the default margin for line numbers. The style for a `\ledsidenote` follows that for a `\ledleftnote` or a `\ledrightnote` depending on the margin it is put in.

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.

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

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

`\ledlsnotesep` The texts are put a distance `\ledlsnotesep` (or `\ledrsnotesep`) into the left
`\ledrsnotesep` (or right) margin. These lengths are initially set to the value of `\linenumsep`.
`\ledlsnotefontsetup` These macros specify how the sidenote texts are to be typeset. The initial
`\ledrsnotefontsetup` definitions are:

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


These can of course be changed to suit.

`\sidenoteseq` 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 `\sidenoteseq`.

11 Familiar footnotes

The `footmisc` package [Fai03] by Robin Fairbairns has an option whereby sequential footnote marks in the text can be separated by commas^{3,4} like so. As a convenience `eledmac` provides this automatically.

`\multfootsep` `\multfootsep` is used as the separator between footnote markers. Its default definition is:

```
\providecommand*{\multfootsep}{\textsuperscript{\normalfont,}}
```

and can be changed if necessary.

`\footnoteA` As well as the standard L^AT_EX 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.

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

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

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

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

```
\newcommand*{\bodyfootmarkA}{%
```

```
\hbox{\textsuperscript{\normalfont\@nameuse{@thefnmarkA}}}}
```

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

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

There are similar command triples for the other series.

Additional footnote series can be easily defined: you just have to use `\newseries`, defined above (see 4.7 p.25).

11.1 Position of the familiar footnotes

`\fnpos` There is a historical incoherence in (e)ledmac. The familiar footnotes are before
`\mpfnpos` 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 L^AT_EX provides the `\index{⟨item⟩}` command for specifying that `⟨item⟩` and the current page number should be added to the raw index (`idx`) file. The `\edindex{⟨item⟩}` macro can be used in numbered text to specify that `⟨item⟩` and the current page & linenum should be added to the raw index file.

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

The minimal version of `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 `imakeidx` or `indextools` .
2. Load `eledmac`.
3. Declare the index with the macro `\makeindex` of `imakeidx`/`indextools`.

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

\edindexlab The `\edindex` process uses a `\label/\ref` mechanism to get the correct line number. It automatically generates labels of the form `\label{\edindexlab N}`, where `N` is a number, and the default definition of `\edindexlab` is:

```
\newcommand*{\edindexlab}{\&}
```

in the hopes that this will not be used by any other labels (`\edindex`'s labels are like `\label{\&27}`). You can change `\edindexlab` to something else if you need to.

13 Tabular material

L^AT_EX's normal `tabular` and `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, `eledmac` provides some simple tabulation environments that can be line numbered. The environments can also be used in normal unnumbered text.

There are six environments; the `edarray*` environments are for math and `edtabular*` for text entries. The final `l`, `c`, or `r` in the environment names indicate that the entries will be flushleft (`l`), centered (`c`) or flushright (`r`). There is no means of specifying different formats for each column, nor for specifying a fixed width for a column. The environments are centered with respect to the surrounding text.

```

edarrayl
edarrayc
edarrayr
edtabularl
edtabularc
edtabularr
\begin{edtabularc}
1 & 2 & 3 \\
a & bb & ccc \\
AAA & BB & C
\end{edtabularc}

```

1	2	3
a	bb	ccc
AAA	BB	C

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 \mathcal{E}) 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:

```

\beginnumbering
\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
\endnumbering

```

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.

for use in edtabular environments.

1 & 2 & 3 & 4 \\\

1 2 3 4

$$F + G + C$$
 $a \quad bb \quad ccc \quad dddd$

example `\hrulefill` or `\upbracefill`.

would not appear to be necessary.

The typeset appearance of the following code is shown below.

1 & 2 & 3 & 4 & 5 \\\

Q & & fd & h & qwertziohg \\\

```
v      & wptz & x  & y  & vb \\
```

```
& nnn & \edrowfill{3}{5}{\upbracefill} & & \\
```

$$\downarrow \text{edrowfill}\{1\}\{3\}\{\downarrow \text{downbracefill}\} \quad \& \quad \& \quad \& \text{pq} \quad \& \text{dgh} \quad \backslash \backslash$$

```
k          &          & l  & co & ghweropjklmnbvcxys \\
```

```
1 & 2 & 3 & \edrowfill{4}{5}{\hrulefill} &
```

$$\end{tabular}$$

1	2	3	4	5
Q		fd	h	qwertziohg
v	wptz	x	y	vb
g	nnn			
		pq dgh		
k		l	co	ghweropjklmnbvexys
1	2	3		

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

$$\backslash newcommand*{\upbracketfill}{\%$$

instead of a brace. It can be used like this:

1 & 2 & 3 & 4 \\\

a & \edrowfill{2}{3}{\upbracketfill} & & d \\

A & B & C & D

\end{edarrayc}

$$\begin{array}{cccc} 1 & 2 & 3 & 4 \\ a & \rule{1cm}{0.4pt} & d & \\ A & B & C & D \end{array}$$

`\edatleft` `\edatleft[$\langle math \rangle\{\langle symbol \rangle\}\{\langle halfheight \rangle\}$]` typesets the math $\langle symbol \rangle$ as `\left<symbol>` with the optional $\langle math \rangle$ centered before it. The $\langle symbol \rangle$ is twice $\langle halfheight \rangle$ tall. The `\edatright` macro is similar and it typesets `\right<symbol>` with $\langle math \rangle$ centered after it.

```
\begin{edarrayc}
& 1 & 2 & 3 & \\
& 4 & 5 & 6 & \\
\edatleft[left =]{\{1.5\baselineskip}
& 7 & 8 & 9 & \\
\edatright[= right]{\{1.5\baselineskip}
\end{edarrayc}
```

$$left = \left(\begin{array}{ccc} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{array} \right) = right$$

`\edbeforetab` `\edbeforetab{ $\langle text \rangle\{\langle entry \rangle\}$ }`, where $\langle entry \rangle$ is an entry in the leftmost column, typesets $\langle text \rangle$ left justified before the $\langle entry \rangle$. Similarly `\edaftertab{ $\langle entry \rangle\{\langle text \rangle\}$ }`, 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{8}{After} & \\
D & 1 & 5 & 0 & \\
\end{edarrayl}
```

	$\begin{array}{cccc} A & 1 & 2 & 3 \\ B & 1 & 3 & 6 \\ C & 1 & 4 & 8 \\ D & 1 & 5 & 0 \end{array}$		After
Before			

`\edvertline` The macro `\edvertline{ $\langle height \rangle\}$` 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 & & \end{arrayr}
```

```

m & n & o & p & \\\
k & & L & cvb & \edvertline{4pc}
\end{edarrayr}

```

<i>a</i>	<i>b</i>	<i>C</i>	<i>d</i>	
<i>v</i>	<i>w</i>	<i>x</i>	<i>y</i>	
<i>m</i>	<i>n</i>	<i>o</i>	<i>p</i>	
<i>k</i>		<i>L</i>	<i>cvb</i>	

The `\edvertdots` macro is similar to `\edvertline` except that it produces a vertical dotted instead of a solid line.

14 Sectioning commands

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

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

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

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

- `\eledsubsubsection[⟨text⟩]{⟨critical text⟩}`
- `\eledsubsubsection*`

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

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

At the first run, you will see only the text. It's normal. At the second run, you will see the formatting. And consequently, at the third run, you will see the table of contents.

For technical reasons, the page break before `\elechapter` can't be added automatically. You have to insert it manually via `\beforeeledchapter`, 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.

`\ledpbsetting` 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 `\ledpbsetting{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. $n + 1$.

`\lednopbinversetrue` 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 (better: in your preamble). This feature works only with verse of 2 lines, not more. It works at the third run, or at fourth run with `eledpar`. By default, when a long verse runs normally between two pages, a page break will be placed at the beginning of the verse. However, if you have added `\ledpbsetting{after}`, the page break will be placed at the end of the long verse, and the page containing the long verse will have one extra line.

17 Miscellaneous

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

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

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

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

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

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

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

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

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


17.1 Known and suspected limitations

In general, **eledmac**'s system for adding marginal line numbers breaks anything that makes direct use of the **L^AT_EX** insert system, which includes `marginpars`, footnotes and floats.

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

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

`\ballast` **L^AT_EX** 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 **T_EX** never settle down. At each successive run, **eledmac** may oscillate between two different sets of page decisions. To stop this happening, should it arise, Wayne Sullivan suggested the inclusion of the quantity `\ballast`. The amount of `\ballast` will be subtracted from the penalties which apply to the page breaks calculated on the *previous* run through **T_EX**, thus reinforcing these breaks. So if you find your page breaks oscillating, say

```
\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 16, p. 24, and described in more detail on p. 117, really is a nuisance if that's something you need to do. There are some possible solutions, described by Michael Downes, but this area remains unsatisfactory.

L^AT_EX has a reputation for putting things in the wrong margin after a page break. The **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.

`\pageparbreak` If you can't resist flipping the numbers or numbering by the page, then you might find that judicious use of `\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 `\pageparbreak` accordingly.

`\footfudgefiddle` For paragraphed footnotes **T_EX** 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 68) to increase the estimate. You have to use `\renewcommand` for this, like:

```
\renewcommand{\footfudgefiddle}{68}
```

Help, suggestions and corrections will be gratefully received.

17.2 Use with other packages

Because of `eledmac`'s complexity it may not play well with other packages. In particular `eledmac` is sensitive to commands in the arguments to the `\edtext` and `*footnote` macros (this is discussed in more detail in section 22, and in particular the discussion about `\no@expands` and `\morenoexpands`). You will have to see what works or doesn't work in your particular case.

It is possible that `eledmac` and the `hyperref` package may work together. I have not tried this combination but past experience with `hyperref` suggests that cooperation is unlikely; `hyperref` changes many L^AT_EX internals and `eledmac` does things that are not normally seen in L^AT_EX.

If you want to use the option *bottom* of the `footmisc` package, you must load this package *before* the `eledmac` package.

`\morenoexpands` 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:

```
... \edtext{\colorbox{mycolor}{lemma}}{\Afootnote{...\colorbox{...}}}
```

If you actually try this¹⁹ you will find L^AT_EX whinging 'Missing { inserted', and then things start to fall apart. The trick in this case is to specify either:

```
\newcommand{\morenoexpands}{\let\colorbox=0}
```

or

```
\makeatletter
\newcommand{\morenoexpands}{\let\colorbox\@secondoftwo}
\makeatother
```

(`\@secondoftwo` is an internal L^AT_EX 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

```
... \edtext{\textcolor{mycolor}{lemma}}{\Afootnote{...\textcolor{...}}}
```

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:

```
\makeatletter
```

¹⁹Reported by Dirk-Jan Dekker in the CTT thread 'Incompatibility of "color" package' on 2003/08/28.

```
\newcommand{\morenoexpands}{\let\textcolor\@secondoftwo}
\makeatother
```

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` / `polyglossia` packages for typesetting in multiple languages. The package has been called *eledpar* since September 2012.

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

17.4 Notes for EDMAC users

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 changed²⁰ 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 <code>\critext{Smith}</code>	1 I saw my friend
<code>\Afootnote{Jones C, D.}</code>	2 Smith on Tuesday.
on Tuesday.	2 Smith] Jones C, D.

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.

²⁰A name like `\text` is likely to be defined by other L^AT_EX packages (it certainly is by the AMS packages) and it seems sensible to try and avoid clashes with other definitions.

The $\langle lemma \rangle$ 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:

<code>\critext{I saw my friend</code>	1 I saw my friend
<code>\critext{Smith}{\Afootnote{Jones</code>	2 Smith on Tuesday.
<code>C, D.}/ on Tuesday.}</code>	<u>2 Smith</u>] Jones C, D.
<code>\Bfootnote{The date was</code>	
<code>July 16, 1954.}</code>	<u>1–2</u> I saw my friend
<code>/</code>	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 $\langle lemma \rangle$ argument of another `\critext` must end there, too. (The `\lemma` and `\linenum` commands may be used to generate overlapping notes if necessary.)

The second argument of the `\critext` macro, $\langle commands \rangle$, 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:

```
\catcode'\<=\active
\let<=\critext
```

Then you might say `<{Smith}\Afootnote{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:

```
\catcode'\<=\active
\def\xtext#1#2>{\critext{#1}{#2}/}
\let<=\xtext
```

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`. (See section 22 to find out why.) Aliases of the second kind may be nested without any problem.

If you really have to use `\critext` in any of the tabular or array environments, then `\edtext` must not be used in the same environment. If you use `\critext` in one of these environments then you have to issue the declaration `\usingcritext` beforehand. The declaration `\usingedtext` must be issued to revert to the default assumption that `\edtext` will be used.

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 $\text{\LaTeX}2\text{\epsilon}$.

```
1 \code
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{eledmac}[2015/01/23 v1.16.0 LaTeX port of EDMAC]%
```

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

- Replace as many `\def`'s by `\newcommand`'s as possible to avoid overwriting \LaTeX macros.
- Replace user-level \TeX counts by \LaTeX counters.
- Use the \LaTeX font handling mechanisms.
- Use \LaTeX messaging and file facilities.

19.1 Package options

```

\ifledfinal      Use this to remember which option is used, set and execute the options with final
\ifparapparatus@ as the default.
\ifnoquotation@ 4 \newif\ifledfinal
\iflednopbinverse 5 \newif\ifparapparatus@
\ifparledgroup    6 \newif\ifnoquotation@
\ifwidthliketwocolumns 7 \newif\iflednopbinverse
\ifledsecnolinenumber 8 \newif\ifparledgroup
                  9 \newif\ifwidthliketwocolumns%
                  10 \newif\ifledsecnolinenumber
                  11 \parapparatus@false
                  12 \DeclareOption{noquotation}{\noquotation@true}
                  13 \DeclareOption{final}{\ledfinaltrue}
                  14 \DeclareOption{draft}{\ledfinalfalse}
                  15 \DeclareOption{parapparatus}{\parapparatus@true}
                  16 \DeclareOption{nopbinverse}{\lednopbinversetrue}
                  17 \DeclareOption{ledsecnolinenumber}{\ledsecnolinenumbertrue}
                  18 \DeclareOption{widthliketwocolumns}{\widthliketwocolumnstrue}%
                  19 \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.

20 \ProcessOptions*\relax
21

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 iflutex to test if LuaLaTeX is running,
and ragged2e to manage ragging for paragraphed notes.

22 \RequirePackage{xargs}
23 \RequirePackage{etoolbox}
24 \reserveinserts{32}
25 \RequirePackage{suffix}
26 \RequirePackage{xstring}
27 \RequirePackage{iflutex}
28 \RequirePackage{ragged2e}

\if@RTL The \if@RTL is defined by the bidi package, which is sometimes loaded by poly-
glossia. But we define it if the bidi package is not loaded.
29 \ifdef{\if@RTL}{\newif\if@RTL}

\showlemma \showlemma{<lemma>} typesets the lemma text in the body. It depends on the
option.
30 \ifledfinal

```

```

31 \newcommand*{\showlemma}[1]{#1}
32 \else
33 \newcommand*{\showlemma}[1]{\underline{#1}}
34 \fi
35

```

19.2 Loading packages

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, *ifluatex* to test if LuaLaTeX is running, and *ragged2e* to manage ragged for paragraphed notes.

```

36 \RequirePackage{xargs}
37 \RequirePackage{etoolbox}
38 \reserveinserts{32}
39 \RequirePackage{suffix}
40 \RequirePackage{xstring}
41 \RequirePackage{ifluatex}
42 \RequirePackage{ragged2e}

```

19.3 Boolean flags

`\ifl@dmemoir` Define a flag for if the memoir class has been used.

```

43 \newif\ifl@dmemoir
44 \@ifclassloaded{memoir}{\l@dmemoirtrue}{\l@dmemoirfalse}
45

```

`\ifl@imakeidx` Define a flag for if the imakeidx package has been used.

```

46 \newif\ifl@imakeidx
47 \@ifpackageloaded{imakeidx}{\l@imakeidxtrue}{\l@imakeidxfalse}%False is the default value

```

`\ifl@indextools` Define a flag for if the indextools package has been used.

```

48 \newif\ifl@indextools%
49 \@ifpackageloaded{indextools}{%
50 \l@indextoolstrue%
51 \l@imakeidxtrue%
52 \let\imki@wrindexentry\indtl@wrindexentry%
53 }{}%False is the default value. We consider indextools as a variant of imakeidx. That's why we set

```

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

```

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

```

19.4 Messages

All the messages are grouped here as macros. This saves TeX's memory when the same message is repeated and also lets them be edited easily.

`\eledmac@warning` Write a warning message.

```
55 \newcommand{\eledmac@warning}[1]{\PackageWarning{eledmac}{#1}}
```

`\eledmac@error` Write an error message.

```
56 \newcommand{\eledmac@error}[2]{\PackageError{eledmac}{#1}{#2}}
```

`\led@err@NumberingStarted`

`\led@err@NumberingNotStarted`

`\led@err@NumberingShouldHaveStarted`

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

`\led@mess@NotesChanged`

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

`\led@mess@SectionContinued`

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

`\led@err@LineationInNumbered`

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

`\led@warn@BadLineation`

`\led@warn@BadLinenummargin`

`\led@warn@BadLockdisp`

`\led@warn@BadSubblockdisp`

```
76 \newcommand*{\led@warn@BadLineation}{%
77 \eledmac@warning{Bad \string\lineation\space argument}}
78 \newcommand*{\led@warn@BadLinenummargin}{%
79 \eledmac@warning{Bad \string\linenummargin\space argument}}
80 \newcommand*{\led@warn@BadLockdisp}{%
81 \eledmac@warning{Bad \string\lockdisp\space argument}}
82 \newcommand*{\led@warn@BadSubblockdisp}{%
83 \eledmac@warning{Bad \string\subblockdisp\space argument}}
```

`\led@warn@NoLineFile`

```
84 \newcommand*{\led@warn@NoLineFile}[1]{%
85 \eledmac@warning{Can't find line-list file #1}}
```



```

arn@BadAdvancelineSubline
d@warn@BadAdvancelineLine 86 \newcommand*{\led@warn@BadAdvancelineSubline}{%
87 \eledmac@warning{\string\advanceline\space produced a sub-line
88 number less than zero.}}
89 \newcommand*{\led@warn@BadAdvancelineLine}{%
90 \eledmac@warning{\string\advanceline\space produced a line
91 number less than zero.}}

\led@warn@BadSetline
\led@warn@BadSetlinenum 92 \newcommand*{\led@warn@BadSetline}{%
93 \eledmac@warning{Bad \string\setline\space argument}}
94 \newcommand*{\led@warn@BadSetlinenum}{%
95 \eledmac@warning{Bad \string\setlinenum\space argument}}

\led@err@PstartNotNumbered
\led@err@PstartInPstart 96 \newcommand*{\led@err@PstartNotNumbered}{%
\led@err@PendNotNumbered 97 \eledmac@error{\string\pstart\space must be used within a
\led@err@PendNoPstart 98 numbered section}{\@ehc}}
ed@err@AutoparNotNumbered 99 \newcommand*{\led@err@PstartInPstart}{%
100 \eledmac@error{\string\pstart\space encountered while another
101 \string\pstart\space was in effect}{\@ehc}}
102 \newcommand*{\led@err@PendNotNumbered}{%
103 \eledmac@error{\string\pend\space must be used within a
104 numbered section}{\@ehc}}
105 \newcommand*{\led@err@PendNoPstart}{%
106 \eledmac@error{\string\pend\space must follow a \string\pstart}{\@ehc}}
107 \newcommand*{\led@err@AutoparNotNumbered}{%
108 \eledmac@error{\string\autopar\space must be used within a
109 numbered section}{\@ehc}}

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

\led@warn@DuplicateLabel
\led@warn@RefUndefined 112 \newcommand*{\led@warn@DuplicateLabel}[1]{%
113 \eledmac@warning{Duplicate definition of label ‘#1’ on page \the\pageno.}}
114 \newcommand*{\led@warn@RefUndefined}[1]{%
115 \eledmac@warning{Reference ‘#1’ on page \the\pageno\space undefined.
116 Using ‘000’.}}

\led@warn@NoMarginpars
117 \newcommand*{\led@warn@NoMarginpars}{%
118 \eledmac@warning{You can’t use \string\marginpar\space in numbered text}}

ed@warn@BadSidenotemargin
119 \newcommand*{\led@warn@BadSidenotemargin}{%
120 \eledmac@warning{Bad \string\sidenotemmargin\space argument}}

```

```

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

\led@warn@AddfootinsXobsolete
\led@warn@Addfootinsobsolete 123 \newcommand{\led@warn@AddfootinsXobsolete}{%
124   \eledmac@warning{AddfootinsX is obsolete in eledmac 1.0. Use newseries instead.}%
125 }%
126 \newcommand{\led@warn@AddfootinsObsolete}{%
127   \eledmac@warning{Addfootins is obsolete in eledmac 1.0. Use newseries instead.}%
128 }%

\led@warn@SeriesStillExist
129 \newcommand{\led@warn@SeriesStillExist}[1]{%
130   \eledmac@warning{Series #1 is still existing !}%
131 }%

\led@err@ManySidenotes
\led@err@ManyLeftnotes 132 \newcommand{\led@err@ManySidenotes}{%
\led@err@ManyRightnotes 133   \ifledRcol%
134     \eledmac@warning{\itemcount@space sidenotes on line \the\line@numRspace p. \the\pag
135   }%
136     \eledmac@warning{\itemcount@space sidenotes on line \the\line@numspace p. \the\pag
137   }%
138 }%
139 \newcommand{\led@err@ManyLeftnotes}{%
140   \ifledRcol%
141     \eledmac@warning{\itemcount@space leftnotes on line \the\line@numRspace p. \the\pag
142   }%
143     \eledmac@warning{\itemcount@space leftnotes on line \the\line@numspace p. \the\pag
144   }%
145 }%
146 \newcommand{\led@err@ManyRightnotes}{%
147   \ifledRcol%
148     \eledmac@warning{\itemcount@space rightnotes on line \the\line@numRspace p. \the\pag
149   }%
150     \eledmac@warning{\itemcount@space rightnotes on line \the\line@numspace p. \the\pag
151   }%
152 }%

\led@war@FalseverseDeprecated
\led@war@ledxxxDeprecated 153 \newcommand{\led@war@FalseverseDeprecated}{%
154   \eledmac@warning{\string\falseverse\space deprecated. Look at \string\newverse\space ins
155 }%
156 \newcommand{\led@war@ledxxxDeprecated}[1]{%
157   \eledmac@warning{\string\led#1\space deprecated. Look at \string\led#1 instead.}%
158 }%

\led@err@TooManyColumns
\led@err@UnequalColumns
\led@err@LowStartColumn
\led@err@HighEndColumn
\led@err@ReverseColumns

```

```

159 \newcommand*\led@err@TooManyColumns}{%
160   \eledmac@error{Too many columns}{\@ehc}}
161 \newcommand*\led@err@UnequalColumns}{%
162   \eledmac@error{Number of columns is not equal to the number
163     in the previous row (or \protect\\ \space forgotten?)}{\@ehc}}
164 \newcommand*\led@err@LowStartColumn}{%
165   \eledmac@error{Start column is too low}{\@ehc}}
166 \newcommand*\led@err@HighEndColumn}{%
167   \eledmac@error{End column is too high}{\@ehc}}
168 \newcommand*\led@err@ReverseColumns}{%
169   \eledmac@error{Start column is greater than end column}{\@ehc}}

```

err@EdtextWithoutFootnote

```

170 \newcommand*\led@err@EdtextWithoutFootnote}{%
171   \eledmac@error{edtext without Xfootnote. Check syntax.}{\@ehd}%
172 }%

```

ror@ImakeidxAfterEledmac

```

173 \newcommand*\led@error@ImakeidxAfterEledmac}{%
174   \eledmac@error{Imakeidx must be loaded before eledmac.}{\@ehd}%
175 }%

```

or@IndextoolsAfterEledmac

```

176 \newcommand*\led@error@IndextoolsAfterEledmac}{%
177   \eledmac@error{Indextools must be loaded before eledmac.}{\@ehd}%
178 }%

```

19.5 Gobbling

\@gobblethree

```

\@gobblefour 179 \providecommand*\@gobblethree}[3]{
180 \providecommand*\@gobblefour}[4]{

```

Here, we define some commands which gobble their arguments.

19.6 Miscellaneous commands

\linenumberlist The code for the \linenumberlist mechanism was given to Peter Wilson by Wayne Sullivan on 2004/02/11.

Initialize it as \empty

```

181 \let\linenumberlist=\empty
182

```

\@l@tempcnta In imitation of L^AT_EX, we create a couple of scratch counters.

\@l@tempcntb L^AT_EX already defines \@tempcnta and \@tempcntb but Peter Wilson have found in the past that it can be dangerous to use these (for example one of the AMS packages did something nasty to the ccaption package's use of one of these).

```

183 \newcount\@l@tempcnta \newcount\@l@tempcntb

```

20 Sectioning commands

<code>\section@num</code>	You use <code>\beginnumbering</code> and <code>\endnumbering</code> 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. L ^A T _E X will maintain and display a ‘section number’ as a count named <code>\section@num</code> that counts how many <code>\beginnumbering</code> and <code>\resumenumbers</code> commands have appeared; it needn’t be related to the logical divisions of your text.
<code>\extensionchars</code>	Each section will read and write an associated ‘line-list file’, containing information used to do the numbering; the file will be called <code><jobname>.nn</code> , where <code>nn</code> is the section number. However, you may direct that an extra string be added before the <code>nn</code> in that filename, in order to distinguish these temporary files from others: that string is called <code>\extensionchars</code> . Initially it’s empty, since different operating systems have greatly varying ideas about what characters are permitted in file names. So <code>\renewcommand{\extensionchars}{-}</code> gives temporary files called <code>jobname.-1</code> , <code>jobname.-2</code> , etc. <pre> 184 \newcount\section@num 185 \section@num=0 186 \let\extensionchars=\empty </pre>
<code>\ifnumbering</code> <code>\numberingtrue</code> <code>\numberingfalse</code>	The <code>\ifnumbering</code> flag is set to <code>true</code> if we’re within a numbered section (that is, between <code>\beginnumbering</code> and <code>\endnumbering</code>). You can use <code>\ifnumbering</code> in your own code to check whether you’re in a numbered section, but don’t change the flag’s value. <pre> 187 \newif\ifnumbering </pre>
<code>\ifnumberingR</code> <code>\ifl@dpairing</code> <code>\ifl@dpaging</code> <code>\l@dpagingtrue</code> <code>\l@dpagingfalse</code> <code>\l@dpairingtrue</code> <code>\l@dpairingfalse</code> <code>\ifpst@rtedL</code> <code>\pst@rtedLtrue</code> <code>\pst@rtedLfalse</code> <code>\l@dnumpstartsL</code> <code>\ifledRcol</code> <code>\ifledRcol@</code>	In preparation for the <code>eledpar</code> package, these are related to the ‘left’ text of parallel texts (when <code>\ifl@dpairing</code> is <code>TRUE</code>). They are explained in the <code>eledpar</code> manual. <pre> 188 \newif\ifl@dpairing 189 \l@dpairingfalse 190 \newif\ifl@dpaging% 191 \l@dpagingfalse% 192 \newif\ifpst@rtedL 193 \pst@rtedLfalse 194 \newcount\l@dnumpstartsL </pre> <p><code>\ifledRcol</code> is set to <code>true</code> in the <code>Rightside</code> environnement. It must be distinguished from <code>\ifledRcol@</code> which is set to <code>true</code> when a right line is processed, in <code>\Pages</code> or <code>\Columns</code>.</p> <pre> 195 \newif\ifledRcol 196 \newif\ifledRcol@ </pre>
	The <code>\ifnumberingR</code> flag is set to <code>true</code> if we’re within a right text numbered section. <pre> 197 \newif\ifnumberingR </pre>

`\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.

```

198 \newcommand*{\beginnumbering}{%
199   \ifnumbering
200     \led@err@NumberingStarted
201     \endnumbering
202   \fi
203   \global\numberingtrue
204   \global\advance\section@num \@ne
205   \initnumbering@reg
206   \message{Section \the\section@num }%
207   \line@list@stuff{\jobname.\extensionchars\the\section@num}%
208   \l@dend@stuff
209   \setcounter{pstart}{1}
210   \ifl@dpairing
211     \global\l@dnumpstartsL \z@
212     \global\pst@rtedLfalse

```

The tools for section's title commands are called:

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

```

213 \else
214   \begingroup
215   \initnumbering@sectcmd
216   \ifwidthliketwocolumns%
217     \csuse{setwidthliketwocolumns@\columns@position}%
218     \csuse{setpositionliketwocolumns@\columns@position}%
219   \fi%
220 \fi

```

```

221 \gdef\eled@sections@{}%
222 \if@noeled@sec\else%
223   \makeatletter\inputIfFileExists{\jobname.eledsec\the\section@num}{-}{\makeatother}%
224   \immediate\openout\eled@sectioning@out=\jobname.eledsec\the\section@num\relax%
225 \fi%
226 }
227 \newcommand*\initnumbering@reg{%
228   \global\pst@rtedLfalse
229   \global\l@dnumpstartsL \z@
230   \global\absline@num \z@
231   \gdef\normal@page@break{}
232   \gdef\l@prev@pb{}
233   \gdef\l@prev@nopb{}
234   \global\line@num \z@
235   \global\subline@num \z@
236   \global\@lock \z@
237   \global\sub@lock \z@
238   \global\sublines@false
239   \global\let\next@page@num=\relax
240   \global\let\sub@change=\relax
241   \resetprevline@
242   \resetprevpage@num
243 }
244

```

`\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.

```

245 \def\endnumbering{%
246   \ifnumbering
247     \global\numberingfalse
248     \normal@pars
249     \ifl@dpairing
250       \global\pst@rtedLfalse
251     \else
252       \ifx\insertlines@list\empty\else
253         \global\noteschanged@true
254       \fi
255       \ifx\line@list\empty\else
256         \global\noteschanged@true
257       \fi
258     \fi
259     \ifnoteschanged@
260       \led@mess@NotesChanged
261     \fi
262   \else
263     \led@err@NumberingNotStarted
264   \fi
265   \autoparfalse
266   \if@noeled@sec\else%

```

```

267 \immediate\closeout\eled@sectioning@out%
268 \fi%
269 \ifl@dpairing\else
270 \global\l@dnumstartsL=\z@%
271 \endgroup
272 \fi
273 }

```

`\pausenumbering` The `\pausenumbering` macro is just the same as `\endnumbering`, but with the `\resumenumbering` `\ifnumbering` flag set to true, to show that numbering continues across the gap.²¹

```

274 \newcommand{\pausenumbering}{%
275 \ifautopar\global\autopar@pausetrue\fi%
276 \endnumbering\global\numberingtrue}

```

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

```

277 \newcommand*{\resumenumbering}{%
278 \ifnumbering
279 \ifautopar@pause\autopar\fi
280 \global\pst@rtedLtrue
281 \global\advance\section@num \@ne
282 \led@mess@SectionContinued{\the\section@num}%
283 \line@list@stuff{\jobname.\extensionchars\the\section@num}%
284 \l@dend@stuff
285 \ifl@dpairing\else%
286 \begingroup%
287 \initnumbering@sectcmd%
288 \ifwidthliketwocolumns%
289 \csuse{setwidthliketwocolumns@\columns@position}%
290 \csuse{setpositionliketwocolumns@\columns@position}%
291 \fi%
292 \fi%
293 \else
294 \led@err@NumberingShouldHaveStarted
295 \endnumbering
296 \beginnumbering
297 \fi}
298
299

```

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

²¹Our thanks to Wayne Sullivan, who suggested the idea behind these macros.

numbers that start at 1 at the start of each section and increase regardless of page breaks. `eledmac` can do it either way, and you can switch from one to the other within one work. But you have to choose one or the other for all line numbers and line references within each section. Here we will define internal codes for these systems and the macros you use to select them.

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

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

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

```

\ifbypstart@
\byppstart@true
\byppstart@false
  \ifbypage@
  \bypage@true
  \bypage@false
300 \newif\ifbypage@
301 \newif\ifbypstart@

\lineation \lineation{<word>} is the macro you use to select the lineation system. Its
argument is a string: either page or section or pstart.
302 \newcommand*{\lineation}[1]{%
303   \ifnumbering
304     \led@err@LineationInNumbered
305   \else
306     \def\@tempa{#1}\def\@tempb{page}%
307     \ifx\@tempa\@tempb
308       \global\bypage@true
309       \global\byppstart@false
310       \pstartinfootnote[] [false]
311     \else
312       \def\@tempb{pstart}%
313       \ifx\@tempa\@tempb
314         \global\bypage@false
315         \global\byppstart@true
316         \pstartinfootnote
317       \else
318         \def\@tempb{section}
319         \ifx\@tempa\@tempb
320           \global\bypage@false
321           \global\byppstart@false
322           \pstartinfootnote[] [false]
323         \else
324           \led@warn@BadLineation
325         \fi
326       \fi
327     \fi
328   \fi}}

\linenummargin You call \linenummargin{<word>} to specify which margin you want your line
\line@margin numbers in; it takes one argument, a string. You can put the line numbers in
\l@getline@margin the same margin on every page using left or right; or you can use inner or
outer to get them in the inner or outer margins. (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 `\line@margin`: 0 for left, 1 for right, 2 for outer, and 3 for inner.

```

329 \newcount\line@margin
330 \newcommand*\linenummargin{1}{%
331   \l@getline@margin{#1}%
332   \ifnum\l@dttempcntb>\m@ne
333     \global\line@margin=\l@dttempcntb
334   \fi}}
335 \newcommand*\l@getline@margin{1}{%
336   \def\@tempa{#1}\def\@tempb{left}%
337   \ifx\@tempa\@tempb
338     \l@dttempcntb \z@
339   \else
340     \def\@tempb{right}%
341     \ifx\@tempa\@tempb
342       \l@dttempcntb \@ne
343     \else
344       \def\@tempb{outer}%
345       \ifx\@tempa\@tempb
346         \l@dttempcntb \tw@
347       \else
348         \def\@tempb{inner}%
349         \ifx\@tempa\@tempb
350           \l@dttempcntb \thr@@
351         \else
352           \led@warn@BadLinenummargin
353           \l@dttempcntb \m@ne
354         \fi
355       \fi
356     \fi
357   \fi}
358
```

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

```

359 \newcounter{firstlinenum}
360 \setcounter{firstlinenum}{5}
361 \newcounter{linenumincrement}
362 \setcounter{linenumincrement}{5}

```

`\c@firstsublinenum` The following parameters are just like `firstlinenum` and `linenumincrement`, but
`\c@sublinenumincrement` for sub-line numbers. `sublinenumincrement` must be at least 1.

```

363 \newcounter{firstsublinenum}
364 \setcounter{firstsublinenum}{5}
365 \newcounter{sublinenumincrement}
366 \setcounter{sublinenumincrement}{5}
367

```

`\firstlinenum` These macros can be used to set the corresponding counters.

```

\linenumincrement 368 \newcommand*{\firstlinenum}[1]{\setcounter{firstlinenum}{#1}}
\firstsublinenum 369 \newcommand*{\linenumincrement}[1]{\setcounter{linenumincrement}{#1}}
\sublinenumincrement 370 \newcommand*{\firstsublinenum}[1]{\setcounter{firstsublinenum}{#1}}
371 \newcommand*{\sublinenumincrement}[1]{\setcounter{sublinenumincrement}{#1}}
372

```

`\lockdisp` 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.

```

373 \newcount\lock@disp
374 \newcommand{\lockdisp}[1]{%
375   \l@getlock@disp{#1}%
376   \ifnum\l@tempcntb>\m@ne
377     \global\lock@disp=\l@tempcntb
378   \else
379     \led@warn@BadLockdisp
380   \fi}}
381 \newcommand*{\l@getlock@disp}[1]{
382   \def\@tempa{#1}\def\@tempb{first}%
383   \ifx\@tempa\@tempb
384     \l@tempcntb \z@
385   \else
386     \def\@tempb{last}%
387     \ifx\@tempa\@tempb
388       \l@tempcntb \@ne
389     \else
390       \def\@tempb{all}%
391       \ifx\@tempa\@tempb
392         \l@tempcntb \tw@
393       \else
394         \l@tempcntb \m@ne
395       \fi
396     \fi
397   \fi}
398

```

`\sublockdisp` 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.

```

399 \newcount\sublock@disp
400 \newcommand{\sublockdisp}[1]{%

```

```

401 \l@edgetlock@disp{#1}%
402 \ifnum\@l@dttempcntb>\m@ne
403   \global\sublock@disp=\@l@dttempcntb
404 \else
405   \led@warn@BadSublockdisp
406 \fi}}
407

```

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

`\linenumrep` NOTE: In v0.7 `\linenumrep` and `\sublinenumrep` replaced the internal

`\sublinenumberstyle` `\linenumr@p` and `\sublinenumr@p`.

`\sublinenumrep` `\linenumberstyle` and `\sublinenumberstyle` are user level macros for setting the number representation (`\linenumrep` and `\sublinenumrep`) for line and sub-line numbers.

`\sublinenumr@p`

```

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

```

Initialise the number styles to arabic.

```

412 \linenumberstyle{arabic}
413 \let\linenumr@p\linenumrep
414 \sublinenumberstyle{arabic}
415 \let\sublinenumr@p\sublinenumrep
416

```

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

`\rightlinenum`

`\linenumsep` 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.

`\numlabfont`

`\ledlinenum`

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 L^AT_EX `\scriptsize` for a 10pt document.

```

417 \newlength{\linenumsep}
418 \setlength{\linenumsep}{1pc}
419 \newcommand*{\numlabfont}{\normalfont\scriptsize}
420 \newcommand*{\ledlinenum}{%
421   \numlabfont\linenumrep{\line@num}%

```

```

422 \ifsublines@
423   \ifnum\subline@num>0\relax
424     \unskip\fullstop\sublinenumrep{\subline@num}%
425   \fi
426 \fi}
427 \newcommand*{\leftlinenum}{%
428   \ledlinenum
429   \kern\linenumsep}
430 \newcommand*{\rightlinenum}{%
431   \kern\linenumsep
432   \ledlinenum}
433

```

21.2 List macros

Reminder: compare these with the L^AT_EX 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.

```

434 \newcommand*{\list@create}[1]{\global\let#1=\empty}

```

`\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`.

```

435 \newcommand*{\list@clear}[1]{\global\let#1=\empty}

```

`\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`.

```

436 \newtoks\led@toksa \newtoks\led@toksb
437 \global\led@toksa={\}
438 \long\def\xright@appenditem#1\to#2{%
439   \global\led@toksb=\expandafter{#2}%
440   \xdef#2{\the\led@toksb\the\led@toksa\expandafter{#1}}%
441   \global\led@toksb={}}

```

`\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`.

```

442 \long\def\xleft@appenditem#1\to#2{%

```

```

443 \global\led@toksb=\expandafter{#2}%
444 \xdef#2{\the\led@toksa\expandafter{#1}\the\led@toksb}%
445 \global\led@toksb={}}

```

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

```

446 \def\gl@p#1\to#2{\expandafter\gl@poff#1\gl@poff#1#2}
447 \long\def\gl@poff\#1#2\gl@poff#3#4{\gdef#4{#1}\gdef#3{#2}}
448

```

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 **L^AT_EX** 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**.

```

449 \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.

```

450 \newcount\subline@num

```

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

\sublines@true 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.

451 `\newif\ifsublines@`

`\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.

452 `\newcount\absline@num`

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

`\@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.

453 `\newcount\@lock`

454 `\newcount\sub@lock`

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

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

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

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

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

There is one item in this list for every lemma marked by `\edtext`, even if there are several notes to that lemma, or no notes at all. `\edtext` reads the data in this list, making it available for use in the text of notes.

- `\insertlines@list`: the line numbers of lines that have footnotes or other insertions. These are the absolute numbers where the corresponding lemmas begin. This list contains one entry for every footnote in the section; one lemma may contribute no footnotes or many footnotes. This list is used by `\add@inserts` within `\do@line`, to tell it where to insert notes.
- `\actionlines@list`: a list of absolute line numbers at which we are to perform special actions; these actions are specified by the `\actions@list` list defined below.
- `\actions@list`: action codes corresponding to the line numbers in `\actionlines@list`. These codes tell `eledmac` what action it's supposed to take at each of these lines. One action, the page-start action, is generated behind the scenes by `eledmac` itself; the others, for specifying sub-lineation, line-number locking, and line-number alteration, are generated only by explicit commands in your input file. The page-start and line-number-alteration actions require arguments, to specify the new values for the page or line numbers; instead of storing those arguments in another list, we have chosen the action-code values so that they can encode both the action and the argument in these cases. Action codes greater than -1000 are page-start actions, and the code value is the page number; action codes less than -5000 specify line numbers, and the code value is a transformed version of the line number; action codes between these two values specify other actions which require no argument.

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

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

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

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

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

The action code -1004 specifies the end of line number locking.

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

The action code `-1006` specifies the end of sub-line number locking.

The four action codes for line and sub-line number locking are added to the list by the `\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 advanced) to a specific positive value. The value of the code is $-(5000 + n)$, where n is the value (always ≥ 0) assigned to the current line number. Action 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:

```

455     \list@create{\line@list}
456     \list@create{\insertlines@list}
457     \list@create{\actionlines@list}
458     \list@create{\actions@list}
459
    \page@num We'll need some counts while we read the line-list, for the page number and the
    \endpage@num ending page, line, and sub-line numbers. Some of these will be used again later
    \endline@num on, when we are acting on the data in our list macros.
    \endsubline@num
460 \newcount\page@num
461 \newcount\endpage@num
462 \newcount\endline@num
463 \newcount\endsubline@num

    \ifnoteschanged@ If the number of the footnotes in a section is different from what it was during
    \noteschanged@true the last run, or if this is the very first time you've run LATEX, on this file, the
    \noteschanged@false 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 LATEX 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.
464 \newif\ifnoteschanged@

```


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

```
\resetprevline@
465 \newcommand*{\resetprevline@}{%
466   \def\do##1{\global\csundef{prevline##1}}%
467   \dolistloop{\@series}%
468 }
```

`\resetprevpage@num` Inside the apparatus, at each note, the page number is stored in a macro called `\prevpageX@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@
469 \newcommand*{\resetprevpage@num}{%
470   \def\do##1{\ifcsdef{prevpage##1@num}{\global\csname prevpage##1@num\endcsname=0}{}}%
471   \dolistloop{\@series}%
472 }
```

21.4 Reading the line-list file

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

```
473 \newread\@inputcheck
474 \newcommand*{\read@linelist}[1]{%
475   \list@clearing@reg
```

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 L^AT_EX 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 `\resumenummering`, 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.

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

```
477 \endgroup
```

When the reading is done, we're all through with the line-list file. All the information we needed from it will now be encoded in our list macros.

Finally, we initialize the `\next@actionline` and `\next@action` macros, which specify where and what the next action to be taken is.

```
478 \global\page@num=\m@ne
479 \ifx\actionlines@list\empty
480   \gdef\next@actionline{1000000}%
481 \else
482   \glp\actionlines@list\to\next@actionline
483   \glp\actions@list\to\next@action
484 \fi}
485
```

`\list@clearing@reg` Clears the lists for `\read@linelist`

```
486 \newcommand*{\list@clearing@reg}{%
487   \list@clear{\line@list}%
488   \list@clear{\insertlines@list}%
489   \list@clear{\actionlines@list}%
490   \list@clear{\actions@list}%
491   \list@clear{\sw@list}%
492   \list@clear{\sw@list@inedtext}%
493 }
```

`\get@linelistfile` `eledmac` can take advantage of the L^AT_EX 'safe file input' macros to get the line-list file.

```
494 \newcommand*{\get@linelistfile}[1]{%
495   \InputIfFileExists{#1}{%
496     \global\noteschanged@false
497     \begingroup
498       \catcode'\[=1 \catcode'\]=2
499       \makeatletter \catcode'\^M=9}%
500     \led@warn@NoLineFile{#1}%
501     \global\noteschanged@true
502     \begingroup}%
503 }
504
```

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

21.5 Commands within the line-list file

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

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

The macros with `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 `\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 `\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 `\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 `\thepage`) but it may come in handy later. The macro `\fix@page` checks if a new page has started.

```

505 \newcommand{\@nl}[2]{%
506   \fix@page{#1}%
507   \@nl@reg}
508 \newcommand*{\@nl@reg}{%
509   \ifx\l@dchset@num\relax \else
510     \advance\absline@num \@ne
511     \set@line@action
512     \let\l@dchset@num=\relax
513     \advance\absline@num \m@ne
514     \advance\line@num \m@ne
515   \fi

First increment the absolute line-number, and perform deferred actions relating
to page starts and sub-lines.

516   \advance\absline@num \@ne
517     \ifx\next@page@num\relax \else
518       \page@action
519       \let\next@page@num=\relax
520     \fi
521     \ifx\sub@change\relax \else

```

```

522         \ifnum\sub@change>\z@
523             \sublines@true
524         \else
525             \sublines@false
526         \fi
527         \sub@action
528         \let\sub@change=\relax
529     \fi

```

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

```

530     \ifcase\@lock
531     \or
532         \@lock \tw@
533     \or \or
534         \@lock \z@
535     \fi
536     \ifcase\sub@lock
537     \or
538         \sub@lock \tw@
539     \or \or
540         \sub@lock \z@
541     \fi

```

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

```

542     \ifsublines@
543         \ifnum\sub@lock<\tw@
544             \advance\subline@num \@ne
545         \fi
546     \else
547         \ifnum\@lock<\tw@
548             \advance\line@num \@ne \subline@num \z@
549         \fi
550     \fi}
551

```

`\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`.

```

552 \newcount\last@page@num
553 \last@page@num=-10000
554 \newcommand*{\fix@page}[1]{%
555     \ifnum #1=\last@page@num
556     \else
557         \ifbypage@
558             \line@num=\z@ \subline@num=\z@
559         \fi
560         \page@num=#1\relax
561         \last@page@num=#1\relax
562         \def\next@page@num{#1}%
563         \listxadd{\normal@page@break}{\the\absline@num}

```

```

564 \fi}
565

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

566 \newcommand*\@pend}[1]{}
567 \newcommand*\@pendR}[1]{}
568 \newcommand*\@lopL}[1]{}
569 \newcommand*\@lopR}[1]{}
570

\sub@on The \sub@on and \sub@off macros turn sub-lineation on and off: but not directly,
\sub@off 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.

571 \newcommand*\sub@on{\ifsublines@
572     \let\sub@change=\relax
573 \else
574     \def\sub@change{1}%
575 \fi}
576 \newcommand*\sub@off{\ifsublines@
577     \def\sub@change{-1}%
578 \else
579     \let\sub@change=\relax
580 \fi}
581

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

582 \newcommand*\@adv}[1]{\ifsublines@
583     \advance\subline@num by #1\relax
584     \ifnum\subline@num<\z@
585         \led@warn@BadAdvancelineSubline
586         \subline@num \z@
587     \fi
588 \else
589     \advance\line@num by #1\relax
590     \ifnum\line@num<\z@
591         \led@warn@BadAdvancelineLine
592         \line@num \z@
593     \fi
594 \fi
595 \set@line@action}
596

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

597 \newcommand*\@set}[1]{\ifsublines@

```

```

598   \subline@num=#1\relax
599   \else
600     \line@num=#1\relax
601   \fi
602   \set@line@action}
603

```

`\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 line number change is to be done.

```

604 \newcommand*{\l@d@set}[1]{%
605   \line@num=#1\relax
606   \advance\line@num \@ne
607   \def\l@dchset@num{#1}}
608 \let\l@dchset@num\relax
609

```

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

```

610 \newcommand*{\page@action}{%
611   \xright@appenditem{\the\absline@num}\to\actionlines@list
612   \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.

```

613 \newcommand*{\set@line@action}{%
614   \xright@appenditem{\the\absline@num}\to\actionlines@list
615   \ifsublines@
616     \@l@dttempcnta=-\subline@num
617   \else
618     \@l@dttempcnta=-\line@num
619   \fi
620   \advance\@l@dttempcnta by -5000
621   \xright@appenditem{\the\@l@dttempcnta}\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 `\ifsublines@` flag.

```

622 \newcommand*{\sub@action}{%
623   \xright@appenditem{\the\absline@num}\to\actionlines@list
624   \ifsublines@
625     \xright@appenditem{-1001}\to\actions@list
626   \else
627     \xright@appenditem{-1002}\to\actions@list
628   \fi}

```

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

`\do@lockon` The current setting of the sub-lineation flag tells us whether this applies to line numbers or sub-line numbers.

`\do@lockonL`

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.

```

629 \newcommand*{\lock@on}{\futurelet\next\do@lockon}
630 \newcommand*{\do@lockon}{%
631   \ifx\next\lock@off
632     \global\let\lock@off=\skip@lockoff
633   \else
634     \do@lockonL
635   \fi}
636 \newcommand*{\do@lockonL}{%
637   \xright@appenditem{\the\absline@num}\to\actionlines@list
638   \ifsublines@
639     \xright@appenditem{-1005}\to\actions@list
640     \ifnum\sub@lock=\z@
641       \sub@lock \@ne
642     \else
643       \ifnum\sub@lock=\thr@@
644         \sub@lock \@ne
645       \fi
646     \fi
647   \else
648     \xright@appenditem{-1003}\to\actions@list
649     \ifnum\@lock=\z@
650       \@lock \@ne
651     \else
652       \ifnum\@lock=\thr@@
653         \@lock \@ne
654       \fi
655     \fi
656   \fi}
657
```

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

```

\do@lockoff 658 \newcommand*{\do@lockoffL}{%
\do@lockoffL 659   \xright@appenditem{\the\absline@num}\to\actionlines@list
\skip@lockoff 660   \ifsublines@
661     \xright@appenditem{-1006}\to\actions@list
662     \ifnum\sub@lock=\tw@
663       \sub@lock \thr@@
664     \else
665       \sub@lock \z@
666     \fi
667   \else
668     \xright@appenditem{-1004}\to\actions@list
669     \ifnum\@lock=\tw@
670       \@lock \thr@@
671     \else
```

```

672      \@lock \z@
673      \fi
674    \fi}
675 \newcommand*{\do@lockoff}{\do@lockoffL}
676 \newcommand*{\skip@lockoff}{\global\let\lock@off=\do@lockoff}
677 \global\let\lock@off=\do@lockoff
678

```

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

```

679 \newcommand*{\n@num}{\n@num@reg}
680 \newcommand*{\n@num@reg}{%
681   \xright@appenditem{\the\absline@num}\to\actionlines@list
682   \xright@appenditem{-1007}\to\actions@list}
683

```

`\@ref` `\@ref` marks the start of a passage, for creation of a footnote reference. It takes `\insert@count` 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`.

```

684   \newcount\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.

```

685 \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.

```

686 \newcommand*{\@ref}[2]{%
687   \@ref@reg{#1}{#2}}
688 \newcommand*{\@ref@reg}[2]{%
689   \global\insert@count=#1\relax
690   \loop\ifnum\insert@count>\z@
691     \xright@appenditem{\the\absline@num}\to\insertlines@list
692     \global\advance\insert@count \m@ne
693   \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.


```

694 \begingroup
695   \let\@ref=\dummy@ref
696   \let\@lopL\@gobble
697   \let\page@action=\relax
698   \let\sub@action=\relax
699   \let\set@line@action=\relax
700   \let\@lab=\relax
701   \let\@sw\@gobbletwo%
702   #2
703   \global\endpage@num=\page@num
704   \global\endline@num=\line@num
705   \global\endsubline@num=\subline@num
706 \endgroup

```

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

```

707   \xright@appenditem%
708     {\the\page@num|\the\line@num|%
709     \ifsublines@ \the\subline@num \else 0\fi}%
710     {\the\endpage@num|\the\endline@num|%
711     \ifsublines@ \the\endsubline@num \else 0\fi}}\to\line@list

```

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

```

712   #2}
713

```

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.

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

```

714 \newwrite\linenum@out

```

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

We can manage pretty well by means of the `\iffirst@linenum@out@` flag; its inelegant name suggests the nature of the problem that made its creation necessary. It's set to be `true` before any `\linenum@out` file is opened. When such a file is opened for the first time, it's done using `\immediate`, so that it will at once be safe for the output routine to write to it; we then set this flag to `false`.

```

715 \newif\iffirst@linenum@out@

```

```
716 \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.

```
717 \newcommand*{\line@list@stuff}[1]{%
```

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

```
718 \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.

```
719 \iffirst@linenum@out@
```

```
720 \immediate\closeout\linenum@out%
```

```
721 \global\first@linenum@out@false%
```

```
722 \immediate\openout\linenum@out=#1\relax%
```

```
723 \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.

```
724 \if@minipage%
```

```
725 \leavevmode%
```

```
726 \fi%
```

```
727 \closeout\linenum@out%
```

```
728 \openout\linenum@out=#1\relax%
```

```
729 \fi}
```

```
730
```

`\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.

```
731 \newcommand*{\new@line}{%
```

```
732 \IfStrEq{\led@pb@setting}{after}%
```

```
733 {\xifinlist{\the\absline@num}{\l@prev@nopb}%
```

```
734 {\xifinlist{\the\absline@num}{\normal@page@break}%
```

```
735 {\numgdef{\@next@page}{\thepage+1}%
```

```
736 \write\linenum@out{\string\@nl[\@next@page][\@next@page]}%
```

```
737 }%
```

```
738 {\write\linenum@out{\string\@nl[\the\c@page][\thepage]}}%
```

```
739 }%
```

```
740 {\write\linenum@out{\string\@nl[\the\c@page][\thepage]}}%
```

```
741 }%
```

```
742 \IfStrEq{\led@pb@setting}{before}%
```

```
743 {\numdef{\next@absline}{\the\absline@num+1}%
```

```
744 \xifinlist{\next@absline}{\l@prev@nopb}%
```

```
745 {\xifinlist{\the\absline@num}{\normal@page@break}%
```

```
746 {\numgdef{\nc@page}{\c@page+1}%
```

```
747 \write\linenum@out{\string\@nl[\nc@page][\nc@page]}%
```

```
748 }%
```

```

749      {\write\linenum@out{\string\n@l[\the\c@page][\thepage]}}%
750    }%
751    {\write\linenum@out{\string\@nl[\the\c@page][\thepage]}}%
752  }%
753  {}%
754  \IfStrEqCase{\led@pb@setting}{{before}{\relax}{after}{\relax}}{\write\linenum@out{\string\@nl[\the\c@page][\thepage]}}
755 }
756

```

`\if@noneed@Footnote` We enclose a lemma marked by `\edtext` in `\flag@start` and `\flag@end`: these
`\flag@start` send the `\@ref` command to the line-list file. `\edtext` is responsible for setting
`\flag@end` the value of `\insert@count` appropriately; it actually gets done by the various
footnote macros.

```

757 \newif\if@noneed@Footnote%Bool to check if we have to print a error message
758
759 \newcommand*{\flag@start}{%
760   \ifledRcol%
761     \edef\next{\write\linenum@outR{%
762       \string\@ref[\the\insert@countR] []}%
763     \next%
764     \ifnum\insert@countR<1%
765       \if@noneed@Footnote\else%
766         \led@err@EdtextWithoutFootnote%
767       \fi%
768     \fi%
769   \else%
770     \edef\next{\write\linenum@out{%
771       \string\@ref[\the\insert@count] []}%
772     \next%
773     \ifnum\insert@count<1%
774       \if@noneed@Footnote\else%
775         \led@err@EdtextWithoutFootnote%
776       \fi%
777     \fi%
778   \fi}%
779

```

`\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 `\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 `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.

```

780 \newcommand*{\page@start}{%
781

```

`\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.

```
782 \newcommand*{\startsub}{\dimen0\lastskip
783   \ifdim\dimen0>0pt \unskip \fi
784   \write\linenum@out{\string\sub@on}%
785   \ifdim\dimen0>0pt \hskip\dimen0 \fi}
786 \def\endsub{\dimen0\lastskip
787   \ifdim\dimen0>0pt \unskip \fi
788   \write\linenum@out{\string\sub@off}%
789   \ifdim\dimen0>0pt \hskip\dimen0 \fi}
790
```

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

```
791 \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.

```
792 \newcommand*{\setline}[1]{%
793   \ifnum#1<\z@
794     \led@warn@BadSetline
795   \else
796     \write\linenum@out{\string\@set[#1]}%
797   \fi}
798
```

`\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.

```
799 \newcommand*{\setlinenum}[1]{%
800   \ifnum#1<\z@
801     \led@warn@BadSetlinenum
802   \else
803     \write\linenum@out{\string\l@d@set[#1]}%
804   \fi}
805
```

`\startlock` 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.

```
806 \newcommand*{\startlock}{\write\linenum@out{\string\lock@on}}
807 \def\endlock{\write\linenum@out{\string\lock@off}}
808
```

`\ifl@dskipnumber` In numbered text `\skipnumbering` will suspend the numbering for that particular line.
`\l@dskipnumbertrue`

```
\l@dskipnumberfalse 809 \newif\ifl@dskipnumber
\skipnumbering 810 \l@dskipnumberfalse
\skipnumbering@reg 811 \newcommand*{\skipnumbering}{\skipnumbering@reg}
812 \newcommand*{\skipnumbering@reg}{%
813 \write\linenum@out{\string\n@num}%
814 \advanceline{-1}}
815
```

22 Marking text for notes

The `\edtext` (or `\critext`) macro is used to create all footnotes and endnotes, as well as to print the portion of the main text to which a given note or notes is keyed. The idea is to have that lemma appear only once in the `.tex` file: all instances of it in the main text and in the notes are copied from that one appearance.

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

`\critext` requires two arguments. At any point within numbered text, you use it by saying:

```
\critext{#1}#2/
```

Similarly `\edtext` requires the same two arguments but you use it by saying:

```
\edtext{#1}{#2}
```

- `#1` is the piece of the main text being glossed; it gets added to the main text, and is also used as a lemma for notes to it.
- `#2` is a series of subsidiary macros that generate various kinds of notes. With `\critext` the `/` after `#2` *must* appear: it marks the end of the macro. (*The TeXbook*, p. 204, points out that when additional text to be matched follows the arguments like this, spaces following the macro are not skipped, which is very desirable since this macro will never be used except within text. Having an explicit terminator also helps keep things straight when nested calls to `\critext` are used.) Braces around `#2` are optional with `\critext` and required for `\edtext`.

The `*text` macro may be used (somewhat) recursively; that is, `*text` may be used within its own first argument. The code would be much simpler without this feature, but nested notes will commonly be necessary: it's quite likely that we'll have an explanatory note for a long passage and notes on variants for individual words within that passage. The situation we can't handle is overlapping notes that aren't nested: for example, one note covering lines 10–15, and another covering 12–18. You can handle such cases by using the `\lemma` and `\linenum` macros within `#2`: they alter the copy of the lemma and the line numbers that are passed to the notes, and hence allow you to overcome any limitations of this system, albeit with extra effort.

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

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

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

22.1 `\edtext` and `\critext` themselves

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.

`\end@lemmas` To accomodate this, we provide a list macro to which macros may add commands that should subsequently be executed at the end of the lemma when that lemma is added to the text of the paragraph. A macro should add its contribution to `\end@lemmas` by using `\xleft@appenditem`. (Anything that needs to be done at the *start* of the lemma may be handled using `\aftergroup`, since the commands

specified within `\critext`'s second argument are executed within a group that ends just before the lemma is added to the main text.)

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

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

```
816 \list@create{\end@lemmas}
```

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

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

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

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

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

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

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

We're going to need another macro that takes one argument and ignores it entirely. This is supplied by the L^AT_EX `\@gobble{<arg>}`.

`\no@expands` 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.²² This is done because we want to pass into our notes the generic commands to change to roman or whatever, and not their expansions that will ask for a particular style at a specified size. The notes may well be in a smaller font, so the command should be expanded later, when the note's environment is in effect.

A second sort to turn off includes a few of the accent macros. Most are not a problem: an accent that's expanded to an `\accent` command may be harder to read but it works just the same. The ones that cause problems are: those that use

²²Since 'control sequences equivalent to characters are not expandable'—*The TeXbook*, answer to Exercise 20.14.

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 \critext 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 \critext is used.

(A related problem, not addressed by these two macros, is that of characters whose category code is changed by any the macros used in the arguments to \critext . 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.)

```

820 \newcommand*{\no@expands}{%
821   \let\select@lemmafонт=0%
822   \let\startsub=\relax \let\endsub=\relax
823   \let\startlock=\relax \let\endlock=\relax
824   \let\edlabel=\@gobble
825   \let\setline=\@gobble \let\advanceline=\@gobble
826   \let\critext=\dummy@text
827   \let\sameword\sameword@inedtext%
828   \let\edtext=\dummy@edtext
829   \l@dtabnoexpands
830   \morenoexpands}
831 \let\morenoexpands=\relax
832
```

\@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.

```

833 \newcommand{\@tag}{}
834 % \end{macrocode}
835 % \end{macro}
836 % \begin{macro}{\if@edtext@}
```



```

837 % \changes{v1.15.0}{2015/01/12}{New boolean \cs{if@edtext@}.}
838 % This boolean is set to TRUE inside a \cs{edtext} (or \cs{critext}).
839 % That is useful for some commands which can have a different behavior if called inside or outside of
840 %   \begin{macrocode}
841 \newif\if@edtext%

```

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

When executed, `\critext` first ensures that we're in horizontal mode.

```

842 \long\def\critext#1#2/{\leavevmode
843   \edtext@true%

```

`\@tag` Our normal lemma is just argument #1; but that argument could have further invocations of `\critext` within it. We get a copy of the lemma without any `\critext` macros within it by temporarily redefining `\critext` 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 `\critext` 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.

```

844   \begingroup
845     \global\renewcommand{\@tag}{\no@expands #1}%%

```

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

```

846     \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 `\critext`. If we are in a right column (eledpar), we use `\insert@countR` instead of `\insert@count`.

```

847     \ifledRcol \global\insert@countR \z@%
848     \else      \global\insert@count \z@ \fi%

```

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

```

849     \ignorespaces #2\relax

```

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.

```

850 \ifundefined{xpg@main@language}{%if not polyglossia
851   \flag@start}%
852   {\ifRTL\flag@end\else\flag@start\fi% With polyglossia, you must track whether the
853   }
854 \endgroup
855 \showlemma{#1}%

```

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

```

856 \ifx\end@lemmas\empty \else
857   \glp\end@lemmas\to\x@lemma
858   \x@lemma
859   \global\let\x@lemma=\relax
860 \fi
861 \ifundefined{xpg@main@language}{%if not polyglossia
862   \flag@end}%
863   {\ifRTL\flag@start\else\flag@end\fi% With polyglossia, you must track whether the
864   }
865 \edtext@false%
866 \global\@lemmacommand@false%
867 }

```

`\edtext`

```

868 \newcommand{\edtext}[2]{\leavevmode
869   \edtext@true%
870   \begingroup%
871     \global\renewcommand{\@tag}{\no@expands #1}%
872     \set@line%
873     \ifledRcol \global\insert@countR \z@%
874     \else      \global\insert@count \z@ \fi%
875     \ignorespaces #2\relax%
876     \ifundefined{xpg@main@language}{%if not polyglossia
877       \flag@start}%
878       {\ifRTL\flag@end\else\flag@start\fi% With polyglossia, you must track whether the
879       }%
880   \endgroup%
881   \showlemma{#1}%
882   \ifx\end@lemmas\empty \else%
883     \glp\end@lemmas\to\x@lemma%
884     \x@lemma%
885     \global\let\x@lemma=\relax%
886   \fi%
887   \ifundefined{xpg@main@language}{%if not polyglossia
888     \flag@end}%
889     {\ifRTL\flag@start\else\flag@end\fi% With polyglossia, you must track whether the
890     }%

```

```

891 \global\@noneed@Footnotefalse%
892 \@edtext@false%
893 \global\@lemmacommand@false%
894 }%
895
896 \newcommand*{\flag@end}{%
897   \ifledRcol%
898     \write\linenum@outR[]}%
899   \else%
900     \write\linenum@out[]}%
901   \fi}%
902

```

`\ifnumberline` The `\ifnumberline` option can be set to FALSE to disable line numbering.

```

903 \newif\ifnumberline
904 \numberlinetrue

```

`\set@line` 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.

```

905 \newcommand*{\set@line}{%

```

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.

```

906   \ifx\line@list\empty
907     \global\noteschanged@true
908     \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
909   \else
910     \gl@p\line@list\to\@tempb
911     \xdef\l@d@nums{\@tempb|\edfont@info}%
912     \global\let\@tempb=\undefined
913   \fi}
914

```

`\edfont@info` The macro `\edfont@info` returns coded information about the current font.

```

915 \newcommand*{\edfont@info}{\f@encoding/\f@family/\f@series/\f@shape}
916

```

22.2 Substitute lemma

`\lemma` The `\lemma{<text>}` macro allows you to change the lemma that's passed on to the notes.

```

917 \newcommand*{\lemma}[1]{%
918   \global\@lemmacommand@true%
919   \global\renewcommand{\@tag}{\no@expands #1}}%

```

`\if@lemmacommand@` 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.

920 `\newif\if@lemmacommand@%`

22.3 Substitute line numbers

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

As argument `\linenum` takes a set of seven parameters separated by vertical bars, in the format used internally for `\l@d@nums` (see p.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 `\linenum{18|4|0|18|7|1|0}` is an invocation that changes all the parameters, but `\linenum{|3}` only changes the starting line number, and leaves the rest unaltered.

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

```
921 \newcommand*{\linenum}[1]{%
922   \xdef\@tempa{#1|||||\\noexpand\\l@d@nums}%
923   \global\let\l@d@nums=\empty
924   \expandafter\line@set\@tempa|\\ignorespaces}
```

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

```
925 \def\line@set#1|#2\\#3|#4\\{%
926   \gdef\@tempb{#1}%
927   \ifx\@tempb\empty
928     \l@d@add{#3}%
929   \else
930     \l@d@add{#1}%
931   \fi
932   \gdef\@tempb{#4}%
933   \ifx\@tempb\empty\else
934     \l@d@add{|}\line@set#2\\#4\\%
935   \fi}
```

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

```
936 \newcommand{\l@d@add}[1]{\xdef\l@d@nums{\l@d@nums#1}}
937
```

22.4 Lemma disambiguation

The mechanism which counts the occurrence of a same word in a same line is quite complex, because, when L^AT_EX reads a command between a `\pstart` and a `\pend`,

it does not know yet which are the line numbers.

The general mechanism is the following:

- **At the first run**, each `\sameword` command increments an `etoolbox` counter the name of which contains the argument of the `\sameword` commands.
- Then this counter, associated with the argument of `\sameword` is stored (`\@sw` command) in the auxiliary file of the current `eledmac` section (the `.1`, `.2...` file).
- **When this auxiliary file is read at the second run**, different operations are achieved:
 - For each paired `\sameword` argument and absolute line number, a counter is defined. Its value corresponds to the number of times `\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 `\sameword{argument}`.
 - A `\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 `\sameword` argument and previous absolute line number.
- At the second run, when the `\sameword` command is called, new operations happen. We first read the first element of the `\sw@list`, then delete it from this list, and, if we are inside a `\edtext` command, we store it in a `\sw@list@inedtext` list.
- At the second run, when the critical notes are built, the `\sameword@inedtext` command is used instead of `\sameword`. Then, we read the next value of `\sw@list@inedtext` list and remove it from this list. We send it to the `\showwordrank` to be printed after the lemma, but only if the current line has more than one value for the argument of `\sameword`. Otherwise, we just print the lemma, with no number.

```

\sw@list So, first, the lists.
\sw@list@inedtext 938 \list@create{\sw@list}%
                  939 \list@create{\sw@list@inedtext}%

\sameword The hight level macro \sameword, used by the editor.
940 \newcommandx{\sameword}[2][1,usedefault]{\leavevmode%
First, increament the counter corresponding to the argument.
941 \unless\ifledRcol%
942 \csnumgdef{sw@#2}{\csuse{sw@#2}+1}%
Then, write its value to the numbered file
943 \protected@write\linenum@out{}{\string\@sw{#2}{\csuse{sw@#2}}}%

```

At the second run, read the `\sw@list` next item and, if we are in `\edtext`, put it to `\sw@list@inedtext`.

```

944 \unless\ifx\sw@list\empty%
945 \gl@p\sw@list\to\@tempb%
946 \if@edtext@%
947 \unless\if@lemmacommand@%
948 \xright@appenditem{\@tempb}\to\sw@list@inedtext%
949 \else%
950 \ifstrequal{#1}{inlemma}{\xright@appenditem{\@tempb}\to\sw@list@inedtext}{}%
951 \fi%
952 \fi%
953 \global\let\@tempb=\undefined%
954 \fi%
955 % \end{macrocode}
956 % Do the same thing if we are in the right columns.
957 % \begin{macrocode}
958 \else%
959 \csnumgdef{sw@#2@R}{\csuse{sw@#2@R}+1}%
960 \protected@write\linenum@outR{\string\@sw{#2}{\csuse{sw@#2@R}}}%
961 \unless\ifx\sw@listR\empty%
962 \gl@p\sw@listR\to\@tempb%
963 \if@edtext@%
964 \unless\if@lemmacommand@%
965 \xright@appenditem{\@tempb}\to\sw@list@inedtextR%
966 \else%
967 \ifstrequal{#1}{inlemma}{\xright@appenditem{\@tempb}\to\sw@list@inedtextR}{}%
968 \fi%
969 \fi%
970 \global\let\@tempb=\undefined%
971 \fi%
972 \fi%
973 % \end{macrocode}
974 % In any case, print the word.
975 % \begin{macrocode}
976 #2%
977 }%
```

`\@sw` The command printed in the auxiliary files.

```

978 \newcommand{\@sw}[2]{%
979 \unless\ifledRcol%
```

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

```
980 \csxdef{sw@#1@the\absline@num @\the\section@num}{#2}%
```

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

```

981 \numdef{\prev@line}{\the\absline@num-1}%
982 \ifcsundef{sw@#1@prev@line @\the\section@num}{%
983 \csnumgdef{sw@#1@prev@line @\the\section@num}{#2-1}%
984 }{}
```

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

```
985 \numdef{\the@sw}{#2-\csuse{sw@#1@prev@line @\the\section@num}}%
986 \xright@appenditem{\the@sw}\to\sw@list%
```

And do the same thing for the right side.

```
987 \else%
988 \csxdef{sw@#1@the\absline@numR @\the\section@numR @R}{#2}%
989 \numdef{\prev@line}{\the\absline@numR-1}%
990 \ifcsundef{sw@#1@prev@line @\the\section@numR @R}{%
991 \csnumgdef{sw@#1@prev@line @\the\section@numR @R}{#2-1}%
992 }{}%
993 \numdef{\the@sw}{#2-\csuse{sw@#1@prev@line @\the\section@numR @R}}%
994 \xright@appenditem{\the@sw}\to\sw@listR%
995 \fi%
996 }%
```

`\sameword@inedtext` The command called when `\sameword` is called in a edtext.

```
997 \def\sameword@inedtext#1{%
998 \unless\ifledRcol%
```

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

```
999 \ifcsdef{sw@#1@the\absline@num @\the\section@num}{%
1000 \numdef{\prev@line}{\the\absline@num-1}%
1001 \numdef{\sw@atthisline}{\csuse{sw@#1@the\absline@num @\the\section@num}-\csuse{sw@#1@prev@line}}%
1002 }%
1003 {\numdef{\sw@atthisline}{0}}%
```

Just a precaution.

```
1004 \ifx\sw@list@inedtext\empty%
1005 \def\the@sw{999}%
1006 \else%
```

But in many cases, at this step, we should have some content in the list `\sw@list@inedtext`, which contains the reference for edtext.

```
1007 \glp\sw@list@inedtext\to\the@sw%
1008 \fi%
```

Then, print the rank, but only if there is more than one occurrence of the word in the current line.

```
1009 \ifnumgreater{\sw@atthisline}{1}%
1010 {\showwordrank{#1}{\the@sw}}%
1011 {#1}%
```

And the same for right side.

```
1012 \else%
1013 \ifcsdef{sw@#1@the\absline@numR @\the\section@numR @R}{%
1014 \numdef{\prev@line}{\the\absline@numR-1}%
1015 \numdef{\sw@atthisline}{\csuse{sw@#1@the\absline@numR @\the\section@numR @R}-\csuse{sw@#1@prev@line}}%
1016 }%
1017 {\numdef{\sw@atthisline}{0}}%
1018 \ifx\sw@list@inedtextR\empty%
1019 \def\the@sw{999}%
```

```

1020      \else%
1021        \gl@p\sw@list@inedtextR\to\the@sw%
1022      \fi%
1023      \ifnumgreater{\sw@atthisline}{1}%
1024        {\showwordrank{#1}{\the@sw}}%
1025        {#1}%
1026    \fi%
1027 }%

\showwordrank

1028 % Finally, the way the rank will be printed.
1029 \newcommand{\showwordrank}[2]{%
1030   #1\textsuperscript{#2}%
1031 }%
```

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

```

\raw@text Here are numbers and flags that are used internally in the course of the paragraph
\ifnumberedpar@ decomposition.
\numberedpar@true When we first form the paragraph, it goes into a box register, \raw@text,
\numberedpar@false instead of onto the current vertical list. The \ifnumberedpar@ flag will be true
\num@lines while a paragraph is being processed in that way. \num@lines will store the
\one@line number of lines in the paragraph when it's complete. When we chop it up into
\par@line lines, each line in turn goes into the \one@line register, and \par@line will be
the number of that line within the paragraph.

1032 \newbox\raw@text
1033 \newif\ifnumberedpar@
1034 \newcount\num@lines
1035 \newbox\one@line
1036 \newcount\par@line

\pstart \pstart starts the paragraph by clearing the \inserts@list list and other rele-
\AtEveryPstart vant variables, and then arranges for the subsequent text to go into the \raw@text
\numberpstarttrue box. \pstart needs to appear at the start of every paragraph that's to be num-
\numberpstartfalse bered; the \autopar command below may be used to insert these commands
\labelpstarttrue automatically.
\labelpstartfalse
\thepstart
```


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.

```

1037
1038 \newcommand{\AtEveryPstart}[1]{\xdef\at@every@pstart{\noindent\unexpanded{#1}}}%
1039 \xdef\at@every@pstart{}%
1040
1041 \newcounter{pstart}
1042 \renewcommand{\thepstart}{\bfseries\@arabic\c@pstart}. }
1043 \newif\ifnumberpstart
1044 \numberpstartfalse
1045 \newif\iflabelpstart
1046 \labelpstartfalse
1047 \newcommand*{\pstart}[1][1]{%
1048   \ifstrempy{#1}{\at@every@pstart}{\noindent#1}%
1049   \if@nobreak%
1050     \let\@oldnobreak\@nobreaktrue%
1051   \else%
1052     \let\@oldnobreak\@nobreakfalse%
1053   \fi%
1054   \@nobreaktrue%
1055   \ifnumbering \else%
1056     \led@err@PstartNotNumbered%
1057     \beginnumbering%
1058   \fi%
1059   \ifnumberedpar@%
1060     \led@err@PstartInPstart%
1061   \pend%
1062   \fi%
1063   \list@clear{\inserts@list}%
1064   \global\let\next@insert=\empty%
1065   \begingroup\normal@pars%
1066   \global\advance \l@dnumpstartsL\@ne
1067   \global\setbox\raw@text=\vbox\bgroup%
1068     \ifautopar\else%
1069       \ifnumberpstart%
1070         \ifinstanza\else%
1071           \ifsidepstartnum\else%
1072             \thepstart%
1073           \fi%
1074         \fi%
1075       \fi%
1076     \fi%
1077   \numberedpar@true%
1078   \iflabelpstart\protected@edef\@currentlabel%
1079     {\p@pstart\thepstart}%
1080   \fi%
1081   \l@dzeropenalties%
1082 }
```

`\pend` `\pend` must be used to end a numbered paragraph.

```

1083 \newcommand*{\pend}[1][1]{\ifnumbering \else%
1084   \led@err@PendNotNumbered%
1085   \fi%
1086   \ifnumberedpar@ \else%
1087   \led@err@PendNoPstart%
1088   \fi%

```

We set all the usual interline penalties to zero and then immediately call `\endgraf` to end the paragraph; this ensures that there'll be no large interline penalties to prevent us from slicing the paragraph into pieces. These penalties revert to the values that you set when the group for the `\vbox` ends. Then we call `\do@line` to slice a line off the top of the paragraph, add a line number and footnotes, and restore it to the page; we keep doing this until there aren't any more lines left.

```

1089 \l@dzero penalties%
1090 \endgraf\global\num@lines=\prevgraf\egroup%
1091 \global\par@line=0%

```

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

```

1092 \csnumdef{pstartline}{0}%
1093 \loop\ifvbox\raw@text%
1094   \csnumdef{pstartline}{\pstartline+1}%
1095   \do@line%
1096   \ifbypstart@%
1097     \ifnumequal{pstartline}{1}{\setline{1}\resetprevline@}{}%
1098   \fi%
1099 \repeat%

```

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

```

1100 \flush@notes%
1101 \endgroup%
1102 \ignorespaces%
1103 \ifnumberpstart%
1104   \pstartnumtrue%
1105   \fi%
1106   \@oldnobreak%
1107   \addtocounter{pstart}{1}%
1108   \ifstrempy{#1}{\at@every@pend}{\noindent#1}%
1109 }
1110

```

```

\AtEveryPend
\at@every@pend 1111
1112 \newcommand{\AtEveryPend}[1]{\xdef\at@every@pend{\noindent\unexpanded{#1}}}%
1113 \xdef\at@every@pend{}%
1114

```

`\l@dzzeropenalties` A macro to zero penalties for `\pend`.

```

1115 \newcommand*\l@dzzeropenalties{%
1116   \brokenpenalty \z@ \clubpenalty \z@
1117   \displaywidowpenalty \z@ \interlinepenalty \z@ \predisplaypenalty \z@
1118   \postdisplaypenalty \z@ \widowpenalty \z@}
1119
```

`\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.

```

1120 \newif\ifautopar
1121 \autoparfalse
1122 \newcommand*\autopar{%
1123   \ifledRcol
1124     \ifnumberingR \else
1125       \led@err@AutoparNotNumbered
1126       \beginnumberingR
1127       \fi
1128     \else
1129       \ifnumbering \else
1130         \led@err@AutoparNotNumbered
1131         \beginnumbering
1132         \fi
1133       \fi
1134       \autopartrue
1135       \everypar{\setbox0=\lastbox
1136         \endgraf \vskip-\parskip

```

```

1137 \pstart \noindent \kern\wd0 \ifnumberpstart\ifinstanza\else\thepstart\fi\fi
1138 \let\par=\pend}%
1139 \ignorespaces}

```

`\normal@pars` 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.

```

1140 \newcommand*{\normal@pars}{\everypar{}\let\par\endgraf}
1141

```

`\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.

```

1142 \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.

```

1143 \newcommand*{\l@dunhbox@line}[1]{\unhbox #1}
1144 \newcommand*{\do@line}{%
1145   {\vbadness=10000
1146     \splittopskip=\z@
1147     \do@linehook
1148   \l@demptyd@ta
1149     \global\setbox\one@line=\vsplit\raw@text to\baselineskip}%
1150   \unvbox\one@line \global\setbox\one@line=\lastbox
1151   \getline@num
1152   \IfStrEq{\led@pb@setting}{before}{\led@check@pb\led@check@nopb}{%
1153     \ifnum\@lock>\@ne
1154       \inserthangingsymboltrue
1155     \else
1156       \inserthangingsymbolfalse
1157     \fi
1158     \check@pb@in@verse
1159     \affixline@num

```

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

```

1160   \xifinlist{\the\l@dnumpstartsL}{\eled@sections@}%
1161     {\print@eledsection}%
1162     {\print@line}%
1163   \IfStrEq{\led@pb@setting}{after}{\led@check@pb\led@check@nopb}{%
1164     }%

```

`\print@line`

```

1165 % \cs{print@line} is for normal line, i. e line without sectioning command.
1166 \def\print@line{

```

```

1167 \affixpstart@num%
1168 \hb@xt@ \linewidth{\do@insidelinehook\l@dld@ta%
1169 \add@inserts\affixside@note%
1170 \l@dlsn@te
1171 {\ledllfill\hb@xt@ \wd\one@line{\new@line\inserthangingsymbol \l@dunhbox@line{\one@line}}\led
1172 \l@drsn@te
1173 }}%
1174 \add@penalties%
1175 }

```

`\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.

```

1176 \def\print@eledsection{%
1177 \add@inserts\affixside@note%
1178 \numdef{\temp@}{\l@dnumpstartsL-1}%
1179 \xifinlist{\temp@}{\eled@sections@}{\@nbreaktrue}{\@nbreakfalse}%
1180 \@eled@sectioningtrue%
1181 \csuse{eled@sectioning@the\l@dnumpstartsL}%
1182 \@eled@sectioningfalse%
1183 \global\csundef{eled@sectioning@the\l@dnumpstartsL}%
1184 \if@RTL%
1185 \hspace{-3\paperwidth}%
1186 {\hbox{\l@dunhbox@line{\one@line}}\new@line}%
1187 \else%
1188 \hspace{3\paperwidth}%
1189 {\new@line \hbox{\l@dunhbox@line{\one@line}}}%
1190 \fi%
1191 \vskip-\baselineskip%
1192 }

```

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

```

1193 \newcommand*\dolinehook[1]{\gdef\do@linehook{#1}}%
1194 \newcommand*\doinsidelinehook[1]{\gdef\do@insidelinehook{#1}}%
1195

```

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

```

1196 \newcommand*\do@linehook{}
1197 \newcommand*\do@insidelinehook{}

```

`\l@emptyd@ta` Nulls the `\...d@ta`, which may later hold line numbers. Similarly for `\l@dcstotetext`, `\l@dld@ta` `\l@dcstotetext@l`, `\l@dcstotetext@r` for the texts of the sidenotes, left and `\l@drd@ta` right notes.

```

\l@dcstotetext
\l@dcstotetext@l
\l@dcstotetext@r

```

```

1198 \newcommand*{\l@emptyd@ta}{%
1199   \gdef\l@dld@ta}{%
1200   \gdef\l@drd@ta}{%
1201   \gdef\l@dcsnotetext@l}{%
1202   \gdef\l@dcsnotetext@r}{%
1203   \gdef\l@dcsnotetext{}%
1204

```

`\l@dlsn@te` Zero width boxes of the left and right side notes, together with their kerns.

```

\l@drsn@te 1205 \newcommand*{\l@dlsn@te}{%
1206   \hb@xt@ \z@{\hss\box\l@dldp@rbox\kern\ledlsnotesep}}
1207 \newcommand*{\l@drsn@te}{%
1208   \hb@xt@ \z@{\kern\ledrsnotesep\box\l@drp@rbox\hss}}
1209

```

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

```

1210 \newcommand*{\ledllfill}{\hfil}
1211 \newcommand*{\ledrlfill}{\hfil}
1212

```

23.3 Line and page number computation

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

```

1213 \newcommand*{\getline@num}{%
1214   \global\advance\absline@num \@ne
1215   \do@actions
1216   \do@ballast
1217   \ifnumberline
1218   \ifsublines@
1219     \ifnum\sub@lock<\tw@
1220       \global\advance\subline@num \@ne
1221       \fi
1222   \else
1223     \ifnum\@lock<\tw@
1224       \global\advance\line@num \@ne
1225       \global\subline@num \z@
1226     \fi
1227   \fi
1228   \fi
1229 }

```

`\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).

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

```
1230 \newcount\ballast@count
1231 \newcounter{ballast}
1232 \setcounter{ballast}{0}
```

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

```
1233 \newcommand*\do@ballast{\global\ballast@count \z@
1234 \begingroup
1235   \advance\absline@num \@ne
1236   \ifnum\next@actionline=\absline@num
1237     \ifnum\next@action>-1001\relax
1238     \global\advance\ballast@count by -\c@ballast
1239   \fi
1240 \fi
1241 \endgroup}
```

$\backslash\text{do@actions}$ The $\backslash\text{do@actions}$ macro looks at the list of actions to take at particular absolute
 $\backslash\text{do@actions@next}$ 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 $\backslash\text{do@actions@next}$ that is always the last thing that $\backslash\text{do@actions}$ does. If there could be more actions to process for this line, $\backslash\text{do@actions@next}$ is set equal to $\backslash\text{do@actions}$; otherwise it's just $\backslash\text{relax}$.

```
1242 \newcommand*\do@actions{\%
1243   \global\let\do@actions@next=\relax
1244   \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.

```
1245   \ifnum\next@action>-1001
1246     \global\page@num=\next@action
1247     \ifbypage@
1248       \global\line@num=\z@ \global\subline@num=\z@
1249     \resetprevline@
1250   \fi
```

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

```
1251   \else
1252     \ifnum\next@action<-4999
1253       \@l@dttempcnta=-\next@action
1254       \advance\@l@dttempcnta by -5001
1255     \ifsublines@
```

```

1256         \global\subline@num=\@l@dttempcnta
1257     \else
1258         \global\line@num=\@l@dttempcnta
1259     \fi

```

It's one of the fixed codes. We rescale the value in `\@l@dttempcnta` so that we can use a case statement.

```

1260     \else
1261         \@l@dttempcnta=-\next@action
1262         \advance\@l@dttempcnta by -1000
1263         \do@actions@fixedcode
1264     \fi
1265 \fi

```

Now we get information about the next action off the list, and then set `\do@actions@next` so that we'll call ourselves recursively: the next action might also be for this line.

There's no warning if we find `\actionlines@list` empty, since that will always happen near the end of the section.

```

1266     \ifx\actionlines@list\empty
1267         \gdef\next@actionline{1000000}%
1268     \else
1269         \glp\actionlines@list\to\next@actionline
1270         \glp\actions@list\to\next@action
1271         \global\let\do@actions@next=\do@actions
1272     \fi
1273 \fi

```

Make the recursive call, if necessary.

```

1274 \do@actions@next}
1275

```

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

```

1276 \newcommand*{\do@actions@fixedcode}{%
1277     \ifcase\@l@dttempcnta
1278     \or%                % 1001
1279         \global\sublines@true
1280     \or%                % 1002
1281         \global\sublines@false
1282     \or%                % 1003
1283         \global\@lock=\@ne
1284     \or%                % 1004
1285         \ifnum\@lock=\tw@
1286             \global\@lock=\thr@@
1287         \else
1288             \global\@lock=\z@
1289         \fi
1290     \or%                % 1005
1291         \global\sub@lock=\@ne

```



```

1292 \or% % 1006
1293 \ifnum\sub@lock=\tw@
1294 \global\sub@lock=\thr@@
1295 \else
1296 \global\sub@lock=\z@
1297 \fi
1298 \or% % 1007
1299 \l@dskipnumbertrue
1300 \else
1301 \led@warn@BadAction
1302 \fi}
1303
1304

```

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@dld@ta` or a right line number into `\l@drd@ta` if required.

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

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

$$m = \text{firstlinenum} + (n \times \text{linenumincrement})$$

(where *int* truncates a real number to an integer). *m* will be equal to *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 ≤ \firstlinenum`, we compare the two directly instead of making these calculations.

We compute, in the scratch counter `\@l@tempcnta`, the number of the next line that should be printed with a number (*m* in the above discussion), and move the current line number into the counter `\@l@tempcntb` for comparison.

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

```

1305 \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).

```

1306 \ifledgroupnotesL@\else\ifnumberline
1307 \ifl@dskipnumber
1308 \global\l@dskipnumberfalse
1309 \else
1310 \ifsublines@
1311 \@l@tempcntb=\subline@num
1312 \ifnum\subline@num>\c@firstsublinenum
1313 \@l@tempcnta=\subline@num
1314 \advance\@l@tempcnta by-\c@firstsublinenum

```

```

1315     \divide\@l@tempcnta by\c@sublinenumincrement
1316     \multiply\@l@tempcnta by\c@sublinenumincrement
1317     \advance\@l@tempcnta by\c@firstsublinenum
1318   \else
1319     \@l@tempcnta=\c@firstsublinenum
1320   \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.

```

1321   \ch@cksub@l@ck

```

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

```

1322   \else
1323     \@l@tempcntb=\line@num

```

Check on the `\linenumberlist` If it's `\empty` use the standard algorithm.

```

1324   \ifx\linenumberlist\empty
1325     \ifnum\line@num>\c@firstlinenum
1326       \@l@tempcnta=\line@num
1327       \advance\@l@tempcnta by-\c@firstlinenum
1328       \divide\@l@tempcnta by\c@linenumincrement
1329       \multiply\@l@tempcnta by\c@linenumincrement
1330       \advance\@l@tempcnta by\c@firstlinenum
1331     \else
1332       \@l@tempcnta=\c@firstlinenum
1333     \fi
1334   \else

```

The `\linenumberlist` wasn't `\empty`, so here's Wayne's numbering mechanism. This takes place in TeX's mouth.

```

1335     \@l@tempcnta=\line@num
1336     \edef\rem@inder{\,\linenumberlist,\number\line@num,}%
1337     \edef\sc@n@list{\def\noexpand\sc@n@list
1338       ####1,\number\@l@tempcnta,####2|{\def\noexpand\rem@inder{####2}}}%
1339     \sc@n@list\expandafter\sc@n@list\rem@inder|
1340     \ifx\rem@inder\empty\advance\@l@tempcnta\@ne\fi
1341   \fi

```

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

```

1342   \ch@ck@l@ck
1343   \fi

```

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

```

1344   \ifnum\@l@tempcnta=\@l@tempcntb

```

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 L^AT_EX 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 twocolumn stuff before going on with the original code.

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

```
\l@drd@ta 1345 \if@twocolumn
1346 \if@firstcolumn
1347 \gdef\l@dld@ta{\llap{\leftlinenum}}}%
1348 \else
1349 \gdef\l@drd@ta{\rlap{\rightlinenum}}}%
1350 \fi
1351 \else
```

Continuing the original code ...

```
1352 \@l@tempcntb=\line@margin
1353 \ifnum\@l@tempcntb>\@ne
1354 \advance\@l@tempcntb \page@num
1355 \fi
```

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

```
1356 \ifodd\@l@tempcntb
1357 \gdef\l@drd@ta{\rlap{\rightlinenum}}}%
1358 \else
1359 \gdef\l@dld@ta{\llap{\leftlinenum}}}%
1360 \fi
1361 \fi
1362 \else
```

As no line number is to be appended, we just print the line as is.

```
1363 %% #1%
1364 \fi
```

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

```
1365 \f@x@l@cks
1366 \fi
1367 \fi
1368 \fi
1369 }
1370
```

`\ch@cksub@l@ck` These macros handle line number locking for `\affixline@num`. `\ch@cksub@l@ck`
`\ch@ck@l@ck` checks subtitle locking. If it fails, then we disable the line-number display by setting
`\f@x@l@cks` the counters to arbitrary but unequal values.

```
1371 \newcommand*{\ch@cksub@l@ck}{%
1372 \ifcase\sub@lock
1373 \or
```

```

1374     \ifnum\sublock@disp=\@ne
1375         \@l@dttempcntb=\z@ \@l@dttempcnta=\@ne
1376     \fi
1377     \or
1378         \ifnum\sublock@disp=\tw@ \else
1379             \@l@dttempcntb=\z@ \@l@dttempcnta=\@ne
1380         \fi
1381     \or
1382         \ifnum\sublock@disp=\z@
1383             \@l@dttempcntb=\z@ \@l@dttempcnta=\@ne
1384         \fi
1385     \fi}

```

Similarly for line numbers.

```

1386 \newcommand*{\ch@ck@l@ck}{%
1387     \ifcase\@lock
1388     \or
1389         \ifnum\lock@disp=\@ne
1390             \@l@dttempcntb=\z@ \@l@dttempcnta=\@ne
1391         \fi
1392     \or
1393         \ifnum\lock@disp=\tw@ \else
1394             \@l@dttempcntb=\z@ \@l@dttempcnta=\@ne
1395         \fi
1396     \or
1397         \ifnum\lock@disp=\z@
1398             \@l@dttempcntb=\z@ \@l@dttempcnta=\@ne
1399         \fi
1400     \fi}

```

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

```

1401 \newcommand*{\f@x@l@cks}{%
1402     \ifcase\@lock
1403     \or
1404         \global\@lock=\tw@
1405     \or \or
1406         \global\@lock=\z@
1407     \fi
1408     \ifcase\sub@lock
1409     \or
1410         \global\sub@lock=\tw@
1411     \or \or
1412         \global\sub@lock=\z@
1413     \fi}
1414

```

`\pageparbreak` Because of TeX's asynchronous page breaking mechanism we can never be sure juust 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.

```
1415 \newcommand{\pageparbreak}{\pend\newpage\pstart\noindent}
1416
```

23.5 Pstart number printing in side

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

- The pstarts counter is upgrade in the `\pend` command. Consequently, the `\affixpstart@num` command has not to upgrade it, unlike the `\affixline@num` which upgrades the lines counter.
- To print the pstart number only at the beginning of a pstart, and not in every line, a boolean test is made. The `\pstartnum` boolean is set to TRUE at every `\pend`. It's tried in the `\leftpstartnum` and `\rightpstartnum` commands. After the try, it is set to FALSE.

```
\leftpstartnum
\rightpstartnum 1417
\ifsidepstartnum 1418 \newif\ifsidepstartnum
1419 \newcommand*{\affixpstart@num}{%
1420     \ifsidepstartnum
1421         \if@twocolumn
1422             \if@firstcolumn
1423                 \gdef\l@dld@ta{\llap{\leftpstartnum}}}%
1424             \else
1425                 \gdef\l@drd@ta{\rlap{\rightpstartnum}}}%
1426             \fi
1427         \else
1428             \@l@tempcntb=\line@margin
1429             \ifnum\@l@tempcntb>\@ne
1430                 \advance\@l@tempcntb \page@num
1431             \fi
1432             \ifodd\@l@tempcntb
1433                 \gdef\l@drd@ta{\rlap{\rightpstartnum}}}%
1434             \else
1435                 \gdef\l@dld@ta{\llap{\leftpstartnum}}}%
1436             \fi
1437         \fi
1438     \fi
1439 }
1440 }
1441 %
1442
1443 \newif\ifpstartnum
1444 \pstartnumtrue
```

```

1445 \newcommand*{\leftpstartnum}{
1446     \ifpstartnum\thepstart
1447     \kern\linenumsep\fi
1448     \global\pstartnumfalse
1449 }
1450 \newcommand*{\rightpstartnum}{
1451     \ifpstartnum
1452     \kern\linenumsep
1453     \thepstart
1454     \fi
1455     \global\pstartnumfalse
1456 }

```

23.6 Add insertions to the vertical list

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

```
1457 \list@create{\inserts@list}
```

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

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

```

1458 \newcommand*{\add@inserts}{%
1459     \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.

```
1460 \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.

```

1461 \ifx\next@insert\empty
1462     \ifx\insertlines@list\empty
1463         \global\noteschanged@true
1464         \gdef\next@insert{100000}%
1465     \else
1466         \gl@p\insertlines@list\to\next@insert
1467     \fi
1468 \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.

```

1469 \ifnum\next@insert=\absline@num
1470 \gl@p\inserts@list\to\@insert
1471 \@insert
1472 \global\let\@insert=\undefined
1473 \global\let\next@insert=\empty
1474 \global\let\add@inserts@next=\add@inserts
1475 \fi
1476 \fi

```

Make the recursive call, if necessary.

```

1477 \add@inserts@next}
1478

```

23.7 Penalties

`\add@penalties` `\add@penalties` is the last macro used by `\do@line`. It adds up the club, widow, and interline penalties, and puts a single penalty of the appropriate size back into the paragraph; these penalties get removed by the `\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@dttempcnta` 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.94). Finally, the penalty is checked to see that it doesn't go below -10000 .

```

1479 \newcommand*{\add@penalties}{\@l@dttempcnta=\ballast@count
1480 \ifnum\num@lines>\@ne
1481 \global\advance\par@line \@ne
1482 \ifnum\par@line=\@ne
1483 \advance\@l@dttempcnta \clubpenalty
1484 \fi
1485 \@l@dttempcntb=\par@line \advance\@l@dttempcntb \@ne
1486 \ifnum\@l@dttempcntb=num@lines
1487 \advance\@l@dttempcnta \widowpenalty
1488 \fi
1489 \ifnum\par@line<\num@lines
1490 \advance\@l@dttempcnta \interlinepenalty
1491 \fi
1492 \fi
1493 \ifnum\@l@dttempcnta=z@
1494 \relax
1495 \else
1496 \ifnum\@l@dttempcnta>-10000
1497 \penalty\@l@dttempcnta
1498 \else
1499 \penalty -10000
1500 \fi
1501 \fi}
1502

```

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.

```
1503 \newcommand*{\flush@notes}{%
1504   \@xloop
1505     \ifx\inserts@list\empty \else
1506       \gl@p\inserts@list\to\@insert
1507       \@insert
1508       \global\let\@insert=\undefined
1509   \repeat}
1510
```

`\@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.

```
1511 \def\@xloop#1\repeat{%
1512   \def\body{#1\expandafter\body\fi}%
1513   \body}
1514
```

24 Critical footnotes

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

24.1 Fonts

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

`\select@lemmafont` `\select@lemmafont` is provided to set the right font for the lemma in a note.
`\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.

```
1515 \def\select@lemmafont#1|#2|#3|#4|#5|#6|#7|{\select@@lemmafont#7|}
1516 \def\select@@lemmafont#1/#2/#3/#4|{%
1517   {\fontencoding{#1}\fontfamily{#2}\fontseries{#3}\fontshape{#4}%
1518    \selectfont}
1519
```

24.2 Outer-level footnote commands

`\footnoteoptions@` The `\footnoteoption@`[\side][\options]{\value} change the value of on options of Xfootnote, to switch between true and false.

```
1520 \newcommandx*{\footnoteoptions@}[3][1=L,usedefault]{%
1521   \def\do##1{%
1522     \ifstrequal{#1}{L}{% In Leftside
1523       \xright@appenditem{\global\noexpand\settoggle{##1@}{#3}}\to\inserts@list% Switch toogle, in
1524       \global\advance\insert@count \@ne% Increment the left insert counter.
1525     }%
1526     {%
1527       \xright@appenditem{\global\noexpand\settoggle{##1@}{#3}}\to\inserts@listR% Switch toogle, in
1528       \global\advance\insert@countR \@ne% Increment the right insert counter insert.
1529     }%
1530   }%
1531   \notblank{#2}{\docsvlist{#2}}{\}% Parsing all options
1532 }
```

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

```
1533 \newcommandx*{\footnotelang@lua}[1][1=L,usedefault]{%
1534   \ifstrequal{#1}{L}{%
1535     \xright@appenditem{{\csxdef{footnote@luatextextdir}{\the\luatextextdir}}}\to\inserts@list%Know the
1536     \global\advance\insert@count \@ne%
1537     \xright@appenditem{{\csxdef{footnote@luatexpardir}{\the\luatexpardir}}}\to\inserts@list%Know the
1538     \global\advance\insert@count \@ne%
1539   }%
1540   {%
1541     \xright@appenditem{{\csxdef{footnote@luatextextdir}{\the\luatextextdir}}}\to\inserts@listR%Know the
1542     \global\advance\insert@countR \@ne%
1543     \xright@appenditem{{\csxdef{footnote@luatexpardir}{\the\luatexpardir}}}\to\inserts@listR%Know the
1544     \global\advance\insert@countR \@ne%
1545   }%
1546 }
```

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

```
1547 \newcommandx*{\footnotelang@poly}[1][1=L,usedefault]{%
```

```

1548 \ifstrequal{#1}{L}{%
1549 \if@RTL%
1550 \xright@appenditem{{\csxdef{footnote@dir}{@RTLtrue}}}\to\inserts@list%Know the lan
1551 \global\advance\insert@count \@ne%
1552 \else
1553 \xright@appenditem{{\csxdef{footnote@dir}{@RTLfalse}}}\to\inserts@list%Know the l
1554 \global\advance\insert@count \@ne%
1555 \fi%
1556 \xright@appenditem{{\csxdef{footnote@lang}{\expandonce\language\language}}}\to\inserts@list
1557 \global\advance\insert@count \@ne%
1558 }%
1559 {%
1560 \if@RTL
1561 \xright@appenditem{{\csxdef{footnote@dir}{@RTLtrue}}}\to\inserts@listR%Know the lan
1562 \global\advance\insert@countR \@ne%
1563 \else
1564 \xright@appenditem{{\csxdef{footnote@dir}{@RTLfalse}}}\to\inserts@listR%Know the l
1565 \global\advance\insert@countR \@ne%
1566 \fi
1567 \xright@appenditem{{\csxdef{footnote@lang}{\expandonce\language\language}}}\to\inserts@list
1568 \global\advance\insert@countR \@ne%
1569 }%
1570 }

```

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.

```

1571 \notbool{parapparatus@}{\newcommand*}{\newcommand}{\normalvfootnote}[2]{%
1572 \insert\csname #1footins\endcsname\bgroup

```

```

1573 \csuse{bhookXnote@#1}
1574 \csuse{Xnotefontsize@#1}
1575 \footssplitskips
1576 \ifl@dpairing\ifl@dpaging\else%
1577   \setXnoteswidthliketwocolumns@{#1}%
1578 \fi\fi%
1579 \setXnotespositionliketwocolumns@{#1}%
1580 \spaceskip=\z@skip \xspaceskip=\z@skip
1581 \csname #1footfmt\endcsname #2[#1]\egroup}

```

`\footssplitskips` Some setup code that is common for a variety of the footnotes.

```

1582 \newcommand*{\footssplitskips}{%
1583   \interlinepenalty=\interfootnotelinepenalty
1584   \floatingpenalty=\@MM
1585   \splittopskip=\ht\strutbox \splitmaxdepth=\dp\strutbox
1586   \leftskip=\z@skip \rightskip=\z@skip}
1587

```

`\mpnormalvfootnote` And a somewhat different version for minipages.

```

1588 \notbool{parapparatus@}{\newcommand*{\newcommand}{\mpnormalvfootnote}[2]{%
1589   \global\setbox\@nameuse{mp#1footins}\vbox{%
1590     \unvbox\@nameuse{mp#1footins}
1591     \csuse{bhookXnote@#1}
1592     \csuse{Xnotefontsize@#1}
1593     \hsize\columnwidth
1594     \@parboxrestore
1595     \color@begingroup
1596     \csname #1footfmt\endcsname #2[#1]\color@endgroup}}
1597

```

`\ledsetnormalparstuff` `\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 `\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.

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

```

1598 \newcommand*{\ledsetnormalparstuff}{%
1599   \ifluatex%
1600     \luatextextdir\footnote@luatextextdir%
1601     \luatexpardir\footnote@luatexpardir%
1602   \fi%
1603   \csuse{\csuse{footnote@dir}}%
1604   \normal@pars%
1605   \noindent \parfillskip \z@ \@plus 1fil}

```

```

1606
1607 \notbool{parapparatus@}{\newcommandx*}{\newcommandx}{\normalfootfmt}[4][4=Z]{% 4th arg is
1608   \ledsetnormalparstuff%
1609   \hangindent=\csuse{Xhangindent@#4}
1610   \strut{\printlinefootnote{#1}{#4}}}%
1611   {\nottoggle{Xlemmadisablefontselection@#4}{\select@lemmafont#1|#2}{#2}}}%
1612   \iftoggle{nosep@}{\hskip\csuse{inplaceoflemmaseparator@#4}}{\ifcsemtyp{lemmaseparator@#4}
1613     {\hskip\csuse{inplaceoflemmaseparator@#4}}}%
1614     {\nobreak\hskip\csuse{beforelemmaseparator@#4}\csuse{lemmaseparator@#4}\hskip\csuse{a
1615     }}%
1616   #3\strut\par}

```

\endashchar 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.

```

1617 \def\endashchar{\textnormal{--}}
1618 \newcommand*{\fullstop}{\textnormal{.}}
1619 \newcommand*{\rbracket}{\textnormal{}}
1620   \csuse{text\csuse{footnote@lang}}{%
1621     \ifluatex%
1622     \ifdefstring{\footnote@luatextextdir}{TRT}{\thinspace[]\thinspace[]}%
1623     \else%
1624     \thinspace}%
1625     \fi}%
1626   }%
1627 }
1628

```

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

```

1629 \newcommand{\printpstart}[0]{%
1630   \ifl@dpairing%
1631     \ifledRcol%
1632       \thepstartR%
1633     \else%
1634       \thepstartL%
1635     \fi%
1636   \else%
1637     \thepstart%

```

```

1638     \fi%
1639 }

```

The `\printlines` macro prints the line numbers for a note—which, in the general case, is a rather complicated task. The seven parameters of the argument are the line numbers as stored in `\l@d@nums`, in the form described on page 62: the starting page, line, and sub-line numbers, followed by the ending page, line, and sub-line numbers, and then the font specifier for the lemma.

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

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

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

There’s no counter for the line number because it’s always printed.

L^AT_EX tends to use a lot of counters and packages should try and minimise the number of new ones they create. In line with this Peter Wilson have reverted to traditional booleans.

```

\ifl@d@pnum
\ifl@d@ssub 1640 \newif\ifl@d@pnum
\ifl@d@elin 1641 \l@d@pnumfalse
\ifl@d@esl 1642 \newif\ifl@d@ssub
\ifl@d@dash 1643 \l@d@ssubfalse
1644 \newif\ifl@d@elin
1645 \l@d@elinfalse
1646 \newif\ifl@d@esl
1647 \l@d@eslfalse
1648 \newif\ifl@d@dash
1649 \l@d@dashfalse

\l@dp@rsefootsspec \l@dp@rsefootsspec{<spec>}{<lemma>}{<text>} parses a footnote specification.
\l@dp@rsefootsspec <lemma> and <text> are the lemma and text respectively. <spec> is the line and
\l@dp@rsefootsspec page number and lemma font specifier in \l@d@nums style format. The real work
\l@dp@rsefootsspec is done by \l@dp@rsefootsspec which defines macros holding the numeric values.
\l@dp@rsefootsspec 1650 \newcommand*{\l@dp@rsefootsspec}[3]{\l@dp@rsefootsspec#1|}
\l@dp@rsefootsspec 1651 \def\l@dp@rsefootsspec#1|#2|#3|#4|#5|#6|#7|{%
\l@dp@rsefootsspec 1652 \gdef\l@dp@rsefootsspec#1|}%
\l@dp@rsefootsspec 1653 \gdef\l@dp@rsefootsspec#1|#2|}%
\l@dp@rsefootsspec 1654 \gdef\l@dp@rsefootsspec#1|#2|#3|}%
\l@dp@rsefootsspec 1655 \gdef\l@dp@rsefootsspec#1|#2|#3|#4|}%
\l@dp@rsefootsspec 1656 \gdef\l@dp@rsefootsspec#1|#2|#3|#4|#5|}%
\l@dp@rsefootsspec 1657 \gdef\l@dp@rsefootsspec#1|#2|#3|#4|#5|#6|}%
1658 }

```

Initialise the several number value macros.

```
1659 \def\l@dparsedstartpage{0}%
1660 \def\l@dparsedstartline{0}%
1661 \def\l@dparsedstartsub{0}%
1662 \def\l@dparsedendpage{0}%
1663 \def\l@dparsedendline{0}%
1664 \def\l@dparsedendsub{0}%
1665
```

\setprintlines First of all, we print the page numbers only if: 1) we're doing the lineation by page, and 2) the ending page number is different from the starting page number.

Just a reminder of the arguments:

```
\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**.

```
1666 \newcommand*\setprintlines}[6]{%
1667   \l@dpnumfalse \l@ddashfalse
1668   \ifbypage@
1669     \ifnum#4=#1 \else
1670       \l@dpnumtrue
1671       \l@ddashtrue
1672     \fi
1673   \fi
```

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.

```
1674   \ifl@dpnum \l@d@elintrue \else \l@d@elinfalse \fi
1675   \ifnum#2=#5 \else
1676     \l@d@elintrue
1677     \l@ddashtrue
1678   \fi
```

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

```
1679   \l@d@ssubfalse
1680   \ifnum#3=0 \else
1681     \l@d@ssubtrue
1682   \fi
```

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.

```
1683   \l@d@eslfalse
1684   \ifnum#6=0 \else
1685     \ifnum#6=#3
1686       \ifl@d@elin \l@d@esltrue \else \l@d@eslfalse \fi
1687     \else
1688       \l@d@esltrue
1689       \l@ddashtrue
1690     \fi
1691   \fi}
```

`\printlines` Now we're ready to print it all. If the lineation is by `pstart`, we print the `pstart`.

```

1692 \def\printlines#1|#2|#3|#4|#5|#6|#7|{\begingroup
1693   \setprintlines{#1}{#2}{#3}{#4}{#5}{#6}%

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).

1694   \ifl@d@pnum #1\fullstop\fi
1695   \linenumrep{#2}

1696   \ifl@d@ssub \fullstop \sublinenumrep{#3}\fi
1697   \ifl@d@dash \endashchar\fi
1698   \ifl@d@pnum #4\fullstop\fi
1699   \ifl@d@elin \linenumrep{#5}\fi
1700   \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumrep{#6}\fi
1701 \endgroup}

```

`\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`.

```

1702 \newcommand*{\normalfootstart}[1]{%
1703   \ifdimequal{0pt}{\preXnotes@}{}%
1704   {%
1705     \iftoggle{preXnotes@}{%
1706       \togglefalse{preXnotes@}\skip\csname #1footins\endcsname=\csuse{preXnotes@}}%
1707     }%
1708   }%
1709   \vskip\skip\csname #1footins\endcsname%
1710   \leftskip0pt \rightskip0pt
1711   \ifl@d@pairing\else%
1712     \hsize=\old@hsize%
1713   \fi%
1714   \setXnoteswidthliketwocolumns@{#1}%

```

```

1715 \setXnotespositionliketwocolumns@{#1}%
1716 \print@Xfootnoterule{#1}%%
1717 \vskip\csuse{afterXrule@#1}%
1718 \noindent\leavevmode}

```

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

```
1719 \let\normalfootnoterule=\footnoterule
```

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

```

1720 \newcommand*{\normalfootgroup}[1]{%
1721   {\csuse{Xnotefontsize@#1}\noindent\csuse{txtbeforeXnotes@#1}}%
1722   \unvbox\csname #1footins\endcsname%
1723   \hsize=\old@hsize%
1724   }%
1725

```

`\mpnormalfootgroup` A somewhat different version for minipages.

```

1726 \newcommand*{\mpnormalfootgroup}[1]{%
1727   \vskip\skip\@nameuse{mp#1footins}
1728   \ifl@dpairing\ifparledgroup%
1729     \leavevmode\marks\parledgroup@{begin}%
1730     \marks\parledgroup@series{#1}%
1731     \marks\parledgroup@type{Xfootnote}%
1732     \fi\fi\normalcolor%
1733     \ifparledgroup%
1734       \ifl@dpairing%
1735       \else%
1736         \setXnoteswidthliketwocolumns@{#1}%
1737         \setXnotespositionliketwocolumns@{#1}%
1738         \print@Xfootnoterule{#1}%%
1739         \vskip\csuse{afterXrule@#1}%
1740         \fi%
1741       \else%
1742         \setXnoteswidthliketwocolumns@{#1}%
1743         \setXnotespositionliketwocolumns@{#1}%
1744         \print@Xfootnoterule{#1}%%
1745         \vskip\csuse{afterXrule@#1}%
1746         \fi%
1747       \setlength{\parindent}{0pt}
1748     {\csuse{Xnotefontsize@#1}\csuse{txtbeforeXnotes@#1}}
1749     \unvbox\csname mp#1footins\endcsname}}
1750

```

24.4 Standard footnote definitions

`\footnormal` We can now define all the parameters for the five series of footnotes; initially they use the ‘normal’ footnote formatting, which is set up by calling `\footnormal`. You

can switch to another type of formatting by using `\footparagraph`, `\foottwocol`, or `\footthreecol`.

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

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

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

```
\skip\Afootins=12pt plus5pt minus5pt
\count\Afootins=1000
\dimen\Afootins=0.8\vsiz
\let\vAfootnote=\normalvfootnote \let\Afootfmt=\normalfootfmt
\let\Afootstart=\normalfootstart \let\Afootgroup=\normalfootgroup
\let\Afootnoterule=\normalfootnoterule
```

Instead of repeating ourselves, we define a `\footnormal` macro that makes all these assignments for us, for any given series letter. This also makes it easy to change from any different system of formatting back to the `normal` setting.

`\ledfootinsdim` Have a constant value for the `\dimen\footins`

```
1751 \newcommand*\ledfootinsdim{0.8\vsiz} % kept for backward compatibility, should'nt be used anymore.
```

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

```
1752 \newtoggle{preXnotes@}
1753 \toggletrue{preXnotes@}
1754 \newcommand{\preXnotes@}{0pt}
1755 \newcommand*\preXnotes[1]{\renewcommand{\preXnotes@}{#1}}
```

The same, but for familiar footnotes.

`\preXnotes`

```
\preXnotes@ 1756 \newtoggle{prenotesX@}
1757 \toggletrue{prenotesX@}
1758 \newcommand{\prenotesX@}{0pt}
1759 \newcommand*\prenotesX[1]{\renewcommand{\prenotesX@}{#1}}
```

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

```
1760 \newcommand*\footnormal[1]{%
1761   \csgdef{series@display#1}{normal}
1762   \expandafter\let\csname #1footstart\endcsname=\normalfootstart
1763   \expandafter\let\csname v#1footnote\endcsname=\normalvfootnote
1764   \expandafter\let\csname #1footfmt\endcsname=\normalfootfmt
1765   \expandafter\let\csname #1footgroup\endcsname=\normalfootgroup
```

```

1766 \expandafter\let\csname #1footnoterule\endcsname=%
1767                                     \normalfootnoterule
1768 \count\csname #1footins\endcsname=1000
1769 \dimen\csname #1footins\endcsname=\csuse{maxhXnotes@#1}
1770 \skip\csname #1footins\endcsname=\csuse{beforeXnotes@#1}

```

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).

```

1771 \expandafter\let\csname mpv#1footnote\endcsname=\mpnormalvfootnote
1772 \expandafter\let\csname mp#1footgroup\endcsname=\mpnormalfootgroup
1773 \count\csname mp#1footins\endcsname=1000
1774 \dimen\csname mp#1footins\endcsname=\csuse{maxhXnotes@#1}
1775 \skip\csname mp#1footins\endcsname=\csuse{beforeXnotes@#1}
1776 }
1777

```

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 T_EX to make.

24.5 Paraphrased footnotes

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

`\footparagraph` The `\footparagraph` macro sets up everything for one series of the footnotes so that they'll be paraphrased; 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 paraphrased footnotes after having columnar ones, since they change this value (see below).

It is important to call `\footparagraph` only after `\hsize` has been set for the pages that use this series of notes; otherwise T_EX 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 paraphrased.

```

1778 \newcommand*{\footparagraph}[1]{%
1779   \csgdef{series@display#1}{paragraph}
1780   \expandafter\newcount\csname prevpage#1@num\endcsname
1781   \expandafter\let\csname #1footstart\endcsname=\parafootstart
1782   \expandafter\let\csname v#1footnote\endcsname=\para@vfootnote
1783   \expandafter\let\csname #1footfmt\endcsname=\parafootfmt
1784   \expandafter\let\csname #1footgroup\endcsname=\para@footgroup
1785   \count\csname #1footins\endcsname=1000

```

```

1786 \dimen\csname #1footins\endcsname=\csuse{maxhXnotes@#1}
1787 \skip\csname #1footins\endcsname=\csuse{beforeXnotes@#1}
1788 \para@footsetup{#1}

```

And the extra setup for minipages.

```

1789 \expandafter\let\csname mpv#1footnote\endcsname=\mppara@vfootnote
1790 \expandafter\let\csname mp#1footgroup\endcsname=\mppara@footgroup
1791 \count\csname mp#1footins\endcsname=1000
1792 \dimen\csname mp#1footins\endcsname=\csuse{maxhXnotes@#1}
1793 \skip\csname mp#1footins\endcsname=\csuse{beforeXnotes@#1}
1794 }

```

`\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.

```
1795 \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`.

```

1796 \newcommand*{\para@footsetup}[1]{\csuse{Xnotefontsize@#1}
1797 \setXnoteswidthliketwocolumns@{#1}%
1798 \dimen0=\baselineskip
1799 \multiply\dimen0 by 1024
1800 \divide \dimen0 by \columnwidth \multiply\dimen0 by \footfudgefiddle\relax
1801 \csxdef{#1footfudgefactor}{%
1802 \expandafter\strip@pt\dimen0 }}
1803

```

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`.

`\parafootstart` `\parafootstart` is the same as `\normalfootstart`, but we give it again to ensure that `\rightskip` and `\leftskip` are zeroed (this needs to be done before `\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.

```

1804 \newcommand*{\parafootstart}[1]{%
1805 \rightskip=0pt \leftskip=0pt \parindent=0pt
1806 \ifdimequal{0pt}{\preXnotes@}{}%
1807 }

```

```

1808     \iftoggle{preXnotes@}{%
1809         \togglefalse{preXnotes@}\skip\csname #1footins\endcsname=\csuse{preXnotes@}}%
1810         {}%
1811     }%
1812     \vskip\skip\csname #1footins\endcsname%
1813     \setXnoteswidthliketwocolumns@{#1}%
1814     \setXnotespositionliketwocolumns@{#1}%
1815     \print@Xfootnoterule{#1}%
1816     \vskip\csuse{afterXrule@#1}%
1817     \noindent\leavevmode}

```

`\para@vfootnote` `\para@vfootnote` is a version of the `\vfootnote` command that's used for paragraphed notes. It gets appended to the `\inserts@list` list by an outer-level footnote command like `\Afootnote`. The first argument is the note series letter; the second is the full text of the printed note itself, including line numbers, lemmata, and footnote text.

The initial model for this insertion is, of course, the `\insert\footins` definition in *The 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`s. If you later unbox these hboxes and stick them together, as the *TeXbook* macros do to make these footnotes, you lose the ability to hyphenate after an explicit hyphen. This can lead to overfull `\hboxes` when you would not expect to find them, and to the uninitiated it might be very hard to see why the problem had arisen.²³

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.²⁴ 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.²⁵ Michael's unboxing macro is called `\unvxh`: `unvbox`, extract the last line, and `unhbox` it.

²³Michael Downes, 'Line Breaking in `\unhboxed` Text', *TUGboat* **11** (1990), pp. 605–612.

²⁴See *The TeXbook*, p. 455 (editions after January 1990).

²⁵Wayne 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.²⁶ 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. 111 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.

```

1818 \newcommand*{\para@vfootnote}[2]{%
1819   \insert\csname #1footins\endcsname
1820   \bgroup
1821     \csuse{bhookXnote@#1}
1822     \csuse{Xnotefontsize@#1}
1823     \footplitskips
1824     \setbox0=\vbox{\hsize=\maxdimen
1825       \noindent\csname #1footfmt\endcsname#2[#1]}%
1826     \setbox0=\hbox{\unvxb0[#1]}%
1827     \dp0=0pt
1828     \ht0=\csname #1footfudgefactor\endcsname\wd0

```

Here we produce the contents of the footnote from box 0, and add a penalty of 0 between boxes in this insert.

```

1829   \if@RTL\noindent \leavevmode\fi\box0%
1830   \penalty0
1831 \egroup}
1832

```

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 `\makehboxoffboxes`. 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`).

²⁶'Line Breaking', p. 610.

`\mppara@vfootnote` This version is for minipages.

```

1833 \newcommand*{\mppara@vfootnote}[2]{%
1834   \global\setbox\@nameuse{mp#1footins}\vbox{%
1835     \unvbox\@nameuse{mp#1footins}%
1836     \csuse{bhookXnote@#1}
1837     \csuse{Xnotefontsize@#1}
1838     \footsplitskips
1839     \setbox0=\vbox{\hsize=\maxdimen
1840       \noindent\color@begingroup\csname #1footfmt\endcsname #2[#1]\color@endgroup}%
1841     \setbox0=\hbox{\unvxh0[#1]}%
1842     \dp0=\z@
1843     \ht0=\csname #1footfudgefactor\endcsname\wd0
1844     \box0
1845     \penalty0
1846   }}
1847
```

`\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 T_EX automatically attaches to the end of paragraphs. When T_EX finishes a paragraph, it throws away any remaining glue, and then tacks on the following items: a `\penalty` of 10000, a `\parfillskip` and a `\rightskip` (*The TeXbook*, pp.99–100). `\unvxh` cancels these unwanted paragraph-final items using `\unskip` and `\unpenalty`.

```

1848 \newcommand*{\unvxh}[2][2=Z]{% 2th is optional for retro-compatibility
1849   \setbox0=\vbox{\unvbox#1%
1850     \global\setbox1=\lastbox}%
1851   \unhbox1
1852   \unskip           % remove \rightskip,
1853   \unskip           % remove \parfillskip,
1854   \unpenalty        % remove \penalty of 10000,
1855   \hskip\csuse{afternote@#2}} % but add the glue to go between the notes
1856
```

`\parafootfmt` `\parafootfmt` is `\normalfootfmt` adapted to do the special stuff needed for paragraphed notes—leaving out the `\endgraf` at the end, sticking in special penalties and kern, and leaving out the `\footstrut`. The first argument is the line and page number information, the second is the lemma, the third is the text of the footnote, and the fourth is the series (optional, for backward compatibility).

```

1857 \newcommand*{\parafootfmt}[4][4=Z]{%
1858   \insertparafootsep{#4}%
1859   \ledsetnormalparstuff%
1860   \printlinefootnote{#1}{#4}%
1861   {\nottoggle{Xlemmadisablefontselection@#4}{\select@lemmafont#1|#2}{#2}}%
1862   \iftoggle{nosep@}{\hskip\csuse{inplaceoflemmaseparator@#4}}{\ifcsemtyp{lemmaseparator@#4}}%
1863   {\hskip\csuse{inplaceoflemmaseparator@#4}}%
1864   {\nobreak\hskip\csuse{beforelemmaseparator@#4}\csuse{lemmaseparator@#4}\hskip\csuse{afterlemmaseparator@#4}}%
1865   }%
1866   #3\penalty-10 }

```

Note that in the above definition, the penalty of -10 encourages a line break between notes, so that notes have a slight tendency to begin on new lines. The `\insertparafootsep` command is used to insert the `\parafootsep@series` between each note in the *same* page.

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

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

```

1867 \newcommand*{\para@footgroup}[1]{%
1868   \unvbox\csname #1footins\endcsname
1869   \ifcsstring{Xragged@#1}{L}{\RaggedLeft}{}%
1870   \ifcsstring{Xragged@#1}{R}{\RaggedRight}{}%
1871   \makehboxofhboxes
1872   \setbox0=\hbox{{\csuse{Xnotefontsize@#1}\csuse{txtbeforeXnotes@#1}}\unhbox0 \removehboxes}%
1873   \csuse{Xnotefontsize@#1}
1874   \noindent\unhbox0\par%
1875   \global\hsize=\old@hsize%
1876   }%
1877

```

`\mppara@footgroup` The minipage version.

```

1878 \newcommand*{\mppara@footgroup}[1]{%
1879   \setXnoteswidthliketwocolumns@{#1}%
1880   \vskip\skip\@nameuse{mp#1footins}
1881   \ifl@dpairing\ifparledgroup%
1882     \leavevmode\marks\parledgroup@{begin}%
1883     \marks\parledgroup@series{#1}%
1884     \marks\parledgroup@type{Xfootnote}%
1885   \fi\fi\normalcolor
1886   \ifparledgroup%
1887     \ifl@dpairing%
1888     \else%
1889       \setXnoteswidthliketwocolumns@{#1}%
1890       \setXnotespositionliketwocolumns@{#1}%
1891       \print@Xfootnoterule{#1}%
1892       \vskip\csuse{afterXrule@#1}%
1893     \fi%
1894   \else%
1895     \setXnoteswidthliketwocolumns@{#1}%
1896     \setXnotespositionliketwocolumns@{#1}%
1897     \print@Xfootnoterule{#1}%
1898     \vskip\csuse{afterXrule@#1}%
1899   \fi%
1900   \unvbox\csname mp#1footins\endcsname
1901   \ifcsstring{Xragged@#1}{L}{\RaggedLeft}{}%
1902   \ifcsstring{Xragged@#1}{R}{\RaggedRight}{}%
1903   \makehboxofhboxes

```

```

1904 \setbox0=\hbox{{\csuse{Xnotefontsize@#1}\csuse{txtbeforeXnotes@#1}}\unhbox0 \removehbox
1905 \csuse{Xnotefontsize@#1}
1906 \noindent\unhbox0\par}}
1907

```

`\makehboxofhboxes`

```

\removehboxes 1908 \newcommand*\makehboxofhboxes{\setbox0=\hbox{}%
1909 \loop
1910 \unpenalty
1911 \setbox2=\lastbox
1912 \ifhbox2
1913 \setbox0=\hbox{\box2\unhbox0}%
1914 \repeat}
1915
1916 \newcommand*\removehboxes{\setbox0=\lastbox
1917 \ifhbox0{\removehboxes}\unhbox0 \fi}
1918

```

24.6 Insertion of the footnotes separator

The command `\insertparafootsep{<series>}` must be called at the beginning of `\parafootftm` (and like commands).

```

\prevpage@num
\insertparafootsep 1919 \newcommand{\insertparafootsep}[1]{%
1920 \ifnumequal{\csuse{prevpage#1@num}}{\page@num}%
1921 {\ifcsdef{prevline#1}% Be sur \prevline#1 exists.
1922 {\ifnumequal{\csuse{prevline#1}}{\line@num}%
1923 {\ifcsemt{symp\line@num}{\csuse{parafootsep@#1}}{}}}%
1924 {\csuse{parafootsep@#1}}}%
1925 }%
1926 {\csuse{parafootsep@#1}}}%
1927 }%
1928 {}%
1929 \global\csname prevpage#1@num\endcsname=\page@num%
1930 }

```

24.7 Columnar footnotes

`\rigidbalance` 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`.


```

1931 \newcount\@k \newdimen\@h
1932 \newcommand*\rigidbalance}[3]{\setbox0=\box#1 \@k=#2 \@h=#3
1933 \@@line{\splittopskip=\@h \vbadness=\@M \hfilneg
1934 \valign{##\vfil\cr\dosplits}}}}
1935
1936 \newcommand*\dosplits{\ifnum\@k>0 \noalign{\hfil}\splitoff
1937 \global\advance\@k-1\cr\dosplits\fi}
1938
1939 \newcommand*\splitoff{\dimen0=\ht0
1940 \divide\dimen0 by\@k \advance\dimen0 by\@h
1941 \setbox2 \vsplit0 to \dimen0
1942 \unvbox2 }
1943

```

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.

```

1944 \newcommand*\footthreecol}[1]{%
1945 \csgdef{series@display#1}{threecol}
1946 \expandafter\let\csname v#1footnote\endcsname=\threecolvfootnote
1947 \expandafter\let\csname #1footfmt\endcsname=\threecolfootfmt
1948 \expandafter\let\csname #1footgroup\endcsname=\threecolfootgroup
1949 \threecolfootsetup{#1}

```

The additional setup for minipages.

```

1950 \expandafter\let\csname mpv#1footnote\endcsname=\mpnormalvfootnote
1951 \expandafter\let\csname mp#1footgroup\endcsname=\mpthreecolfootgroup
1952 \mpthreecolfootsetup{#1}
1953 }
1954

```

The `\footstart` and `\footnoterule` macros for these notes assume the normal values (p.111 above).

`\threecolfootsetup` The `\threecolfootsetup` macro calculates and sets some numbers for three-column footnotes.

We set the `\count` of the foot insert to 333. Each footnote can be thought of as contributing only one third of its height to the page, since the footnote insertion has been made as a long narrow column, which then gets trisected by the `\rigidbalance` routine (inside `\threecolfootgroup`). These new, shorter columns are saved in a box, and then that box is *put back* into the footnote insert, replacing the original collection of the footnotes. This new box is, therefore, only about a third of the height of the original one.

The `\dimen` value for this note series has to change in the inverse way: it needs to be three times the actual limit on the amount of space these notes are allowed to fill on the page, because when `TEX` is accumulating material for the page and checking that limit, it doesn't apply the `\count` scaling.

```

1955 \newcommand*{\threecolfootsetup}[1]{%
1956   \count\csname #1footins\endcsname 333
1957   \multiply\dimen\csname #1footins\endcsname \thr@@}

```

`\mpthreecolfootsetup` The setup for minipages.

```

1958 \newcommand*{\mpthreecolfootsetup}[1]{%
1959   \count\csname mp#1footins\endcsname 333
1960   \multiply\dimen\csname mp#1footins\endcsname \thr@@}
1961

```

`\threecolvf footnote` `\threecolvf footnote` is the `\vf footnote` 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).

```

1962 \notbool{parapparatus@}{\newcommand*}{\newcommand}{\threecolvf footnote}[2]{%
1963   \insert\csname #1footins\endcsname\bgroup
1964   \csuse{Xnotefontsize@#1}
1965   \footsplittopskip
1966   \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.

```

1967 \notbool{parapparatus@}{\newcommandx*}{\newcommandx}{\threecolfootfmt}[4][4=Z]{%
1968   \normal@pars
1969   \hsize \csuse{hsizethreecol@#4}
1970   \parindent=0pt
1971   \tolerance=5000
1972   \raggedright
1973   \hangindent=\csuse{Xhangindent@#4}
1974   \leavevmode
1975   \strut{\printlinefootnote{#1}{#4}}%
1976   {\nottoggle{Xlemmadisablefontselection@#4}{\select@lemmafont#1|#2}{#2}}%
1977   \iftoggle{nosep@}{\hskip\csuse{inplaceoflemmaseparator@#4}}{\ifcsempy{lemmaseparator@#4}
1978     {\hskip\csuse{inplaceoflemmaseparator@#4}}}%
1979     {\nobreak\hskip\csuse{beforelemmaseparator@#4}\csuse{lemmaseparator@#4}\hskip\csuse{a
1980     }}%
1981   #3\strut\par\allowbreak}

```

`\threecolfootgroup` And here is the `footgroup` macro that's called within the output routine to re-group the notes into three columns. Once again, the call to `\notefontsetup` is

there to ensure that it is the right `\splittopskip`—the one used in footnotes—which is used to provide the third argument for `\rigidbalance`. This third argument (`\@h`) is the `topskip` for the box containing the text of the footnotes, and does the job of making sure the top lines of the columns line up horizontally. In *The TeXbook*, p.398, Donald Knuth suggests retrieving the output of `\rigidbalance`, putting it back into the insertion box, and then printing the box. Here, we just print the `\line` which comes out of `\rigidbalance` directly, without any re-boxing.

```
1982 \newcommand*{\threecolfootgroup}[1]{\notefontsetup
1983   {\csuse{Xnotefontsize@#1}\noindent\csuse{txtbeforeXnotes@#1}}\par
1984   \splittopskip=\ht\strutbox
1985   \expandafter
1986   \rigidbalance\csname #1footins\endcsname \thr@@ \splittopskip}}
```

`\mpthreecolfootgroup` The setup for minipages.

```
1987 \newcommand*{\mpthreecolfootgroup}[1]{%
1988   \vskip\skip\@nameuse{mp#1footins}
1989   \ifl@dpairing\ifparledgroup%
1990     \leavevmode\marks\parledgroup@{begin}%
1991     \marks\parledgroup@series{#1}%
1992     \marks\parledgroup@type{Xfootnote}%
1993   \fi\fi\normalcolor
1994   \ifparledgroup%
1995     \ifl@dpairing%
1996     \else%
1997       \setXnoteswidthliketwocolumns@{#1}%
1998       \setXnotespositionliketwocolumns@{#1}%
1999       \print@Xfootnoterule{#1}%
2000       \vskip\csuse{afterXrule@#1}%
2001     \fi%
2002   \else%
2003     \setXnoteswidthliketwocolumns@{#1}%
2004     \setXnotespositionliketwocolumns@{#1}%
2005     \print@Xfootnoterule{#1}%
2006     \vskip\csuse{afterXrule@#1}%
2007   \fi%
2008   {\csuse{Xnotefontsize@#1}\noindent\csuse{txtbeforeXnotes@#1}}\par
2009   \splittopskip=\ht\strutbox
2010   \expandafter
2011   \rigidbalance\csname mp#1footins\endcsname \thr@@ \splittopskip}}
2012
```

Two columns

`\foottwocol` You say `\foottwocol{A}` to have the A series of the footnotes typeset in two columns. It is important to call this only after `\hsize` has been set for the document.

```
2013 \newcommand*{\foottwocol}[1]{%
```

```

2014 \csgdef{series@display#1}{twocol}
2015 \expandafter\let\csname v#1footnote\endcsname=\twocolvfootnote
2016 \expandafter\let\csname #1footfmt\endcsname=\twocolfootfmt
2017 \expandafter\let\csname #1footgroup\endcsname=\twocolfootgroup
2018 \twocolfootsetup{#1}

```

The additional setup for minipages.

```

2019 \expandafter\let\csname mpv#1footnote\endcsname=\mpnormalvfootnote
2020 \expandafter\let\csname mp#1footgroup\endcsname=\mptwocolfootgroup
2021 \mptwocolfootsetup{#1}
2022 }
2023

```

`\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`

```

2024 \newcommand*{\twocolfootsetup}[1]{%
2025   \count\csname #1footins\endcsname 500
2026   \multiply\dimen\csname #1footins\endcsname \tw@}

2027 \notbool{parapparatus@}{\newcommand*}{\newcommand}{\twocolvfootnote}[2]{\insert\csname #1
2028   \csuse{Xnotefontsize@#1}
2029   \footsplitskips
2030   \csname #1footfmt\endcsname #2[#1]\egroup}

2031 \notbool{parapparatus@}{\newcommandx*}{\newcommandx}{\twocolfootfmt}[4][4=Z]{% 4th arg is
2032   \normal@pars
2033   \hsize \csuse{hsizetwocol@#4}
2034   \parindent=0pt
2035   \tolerance=5000
2036   \raggedright
2037   \hangindent=\csuse{Xhangindent@#4}
2038   \leavevmode
2039   \strut{\printlinefootnote{#1}{#4}}}%
2040   {\nottoggle{Xlemmadisablefontselection@#4}{\select@lemmafont#1|#2}{#2}}}%
2041   \iftoggle{nosep@}{\hskip\csuse{inplaceoflemmaseparator@#4}}{\ifcsemt{lemmaseparator@#4}
2042     {\hskip\csuse{inplaceoflemmaseparator@#4}}}%
2043     {\nobreak\hskip\csuse{beforelemmaseparator@#4}\csuse{lemmaseparator@#4}\hskip\csuse{a
2044   }}%
2045   #3\strut\par\allowbreak}

2046 \newcommand*{\twocolfootgroup}[1]{\csuse{Xnotefontsize@#1}
2047   {\csuse{Xnotefontsize@#1}\noindent\csuse{txtbeforeXnotes@#1}}\par
2048   \splittopskip=\ht\strutbox
2049   \expandafter
2050   \rigidbalance\csname #1footins\endcsname \tw@ \splittopskip}}
2051

```

`\mptwocolfootsetup` The versions for minipages.

```

\mptwocolfootgroup 2052 \newcommand*{\mptwocolfootsetup}[1]{%
2053   \count\csname mp#1footins\endcsname 500
2054   \multiply\dimen\csname mp#1footins\endcsname \tw@}

```

```

2055 \newcommand*{\mptwocolfootgroup}[1]{%
2056   \vskip\skip\@nameuse{mp#1footins}
2057   \ifl@dpairing\ifparledgroup%
2058     \leavevmode\marks\parledgroup@{begin}%
2059     \marks\parledgroup@series{#1}%
2060     \marks\parledgroup@type{Xfootnote}%
2061   \fi\fi\normalcolor
2062   \ifparledgroup%
2063     \ifl@dpairing%
2064     \else%
2065       \setXnoteswidthliketwocolumns@{#1}%
2066       \setXnotespositionliketwocolumns@{#1}%
2067       \print@Xfootnoterule{#1}%%
2068       \vskip\csuse{afterXrule@#1}%
2069     \fi%
2070   \else%
2071     \setXnoteswidthliketwocolumns@{#1}%
2072     \setXnotespositionliketwocolumns@{#1}%
2073     \print@Xfootnoterule{#1}%%
2074     \vskip\csuse{afterXrule@#1}%
2075   \fi%
2076   {\csuse{Xnotefontsize@#1}\noindent\csuse{txtbeforeXnotes@#1}}\par
2077   \splittopskip=\ht\strutbox
2078   \expandafter
2079   \rigidbalance\csname mp#1footins\endcsname \tw@ \splittopskip}}
2080

```

25 Familiar footnotes

25.1 Generality

The original EDMAC provided users with five series of critical footnotes (`\Afootnote` `\Bfootnote` `\Cfootnote` `\Dfootnote` `\Efootnote`), and L^AT_EX provides a single numbered footnote. The `eledmac` package uses the EDMAC mechanism to provide five series of numbered footnotes.

First, though, the `footmisc` package has an option whereby two or more consecutive `\footnotes` have their marks separated by commas. This seems such a useful ability that it is provided automatically by `eledmac`.

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

```

2081 \providecommand*{\multiplefootnotemarker}{3sp}
2082 \providecommand*{\multfootsep}{\textsuperscript{\normalfont,}}
2083

```

`\m@mmf@prepare` A pair of self-cancelling kerns. This may have been defined in the `memoir` class.

```

2084 \providecommand*{\m@mmf@prepare}{%
2085   \kern-\multiplefootnotemarker

```

```
2086 \kern\multiplefootnotemarker\relax}
```

\m@mmf@check This may have been defined in the memoir class. If it recognises the last kern as \multiplefootnotemarker it typesets \multfootsep.

```
2087 \providecommand*\m@mmf@check}{%
2088 \ifdim\lastkern=\multiplefootnotemarker\relax
2089 \edef\x@sf{\the\spacefactor}%
2090 \unkern
2091 \multfootsep
2092 \spacefactor\x@sf\relax
2093 \fi}
2094
```

We have to modify \@footnotetext and \@footnotemark. However, if memoir is used the modifications have already been made.

```
2095 \ifclassloaded{memoir}{\%
```

\@footnotetext Add \m@mmf@prepare at the end of \@footnotetext.

```
2096 \apptocmd{\@footnotetext}{\m@mmf@prepare}{\%}
```

\@footnotemark Modify \@footnotemark to cater for adjacent \footnotes.

```
2097 \renewcommand*\@footnotemark}{%
2098 \leavevmode
2099 \ifhmode
2100 \edef\x@sf{\the\spacefactor}%
2101 \m@mmf@check
2102 \nobreak
2103 \fi
2104 \@makefnmark
2105 \m@mmf@prepare
2106 \ifhmode\spacefactor\x@sf\fi
2107 \relax}
```

Finished the modifications for the non-memoir case.

```
2108 }
```

```
2109
```

\l@doldold@footnotetext In order to enable the regular \footnotes in numbered text we have to play around with its \@footnotetext, using different forms for when in numbered or regular text.

```
2110 \pretocmd{\@footnotetext}{%
2111 \ifnumberedpar@
2112 \edtext{\l@dbfnote{#1}}}%
2113 \else
2114 }{}{}
2115 \apptocmd{\@footnotetext}{\fi}{\%}
```

```

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

2116 \newcommand{\l@dbfnote}[1]{%
2117   \ifnumberedpar@
2118   \gdef\@tag{#1\relax}%
2119   \xright@appenditem{\noexpand\vl@dbfnote{\@expandonce\@tag}}{\@thefnmark}}%
2120   \to\inserts@list
2121   \global\advance\insert@count \@ne
2122   \fi\ignorespaces}
2123 \newcommand{\vl@dbfnote}[2]{%
2124   \def\@thefnmark{#2}%
2125   \@footnotetext{#1}%
2126   }%

```

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 2127 \newcommand*\prebodyfootmark{%
2128   \leavevmode
2129   \ifhmode
2130     \edef\@x@sf{\the\spacefactor}%
2131     \m@mmf@check
2132     \nobreak
2133     \fi}
2134 \newcommand*\postbodyfootmark{%
2135   \m@mmf@prepare
2136   \ifhmode\spacefactor\@x@sf\fi\relax}
2137

\normal@footnotemarkX \normal@footnotemarkX{<series>} sets up the typesetting of the marker at the
point where the footnote is called for.

2138 \newcommand*\normal@footnotemarkX[1]{%
2139   \prebodyfootmark
2140   \@nameuse{bodyfootmark#1}%
2141   \postbodyfootmark}
2142

\normalbodyfootmarkX The \normalbodyfootmarkX{<series>} really typesets the in-text marker. The
style is the normal superscript.

2143 \newcommand*\normalbodyfootmarkX[1]{%
2144   \hbox{\textsuperscript{\normalfont\@nameuse{\@thefnmark#1}}}}

\normalvfootnoteX \normalvfootnoteX{<series>}{<text>} does the \insert for the <series> and calls
the series’ \footfmt... to format the <text>.

```

```

2145 \newcommand*{\normalvfootnoteX}[2]{%
2146   \insert\@nameuse{footins#1}\bgroup
2147     \csuse{bhooknoteX@#1}
2148     \csuse{notefontsizeX@#1}
2149     \footsplitskips
2150     \ifl@dpairing\ifl@dpaging\else%
2151       \setnotesXwidthliketwocolumns@{#1}%
2152     \fi\fi%
2153     \setnotesXpositionliketwocolumns@{#1}%
2154     \spaceskip=\z@skip \xspaceskip=\z@skip
2155     \csuse{\csuse{footnote@dir}}\if@RTL\else\noindent\leavevmode\fi\@nameuse{footfmt#1}-#1%
2156

```

`\mpnormalvfootnoteX` The minipage version.

```

2157 \newcommand*{\mpnormalvfootnoteX}[2]{%
2158   \global\setbox\@nameuse{mpfootins#1}\vbox{%
2159     \unvbox\@nameuse{mpfootins#1}
2160     \csuse{bhooknoteX@#1}
2161     \csuse{notefontsizeX@#1}
2162     \hsize\columnwidth
2163     \@parboxrestore
2164     \color@begingroup
2165     \@nameuse{footfmt#1}-#1-#2\color@endgroup}}
2166

```

`\normalfootfmtX` `\normalfootfmtX{<series>}{<text>}` typesets the footnote text, prepended by the marker.

```

2167 \newcommand*{\normalfootfmtX}[2]{%
2168   \protected@edef\@currentlabel{%
2169     \@nameuse{@thefnmark#1}}%
2170   }%
2171   \ledsetnormalparstuff
2172   \hangindent=\csuse{hangindentX@#1}%
2173   {\csuse{notenumfontX@#1}\@nameuse{footfootmark#1}}\strut%\enspace
2174   #2\strut\par}}
2175

```

`\normalfootfootmarkX` `\normalfootfootmarkX{<series>}` is called by `\normalfootfmtX` to typeset the footnote marker in the footer before the footnote text.

```

2176 \newcommand*{\normalfootfootmarkX}[1]{%
2177   \textsuperscript{\@nameuse{@thefnmark#1}}}
2178

```

`\normalfootstartX` `\normalfootstartX{<series>}` is the `<series>` footnote starting macro used in the output routine.

```

2179 \newcommand*{\normalfootstartX}[1]{%
2180   \ifdimequal{0pt}{\prenotesX@}{}%
2181   {%
2182     \iftoggle{prenotesX@}{%

```



```

2183         \togglefalse{prenotesX@} \skip\csname footins#1\endcsname=\csuse{prenotesX@}}%
2184         {}%
2185     }%
2186 \vskip\skip\csname footins#1\endcsname%
2187 \leftskip=\z@
2188 \rightskip=\z@
2189 \ifl@dpairing\else%
2190     \hsize=\old@hsize%
2191 \fi%
2192 \setnotesXwidthliketwocolumns@{#1}%
2193 \setnotesXpositionliketwocolumns@{#1}%
2194 \print@footnoterule{#1}%
2195 \vskip\csuse{afterruleX@#1}%
2196

```

`\normalfootnoteruleX` The rule drawn before the footnote series group.

```

2197 \let\normalfootnoteruleX=\footnoterule
2198

```

`\normalfootgroupX` `\normalfootgroupX{<series>}` sends the contents of the `<series>` insert box to the output page without alteration.

```

2199 \newcommand*\normalfootgroupX[1]{%
2200     \unvbox\@nameuse{footins#1}%
2201     \hsize=\old@hsize%
2202 }%
2203

```

`\mpnormalfootgroupX` The minipage version.

```

2204 \newcommand*\mpnormalfootgroupX[1]{%
2205     \vskip\skip\@nameuse{mpfootins#1}
2206     \ifl@dpairing\ifparledgroup%
2207         \leavevmode\marks\parledgroup@{begin}%
2208         \marks\parledgroup@series{#1}%
2209         \marks\parledgroup@type{footnoteX}%
2210     \fi\fi\normalcolor
2211     \ifparledgroup%
2212         \ifl@dpairing%
2213         \else%
2214             \setnotesXwidthliketwocolumns@{#1}%
2215             \setnotesXpositionliketwocolumns@{#1}%
2216             \print@footnoterule{#1}%
2217             \vskip\csuse{afterruleX@#1}%
2218         \fi%
2219     \else%
2220         \setnotesXwidthliketwocolumns@{#1}%
2221         \setnotesXpositionliketwocolumns@{#1}%
2222         \print@footnoterule{#1}%
2223         \vskip\csuse{afterruleX@#1}%
2224     \fi%

```

```

2225 \unvbox\@nameuse{mpfootins#1}}
2226

```

`\normalbfnoteX`

```

2227 \newcommand{\normalbfnoteX}[2]{%
2228   \ifnumberedpar@
2229     \protected@xdef\thisfootnote{\csuse{thefootnote#1}}%
2230     \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\expandonce\thisfootnote}}%
2231     \to\inserts@list
2232     \global\advance\insert@count \@ne
2233   \fi\ignorespaces}
2234

```

`\vbfnoteX`

```

2235 \newcommand{\vbfnoteX}[3]{%
2236   \@namedef{@thefnmark#1}{#3}%
2237   \@nameuse{regvfootnote#1}{#1}{#2}}
2238

```

`\vnumfootnoteX`

```

2239 \newcommand{\vnumfootnoteX}[2]{%
2240   \ifnumberedpar@
2241     \edtext{}{\normalbfnoteX{#1}{#2}}%
2242   \else
2243     \@nameuse{regvfootnote#1}{#1}{#2}%
2244   \fi}
2245

```

`\footnormalX` `\footnormalX{<series>}` initialises the settings for the `<series>` footnotes. This should always be called for each series.

```

2246 \newcommand*{\footnormalX}[1]{%
2247   \csgdef{series@displayX#1}{normalX}
2248   \expandafter\let\csname footstart#1\endcsname=\normalfootstartX
2249   \@namedef{@footnotemark#1}{\normal@footnotemarkX{#1}}
2250   \@namedef{bodyfootmark#1}{\normalbodyfootmarkX{#1}}
2251   \expandafter\let\csname regvfootnote#1\endcsname=\normalvfootnoteX
2252   \expandafter\let\csname vfootnote#1\endcsname=\vnumfootnoteX
2253   \expandafter\let\csname footfmt#1\endcsname=\normalfootfmtX
2254   \@namedef{footfootmark#1}{\normalfootfootmarkX{#1}}
2255   \expandafter\let\csname footgroup#1\endcsname=\normalfootgroupX
2256   \expandafter\let\csname footnoterule#1\endcsname=\normalfootnoteruleX
2257   \count\csname footins#1\endcsname=1000
2258   \dimen\csname footins#1\endcsname=\csuse{maxhnotesX@#1}
2259   \skip\csname footins#1\endcsname=\csuse{beforenotesX@#1}

```

Additions for minipages.

```

2260   \expandafter\let\csname mpvfootnote#1\endcsname=\mpnormalvfootnoteX
2261   \expandafter\let\csname mpfootgroup#1\endcsname=\mpnormalfootgroupX
2262   \count\csname mpfootins#1\endcsname=1000
2263   \dimen\csname mpfootins#1\endcsname=\csuse{maxhnotesX@#1}

```

```

2264 \skip\csname mpfootins#1\endcsname=\csuse{beforenotesX@#1}
2265 }
2266

```

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>}
2267 \newcommand*\foottwocolX[1]{%
2268 \csgdef{series@displayX#1}{twocol}
2269 \expandafter\let\csname regvfootnote#1\endcsname=\twocolvfootnoteX
2270 \expandafter\let\csname footfmt#1\endcsname=\twocolfootfmtX
2271 \expandafter\let\csname footgroup#1\endcsname=\twocolfootgroupX
2272 \twocolfootsetupX{#1}
2273 \expandafter\let\csname mpvfootnote#1\endcsname=\mpnormalvfootnoteX
2274 \expandafter\let\csname mpfootgroup#1\endcsname=\mptwocolfootgroupX
2275 \mptwocolfootsetupX{#1}}
2276

\twocolfootsetupX \twocolfootsetupX{<series>}
\mptwocolfootsetupX 2277 \newcommand*\twocolfootsetupX[1]{%
2278 \count\csname footins#1\endcsname 500
2279 \multiply\dimen\csname footins#1\endcsname by \tw@}
2280 \newcommand*\mptwocolfootsetupX[1]{%
2281 \count\csname mpfootins#1\endcsname 500
2282 \multiply\dimen\csname mpfootins#1\endcsname by \tw@}
2283

\twocolvfootnoteX \twocolvfootnoteX{<series>}
2284 \newcommand*\twocolvfootnoteX[2]{%
2285 \insert\csname footins#1\endcsname\bgroup
2286 \csuse{notefontsizeX@#1}
2287 \footssplitsskip
2288 \spaceskip=\z@skip \xspaceskip=\z@skip
2289 \@nameuse{footfmt#1}{#1}{#2}\egroup}
2290

\twocolfootfmtX \twocolfootfmtX{<series>}
2291 \newcommand*\twocolfootfmtX[2]{%
2292 \protected@edef\@currentlabel{%
2293 \@nameuse{@thefnmark#1}%
2294 }%
2295 \normal@pars
2296 \hangindent=\csuse{hangindentX@#1}%
2297 \hsize \csuse{hsizetwocolX@#1}
2298 \parindent=\z@
2299 %%% \parfillskip=0pt \@plus 1fil

```

```

2300 \tolerance=5000\relax
2301 \raggedright
2302 \leavevmode
2303 {\csuse{notenumfontX@#1}\@nameuse{footfootmark#1}\strut%\enspace
2304 #2\strut\par}\allowbreak}
2305
\twocolfootgroupX \twocolfootgroupX{<series>}
\mptwocolfootgroupX 2306 \newcommand*{\twocolfootgroupX}[1]{\csuse{notefontsizeX@#1}
2307 \splittopskip=\ht\strutbox
2308 \expandafter
2309 \rigidbalance\csname footins#1\endcsname \tw@ \splittopskip}}
2310 \newcommand*{\mptwocolfootgroupX}[1]{\{
2311 \vskip\skip\@nameuse{mpfootins#1}
2312 \ifl@dpairing\ifparledgroup%
2313 \leavevmode\marks\parledgroup@{begin}%
2314 \marks\parledgroup@series{#1}%
2315 \marks\parledgroup@type{footnoteX}%
2316 \fi\fi\normalcolor
2317 \ifparledgroup%
2318 \ifl@dpairing%
2319 \else%
2320 \setnotesXwidthliketwocolumns@{#1}%
2321 \setnotesXpositionliketwocolumns@{#1}%
2322 \print@footnoteXrule{#1}%
2323 \vskip\csuse{afterruleX@#1}%
2324 \fi%
2325 \else%
2326 \setnotesXwidthliketwocolumns@{#1}%
2327 \setnotesXpositionliketwocolumns@{#1}%
2328 \print@footnoteXrule{#1}%
2329 \vskip\csuse{afterruleX@#1}%
2330 \fi%
2331 \splittopskip=\ht\strutbox
2332 \expandafter
2333 \rigidbalance\csname mpfootins#1\endcsname \tw@ \splittopskip}}
2334

```

25.4 Three columns footnotes

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

```

\footthreecolX \footthreecolX{<series>}
2335 \newcommand*{\footthreecolX}[1]{%
2336 \csgdef{series@displayX#1}{threecol}
2337 \expandafter\let\csname regvfootnote#1\endcsname=\threecolvfootnoteX
2338 \expandafter\let\csname footfmt#1\endcsname=\threecolfootfmtX
2339 \expandafter\let\csname footgroup#1\endcsname=\threecolfootgroupX

```

```

2340 \threecolfootsetupX{#1}
2341 \expandafter\let\csname mpvfootnote#1\endcsname=\mpnormalvfootnoteX
2342 \expandafter\let\csname mpfootgroup#1\endcsname=\mpthreecolfootgroupX
2343 \mpthreecolfootsetupX{#1}}
2344

```

```

\threecolfootsetupX \threecolfootsetupX{<series>}
\mpthreecolfootsetupX 2345 \newcommand*{\threecolfootsetupX}[1]{%
2346 \count\csname footins#1\endcsname 333
2347 \multiply\dimen\csname footins#1\endcsname by \thr@@}
2348 \newcommand*{\mpthreecolfootsetupX}[1]{%
2349 \count\csname mpfootins#1\endcsname 333
2350 \multiply\dimen\csname mpfootins#1\endcsname by \thr@@}
2351

```

```

\threecolvfootnoteX \threecolvfootnoteX{<series>}{<text>}
2352 \newcommand*{\threecolvfootnoteX}[2]{%
2353 \insert\csname footins#1\endcsname\bgroup
2354 \csuse{notefontsizeX@#1}
2355 \footsplitskips
2356 \@nameuse{footfmt#1}{#1}{#2}\egroup}
2357

```

```

\threecolfootfmtX \threecolfootfmtX{<series>}
2358 \newcommand*{\threecolfootfmtX}[2]{%
2359 \protected@edef\@currentlabel{%
2360 \@nameuse{@thefnmark#1}%
2361 }%
2362 \hangindent=\csuse{hangindentX@#1}%
2363 \normal@pars
2364 \hsize \csuse{hsizethreecolX@#1}
2365 \parindent=\z@
2366 %% \parfillskip=0pt \@plus 1fil
2367 \tolerance=5000\relax
2368 \raggedright
2369 \leavevmode
2370 {\csuse{notenumfontX@#1}\@nameuse{footfootmark#1}\strut%\enspace
2371 #2\strut\par}\allowbreak}
2372

```

```

\threecolfootgroupX \threecolfootgroupX{<series>}
\mpthreecolfootgroupX 2373 \newcommand*{\threecolfootgroupX}[1]{\csuse{notefontsizeX@#1}
2374 \splittopskip=\ht\strutbox
2375 \expandafter
2376 \rigidbalance\csname footins#1\endcsname \thr@@ \splittopskip}}
2377 \newcommand*{\mpthreecolfootgroupX}[1]{%
2378 \vskip\skip\@nameuse{mpfootins#1}
2379 \ifl@dpairing\ifparledgroup
2380 \leavevmode\marks\parledgroup@{begin}%

```

```

2381 \marks\parledgroup@series{#1}%
2382 \marks\parledgroup@type{footnoteX}%
2383 \fi\fi\normalcolor
2384 \ifparledgroup%
2385 \ifl@dpairing%
2386 \else%
2387 \setnotesXwidthliketwocolumns@{#1}%
2388 \setnotesXpositionliketwocolumns@{#1}%
2389 \print@footnoteXrule{#1}%
2390 \vskip\csuse{afterruleX@#1}%
2391 \fi%
2392 \else%
2393 \setnotesXwidthliketwocolumns@{#1}%
2394 \setnotesXpositionliketwocolumns@{#1}%
2395 \print@footnoteXrule{#1}%
2396 \vskip\csuse{afterruleX@#1}%
2397 \fi%
2398 \splittopskip=\ht\strutbox
2399 \expandafter
2400 \rigidbalance\csname mpfootins#1\endcsname \thr@@ \splittopskip}}
2401

```

25.5 Paragraphed footnotes

The following macros set footnotes as one paragraph.

```

\footparagraphX \footparagraphX{<series>}
2402 \newcommand*{\footparagraphX}[1]{%
2403 \csgdef{series@displayX#1}{paragraphX}%
2404 \expandafter\newcount\csname prevpage#1@num\endcsname
2405 \expandafter\let\csname footstart#1\endcsname=\parafootstartX
2406 \expandafter\let\csname regvfootnote#1\endcsname=\para@vfootnoteX
2407 \expandafter\let\csname footfmt#1\endcsname=\parafootfmtX
2408 \expandafter\let\csname footgroup#1\endcsname=\para@footgroupX
2409 \expandafter\let\csname footnoterule#1\endcsname=\normalfootnoteruleX
2410 \count\csname footins#1\endcsname=1000
2411 \dimen\csname footins#1\endcsname=\csuse{maxhnotesX@#1}
2412 \skip\csname footins#1\endcsname=\csuse{beforenotesX@#1}
2413 \expandafter\let\csname mpvfootnote#1\endcsname=\mppara@vfootnoteX
2414 \expandafter\let\csname mpfootgroup#1\endcsname=\mppara@footgroupX
2415 \count\csname mpfootins#1\endcsname=1000
2416 \dimen\csname mpfootins#1\endcsname=\csuse{maxhnotesX@#1}
2417 \skip\csname mpfootins#1\endcsname=\csuse{beforenotesX@#1}
2418 \para@footsetupX{#1}}
2419
\para@footsetupX \para@footsetupX{<series>}
2420 \newcommand*{\para@footsetupX}[1]{\csuse{notefontsizeX@#1}
2421 \setnotesXwidthliketwocolumns@{#1}%

```

```

2422 \dimen0=\baselineskip
2423 \multiply\dimen0 by 1024
2424 \divide\dimen0 by \columnwidth \multiply\dimen0 by \footfudgefiddle\relax%
2425 \expandafter
2426 \xdef\csname footfudgefactor#1\endcsname{%
2427   \expandafter\strip@pt\dimen0 }%
2428

```

```
\parafootstartX \parafootstartX{<series>}
```

```

2429 \newcommand*{\parafootstartX}[1]{%
2430   \ifdimequal{0pt}{\prenotesX@}{}%
2431   {%
2432     \iftoggle{prenotesX@}{%
2433       \togglefalse{prenotesX@}\skip\csname footins#1\endcsname=\csuse{prenotesX@}}%
2434     }%
2435   }%
2436   \vskip\skip\csname footins#1\endcsname%
2437   \leftskip=\z@
2438   \rightskip=\z@
2439   \parindent=\z@
2440   \vskip\skip\@nameuse{footins#1}%
2441   \setnotesXwidthliketwocolumns@{#1}%
2442   \setnotesXpositionliketwocolumns@{#1}%
2443   \print@footnoterule{#1}%
2444   \vskip\csuse{afterruleX@#1}%
2445   }
2446

```

```
\para@vfootnoteX \para@vfootnoteX{<series>}{<text>}
```

```

\mppara@vfootnoteX 2447 \newcommand*{\para@vfootnoteX}[2]{%
2448   \insert\csname footins#1\endcsname
2449   \bgroup
2450     \csuse{bhooknoteX@#1}
2451     \csuse{notefontsizeX@#1}
2452     \footsplitskips
2453     \setbox0=\vbox{\hsize=\maxdimen
2454       \noindent\@nameuse{footfmt#1}{#1}{#2}}%
2455     \setbox0=\hbox{\unvbox0[#1]}%
2456     \dp0=\z@
2457     \ht0=\csname footfudgefactor#1\endcsname\wd0
2458     \box0
2459     \penalty0
2460   \egroup}
2461 \newcommand*{\mppara@vfootnoteX}[2]{%
2462   \global\setbox\@nameuse{mpfootins#1}\vbox{%
2463     \unvbox\@nameuse{mpfootins#1}
2464     \csuse{bhooknoteX@#1}
2465     \csuse{notefontsizeX@#1}
2466     \footsplitskips
2467     \setbox0=\vbox{\hsize=\maxdimen

```

```

2468      \noindent\color@begingroup\@nameuse{footfmt#1}{#1}{#2}\color@endgroup}%
2469      \setbox0=\hbox{\unvbox0[#1]}%
2470      \dp0=\z@
2471      \ht0=\csname footfudgefactor#1\endcsname\wd0
2472      \box0
2473      \penalty0}}
2474
\parafootfmtX \parafootfmtX{<series>}
2475 \newcommand*{\parafootfmtX}[2]{%
2476   \protected@edef\@currentlabel{%
2477     \@nameuse{@thefnmark#1}%
2478   }%
2479   \insertparafootsep{#1}%
2480   \ledsetnormalparstuff
2481   {\csuse{notenumfontX@#1}\csuse{notenumfontX@#1}\@nameuse{footfootmark#1}\strut%\enspace
2482     #2\penalty-10}}
2483
\para@footgroupX \para@footgroupX{<series>}
\mppara@footgroupX
2484 \newcommand*{\para@footgroupX}[1]{%
2485   \unvbox\csname footins#1\endcsname
2486   \ifcsstring{raggedX@#1}{L}{\RaggedLeft}{}%
2487   \ifcsstring{raggedX@#1}{R}{\RaggedRight}{}%
2488   \makehboxofhboxes
2489   \setbox0=\hbox{\unhbox0 \removehboxes}%
2490   \csuse{notefontsizeX@#1}
2491   \noindent\unhbox0\par}
2492 \newcommand*{\mppara@footgroupX}[1]{%
2493   \setnotesXwidthliketwocolumns@{#1}%
2494   \vskip\skip\@nameuse{mpfootins#1}
2495   \ifl@dpairing\ifparledgroup
2496     \leavevmode%
2497     \leavevmode\marks\parledgroup@{begin}%
2498     \marks\parledgroup@series{#1}%
2499     \marks\parledgroup@type{footnoteX}%
2500   \fi\fi\normalcolor
2501   \ifparledgroup%
2502     \ifl@dpairing%
2503     \else%
2504       \setnotesXwidthliketwocolumns@{#1}%
2505       \setnotesXpositionliketwocolumns@{#1}%
2506       \print@footnoteXrule{#1}%
2507       \vskip\csuse{afterruleX@#1}%
2508     \fi%
2509   \else%
2510     \setnotesXwidthliketwocolumns@{#1}%
2511     \setnotesXpositionliketwocolumns@{#1}%
2512     \print@footnoteXrule{#1}%
2513     \vskip\csuse{afterruleX@#1}%

```



```

2514 \fi%
2515 \unvbox\csname mpfootins#1\endcsname
2516 \ifcsstring{raggedX@#1}{L}{\RaggedLeft}{}%
2517 \ifcsstring{raggedX@#1}{R}{\RaggedRight}{}%
2518 \makehboxofhboxes
2519 \setbox0=\hbox{\unhbox0 \removehboxes}%
2520 \csuse{notefontsizeX@#1}
2521 \noindent\unhbox0\par}}
2522

```

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.

`\old@hsize` 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 `\noteswidthliketwocolumns@` at the on the setup for paragraphed notes.

```

2523
2524 \newdimen\old@hsize%
2525 \old@hsize=\linewidth%
2526
2527 \newcommand{\setXnoteswidthliketwocolumns@}[1]{%
2528   \global\let\hsize@fornote=\linewidth%
2529   \global\old@hsize=\linewidth%
2530   \iftoggle{Xnoteswidthliketwocolumns@#1}%
2531     {%
2532       \csuse{setwidthliketwocolumns@\columns@position}%
2533       \global\let\hsize@fornote=\hsize%
2534     }%
2535   }%
2536   \let\hsize=\hsize@fornote%
2537   \let\columnwidth=\hsize@fornote%
2538 }%
2539
2540 \newcommand{\setnotesXwidthliketwocolumns@}[1]{%
2541   \global\let\hsize@fornote=\hsize%
2542   \global\old@hsize=\linewidth%
2543   \iftoggle{notesXwidthliketwocolumns@#1}%
2544     {%
2545       \csuse{setwidthliketwocolumns@\columns@position}%
2546       \global\let\hsize@fornote=\hsize%
2547     }%
2548   }%
2549   \let\hsize=\hsize@fornote%
2550   \let\columnwidth=\hsize@fornote%
2551 }%
2552

```

`\setXnotespositionliketwocolumns@` These two commands set the position of the critical / familiar footnotes, depending on the hooks `Xnoteswidthliketwocolumns` and `notesXwidthliketwocolumns`.
`\setnotesXpositionliketwocolumns@` They call commands which are defined only in `eledpar`, because this feature has no sens without `eledpar`.

```

2553 \newcommand{\setXnotespositionliketwocolumns@}[1]{%
2554   \iftoggle{Xnoteswidthliketwocolumns@#1}{%
2555     \csuse{setnotespositionliketwocolumns@\columns@position}%
2556   }{}%
2557 }%
2558
2559 \newcommand{\setnotesXpositionliketwocolumns@}[1]{%
2560   \iftoggle{notesXwidthliketwocolumns@#1}{%
2561     \csuse{setnotespositionliketwocolumns@\columns@position}%
2562   }{}%
2563 }%
2564
```

27 Footnotes' order

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

```

\@fnpos 2565 \def\@fnpos{familiar-critical}
\@mpfnpos 2566 \def\@mpfnpos{critical-familiar}
2567 \newcommand{\fnpos}[1]{\xdef\@fnpos{#1}}
2568 \newcommand{\mpfnpos}[1]{\xdef\@mpfnpos{#1}}
```

28 Footnotes' rule

Because the footnotes' rules can be shifted to the right when footnotes are set like two columns, we don't print them directly, but we put them in a `\vbox`.

```

\print@Xfootnoterule
\print@footnotexrule 2569 \newcommand{\print@Xfootnoterule}[1]{%
2570   \nointerlineskip%
2571   \moveleft-\leftskip\vbox{\csuse{#1footnoterule}}%
2572   \nointerlineskip%
2573 }%
2574
2575 \newcommand{\print@footnotexrule}[1]{%
2576   \nointerlineskip%
2577   \moveleft-\leftskip\vbox{\csuse{footnoterule#1}}%
2578   \nointerlineskip%
2579 }%
2580
```

29 Footnotes' output

`\doxtrafeeti` We have to add all the new kinds of familiar footnotes to the output routine.

`\doreinxtrafeeti` These are the class 1 feet.

```

2581 \newcommand*{\doxtrafeeti}{%
2582   \setbox\@outputbox \vbox{%
2583     \unvbox\@outputbox
2584     \def\do##1{\ifvoid\csuse{footins##1}\else\csuse{footstart##1}{##1}\csuse{footgroup##1}{##1}\fi}
2585     \dolistloop{\@series}%
2586   }}
2587
2588 \newcommand{\doreinxtrafeeti}{%
2589   \def\do##1{\ifvoid\csuse{footins##1}\else\insert\csuse{footins##1}{\unvbox\csuse{footins##1}}\fi}%
2590   \dolistloop{\@series}%
2591 }
2592
\addfootinsX Juste for backward compatibility: print a warning message.
2593 \newcommand*{\addfootinsX}[1]{%
2594   \led@warn@AddfootinsX@obsolete%
2595   \footnormalX{#1}%
2596   \g@addto@macro{\doxtrafeeti}{%
2597     \setbox\@outputbox \vbox{%
2598       \unvbox\@outputbox
2599       \ifvoid\@nameuse{footins#1}\else
2600         \@nameuse{footstart#1}{#1}\@nameuse{footgroup#1}{#1}\fi}}%as
2601   \g@addto@macro{\doreinxtrafeeti}{%
2602     \ifvoid\@nameuse{footins#1}\else
2603       \insert\@nameuse{footins#1}{\unvbox\@nameuse{footins#1}}\fi}%
2604   \g@addto@macro{\l@dfambeginmini}{%
2605     \expandafter\expandafter\expandafter\let\expandafter\expandafter
2606       \csname footnote#1\endcsname \csname mpfootnote#1\endcsname}%
2607   \g@addto@macro{\l@dfamendmini}{%
2608     \ifvoid\@nameuse{mpfootins#1}\else\@nameuse{mpfootgroup#1}{#1}\fi}%
2609 }
```

30 Endnotes

`\l@d@end` Endnotes of all varieties are saved up in a file, typically named `\jobname.end`.

`\ifl@dend@` `\l@d@end` is the output stream number for this file, and `\ifl@dend@` is a flag that's

`\l@dend@true` true when the file is open.

`\l@dend@false` `\l@dend@false`

```

2610 \newwrite\l@d@end
2611 \newif\ifl@dend@

\l@dend@open \l@dend@open and \l@dend@close are the macros that are used to open and close
\l@dend@close the endnote file. Note that all our writing to this file is \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.

```
2612 \newcommand{\l@dend@open}[1]{\global\l@dend@true\immediate\openout\l@d@end=#1\relax}
2613 \newcommand{\l@dend@close}{\global\l@dend@false\immediate\closeout\l@d@end}
2614
```

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

```
2615 \newcommand{\l@dend@stuff}{%
2616   \ifl@dend@relax\else
2617     \l@dend@open{\jobname.end}%
2618   \fi
2619   \immediate\write\l@d@end{\string\l@d@section{\the\section@num}}
2620
```

`\endprint` The `\endprint` here is nearly identical in its functioning to `\normalfootfmt`.
`\l@d@section` The endnote file also contains `\l@d@section` commands, which supply the section numbers from the main text; standard `eledmac` does nothing with this information, but it's there if you want to write custom macros to do something with it.

```
2621 \def\endprint#1#2#3#4{{\csuse{bhookXendnote@#4}\csuse{Xendnote@fontsize@#4}{\csuse{Xendnote@font@#4}{\select@font#1|#2}{#3}}\enspace{\nottoggle{Xendlemmadisablefontselection@#4}{\select@font#1|#2}{#3}}\endprint}
2622
2623
2624 \let\l@d@section=\@gobble
2625
```

`\setprintendlines` The `\printendlines` 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.

```
2626 \newcommand*{\setprintendlines}[6]{%
2627   \l@d@pnumfalse \l@d@dashfalse
2628   \ifnum#4=#1 \else
2629     \l@d@pnumtrue
2630     \l@d@dashtrue
2631   \fi
```

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.

```
2632   \ifl@d@pnum \l@d@elintrue \else \l@d@elinfalse \fi
2633   \ifnum#2=#5 \else
2634     \l@d@elintrue
```

```
2635 \l@d@dashtrue
```

```
2636 \fi
```

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

```
2637 \l@d@ssubfalse
```

```
2638 \ifnum#3=0 \else
```

```
2639 \l@d@ssubtrue
```

```
2640 \fi
```

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.

```
2641 \l@d@eslfalse
```

```
2642 \ifnum#6=0 \else
```

```
2643 \ifnum#6=#3
```

```
2644 \ifl@d@elin \l@d@esltrue \else \l@d@eslfalse \fi
```

```
2645 \else
```

```
2646 \l@d@esltrue
```

```
2647 \l@d@dashtrue
```

```
2648 \fi
```

```
2649 \fi}
```

`\printendlines` Now we're ready to print it all.

```
2650 \def\printendlines#1|#2|#3|#4|#5|#6|#7|{\begingroup
```

```
2651 \setprintendlines{#1}{#2}{#3}{#4}{#5}{#6}%
```

The only subtlety left here is when to print a period between numbers. But the only instance in which this is tricky is for the ending sub-line number: it could be coming after the starting sub-line number (in which case we want only the dash) or after an ending line number (in which case we need to insert a period).

```
2652 \printnpnum{#1} \linenumrep{#2}%
```

```
2653 \ifl@d@ssub \fullstop \sublinenumrep{#3}\fi
```

```
2654 \ifl@d@dash \endashchar\fi
```

```
2655 \ifl@d@pnum \printnpnum{#4}\fi
```

```
2656 \ifl@d@elin \linenumrep{#5}\fi
```

```
2657 \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumrep{#6}\fi
```

```
2658 \endgroup}
```

```
2659
```

`\printnpnum` A macro to print a page number in an endnote.

```
2660 \newcommand*{\printnpnum}[1]{p.#1} }
```

```
2661
```

`\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.

```
2662 \newcommand*{\doendnotes}[1]{\l@dend@close
```

```
2663 \begingroup
```

```
2664 \makeatletter
```

```
2665 \expandafter\let\csname #1end\endcsname=\endprint
```

```
2666 \input\jobname.end
```

```
2667 \endgroup}
```

`\noendnotes` You can say `\noendnotes` before the first `\beginnumbering` in your file if if you will not use any of the endnote commands: this will suppress the creation of an `.end` file. If you do have some lingering endnote commands in your file, the notes will be written to your terminal and to the log file.

```
2668 \newcommand*{\noendnotes}{\global\let\l@dend@stuff=\relax
2669 \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@`.

```
2670 \newcommand{\newseries}[1]{%
2671 \def\do##1{\newseries@{##1}}%
2672 \docsvlist{#1}
2673 }
```

`\@series` The `\series@` macro is an etoolbox list, which contains the name of all series.

```
2674 \newcommand{\@series}{}%
```

The command `\newseries@\series` creates a new series of the footnote.

`\newseries@`

```
2675 \newcommand{\newseries@}[1]{%
```

31.1 Test if series is still existing

```
2676 \xifinlist{#1}{\@series}{\led@warn@SeriesStillExist{#1}}%
2677 {%
```

31.2 Create all commands to memorize display options

```
2678 \newtoggle{Xlemmadisablefontselection@#1}
2679 \newtoggle{Xendlemmadisablefontselection@#1}
2680 \csgdef{Xhangindent@#1}{0pt}%
2681 \csgdef{hangindentX@#1}{0pt}%
2682 \csgdef{Xragged@#1}{}%
2683 \csgdef{raggedX@#1}{}%
2684 \csgdef{hsizetwocol@#1}{0.45 \hsize}%
2685 \csgdef{hsizetwocolX@#1}{0.45 \hsize}%
2686 \csgdef{hsizethreecol@#1}{.3 \hsize}%
2687 \csgdef{hsizethreecolX@#1}{.3 \hsize}%
2688 \csgdef{Xnotenumfont@#1}{\notenumfont}%
2689 \csgdef{Xendnotenumfont@#1}{\notenumfont}%
2690 \csgdef{notenumfontX@#1}{\notenumfont}%
2691 \csgdef{Xnotefontsize@#1}{\notefontsetup}%
2692 \csgdef{notefontsizeX@#1}{\notefontsetup}%
2693 \csgdef{Xendnotefontsize@#1}{\notefontsetup}%
```

```

2694 \csgdef{bhooknoteX@#1}{}%
2695 \csgdef{bhookXnote@#1}{}%
2696 \csgdef{bhookXendnote@#1}{}%
2697 \csgdef{boxlinenum@#1}{0pt}%
2698 \csgdef{boxsymlinenum@#1}{0pt}%
2699 \newtoggle{numberonlyfirstinline@#1}%
2700 \newtoggle{numberonlyfirstintwolines@#1}%
2701 \newtoggle{onlypstartinfootnote@#1}%
2702 \newtoggle{pstartinfootnote@#1}%
2703 \csgdef{symlinenum@#1}{\symlinenum}%
2704 \newtoggle{nonnumberinfootnote@#1}%
2705 \csgdef{beforenumberinfootnote@#1}{0pt}%
2706 \csgdef{afternumberinfootnote@#1}{0.5em}%
2707 \newtoggle{nonbreakableafternumber@#1}%
2708 \csgdef{beforesymlinenum@#1}{\csuse{beforenumberinfootnote@#1}}%
2709 \csgdef{aftersymlinenum@#1}{\csuse{afternumberinfootnote@#1}}%
2710 \csgdef{inplaceofnumber@#1}{1em}%
2711 \global\cslet{lemmaseparator@#1}{\rbracket}%
2712 \csgdef{beforelemmaseparator@#1}{0em}%
2713 \csgdef{afterlemmaseparator@#1}{0.5em}%
2714 \csgdef{inplaceoflemmaseparator@#1}{1em}%
2715 \csgdef{afternote@#1}{1em plus .4em minus .4em}%
2716 \csgdef{parafootsep@#1}{\parafootftmsep}%
2717 \csgdef{beforeXnotes@#1}{1.2em \@plus .6em \@minus .6em}%
2718 \csgdef{beforenotesX@#1}{1.2em \@plus .6em \@minus .6em}%
2719 \csgdef{afterXrule@#1}{0pt}%
2720 \csgdef{afterruleX@#1}{0pt}%
2721 \csgdef{txtbeforeXnotes@#1}{}%
2722 \csgdef{maxhnotesX@#1}{\ledfootinsdim}%
2723 \csgdef{maxhXnotes@#1}{\ledfootinsdim}%
2724 \newtoggle{Xnoteswidthliketwocolumns@#1}%
2725 \newtoggle{notesXwidthliketwocolumns@#1}%

```

31.3 Create inserts, needed to add notes in foot

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

```

2726
2727 \expandafter\newinsert\csname mpfootins#1\endcsname
2728 \expandafter\newinsert\csname footins#1\endcsname
2729 \expandafter\newinsert\csname #1footins\endcsname
2730 \expandafter\newinsert\csname mp#1footins\endcsname

```

31.4 Create commands for critical apparatus, \Xfootnote

Note the double # in command: it's because command is made inside another command.

```

2731
2732 \global\newbool{parapparatus@}{\expandafter\newcommand\expandafter *}{\expandafter\newcommand}\cs
2733 \begingroup%
2734 \newcommand{\content}{##2}%

```

```

2735 \ifnumberedpar@
2736 \ifledRcol%
2737 \ifluatex%
2738 \footnotelang@lua[R]%
2739 \fi%
2740 \@ifundefined{xpg@main@language}%if polyglossia
2741 {}%
2742 {\footnotelang@poly[R]}%
2743 \footnoteoptions@R[##1]{true}%
2744 \xright@appenditem{\noexpand\prepare@edindex@fornote{\l@d@nums}}%
2745 \noexpand\csuse{v#1footnote}{#1}%
2746 {{\l@d@nums}{\expandonce\@tag}{\expandonce\content}}}\to\inserts@listR
2747 \footnoteoptions@R[##1]{false}%
2748 \global\advance\insert@count \@ne%
2749 \else%
2750 \ifluatex%
2751 \footnotelang@lua%
2752 \fi%
2753 \@ifundefined{xpg@main@language}%if polyglossia
2754 {}%
2755 {\footnotelang@poly}%
2756 \footnoteoptions@##1{true}%
2757 \xright@appenditem{\noexpand\prepare@edindex@fornote{\l@d@nums}}%
2758 \noexpand\csuse{v#1footnote}{#1}%
2759 {{\l@d@nums}{\expandonce\@tag}{\expandonce\content}}}\to\inserts@list
2760 \global\advance\insert@count \@ne%
2761 \footnoteoptions@##1{false}%
2762 \fi
2763 \else
2764 \csuse{v#1footnote}{#1}{\{0|0|0|0|0|0|0\}}{##1}%
2765 \fi%
2766 \ignorespaces%
2767 \endgroup
2768 }

```

Set standard display and remember the display.

```

2769 \csgdef{series@display#1}{
2770 \footnormal{#1}

```

31.5 Create tools for familiar footnotes (\footnoteX)

First, create the \footnoteX command.

```

2771
2772 \global\expandafter\newcommand\csname footnote#1\endcsname[1]{%
2773 \begingroup%
2774 \newcommand{\content}{##1}%
2775 \stepcounter{footnote#1}%
2776 \protected@csxdef{@thefnmark#1}{\csuse{thefootnote#1}}%
2777 \csuse{@footnotemark#1}%
2778 \csuse{vfootnote#1}{#1}{\expandonce\content}\m@mmf\prepare%

```



```
2779     \endgroup%
2780 }
```

The counters.

```
2781     \newcounter{footnote#1}
2782     \global\expandafter\renewcommand\csname thefootnote#1\endcsname{\arabic{footnote#1}}
2783 % \end{macrocode}
2784 % Don't forget to initialize series
2785 %     \begin{macrocode}
2786     \csgdef{series@displayX#1}{\}
2787     \footnormalX{#1}
```

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

```
2788
2789     \global\expandafter\newcommand\csname #1endnote\endcsname[2]{\newlinechar='40
2790         \global\@noneed@Footnotetrue%
2791         \newcommand{\content}{##1}%
2792         \immediate\write\l@end{\expandafter\string\csname #1end\endcsname%
2793             {\ifnumberedpar@l@d@nums\fi}%
2794             {\ifnumberedpar@\expandonce\@tag\fi}{\expandonce\content}{#1}}\ignorespaces%
2795     }
```

`\Xendnote` 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.²⁷

```
2796
2797     \global\cslet{#1end}{\@gobblefour}
2798 %\end{macrocode}
2799 % We need to be able to modify \Eledmac's footnote macros and restore their
2800
2801     \global\csletcs{#1@footnote}{#1footnote}
2802 % \cs{Stock series in \cs{@series}
2803 %     \begin{macrocode}
2804
2805     \listxadd{\@series}{#1}
2806 }
2807 }% End of \newseries
```

²⁷Christophe Hebeisen (christophe.hebeisen@a3.epfl.ch) emailed on 2003/11/05 to say he had found that `\@gobblethree` was also defined in the `amsfonts` package.

31.7 Init standards series (A,B,C,D,E,Z)

```
2808 \newseries{A,B,C,D,E,Z}
```

31.8 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.

```
2809 \newcommand{\seriesatbegin}[1]{
2810     \edef\series{#1}
2811     \def\new{}
2812     \listadd{\new}{\series}
2813     \def\do##1{\ifcsstring{series}{##1}{-}{\listadd{\new}{##1}}}
2814     \dolistloop{@series}
2815     \xdef\@series{\new}
2816 }
2817 % \end{macrocode}
2818 % \end{macro}
2819 % \begin{macro}{\seriesatend}
2820 % And \cs{seriesatend} moves the series to the end of the list.
2821 % \begin{macrocode}
2822 \newcommand{\seriesatend}[1]{
2823     \edef\series{#1}
2824     \def\new{}
2825     \def\do##1{\ifcsstring{series}{##1}{-}{\listadd{\new}{##1}}}
2826     \dolistloop{@series}
2827     \listadd{\new}{\series}
2828     \xdef\@series{\new}
2829 }
2830 % \end{macrocode}
2831 % \end{macro}
2832 % \subsection{Display}
2833 % \changes{v1.0}{2012/09/15}{New generic commands to customize footnote display.}
2834 % \subsubsection{Options}
2835 % \begin{macro}{\settoggle@series}
2836 % \changes{v1.1}{2012/09/25}{\cs{settoggle@series} switch the global value of the toggle,
2837 % \changes{v1.13.0}{2014/09/16}{\cs{settoggle@series} can take an optional arguments to r
2838 % \cs{settoggle@series}\marg{series}\marg{toggle}\marg{value} is a generic command to swi
2839
2840 % \begin{macrocode}
2841 \newcommandx{\settoggle@series}[4][4]{%
2842     \def\do##1{%
2843         \global\settoggle{#2@##1}{#3}%
2844         \ifstrequal{#4}{reload}%
2845             {%
2846                 \csuse{foot\csuse{series@display##1}}{##1}%
2847                 \csuse{foot\csuse{series@displayX##1}}{##1}%
2848             }%
2849         }%
2850     }%
2851     \ifstrempy{#1}{%
```

```

2852         \dolistloop{\@series}%
2853     }%
2854     {%
2855         \docsvlist{#1}%
2856     }%
2857 }

```

`\setcommand@series` `\setcommand@series{<series>}{<command>}{<value>}` is a generic command to change commands for some series.

```

2858 \newcommandx{\setcommand@series}[4][4]{%
2859     \def\do##1{
2860         \csgdef{#2@##1}{#3}
2861         \ifstrequal{#4}{reload}{
2862             \csuse{foot\csuse{series@display##1}}{##1}
2863             \csuse{foot\csuse{series@displayX##1}}{##1}
2864         }{}}
2865     \ifstreempty{#1}{%
2866         \dolistloop{\@series}%
2867     }%
2868     {%
2869         \docsvlist{#1}%
2870     }%
2871 }%

```

`\newhookcommand@series` `\newhookcommand@series\command names` is a generic command to add new commands for hooks, like `\hsizetwocol`.

```

2872 \newcommand{\newhookcommand@series}[1]{%
2873     \global\expandafter\newcommand\expandafter*\csname #1\endcsname[2][2]{%
2874         \setcommand@series{##1}{#1}{##2}%
2875     }%
2876 }
2877 \newhookcommand@series{Xhangindent}
2878
2879 \newhookcommand@series{hangindentX}
2880
2881 \newhookcommand@series{Xragged}
2882
2883 \newhookcommand@series{raggedX}
2884
2885 \newhookcommand@series{hsizetwocol}
2886
2887 \newhookcommand@series{hsizethreecol}
2888
2889 \newhookcommand@series{hsizetwocolX}
2890
2891 \newhookcommand@series{hsizethreecolX}
2892
2893 \newhookcommand@series{Xnotenumfont}
2894

```

```

2895 \newhookcommand@series{notenumfontX}
2896
2897 \newhookcommand@series{Xendnotenumfont}
2898
2899 \newhookcommand@series{bhooknoteX}
2900
2901 \newhookcommand@series{bhookXnote}
2902
2903 \newhookcommand@series{bhookXendnote}
2904
2905 \newhookcommand@series{Xnotefontsize}
2906
2907 \newhookcommand@series{notefontsizeX}
2908
2909 \newhookcommand@series{Xendnotefontsize}
2910
2911 \newhookcommand@series{boxlinenum}
2912
2913 \newhookcommand@series{boxsymlinenum}
2914
2915 \newhookcommand@series{parafootsep}
2916
2917 \newhookcommand@series{symlinenum}
2918
2919 \newhookcommand@series{beforenumberinfootnote}
2920
2921 \newhookcommand@series{afternumberinfootnote}
2922
2923 \newhookcommand@series{beforesymlinenum}
2924
2925 \newhookcommand@series{aftersymlinenum}
2926
2927 \newhookcommand@series{inplaceofnumber}
2928
2929 \newhookcommand@series{lemmaseparator}
2930
2931 \newhookcommand@series{beforelemmaseparator}
2932
2933 \newhookcommand@series{afterlemmaseparator}
2934
2935 \newhookcommand@series{inplaceoflemmaseparator}
2936
2937 \newhookcommand@series{afternote}
2938
2939 \newhookcommand@series{txtbeforeXnotes}
2940
2941 \newhookcommand@series{afterruleX}
2942
2943 \newhookcommand@series{afterXrule}
2944

```

```

hookcommand@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).

2945 \newcommand{\newhookcommand@series@reload}[1]{%
2946   \global\expandafter\newcommand\expandafter*\csname #1\endcsname[2][1]{%
2947     \setcommand@series{##1}{#1}{##2}[reload]%
2948   }%
2949 }
2950 \newhookcommand@series@reload{beforeXnotes}
2951
2952 \newhookcommand@series@reload{beforenotesX}
2953
2954 \newhookcommand@series@reload{maxhnotesX}
2955
2956 \newhookcommand@series@reload{maxhXnotes}
2957 % \end{macrocode}
2958 % \end{macro}
2959 % \begin{macro}{\newhooktoggle@series}
2960 %\cs{newhooktoggle@series}\cs{command names} is a generic command to add new commands for new toggle
2961 %   \begin{macrocode}
2962 \newcommand{\newhooktoggle@series}[1]{%
2963   \global\expandafter\newcommandx\expandafter*\csname #1\endcsname[2][1,2={true},usedefault]{%
2964     \settoggle@series{##1}{#1}{##2}%
2965   }%
2966 }
2967 \newhooktoggle@series{numberonlyfirstinline}
2968 \newhooktoggle@series{numberonlyfirstintwolines}
2969 \newhooktoggle@series{nonumberinfootnote}
2970 \newhooktoggle@series{pstartinfootnote}
2971 \newhooktoggle@series{onlypstartinfootnote}
2972 \newhooktoggle@series{nonbreakableafternumber}
2973 \newhooktoggle@series{Xlemmadisablefontselection}
2974 \newhooktoggle@series{Xendlemmadisablefontselection}

\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).

2975 \newcommand{\newhooktoggle@series@reload}[1]{%
2976   \global\expandafter\newcommandx\expandafter*\csname #1\endcsname[2][1,2={true},usedefault]{%
2977     \settoggle@series{##1}{#1}{##2}[reload]%
2978   }%
2979 }%
2980
2981 \newhooktoggle@series@reload{Xnoteswidthliketwocolumns}%
2982 \newhooktoggle@series@reload{notesXwidthliketwocolumns}%
2983

```

31.9 Old commands, kept for backward compatibility

The next commands are kept for ascendant compatibility, but should't be used anymore.

```
\notenumfont
\notefontsetup 2984 \newcommand*{\notenumfont}{\normalfont}
\ifledplinenum 2985 \newcommand*{\notefontsetup}{\footnotesize}
\symlinenum 2986 \newif\ifledplinenum
                2987 \ledplinenumtrue
                2988 \newcommand*{\symlinenum}{}
```

31.10 Hooks for a particular footnote

`\nonum@` `\nonum@` toggle is used to disable line number printing in a particular footnote.

```
2989 \newtoggle{\nonum@}
```

`\nosep@` `\nonum@` toggle is used to disable the lemma separator in a particular footnote.

```
2990 \newtoggle{\nosep@}
```

31.11 Alias

`\nolemmaseparator` `\nolemmaseparator[<series>]` is just an alias for `\lemmaseparator[<series>]{}`.

```
2991 \newcommand*{\nolemmaseparator}[1][1]{\lemmaseparator[#1]{}}
```

`\interparanoteglue` The `\ipn@skip` skip and `\interparanoteglue` command are kept for backward compatibility, but should not be used anymore.

```
2992 \newskip\ipn@skip
2993 \newcommand*{\interparanoteglue}[1]{%
2994     {\notefontsetup\global\ipn@skip=#1 \relax}}
2995 \interparanoteglue{1em plus.4em minus.4em}
```

`\parafootftmsep` The `\parafootftmsep` macro is kept for backward compatibility. It is default value of `\parafootsep@series`.

```
2996 \newcommand{\parafootftmsep}{}
```

31.12 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.

```
2997 \newcommand{\printlinefootnote}[2]{%
2998     \def\extractline@##1|##2|##3|##4|##5|##6|##7|{##2}%
2999     \def\extractsubline@##1|##2|##3|##4|##5|##6|##7|{##3}%
3000     \def\extractendline@##1|##2|##3|##4|##5|##6|##7|{##5}%
3001     \def\extractendsubline@##1|##2|##3|##4|##5|##6|##7|{##6}%
3002     \iftoggle{numberonlyfirstintwolines@#2}{%
```

```

3003     \edef\lineinfo@{\extractline@ #1| - \extractsubline@ #1| - \extractendline@ #1| - \extractends
3004     }%
3005     {%
3006     \edef\lineinfo@{\extractline@ #1| - \extractsubline@ #1|}%
3007     }%
3008     \iftoggle{nonum@}{%Try if the line number must printed for this specific not (by default, yes)
3009     \hspace{\csuse{inplaceofnumber@#2}}%
3010     }%
3011     {%
3012     {%
3013     \iftoggle{nonumberinfootnote@#2}%Try if the line number must printed (by default, yes)
3014     {%
3015     \hspace{\csuse{inplaceofnumber@#2}}%
3016     }%
3017     {%
3018     {\iftoggle{numberonlyfirstinline@#2}% If for this series the line number must be printed
3019     {%
3020     \ifcsdef{prevline#2}%
3021     {%Be sure the \prevline exists.
3022     \ifcsequal{prevline#2}{\lineinfo@}%Try it
3023     {%
3024     \ifcsequal{symlinenum@#2}% Try if a symbol is defined
3025     {%
3026     \hspace{\csuse{inplaceofnumber@#2}}%
3027     }%
3028     {\hspace{\csuse{beforenumberinfootnote@#2}}\csuse{Xnotenumfont@#2}%
3029     \ifdimequal{\csuse{boxsymlinenum@#2}}{0pt}{%
3030     {\csuse{symlinenum@#2}}%
3031     {\hbox to \csuse{boxsymlinenum@#2}{\csuse{symlinenum@#2}\hfill}}%
3032     \hspace{\csuse{aftersymlinenum@#2}}}%
3033     }%
3034     {%
3035     \hspace{\csuse{beforenumberinfootnote@#2}}\csuse{Xnotenumfont@#2}%
3036     \ifdimequal{\csuse{boxlinenum@#2}}{0pt}{%
3037     \iftoggle{pstartinfootnote@#2}{\printpstart}{}%
3038     \printlines#1|}%
3039     {%
3040     \hbox to \csuse{boxlinenum@#2}{%
3041     \iftoggle{pstartinfootnote@#2}{\printpstart}{}%
3042     \iftoggle{onlypstartinfootnote@#2}{\printlines#1|}%
3043     \hfill}%
3044     }%
3045     \iftoggle{nonbreakableafternumber@#2}{\nobreak}{\hspace{\csuse{afternumber@#2}}}%
3046     }%
3047     }%
3048     {%
3049     \hspace{\csuse{beforenumberinfootnote@#2}}\csuse{Xnotenumfont@#2}%
3050     \ifdimequal{\csuse{boxlinenum@#2}}{0pt}{%
3051     \iftoggle{pstartinfootnote@#2}{\printpstart}{}%
3052     \iftoggle{onlypstartinfootnote@#2}{\printlines#1|}%

```

```

3053             {%
3054             \hbox to \csuse{boxlinenum@#2}{%
3055                 \iftoggle{pstartinfootnote@#2}{\printpstart}{}%
3056                 \iftoggle{onlypstartinfootnote@#2}{\printlines#1}{}%
3057                 \hfill}%
3058             }%
3059             \iftoggle{nonbreakableafternumber@#2}{\nobreak}{\hspace{\csuse{afternumber@#2}}}
3060             }%
3061         }%
3062     {%
3063     \hspace{\csuse{beforenumberinfootnote@#2}}\csuse{Xnotenumfont@#2}%
3064     \ifdimequal{\csuse{boxlinenum@#2}}{0pt}{%
3065         \iftoggle{pstartinfootnote@#2}{\printpstart}{}%
3066         \iftoggle{onlypstartinfootnote@#2}{\printlines#1}{}%
3067     }%
3068     {%
3069     \hbox to \csuse{boxlinenum@#2}{%
3070         \iftoggle{pstartinfootnote@#2}{\printpstart}{}%
3071         \iftoggle{onlypstartinfootnote@#2}{\printlines#1}{}%
3072         \hfill}%
3073     }%
3074     \iftoggle{nonbreakableafternumber@#2}{\nobreak}{\hspace{\csuse{afternumber@#2}}}
3075     }%
3076     \csxdef{prevline#2}{\lineinfo@}%
3077     }%
3078 }%
3079 }%
3080 }%
3081 }

```

32 Output routine

Now we begin the output routine and associated things.

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

```

3082 \countdef\pageno=0 \pageno=1
3083 \newcommand*{\advancepageno}{\ifnum\pageno<\z@ \global\advance\pageno\m@ne
3084   \else\global\advance\pageno\@ne\fi}
3085

```

The next portion is probably the trickiest part of moving from TeX to L^AT_EX. 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{\edmac@output}
\def\edmac@output{\shipout\vbox{\normal@pars
  \vbox{\makeheadline\pagebody\makefootline}%
}%
\advancepageno
\ifnum\outputpenalty>-\@MM\else\dosupereject\fi}

\def\pagecontents{\page@start
  \ifvoid\topins\else\unvbox\topins\fi
  \dimen@=\dp\@cclv \unvbox\@cclv % open up \box255
  \do@feet
  \ifraggedbottom \kern-\dimen@ \vfil \fi}
```

`\do@feet` ships out all the footnotes. Standard EDMAC has only five feet, but there is nothing in principle 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
      \vskip \skip\footins
      \color@begingroup
        \normalcolor
        \footnoterule
        \unvbox \footins
      \color@endgroup
    }%
  \fi
  \xdef\@freelist{\@freelist\@midlist}%
  \global \let \@midlist \@empty
  \@combinefloats
  \ifvbox\@kludgeins
    \@makespecialcolbox
  \else
    \setbox\@outputbox \vbox to\@colht {%
      \@texttop
      \dimen@ \dp\@outputbox
```

```

        \unvbox\@outputbox
        \vskip -\dimen@
        \@textbottom
    }%
\fi
\global \maxdepth \@maxdepth
}

\gdef \@reinserts{%
  \ifvoid\footins\else\insert\footins{\unvbox\footins}\fi
  \ifvbox\@kludgeins\insert\@kludgeins{\unvbox\@kludgeins}\fi
}

```

Now we start actually changing things.

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

```

\m@m@makecolintro 3086 \providecommand{\m@m@makecolfloats}{%
3087   \xdef\@freelist{\@freelist\@midlist}%
3088   \global \let \@midlist \@empty
3089   \@combinefloats}
3090 \providecommand{\m@m@makecoltext}{%
3091   \ifvbox\@kludgeins
3092     \@makespecialcolbox
3093   \else
3094     \setbox\@outputbox \vbox to\@colht {%
3095       \@texttop
3096       \dimen@ \dp\@outputbox
3097       \unvbox\@outputbox
3098       \vskip -\dimen@
3099       \@textbottom}%
3100   \fi}
3101 \providecommand{\m@m@makecolintro}{%
3102

```

`\l@d@makecol` This is a partitioned version of the ‘standard’ `\@makecol`, with the initial code put into another macro.

```

3103 \gdef\l@d@makecol{%
3104   \l@ddofootinsert
3105   \m@m@makecolfloats
3106   \m@m@makecoltext
3107   \global \maxdepth \@maxdepth}
3108

```

`\ifFN@bottom` The `\ifFN@bottom` macro is defined by the footmisc package. If this package is not loaded, we define it.

```

3109 \AtBeginDocument{\@ifpackageloaded{footmisc}{\newif\ifFN@bottom}}

```

`\l@ddofootinsert` This macro essentially holds the initial portion of the kernel `\@makecol` code.

```

3110 \newcommand*{\l@ddofootinsert}{%
3111 %%% \page@start
3112 \ifvoid\footins
3113 \setbox\@outputbox \box\@cclv
3114 \else
3115 \setbox\@outputbox \vbox {%
3116 \boxmaxdepth \@maxdepth
3117 \@tempdima\dp\@cclv
3118 \unvbox \@cclv
3119 \ifFN@bottom\vfill\fi\vskip \skip\footins%% If the option bottom of loadmisc package is loaded
3120 \color@begingroup
3121 \normalcolor
3122 \footnoterule
3123 \unvbox \footins
3124 \color@endgroup
3125 }%
3126 \fi

```

That's the end of the copy of the kernel code. We finally call a macro to handle all the additional EDMAC feet.

```

3127 \l@ddoxtrafeet
3128 }
3129

```

`\doxtrafeet` `\doxtrafeet` is the code extending `\@makecol` to cater for the extra `eledmac` feet. We have two classes of extra footnotes. By default, we order the footnote inserts so that the regular footnotes are first, then class 1 (familiar footnotes) and finally class 2 (critical footnotes).

```

3130 \newcommand*{\l@ddoxtrafeet}{%
3131 \IfStrEq{familiar-critical}{\@fnpos}
3132 {\doxtrafeeti\doxtrafeetii}%
3133 {%
3134 \IfStrEq{critical-familiar}{\@fnpos}%
3135 {\doxtrafeetii\doxtrafeeti}%
3136 {\doxtrafeeti\doxtrafeetii}%
3137 }%
3138 }%
3139

```

`\doxtrafeetii` `\doxtrafeetii` is the code extending `\@makecol` to cater for the extra critical feet (class 2 feet). NOTE: the code is likely to be ‘featurefull’.

```

3140 \newcommand*{\doxtrafeetii}{%
3141 \setbox\@outputbox \vbox{%
3142 \unvbox\@outputbox
3143 \@opxtrafeetii}}

```

`\@opxtrafeetii` The extra critical feet to be added to the output.

```

3144 \newcommand*{\@opxtrafeetii}{%
3145   \def\do##1{\ifvoid\csuse{##1footins}\else\csuse{##1footstart}{##1}\csuse{##1footgroup}{%
3146     \dolistloop{\@series}}
\l@ddodoreinextrafeet \l@ddodoreinextrafeet 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@ddoxtrafeet.
3147 \newcommand*{\l@ddodoreinextrafeet}{%
3148   \doreinextrafeeti
3149   \doreinextrafeetii}
3150
\doreinextrafeetii \doreinextrafeetii is the code for catering for the class 2 extra critical footnotes
within \@reinserts. The implementation may well have to change.
3151 \newcommand*{\doreinextrafeetii}{%
3152   \def\do##1{\ifvoid\csuse{##1footins}\else\insert\csuse{##1ootins}{\unvbox\csuse{##1foot
3153     \dolistloop{\@series}
3154 }
3155
\l@d@reinserts And here is the modified version of \@reinserts.
3156 \gdef \l@d@reinserts{%
3157   \ifvoid\footins\else\insert\footins{\unvbox\footins}\fi
3158   \l@ddodoreinextrafeet
3159   \ifvbox\@kludgeins\insert\@kludgeins{\unvbox\@kludgeins}\fi
3160 }
3161
The memoir class does not use the ‘standard’ versions of \@makecol and
\@reinserts, due to its sidebar insert. We had better add that code if mem-
oir is used. (It can be awkward dealing with \if code within \if code, so don’t
use \ifl@dmemoir here.)
3162 \@ifclassloaded{memoir}{%
memoir is loaded so we use memoir’s built in hooks.
3163   \g@addto@macro{\m@mdoextrafeet}{\l@ddoxtrafeet}%
3164   \g@addto@macro{\m@mdodoreinextrafeet}{\l@ddodoreinextrafeet}%
3165   }{%
memoir has not been loaded, so redefine @makecol and @reinserts.
3166   \gdef\@makecol{\l@d@makecol}%
3167   \gdef\@reinserts{\l@d@reinserts}%
3168 }
3169
\addfootins \addfootins is for backward compatibility, but should’nt be used anymore.
3170 \newcommand*{\addfootins}[1]{%
3171   \led@warn@AddfootinsObsolete%
3172   \footnormal{#1}

```

```

3173 \g@addto@macro{\@opxtrafeetii}{%
3174   \ifvoid\@nameuse{#1footins}\else
3175     \@nameuse{#1footstart{#1}}\@nameuse{#1footgroup}{#1}\fi}
3176 \g@addto@macro{\doreinxtrafeetii}{%
3177   \ifvoid\@nameuse{#1footins}\else
3178     \insert\@nameuse{#1footins}{\unvbox\@nameuse{#1footins}}\fi}
3179 \g@addto@macro{\l@dedbeginmini}{%
3180   \expandafter\let\csname #1footnote\endcsname = \@nameuse{mp#1footnote}}
3181 \g@addto@macro{\l@dedendmini}{%
3182   \ifvoid\@nameuse{mp#1footins}\else\@nameuse{mpfootgroup#1{#1}}\fi}
3183 }

```

It turns out that `\@doclearpage` also needs modifying.

`\if@led@nofoot` We have to check if there are any leftover feet. `\@led@extranofeet` is a hook for `\@led@extranofeet` handling further footnotes.

```

3184 \newif\if@led@nofoot
3185 \newcommand*{\@led@extranofeet}{}
3186
3187 \@ifclassloaded{memoir}{%

```

If the memoir class is loaded we hook into its modified `\@doclearpage`.

`\@mem@extranofeet`

```

3188 \g@addto@macro{\@mem@extranofeet}{%
3189   \def\do#1{\ifvoid\csuse{#1footins}\else\@mem@nofootfalse\fi%
3190   \ifvoid\csuse{footins#1}\else\@mem@nofootfalse\fi%
3191   }
3192   \dolistloop{\@series}%
3193   \@led@extranofeet}
3194 }{%

```

As memoir is not loaded we have to do it all here.

`\@led@testifnofoot`

```

\@doclearpage 3195 \newcommand*{\@led@testifnofoot}{%
3196   \@led@nofoottrue
3197   \ifvoid\footins\else\@led@nofootfalse\fi
3198   \def\do##1{\ifvoid\csuse{##1footins}\else\@led@nofootfalse\fi%
3199   \ifvoid\csuse{footins##1}\else\@led@nofootfalse\fi}%
3200   \dolistloop{\@series}
3201   \@led@extranofeet}
3202
3203 \renewcommand{\@doclearpage}{%
3204   \@led@testifnofoot
3205   \if@led@nofoot
3206     \setbox\@tempboxa\vsplit\@cclv to\z@ \unvbox\@tempboxa
3207     \setbox\@tempboxa\box\@cclv
3208     \xdef\@deferlist{\@toplist\@botlist\@deferlist}%
3209     \global \let \@toplist \@empty

```

```

3210 \global \let \@botlist \@empty
3211 \global \colroom \@colht
3212 \ifx \@currlist\@empty
3213 \else
3214 \latexerr{Float(s) lost}\@ehb
3215 \global \let \@currlist \@empty
3216 \fi
3217 \makecolumn\deferlist
3218 \whiles\ifcolmade \fi{\opcol\makecolumn\deferlist}%
3219 \iftwocolumn
3220 \iffirstcolumn
3221 \xdef\dbldeferlist{\dbltoplist\dbldeferlist}%
3222 \global \let \dbltoplist \@empty
3223 \global \colht \textheight
3224 \begingroup
3225 \dblfloatplacement
3226 \makecolumn\dbldeferlist
3227 \whiles\ifcolmade \fi{\outputpage
3228 \makecolumn\dbldeferlist}%
3229 \endgroup
3230 \else
3231 \vbox{}\clearpage
3232 \fi
3233 \fi
3234 \else
3235 \setbox\cclv\vbox{\box\cclv\vfil}%
3236 \l@makecol\opcol
3237 \clearpage
3238 \fi}
3239 }
3240

```

33 Cross referencing

Peter Wilson have rewritten portions of the code in this section so that the LaTeX .aux file is used. This will also handle \included files.

Further, I have renamed some of the original EDMAC macros so that they do not clash with the LaTeX label/ref commands (EDMAC and LaTeX use very different mechanisms). In particular, the original EDMAC \label and \pageref have been renamed as \edlabel and \edpageref respectively.

You can mark a place in the text using a command of the form \edlabel{foo}, and later refer to it using the label foo by saying \edpageref{foo}, or \lineref{foo} or \sublineref{foo}. These reference commands will produce, respectively, the page, line and sub-line on which the \edlabel{foo} command occurred.

The reference macros warn you if a reference is made to an undefined label. If foo has been used as a label before, the \edlabel{foo} command will issue

a complaint; subsequent `\edpageref` and `\lineref` commands will refer to the latest occurrence of `\label{foo}`.

`\labelref@list` Set up a new list, `\labelref@list`, to hold the page, line and sub-line numbers for each label.

```
3241 \list@create{\labelref@list}
```

`\zz@@@` A convenience macro to zero two labeling counters in one go.

```
3242 %% \newcommand*{\zz@@@}{000|000|000} % set three counters to zero in one go
```

```
3243 \newcommand*{\zz@@@}{000|000} % set two counters to zero in one go
```

```
3244
```

`\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.²⁸

This version of the original EDMAC `\label` uses `\@bsphack` and `\@esphack` to eliminate extra space problems and also the LaTeX write methods for the `.aux` file.

Jesse Billett²⁹ found that the original code could be off by several pages. This version, hopefully cures that, and also allows for non-arabic page numbering.

```
3245 \newcommand*{\edlabel}[1]{\@bsphack
3246   \write\linenum@out{\string\@lab}%
3247   \ifx\labelref@list\empty
3248     \xdef\label@refs{\zz@@@}%
3249   \else
3250     \glp\labelref@list\to\label@refs
3251   \ifvmode
3252     \advancelabel@refs
3253   \fi
3254   \fi}
```

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

```
3255 \protected@write\@auxout{}%
3256   {\string\l@dmake@labels\space\thepage|\label@refs|\the\c@pstart|{#1}}%
3257   \ifdef\hypertarget{\hypertarget{#1}{}}{}%
3258   \@esphack}
3259
```

`\advancelabel@refs` In cases where `\edlabel` is the first element in a paragraph, we have a problem with line counts, because line counts change only at the first horizontal box of the paragraph. Hence, we need to test `\edlabel` if it occurs at the start of a paragraph.

²⁸The remaining macros in this section were kindly revised by Wayne Sullivan, who substantially improved their efficiency and flexibility.

²⁹(jdb43@cam.ac.uk) via the ctt thread 'ledmac cross referencing', 25 August 2003.

To do so, we use `\ifvmode`. If the test is true, we must advance by one unit the amount of text we write into the `.aux` file. We do so using `\advancelabel@refs` command.

```

3260 \newcounter{line}%
3261 \newcounter{subline}%
3262 \newcommand{\advancelabel@refs}{%
3263     \setcounter{line}{\expandafter\labelrefsparseline\label@refs}%
3264     \stepcounter{line}%
3265     \ifsublines%
3266         \setcounter{subline}{\expandafter\labelrefsparsesubline\label@refs}%
3267         \stepcounter{subline}{1}%
3268         \def\label@refs{\theline|\thesubline}%
3269     \else%
3270         \def\label@refs{\theline|0}%
3271     \fi%
3272 }
3273 \def\labelrefsparseline#1|#2{#1}
3274 \def\labelrefsparsesubline#1|#2{#2}

```

`\l@dmake@labels` The `\l@dmake@labels` macro gets executed when the labels file is read. For each label it defines a macro, whose name is made up partly from the label you supplied, that contains the page, line and sub-line numbers. But first it checks to see whether the label has already been used (and complains if it has).

The initial use of `\newcommand` is to catch if `\l@dmake@labels` has been previously defined (by a class or package).

```

3275 \newcommand*{\l@dmake@labels}{%
3276 \def\l@dmake@labels#1|#2|#3|#4|#5{%
3277     \expandafter\ifx\csname the@label#5\endcsname \relax\else
3278         \led@warn@DuplicateLabel{#5}%
3279     \fi
3280     \expandafter\gdef\csname the@label#5\endcsname{#1|#2|#3|#4}%
3281     \ignorespaces}
3282

```

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.

```

3283 \AtBeginDocument{%
3284     \def\l@dmake@labels#1|#2|#3|#4|#5{}%
3285 }
3286

```

`\@lab` The `\@lab` command, which appears in the `\linenum@out` file, appends the current values of page, line and sub-line to the `\labelref@list`. These values are defined by the earlier `\@page`, `\@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`.

```
3287 \newcommand*{\@lab}{\xright@appenditem
3288   {\linenumrep{\line@num}|%
3289     \ifsublines@ \sublinenumrep{\subline@num}\else 0\fi}\to\labelref@list}
3290
```

`\wrap@edcrossref` `\wrap@edcrossref` is called around all `eledmac` crossref commands, except those which start with `x`. It adds the hyperlink.

```
3291 \newrobustcmd{\wrap@edcrossref}[2]{%
3292   \ifdef{\hyperlink}%
3293     {\hyperlink{#1}{#2}}%
3294     {#2}%
3295 }
```

`\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`.

```
3296 \newcommand*{\edpageref}[1]{\l@dref@undefined{#1}\wrap@edcrossref{#1}{\l@dgetref@num{1}{#1}}}
3297 \newcommand*{\xpageref}[1]{\l@dgetref@num{1}{#1}}
3298
```

`\lineref` If the specified label exists, `\lineref` gives its line number.

```
\xlineref 3299 \newcommand*{\lineref}[1]{\l@dref@undefined{#1}\wrap@edcrossref{#1}{\l@dgetref@num{2}{#1}}}
3300 \newcommand*{\xlineref}[1]{\l@dgetref@num{2}{#1}}
3301
```

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

```
\xsublineref 3302 \newcommand*{\sublineref}[1]{\l@dref@undefined{#1}\wrap@edcrossref{#1}{\l@dgetref@num{3}{#1}}}
3303 \newcommand*{\xsublineref}[1]{\l@dgetref@num{3}{#1}}
3304
```

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

```
\xpstartref 3305 \newcommand*{\pstartref}[1]{\l@dref@undefined{#1}\wrap@edcrossref{#1}{\l@dgetref@num{4}{#1}}}
3306 \newcommand*{\xpstartref}[1]{\l@dgetref@num{4}{#1}}
3307
```

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.

```
3308 \newcommand*{\l@dref@undefined}[1]{%
3309   \expandafter\ifx\csname the@label#1\endcsname\relax
3310     \led@warn@RefUndefined{#1}%
3311   \fi}
3312
```

`\l@dgetref@num` Next, `\l@dgetref@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@dlabel@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.

```
3313 \newcommand*{\l@dgetref@num}[2]{%
3314   \expandafter
3315   \ifx\csname the@label#2\endcsname \relax
3316     000%
3317   \else
3318     \expandafter\expandafter\expandafter
3319     \l@dlabel@parse\csname the@label#2\endcsname|#1%
3320   \fi}
3321
```

`\l@dlabel@parse` Notice that we slipped another `|` delimiter into the penultimate line of `\l@dgetref@num`, to keep the ‘switch-number’ separate from the reference numbers. This `|` is used as another parameter delimiter by `\l@dlabel@parse`, which extracts the 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@dgetref@num`.)

```
3322 \newcommand*{\l@dlabel@parse}{%
3323   \def\l@dlabel@parse#1|#2|#3|#4|#5{%
3324     \ifcase #5%
3325       \or #1%
3326       \or #2%
3327       \or #3%
3328       \or #4%
3329     \fi}
```

`\xxref` 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.

```

3330 \newcommand*{\xxref}[2]{%
3331   {%
3332     \expandafter\ifx\csname the@label#1\endcsname \relax%
3333       \expandafter\let\csname the@@label#1\endcsname\zz@@@%
3334       \else%
3335         \expandafter\def\csname the@@label#1\endcsname{\l@getref@num{1}{#1}|\l@getref@num{2}{#1}|\l@getref@num{3}{#1}}%
3336       \fi%
3337     \expandafter\ifx\csname the@label#2\endcsname \relax%
3338       \expandafter\let\csname the@@label#2\endcsname\zz@@@%
3339     \else%
3340       \expandafter\def\csname the@@label#2\endcsname{\l@getref@num{1}{#2}|\l@getref@num{2}{#2}|\l@getref@num{3}{#2}}%
3341     \fi%
3342     \linenum{\csname the@@label#1\endcsname}%
3343     \csname the@@label#2\endcsname}}
3344

```

`\edmakelabel` Sometimes the `\edlabel` command cannot be used to specify exactly the page and line desired; you can use the `\edmakelabel` macro make your own label. For example, if you say ‘`\edmakelabel{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. `\edmakelabel` 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 `\edmakelabel`.

```

3345 \newcommand*{\edmakelabel}[2]{\expandafter\xdef\csname the@label#1\endcsname{#2}}
3346

```

(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.

`\l@dold@xympar` Changing `\@xympar` a little at least ensures that `\marginpars` in numbered text
`\@xympar` do not disturb the flow.

```

3347 \let\l@dold@xympar\@xympar
3348 \renewcommand{\@xympar}{%
3349   \ifnumberedpar@
3350     \led@warn@NoMarginpars
3351     \esphack
3352   \else
3353     \l@dold@xympar
3354   \fi}

```

3355

We provide side notes as replacement for `\marginpar` in numbered text.

`\sidenote@margin` These are the sidenote equivalents to `\line@margin` and `\linenummargin` for specifying which margin. The default is the right margin (opposite to the default for line numbers). `\l@dgetsidenote@margin` returns the number associated to side note margin:

left : 0

right : 1

outer : 2

inner : 3

```

3356 \newcount\sidenote@margin
3357 \newcommand*{\sidenotemargin}[1]{%
3358   \l@dgetsidenote@margin{#1}%
3359   \ifnum\@l@tempcntb>\m@ne
3360     \ifledRcol
3361       \global\sidenote@marginR=\@l@tempcntb
3362     \else
3363       \global\sidenote@margin=\@l@tempcntb
3364     \fi
3365   \fi}}
3366 \newcommand*{\l@dgetsidenote@margin}[1]{%
3367   \def\@tempa{#1}\def\@tempb{left}%
3368   \ifx\@tempa\@tempb
3369     \@l@tempcntb \z@
3370   \else
3371     \def\@tempb{right}%
3372     \ifx\@tempa\@tempb
3373       \@l@tempcntb \@ne
3374     \else
3375       \def\@tempb{outer}%
3376       \ifx\@tempa\@tempb
3377         \@l@tempcntb \tw@
3378       \else
3379         \def\@tempb{inner}%
3380         \ifx\@tempa\@tempb
3381           \@l@tempcntb \thr@@
3382         \else
3383           \led@warn@BadSidenotemargin
3384           \@l@tempcntb \m@ne
3385         \fi
3386       \fi
3387     \fi
3388   \fi}
3389 \sidenotemargin{right}
3390
```

`\l@dlp@rbox` We need two boxes to store sidenote texts.

```
\l@drp@rbox 3391 \newbox\l@dlp@rbox
              3392 \newbox\l@drp@rbox
              3393
```

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

```
\ledlsnotesep 3394 \newdimen\ledlsnotewidth \ledlsnotewidth=\marginparwidth
\ledrsnotesep 3395 \newdimen\ledrsnotewidth \ledrsnotewidth=\marginparwidth
\ledlsnotefontsetup 3396 \newdimen\ledlsnotesep \ledlsnotesep=\linenumsep
\ledrsnotefontsetup 3397 \newdimen\ledrsnotesep \ledrsnotesep=\linenumsep
                  3398 \newcommand*{\ledlsnotefontsetup}{\raggedleft\footnotesize}
                  3399 \newcommand*{\ledrsnotefontsetup}{\raggedright\footnotesize}
                  3400
```

`\ledleftnote` `\ledleftnote`, `\ledrightnote`, `\ledinnernote`, `\ledouternote` are the user
`\ledrightnote` commands for left, right, inner and outer sidenotes. The two last one are just
`\ledinnernote` alias for the two first one, depending of the page number. `\ledsidenote{<text>}`
`\ledouterote` is the command for a moveable sidenote.

```
\ledsidenote 3401 \newcommand*{\ledleftnote}[1]{\edtext{}{\l@dlsnote{#1}}}
              3402 \newcommand*{\ledrightnote}[1]{\edtext{}{\l@drsnote{#1}}}
              3403
              3404 \newcommand*{\ledinnernote}[1]{%
              3405   \ifodd\c@page% Do not use \page@num, because it is not yet calculated when command is called
              3406     \ledleftnote{#1}%
              3407   \else%
              3408     \ledrightnote{#1}%
              3409   \fi%
              3410 }
              3411
              3412 \newcommand*{\ledouternote}[1]{%
              3413   \ifodd\c@page% Do not use \page@num, because it is not yet calculated when command is called
              3414     \ledrightnote{#1}%
              3415   \else%
              3416     \ledleftnote{#1}%
              3417   \fi%
              3418 }
              3419
              3420 \newcommand*{\ledsidenote}[1]{\edtext{}{\l@dcsnote{#1}}}
```

`\l@dlsnote` . The ‘footnotes’ for left, right, and moveable sidenotes. The whole scheme is

`\l@drsnote` reminiscent of the critical footnotes code.

```
\l@dcsnote 3421 \newif\ifrightnoteup
              3422 \rightnoteuptrue
              3423
              3424 \newcommand*{\l@dlsnote}[1]{%
              3425   \begingroup%
              3426   \newcommand{\content}{#1}%
              3427   \ifnumberedpar@
```

```

3428 \ifledRcol%
3429 \xright@appenditem{\noexpand\vl@dlsnote{\expandonce\content}}%
3430 \to\inserts@listR
3431 \global\advance\insert@countR \@ne%
3432 \else%
3433 \xright@appenditem{\noexpand\vl@dlsnote{\expandonce\content}}%
3434 \to\inserts@list
3435 \global\advance\insert@count \@ne%
3436 \fi
3437 \fi\ignorespaces\endgroup}
3438
3439 \newcommand*{\l@drsnote}[1]{%
3440 \begingroup%
3441 \newcommand{\content}{#1}%
3442 \ifnumberedpar@
3443 \ifledRcol%
3444 \xright@appenditem{\noexpand\vl@drsnote{\expandonce\content}}%
3445 \to\inserts@listR
3446 \global\advance\insert@countR \@ne%
3447 \else%
3448 \xright@appenditem{\noexpand\vl@drsnote{\expandonce\content}}%
3449 \to\inserts@list
3450 \global\advance\insert@count \@ne%
3451 \fi
3452 \fi\ignorespaces\endgroup}
3453
3454 \newcommand*{\l@dcsnote}[1]{%
3455 \begingroup%
3456 \newcommand{\content}{#1}%
3457 \ifnumberedpar@
3458 \ifledRcol%
3459 \xright@appenditem{\noexpand\vl@dcsnote{\expandonce\content}}%
3460 \to\inserts@listR
3461 \global\advance\insert@countR \@ne%
3462 \else%
3463 \xright@appenditem{\noexpand\vl@dcsnote{\expandonce\content}}%
3464 \to\inserts@list
3465 \global\advance\insert@count \@ne%
3466 \fi
3467 \fi\ignorespaces\endgroup}
3468

```

`\vl@dlsnote` Put the left/right text into boxes, but just save the moveable text. `\l@dcsnotetext`, `\vl@drsnote` `\l@dcsnotetext@l` and `\l@dcsnotetext@r` are etoolbox lists which will store the content of side notes. We store the content in lists, because we need to loop later on them, in case many sidenote co-exist for the same line. That is there some special test to do, in order to:

- Store the content of `\ledsidenote` to `\l@dcsnotetext` in any cases.
- Store the content of `\rightsidenote` to:

- \l@dcsnotetext if \ledsidenote is to be put on right.
- \l@dcsnotetext@r if \ledsidenote is to be put on left.
- Store the content of \leftsidenote to:
 - \l@dcsnotetext if \ledsidenote is to be put on left.
 - \l@dcsnotetext@l if \ledsidenote is to be put on right.

```

3469 \newcommand*{\vl@dlsnote}[1]{%
3470   \ifledRcol@%
3471     \@l@tempcntb=\sidenote@marginR%
3472     \ifnum\@l@tempcntb>\@ne%
3473       \advance\@l@tempcntb by\page@numR%
3474     \fi%
3475   \else%
3476     \@l@tempcntb=\sidenote@margin%
3477     \ifnum\@l@tempcntb>\@ne%
3478       \advance\@l@tempcntb by\page@num%
3479     \fi%
3480   \fi%
3481   \ifodd\@l@tempcntb%
3482     \listgadd{\l@dcsnotetext@l}{#1}%
3483   \else%
3484     \listgadd{\l@dcsnotetext}{#1}%
3485   \fi
3486 }
3487 \newcommand*{\vl@drsnote}[1]{%
3488   \ifledRcol@%
3489     \@l@tempcntb=\sidenote@marginR%
3490     \ifnum\@l@tempcntb>\@ne%
3491       \advance\@l@tempcntb by\page@numR%
3492     \fi%
3493   \else%
3494     \@l@tempcntb=\sidenote@margin%
3495     \ifnum\@l@tempcntb>\@ne%
3496       \advance\@l@tempcntb by\page@num%
3497     \fi%
3498   \fi%
3499   \ifodd\@l@tempcntb%
3500     \listgadd{\l@dcsnotetext}{#1}%
3501   \else%
3502     \listgadd{\l@dcsnotetext@r}{#1}%
3503   \fi%
3504 }
3505 \newcommand*{\vl@dcsnote}[1]{\listgadd{\l@dcsnotetext}{#1}}
3506

```

\setl@dlp@rbox \setl@dlprbox{<lednums>}{<tag>}{<text>} puts <text> into the \l@dlp@rbox box.
 \setl@drpr@box And similarly for the right side box. It is these boxes that finally get displayed in the margins.

```

3507 \newcommand*{\setl@dlp@rbox}[1]{%
3508   {\parindent\z@\hspace=\ledlsnotewidth\ledlsnotefontsetup
3509     \global\setbox\l@dlp@rbox
3510     \ifleftnoteup
3511       =\vbox to\z@{\vss #1}%
3512     \else
3513       =\vbox to 0.70\baselineskip{\strut#1\vss}%
3514     \fi}}
3515 \newcommand*{\setl@drp@rbox}[1]{%
3516   {\parindent\z@\hspace=\ledrsnotewidth\ledrsnotefontsetup
3517     \global\setbox\l@drp@rbox
3518     \ifrightrightnoteup
3519       =\vbox to\z@{\vss#1}%
3520     \else
3521       =\vbox to0.7\baselineskip{\strut#1\vss}%
3522     \fi}}
3523 \newif\ifleftnoteup
3524 \leftnoteuptrue

```

`\sidenotesep` This macro is used to separate sidenotes of the same line.

```

3525 \newcommand{\sidenotesep}{, }
3526 % \end{macrocode}
3527 % \end{macro}
3528 % \begin{macro}{\affixside@note}
3529 % This macro puts any moveable sidenote text into the left or right sidenote
3530 % box, depending on which margin it is meant to go in. It's a very much
3531 % stripped down version of \cs{affixlin@num}.
3532 %
3533 % Before do it, we concatenate all moveable sidenotes of the line, using \cs{sidenotesep}
3534 % It's the result that we put on the sidenote.
3535 % \changes{v1.4.1}{2012/11/16}{Remove spurious spaces.}
3536 % \begin{macrocode}
3537 \newcommand*{\affixside@note}{%
3538   \def\sidenotecontent@{%
3539     \numgdef{\itemcount@}{0}%
3540     \def\do##1{%
3541       \ifnumequal{\itemcount@}{0}%
3542         {%
3543           \appto\sidenotecontent@{##1}}% Not print not separator before the 1st note
3544         {\appto\sidenotecontent@{\sidenotesep ##1}%
3545         }%
3546       \numgdef{\itemcount@}{\itemcount@+1}%
3547     }%
3548     \dolistloop{\l@dcstotetext}%
3549     \ifnumgreater{\itemcount@}{1}{\led@err@ManySidenotes}{}%

```

And we do the same for left and right notes (not movable).

```

3550 \gdef\@templ@d{%
3551 \gdef\@templ@n{\l@dcstotetext\l@dcstotetext@1\l@dcstotetext@r}%
3552 \ifx\@templ@d\@templ@n \else%

```



```

3553 \if@twocolumn%
3554 \if@firstcolumn%
3555 \setl@dlp@rbox{##1}{\sidenotecontent@}%
3556 \else%
3557 \setl@drp@rbox{\sidenotecontent@}%
3558 \fi%
3559 \else%
3560 \l@dttempcntb=\sidenote@margin%
3561 \ifnum\l@dttempcntb>\@ne%
3562 \advance\l@dttempcntb by\page@num%
3563 \fi%
3564 \ifodd\l@dttempcntb%
3565 \setl@drp@rbox{\sidenotecontent@}%
3566 \gdef\sidenotecontent@{}%
3567 \numgdef{\itemcount@}{0}%
3568 \dolistloop{\l@dcnotetext@l}%
3569 \ifnumgreater{\itemcount@}{1}{\led@err@ManyLeftnotes}{}%
3570 \setl@dlp@rbox{\sidenotecontent@}%
3571 \else%
3572 \setl@dlp@rbox{\sidenotecontent@}%
3573 \gdef\sidenotecontent@{}%
3574 \numgdef{\itemcount@}{0}%
3575 \dolistloop{\l@dcnotetext@r}%
3576 \ifnumgreater{\itemcount@}{1}{\led@err@ManyRightnotes}{}%
3577 \setl@drp@rbox{\sidenotecontent@}%
3578 \fi%
3579 \fi%
3580 \fi%
3581 }

```

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 `\@iiminipage` and `\endminipage` macros. We'll arrange this so that additional series can be easily added.

`\l@dfeetbeginmini` These will be the hooks in `\@iiminipage` and `\endminipage` They can be extended
`\l@dfeetendmini` to handle other things if necessary.

```

3582 \newcommand*{\l@dfeetbeginmini}{\l@dedbeginmini\l@dfambeginmini}
3583 \newcommand*{\l@dfeetendmini}{%
3584 \IfStrEq{critical-familiar}{\@mpfnpos}%
3585 {\l@dedendmini\l@dfamendmini}%
3586 {%
3587 \IfStrEq{familiar-critical}{\@mpfnpos}%
3588 {\l@dfamendmini\l@dedendmini}%
3589 {\l@dedendmini\l@dfamendmini}%

```

```

3590     }%
3591 }%

```

`\l@dedbeginmini` These handle the initiation and closure of critical footnotes in a minipage environment.
`\l@dedendmini`

```

3592 \newcommand*\l@dedbeginmini{%
3593   \def\do##1{\csletcs{v##1footnote}{mpv##1footnote}}%
3594   \dolistloop{\@series}%
3595 }
3596 \newcommand*\l@dedendmini{%
3597   \ifl@dpairing
3598     \ifledRcol
3599       \flush@notesR
3600     \else
3601       \flush@notes
3602     \fi
3603   \fi
3604   \def\do##1{
3605     \ifvoid\csuse{mp##1footins}\else%
3606       \ifl@dpairing\ifparledgroup%
3607         \ifledRcol%
3608           \dimgdef{\parledgroup@beforenotesR}{\parledgroup@beforenotesR+\skip\@nameuse{mpv##1footnote}}%
3609         \else%
3610           \dimgdef{\parledgroup@beforenotesL}{\parledgroup@beforenotesL+\skip\@nameuse{mpv##1footnote}}%
3611         \fi%
3612       \fi\fi%
3613       \csuse{mp##1footgroup}{##1}%
3614     \fi}%
3615   \dolistloop{\@series}%
3616 }
3617

```

`\l@dfambeginmini` These handle the initiation and closure of familiar footnotes in a minipage environment.
`\l@dfamendmini`

```

3618 \newcommand*\l@dfambeginmini{%
3619   \def\do##1{\csletcs{vfootnote##1}{mpvfootnote##1}}%
3620   \dolistloop{\@series}%
3621 \newcommand*\l@dfamendmini{%
3622   \def\do##1{\ifvoid\csuse{mpfootins##1}\else\csuse{mpfootgroup##1}{##1}\fi}%
3623   \dolistloop{\@series}%

```

`\@iiiminipage` This is our extended form of the kernel `\@iiiminipage` defined in `ltboxes.dtx`.

```

3624 \def\@iiiminipage#1#2[#3]#4{%
3625   \leavevmode
3626   \@pboxswfalse
3627   \setlength\@tempdima{#4}%
3628   \def\@mpargs{#1}{#2}[#3]{#4}}%
3629   \setbox\@tempboxa\vbox\bgroup

```

```

3630 \color@begingroup
3631 \hsize\@tempdima
3632 \textwidth\hsize \columnwidth\hsize
3633 \@parboxrestore
3634 \def\@mpfn{mpfootnote}\def\thempfn{\thempfootnote}\c@mpfootnote\z@
3635 \let\@footnotetext\@mpfootnotetext

```

The next line is our addition to the original.

```

3636 \l@dfetbeginmini% added
3637 \let\@listdepth\@mplistdepth \@mplistdepth\z@
3638 \@minipagerestore
3639 \@setminipage}
3640

```

`\endminipage` This is our extended form of the kernel `\endminipage` defined in `ltboxes.dtx`.

```

3641 \def\endminipage{%
3642 \par
3643 \unskip
3644 \ifvoid\@mpfootins\else
3645 \l@dunboxmpfoot
3646 \fi

```

The next line is our addition to the original.

```

3647 \l@dfetendmini% added
3648 \@minipagefalse
3649 \color@endgroup
3650 \egroup
3651 \expandafter\@iiiparbox\@mpargs{\unvbox\@tempboxa}}
3652

```

`\l@dunboxmpfoot`

```

3653 \newcommand*{\l@dunboxmpfoot}{%
3654 \vskip\skip\@mpfootins
3655 \normalcolor
3656 \footnoterule
3657 \ifparledgroup
3658 \ifl@dpairing
3659 \ifledRcol
3660 \dimgdef{\parledgroup@beforenotesR}{\parledgroup@beforenotesR+\skip\@mpfootins}
3661 \else
3662 \dimgdef{\parledgroup@beforenotesL}{\parledgroup@beforenotesL+\skip\@mpfootins}
3663 \fi
3664 \fi
3665 \fi
3666 \unvbox\@mpfootins}
3667

```

`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.

```

3668 \newenvironment{ledgroup}{%
3669   \resetprevpage@num%
3670   \def\@mpfn{mpfootnote}\def\thempfn{\thempfootnote}\c@mpfootnote\z@%
3671   \let\@footnotetext\@mpfootnotetext
3672   \l@dfetbeginmini%
3673 }{%
3674   \par
3675   \unskip
3676   \ifvoid\@mpfootins\else
3677     \l@dunboxmpfoot
3678   \fi
3679   \l@dfetendmini%
3680 }
3681

```

ledgroupsize `\begin{ledgroupsize}[\langle pos \rangle]{\langle width \rangle}`

This environment puts footnotes at the end, even if that happens to be in the middle of a page, or crossing a page boundary. It is a sort of unboxed, variable $\langle width \rangle$ minipage. The optional $\langle pos \rangle$ controls the sideways position of numbered text.

```

3682 \newenvironment{ledgroupsize}[2][l]{%

```

Set the various text measures.

```

3683   \hsize #2\relax
3684   %% \textwidth #2\relax
3685   %% \columnwidth #2\relax

```

Initialize fills for centering.

```

3686   \let\ledllfill\hfil
3687   \let\ledrlfill\hfil
3688   \def\@tempa{#1}\def\@tempb{1}%

```

Left adjusted numbered lines

```

3689     \ifx\@tempa\@tempb
3690     \let\ledllfill\relax
3691   \else
3692     \def\@tempb{r}%
3693     \ifx\@tempa\@tempb

```

Right adjusted numbered lines

```

3694     \let\ledrlfill\relax
3695   \fi
3696   \fi

```

Set up the footnoting.

```

3697   \def\@mpfn{mpfootnote}\def\thempfn{\thempfootnote}\c@mpfootnote\z@
3698   \let\@footnotetext\@mpfootnotetext
3699   \l@dfetbeginmini%
3700 }{%
3701   \par
3702   \unskip

```

```

3703 \ifvoid\@mpfootins\else
3704   \l@dunboxmpfoot
3705 \fi
3706 \l@dfeetendmini%
3707 }
3708

```

`\ifledgroupnotesL@` These boolean tests check if we are in the notes of a ledgroup. If we are, we don't
`\ifledgroupnotesR@` number the lines.

```

3709 \newif\ifledgroupnotesL@
3710 \newif\ifledgroupnotesR@

```

36 Indexing

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

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

```

3711 \AtBeginDocument{%
3712   \unless\ifl@imakeidx%
3713     \@ifpackageloaded{imakeidx}{\led@error@ImakeidxAfterEledmac}{}%
3714   \fi%
3715   \unless\ifl@indextools%
3716     \@ifpackageloaded{indextools}{\led@error@indextoolsAfterEledmac}{}%
3717   \fi%
3718 }

```

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

```

\c@labidx 3719 \newcommand{\pagelinesep}{-}
          3720 \newcommand{\edindexlab}{\&\&}
          3721 \newcounter{labidx}
          3722 \setcounter{labidx}{0}
          3723

```

`\doedindexlabel` This macro sets an `\edlabel`.

```

3724 \newcommand{\doedindexlabel}{\stepcounter{labidx}%
3725   \edlabel{\edindexlab\thelabidx}}
3726

```

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

```

3727 \newcommand{\thepageline}{%
3728   \thepage\pagelinesep\xlineref{\edindexlab\thelabidx}}

```

`\thestartpageline` These macros make up the page/line start/end number when the `\edindex` com-
`\theendpageline` mand is called in critical notes.

```

3729 \newcommand{\thestartpageline}{\l@dparsedstartpage\pagelinesep\l@dparsedstartline}
3730 \newcommand{\theendpageline}{\l@dparsedendpage\pagelinesep\l@dparsedendline}

```

`\if@edindex@fornote@true` This boolean test is switching at the beginning of each critical note, to allow indexing in this note.

```
3731 \newif\if@edindex@fornote@
```

`\prepare@edindex@fornote` This macro is called at the beginning of each critical note. It switches some parameters, to allow indexing in this note, with reference to page and line number.

```
3732 \newcommand{\prepare@edindex@fornote}[1]{%
3733   \l@dp@rsefootspec#1|%
3734   \@edindex@fornote@true
3735 }
```

`\get@index@command` This macro is used to analyse if a text to be indexed has a command after a |.

```
3736 \def\get@index@command#1|#2+{%
3737   \gdef\@index@txt{#1}%
3738   \gdef\@index@command{#2}%
3739   \xdef\@index@parenthesis{}%
3740   \IfBeginWith{\@index@command}{(}{%
3741     \StrGobbleLeft{\@index@command}{1}[\@index@command]%
3742     \global\let\@index@command\@index@command%
3743     \xdef\@index@parenthesis{(%}%
3744   }{%
3745     \IfBeginWith{\@index@command}{)}{%
3746       \StrGobbleLeft{\@index@command}{1}[\@index@command]%
3747       \global\let\@index@command\@index@command%
3748       \xdef\@index@parenthesis{)%}%
3749     }{%
3750   }
```

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

```
\ledinnotehyperpage 3751 \newcommand{\ledinnote}[2]{\csuse{#1}{#2\emph{n}}}%
3752 \newcommand{\ledinnotehyperpage}[2]{\csuse{#1}{\hyperpage{#2}\emph{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.

`\edindex` 36.1 Memoir compatibility

`\create@edindex@for@memoir` `\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.

```

3753 \def\create@edindex@for@memoir{
3754   \g@addto@macro{\makememindexhook}{%
3755     \def\edindex{\@bsphack%
3756       \@ifnextchar [{\l@d@index}{\l@d@index[\jobname]}}
3757     \newcommand{\edindex}[2][\jobname]{\@bsphack\@esphack}

```

`\l@d@index` `\l@d@index[file]` is the first stage of `\edindex`, handling the `idx` file. This is a virtually a verbatim copy of memoir's `\@index`, the change being calling `\l@dwrindexm@m` instead of `\@wrindexm@m`.

```

3758 \def\l@d@index[##1]{%
3759   \@ifundefined{##1|idxfile}%
3760   {\ifreportnoidxfile
3761     \led@warn@NoIndexFile{##1}%
3762     \fi
3763     \begingroup
3764     \@sanitize
3765     \@nowrindex}%
3766   {\def\@idxfile{##1}%
3767     \doedindexlabel
3768     \begingroup
3769     \@sanitize
3770     \l@d@wrindexm@m}}

```

`\l@d@wrindexm@m` `\l@d@wrindexm@m{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 `\@wrindexm@m` and `\@wrindexhyp`.

```

3771 \newcommand{\l@d@wrindexm@m}[1]{\l@d@wrindexhyp##1||\}
3772 \def\l@d@wrindexhyp##1|##2|##3\{%
3773   \ifshowindexmark\@showidx{##1}\fi
3774   \ifx\##2\%
3775     \if@edindex@for@note@%
3776       \protected@write\@auxout{%
3777         {\string\@wrindexm@m{\@idxfile}{##1}(\ledinnotehyperpage){\thestartpageline}}%
3778         \protected@write\@auxout{%
3779           {\string\@wrindexm@m{\@idxfile}{##1})ledinnotehyperpage}{\theendpageline}}%
3780       \else%
3781         \protected@write\@auxout{%
3782           {\string\@wrindexm@m{\@idxfile}{##1}hyperpage}{\thepageline}}%
3783       \fi%
3784     \else
3785       \def\Hy@temp@A{##2}%
3786       \ifx\Hy@temp@A\HyInd@ParenLeft
3787         \if@edindex@for@note@%
3788           \protected@write\@auxout{%
3789             {\string\@wrindexm@m{\@idxfile}{##1}(\ledinnotehyperpage{##2}){\thestartpageline}}%

```

```

3790         \protected@write\@auxout{}%
3791         {\string\@wrindexm@m{\@idxfile}{##1|)\ledinnotehyperpage{##2}}{\theendpageline}}%
3792     \else%
3793         \protected@write\@auxout{}%
3794         {\string\@wrindexm@m{\@idxfile}{##1|##2hyperpage}{\thepageline}}%
3795     \fi%
3796 \else
3797     \if@edindex@fornote@%
3798         \protected@write\@auxout{}%
3799         {\string\@wrindexm@m{\@idxfile}{##1|(\ledinnote{##2}}{\thestartpageline}}}%
3800         \protected@write\@auxout{}%
3801         {\string\@wrindexm@m{\@idxfile}{##1|)\ledinnote{##2}}{\theendpageline}}}%
3802     \else%
3803         \protected@write\@auxout{}%
3804         {\string\@wrindexm@m{\@idxfile}{##1|##2}{\thepageline}}%
3805     \fi%
3806 \fi
3807 \fi
3808 \endgroup
3809 \esphack}

```

This finishes the memoir-specific code.

```
3810 }
```

36.2 Normal setting

`\create@edindex@notfor@memoir` `\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.

```
3811 \def\create@edindex@notfor@memoir{
```

`\@wredindex` Write the index information to the `idx` file.

```

3812 \newcommandx{\@wredindex}[2][1=\expandonce\jobname,usedefault]{%#1 = the index name, #2
3813 \global\let\old@Rlineflag\Rlineflag%
3814 \gdef\Rlineflag{}%
3815 \ifl@imakeidx%
3816     \if@edindex@fornote@%
3817         \IfSubStr[1]{##2}{|}{\get@index@command##2+}{\get@index@command##2|}%
3818         \expandafter\imki@wrindexentry{##1}{\@index@txt|(\ledinnote{\@index@command}}{\the
3819         \expandafter\imki@wrindexentry{##1}{\@index@txt|)\ledinnote{\@index@command}}{\the
3820     \else%
3821         \get@edindex@hyperref{##2}%
3822         \imki@wrindexentry{##1}{\@index@txt\@edindex@hyperref}{\thepageline}%
3823     \fi%

```



```

3824 \else%
3825 \if@edindex@fornote%
3826 \IfSubStr[1]{##2}{|}{\get@index@command##2+}{\get@index@command##2|+}%
3827 \expandafter\protected@write\@indexfile{%
3828 {\string\indexentry{\@index@txt|(ledinnote{\@index@command}){\thestartpageline}
3829 }%
3830 \expandafter\protected@write\@indexfile{%
3831 {\string\indexentry{\@index@txt|)ledinnote{\@index@command}){\theendpageline}
3832 }%
3833 \else%
3834 \protected@write\@indexfile{%
3835 {\string\indexentry{##2}{\thepageline}
3836 }%
3837 \fi%
3838 \fi%
3839 \endgroup
3840 \global\let\Rlineflag\old\Rlineflag%
3841 \@esphack}

```

Need to add the definition of `\edindex` to `\makeindex`, and initialise `\edindex` to do nothing.

```

3842 \pretocmd{\makeindex}{%
3843 \def\edindex{\@bsphack
3844 \doedindexlabel
3845 \begingroup
3846 \@sanitize
3847 \wredindex}}{}{}
3848 \newcommand{\edindex}[1]{\@bsphack\@esphack}

3849 % That finishes the non-\Lpack{memoir} index code.
3850 }

```

36.3 Choose the right variant

Then call `\create@edindex@for@memoir` or `\create@edindex@notfor@memoir` depending on the use of `memoir` and `imakeidx`

```

3851 \ifclassloaded{memoir}{%
3852 \ifpackageloaded{imakeidx}%
3853 {\create@edindex@notfor@memoir}%
3854 {%
3855 \ifpackageloaded{indextools}%
3856 {\create@edindex@notfor@memoir}%
3857 {\create@edindex@for@memoir}%
3858 }%
3859 }%
3860 {\create@edindex@notfor@memoir}%

```

36.4 Hyperref compatibility

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

```
3861 \newcommand{\hyperlinkformat}[3]{%
3862   \ifstrempy{#1}%
3863     {\hyperlink{#2}{#3}}%
3864     {\csuse{#1}{\hyperlink{#2}{#3}}}%
3865 }
```

`\hyperlinkR` `\hyperlinkR` command is to be used to create a internal hyperlink and `\ledRflag`, when indexing.

```
3866 \newcommand{\hyperlinkR}[2]{%
3867   \hyperlink{#1}{#2\Rlineflag}%
3868 }%
3869
```

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

```
3870 \newcommand{\hyperlinkformatR}[3]{%
3871   \hyperlinkformat{#1}{#2}{#3\Rlineflag}%
3872 }%
3873
```

`\get@edindex@hyperref` `\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`).

```
3874 \newcommand{\get@edindex@hyperref}[1]{%
3875   \ifdef{\hyperlink}%
3876     {%
3877       \edef\temp@{%
3878         \catcode'\ =9 %space need for catcode
3879         #1%
3880         \catcode'\ =10 % space need for catcode
3881       }%
3882       \IfSubStr{\temp@}{|}%
3883         {\get@index@command#1+%
3884          \ifledRcol%
3885            \gdef\@edindex@hyperref{|\@index@parenthesis %space kept
3886             hyperlinkformatR{\@index@command}%
3887             {\edindexlab\thelabidx}}%
3888          \else%
3889            \gdef\@edindex@hyperref{|\@index@parenthesis %space kept
3890             hyperlinkformat{\@index@command}%
3891             {\edindexlab\thelabidx}}%
3892          \fi%
3893        }%
3894        {\get@index@command#1|+}
```

```

3895     \ifledRcol%
3896     \gdef\@edindex@hyperref{|\hyperlinkR{\edindexlab\thelabidx}}%
3897     \else%
3898     \gdef\@edindex@hyperref{|\hyperlink{\edindexlab\thelabidx}}%
3899     \fi%
3900   }%
3901 }%
3902 {%
3903 \gdef\@index@txt{#1}%
3904 \gdef\@edindex@hyperref{}}%
3905 }

```

37 Macro as environment

The following is borrowed, and renamed, from the `amsmath` package. See also the CTT thread ‘`eq and amstex`’, 1995/08/31, started by Keith Reckdahl and ended definitively by David M. Jones.

Several of the `[math]` macros scan their body twice. This means we must collect all text in the body of an environment form before calling the macro.

`\@emptytoks` This is actually defined in the `amsgen` package.

```

3906 \newtoks\@emptytoks
3907

```

The rest is from `amsmath`.

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

```

3908 \newtoks\l@denvbody
3909

```

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

```

3910 \newcommand{\addtol@denvbody}[1]{%
3911   \global\l@denvbody\expandafter{\the\l@denvbody#1}}
3912

```

`\l@dcollect@body` The macro `\l@dcollect@body` starts the scan for the `\end{...}` command of the current environment. It takes a macro name as argument. This macro is supposed to take the whole body of the environment as its argument. For example, given `cenv#1{...}` as a macro that processes `#1`, then the environment form, `\begin{env}` would call `\l@dcollect@body\cenv`.

```

3913 \newcommand{\l@dcollect@body}[1]{%
3914   \l@denvbody{\expandafter#1\expandafter{\the\l@denvbody}}%
3915   \edef\processl@denvbody{\the\l@denvbody\noexpand\end{\@currenenv}}%
3916   \l@denvbody\@emptytoks \def\l@dbegin@stack{b}%
3917   \begingroup
3918     \expandafter\let\csname\@currenenv\endcsname\l@dcollect@body
3919     \edef\processl@denvbody{\expandafter\noexpand\csname\@currenenv\endcsname}%

```

```

3920 \processl@denbody%
3921 }%
3922

```

`\l@dpush@begins` When adding a piece of the current environment's contents to `\l@denbody`, we scan it to check for additional `\begin` tokens, and add a 'b' to the stack for any that we find.

```

3923 \def\l@dpush@begins#1\begin#2{%
3924 \ifx\end#2\else b\expandafter\l@dpush@begins\fi}
3925

```

`\l@dcollect@@body` `\l@dcollect@@body` takes two arguments: the first will consist of all text up to the next `\end` command, and the second will be the `\end` command's argument. If there are any extra `\begin` commands in the body text, a marker is pushed onto a stack by the `\l@dpush@begins` function. Empty state for this stack means we have reached the `\end` that matches our original `\begin`. Otherwise we need to include the `\end` and its argument in the material we are adding to the environment body accumulator.

```

3926 \def\l@dcollect@@body#1\end#2{%
3927 \edef\l@dbegin@stack{\l@dpush@begins#1\begin\end
3928 \expandafter\@gobble\l@dbegin@stack}%
3929 \ifx\@empty\l@dbegin@stack
3930 \endgroup
3931 \@checkend{#2}%
3932 \addtol@denbody{#1}%
3933 \else
3934 \addtol@denbody{#1\end{#2}}%
3935 \fi
3936 \processl@denbody % A little tricky! Note the grouping
3937 }
3938

```

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}}%
  \collect@body\next@
}{%

% ignore spaces at begin and end of environment
\newenvironment{coloredboxII}[1]{%
  \def\next@{\mycoloredbox{#1}}%
  \collect@body\next@
}{%
\newcommand{\mycoloredbox}[2]{%
  \colorbox{#1}{\ignorespaces#2\unskip}%
}

% support of optional color model argument
\newcommand\coloredboxIII\endcsname{}
\def\coloredboxIII#1#2{%
  \@coloredboxIII{#1}%
}
\def\@coloredboxIII#1#2{%
  \def\next@{\mycoloredboxIII{#1}{#2}}%
  \collect@body\next@
}
\newcommand{\mycoloredboxIII}[3]{%
  \colorbox{#1}{#2}{\ignorespaces#3\unskip}%
}
```

```

\makeatother

\begin{document}
  Black text before
  \begin{coloredbox}{blue}
    Hello World
  \end{coloredbox}
  Black text after

  Black text before
  \begin{coloredboxII}{blue}
    Hello World
  \end{coloredboxII}
  Black text after

  Black text before
  \begin{coloredboxIII}[rgb]{0,0,1}
    Hello World
  \end{coloredboxIII}
  Black text after

\end{document}

Yours sincerely
Heiko <oberdiek@uni-freiburg.de>

```

38 Verse

This is principally Wayne Sullivan's code and commentary from EDSTANZA [Sul92].

The macro `\hangingsymbol` is used to insert a symbol on each hanging of verses. For example, in french typographie the symbol is ']. We obtain it by the next code:

```
\renewcommand{\hangingsymbol}{[,}
```

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

```

\hangingsymbol
\ifinstanza 3939 \newcommand*{\hangingsymbol}{}
            3940 \newif\ifinstanza
            3941 \instanzafalse

\inserthangingsymbol The boolean \inserthangingsymbol is set to TRUE when \@lock is greater
\ifinserthangingsymbol than 1, i.e. when we are not in the first line of a verse. The switch of
\ifinserthangingsymbol \inserthangingsymbol is made in \do@line before the printing of line but
                        after the line number calculation.
3942 \newif\ifinserthangingsymbol

```

```

3943 \newcommand{\inserthangingsymbol}{%
3944 \ifinserthangingsymbol%
3945   \ifinstanza%
3946     \hangingsymbol%
3947   \fi%
3948 \fi%
3949 }

```

\ampersand Within a stanza the `\&` macro is going to be usurped. We need an alias in case an `&` needs to be typeset in a stanza. Define it rather than letting it in case some other package has already defined it.

```

3950 \newcommand*{\ampersand}{\char'\&}
3951

```

\stanza@count Before we can define the main macros we need to save and reset some category codes. To save the current values we use `\next` and `\body` from the `\loop` macro.

```

3952 \chardef\body=\catcode'\@
3953 \catcode'\@=11
3954 \chardef\next=\catcode'\&
3955 \catcode'\&=\active
3956

```

A count register is allocated for counting lines in a stanza; also allocated is a dimension register which is used to specify the base value for line indentation; all stanza indentations are multiples of this value. The default value of `\stanzaindentbase` is 20pt.

```

3957 \newcount\stanza@count
3958 \newlength{\stanzaindentbase}
3959 \setlength{\stanzaindentbase}{20pt}
3960

```

\strip@szacnt The indentations of stanza lines are non-negative integer multiples of the unit called `\stanzaindentbase`. To make it easier for the user to specify these numbers, some list macros are defined. These take numerical values in a list separated by commas and assign the values to special control sequences using `\mathchardef`. Though this does limit the range from 0 to 32767, it should suffice for most applications, including *penalties*, which will be discussed below.

```

3961 \def\strip@szacnt#1,#2|{\def\@tempb{#1}\def\@tempa{#2|}}
3962 \newcommand*{\setstanzavalues}[2]{\def\@tempa{#2,,|}%
3963   \stanza@count\z@
3964   \def\next{\expandafter\strip@szacnt\@tempa
3965     \ifx\@tempb\empty\let\next\relax\else
3966     \expandafter\mathchardef\csname #1@\number\stanza@count
3967     \@endcsname\@tempb\relax
3968     \advance\stanza@count\@ne\fi\next}%
3969   \next}
3970

```

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

```

3971 \newcommand*\setstanzaindents[1]{\setstanzavalues{sza}{#1}}
3972 \newcommand*\setstanzapenalties[1]{\setstanzavalues{szp}{#1}}
3973
3974 \newcounter{stanzaindentsrepetition}
3975 \newcount\stanza@modulo
3976
3977 \newcommand*\managestanza@modulo[0]{
3978     \advance\stanza@modulo\@ne
3979     \ifnum\stanza@modulo>\value{stanzaindentsrepetition}
3980         \stanza@modulo\@ne
3981     \fi
3982 }
```

`\stanzaindent` The macro `\stanzaindent`, when called at the beginning of a verse, change the indentation normally defined for this verse by `\setstanzaindent`. The starred version skip the current verse for the repetition of stanza indent.

```

3983 \newcommand{\stanzaindent}[1]{%
3984     \hspace{\dimexpr#1\stanzaindentbase-\parindent\relax}%
3985     \ignorespaces%
3986 }%
3987 \WithSuffix\newcommand\stanzaindent*[1]{%
3988     \stanzaindent{#1}%
3989     \global\advance\stanza@modulo-\@ne%
3990     \ifnum\stanza@modulo=0%
3991         \global\stanza@modulo=\value{stanzaindentsrepetition}%
3992     \fi%
3993     \ignorespaces%
3994 }%
```

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

`\stanza@hang` `\stanza@hang` sets the hanging indentation to be used if the stanza line requires more than one print line. If it is known that each stanza line will fit on one print line, it is advisable to set the hanging indentation to zero. `\sza@penalty` places the specified penalty following each stanza line. By default, this facility is turned off so that no penalty is included. However, the user may initiate these penalties to indicate good and bad places in the stanza for page breaking.

```

3995 \newcommandx{\stanza@line}[1][1]{
3996     \ifnum\value{stanzaindentsrepetition}=0
3997         \parindent=\cename{sza@number}\stanza@count
```



```

3998             @\endcsname\stanzaindentbase
3999     \else
4000         \parindent=\csname sza@\number\stanza@modulo
4001             @\endcsname\stanzaindentbase
4002         \managestanza@modulo
4003     \fi
4004     \pstart[#1]\stanza@hang\ignorespaces}
4005 \xdef\stanza@hang{\noexpand\leavevmode\noexpand\startlock
4006     \hangindent\expandafter
4007     \noexpand\csname sza@0@\endcsname\stanzaindentbase
4008     \hangafter\@ne}
4009 \def\sza@penalty{\count@\csname szp@\number\stanza@count @\endcsname
4010     \ifnum\count@>\@M\advance\count@-\@M\penalty-\else
4011     \penalty\fi\count@}

```

Now we have the components of the `\stanza` macro, which appears at the start of a group of lines. This macro initializes the count and checks to see if hanging indentation and penalties are to be included. Hanging indentation suspends the line count, so that the enumeration is by verse line rather than by print line. If the print line count is desired, invoke `\let\startlock=\relax` and do the same for `\endlock`. Here and above we have used `\xdef` to make the stored macros take up a bit less space, but it also makes them more obscure to the reader. Lines of the stanza are delimited by ampersands `&`. The last line of the stanza must end with `&`. For convenience the macro `\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 beginning of a stanza. This can be defined to do something useful.

```

4012 \let\startstanzahook\relax
4013 \let\endstanzaextra\relax
4014 \xdef\@startstanza[#1]{%
4015     \noexpand\instanzatrue\expandafter
4016     \begingroup\startstanzahook%
4017     \catcode'\noexpand\&\active%
4018     \global\stanza@count\@ne\stanza@modulo\@ne
4019     \noexpand\ifnum\expandafter\noexpand
4020     \csname sza@0@\endcsname=z@\let\noexpand\stanza@hang\relax
4021     \let\noexpand\endlock\relax\noexpand\else\interlinepenalty
4022     \@M\rightskip\z@ plus 1fil\relax\noexpand\fi\noexpand\ifnum
4023     \expandafter\noexpand\csname szp@0@\endcsname=z@
4024     \let\noexpand\sza@penalty\relax\noexpand\fi%
4025     \def\noexpand\falseverse{%
4026         \noexpand\led@war@FalseverseDeprecated%
4027         \global\advance\stanza@modulo-\@ne%
4028         \global\advance\stanza@count-\@ne%
4029         \relax\noexpand&\leavevmode\skipnumbering}
4030     \def\noexpand&{%
4031         \noexpand\newverse [] []}%
4032     \def\noexpand\&{\noexpand\@stopstanza}%
4033     \noexpand\stanza@line[#1]}

```

```

4034
4035 \newcommandx{\stanza}[1][1,usedefault]{\@startstanza[#1]}
4036
4037 \newcommandx{\@stopstanza}[1][1,usedefault]{%
4038   \endlock%
4039   \pend[#1]%
4040   \endgroup%
4041   \instanzafalse%
4042   \endstanzaextra%
4043 }
4044
4045 \newcommand*{\newverse}[2][1,2,usedefault]{%
4046   \endlock\pend[#1]\sza@penalty\global%
4047   \advance\stanza@count\@ne\stanza@line[#2]%
4048 }
4049

```

`\flagstanza` Use `\flagstanza[len]{text}` at the start of a line to put *text* a distance *len* before the start of the line. The default for *len* is `\stanzaindentbase`.

```

4050 \newcommand*{\flagstanza}[2][\stanzaindentbase]{%
4051   \hskip -#1\llap{#2}\hskip #1\ignorespaces}
4052

```

The ampersand & is used to mark the end of each stanza line, except the last, which is marked with `\&`. This means that `\halign` may not be used directly within a stanza line. This does not affect macros involving alignments defined outside `\stanza` `\&`. Since these macros usurp the control sequence `\&`, the replacement `\ampersand` is defined to be used if this symbol is needed in a stanza. Also we reset the modified category codes and initialize the penalty default.

```

4053 \catcode'\&=\next
4054 \catcode'\@=\body
4055 %% \let\ampersand=\&
4056 \setstanzavalues{szp}{0}
4057

```

39 Arrays and tables

This is based on the work by Herbert Breger in developing `tabmac.tex`.

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 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?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 `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..

```

4058 \newcommand*{\l@dtabnoexpands}{%
4059   \let\rtab=0%
4060   \let\ctab=0%
4061   \let\ltab=0%
4062   \let\rtabtext=0%
4063   \let\ltabtext=0%
4064   \let\ctabtext=0%
4065   \let\edbeforetab=0%
4066   \let\edaftertab=0%
4067   \let\edatab=0%
4068   \let\edatabell=0%
4069   \let\edatleft=0%
4070   \let\edatright=0%
4071   \let\edvertline=0%
4072   \let\edvertdots=0%
4073   \let\edrowfill=0%
4074 }
4075

```

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

```

4076 \newcommand{\disable@familiarnotes}{%
4077   \def\do##1{%
4078     \csletcs{footnote@##1}{footnote##1}%
4079     \expandafter\renewcommand \csname footnote##1\endcsname[1]{%
4080       \protected@csxdef{@thefnmark##1}{\csuse{thefootnote##1}}%
4081       \csuse{@footnotemark##1}%
4082     }%
4083   }%
4084   \dolistloop{\@series}%
4085 }%

```

```

4086 \newcommand{\restore@familiarnotes}{%
4087   \def\do##1{%
4088     \csletcs{footnote##1}{footnote@@##1}%
4089   }%
4090   \dolistloop{\@series}%
4091 }%
4092

\disable@sidenotes The sames, for side notes.
\restore@sidenotes 4093 \newcommand{\disable@sidenotes}{%
4094   \let\@@ledrightnote\ledrightnote%
4095   \let\@@ledleftnote\ledleftnote%
4096   \let\@@ledsidenote\ledsidenote%
4097   \let\ledrightnote\@gobble%
4098   \let\ledleftnote\@gobble%
4099   \let\ledsidenote\@gobble%
4100 }%
4101 \newcommand{\restore@sidenotes}{%
4102   \let\ledrightnote\@@ledrightnote%
4103   \let\ledleftnote\@@ledleftnote%
4104   \let\ledsidenote\@@ledsidenote%
4105 }%

\disable@notes Disable/restore side and familiar notes.
\restore@notes 4106 \newcommand{\disable@notes}{%
4107   \disable@sidenotes%
4108   \disable@familiarnotes%
4109 }%
4110 \newcommand{\restore@notes}{%
4111   \restore@sidenotes%
4112   \restore@familiarnotes%
4113 }%

\l@dampcount \l@dampcount is a counter for the & column dividers and \l@dcolcount is a
\l@dcolcount counter for the columns. These were \Undcount and \stellencount respectively.
4114 \newcount\l@dampcount
4115 \l@dampcount=1\relax
4116 \newcount\l@dcolcount
4117 \l@dcolcount=0\relax
4118

\hilfsbox Some (temporary) helper items.
\hilfsskip 4119 \newbox\hilfsbox
\Hilfsbox 4120 \newskip\hilfsskip
\hilfscount 4121 \newbox\Hilfsbox
4122 \newcount\hilfscount
4123

```

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).

```

4124 \newdimen\dcoli
4125 \newdimen\dcolii
4126 \newdimen\dcoliii
4127 \newdimen\dcoliv
4128 \newdimen\dcolv
4129 \newdimen\dcolvi
4130 \newdimen\dcolvii
4131 \newdimen\dcolviii
4132 \newdimen\dcolix
4133 \newdimen\dcolx
4134 \newdimen\dcolxi
4135 \newdimen\dcolxii
4136 \newdimen\dcolxiii
4137 \newdimen\dcolxiv
4138 \newdimen\dcolxv
4139 \newdimen\dcolxvi
4140 \newdimen\dcolxvii
4141 \newdimen\dcolxviii
4142 \newdimen\dcolxix
4143 \newdimen\dcolxx
4144 \newdimen\dcolxxi
4145 \newdimen\dcolxxii
4146 \newdimen\dcolxxiii
4147 \newdimen\dcolxxiv
4148 \newdimen\dcolxxv
4149 \newdimen\dcolxxvi
4150 \newdimen\dcolxxvii
4151 \newdimen\dcolxxviii
4152 \newdimen\dcolxxix
4153 \newdimen\dcolxxx
4154 \newdimen\dcolerr    % added for error handling
4155

```

`\l@dcolwidth` This is a cunning way of storing the columnwidths indexed by the column number `\l@dcolcount`, like an array. (was `\Dimenzuordnung`)

```

4156 \newcommand{\l@dcolwidth}{\ifcase \the\l@dcolcount \dcoli %???
4157 \or \dcoli \or \dcolii \or \dcoliii
4158 \or \dcoliv \or \dcolv \or \dcolvi
4159 \or \dcolvii \or \dcolviii \or \dcolix \or \dcolx
4160 \or \dcolxi \or \dcolxii \or \dcolxiii
4161 \or \dcolxiv \or \dcolxv \or \dcolxvi
4162 \or \dcolxvii \or \dcolxviii \or \dcolxix \or \dcolxx
4163 \or \dcolxxi \or \dcolxxii \or \dcolxxiii
4164 \or \dcolxxiv \or \dcolxxv \or \dcolxxvi
4165 \or \dcolxxvii \or \dcolxxviii \or \dcolxxix \or \dcolxxx
4166 \else \dcolerr \fi}

```

4167

`\step1@dcolcount` This increments the column counter, and issues an error message if it is too large.

```
4168 \newcommand*\step1@dcolcount{\advance\l@dcolcount\@ne
4169 \ifnum\l@dcolcount>30\relax
4170   \led@err@TooManyColumns
4171 \fi}
4172
```

`\l@dsetmaxcolwidth` Sets the column width to the maximum value seen so far. (was `\dimenzuordnung`)

```
4173 \newcommand*\l@dsetmaxcolwidth{%
4174   \ifdim\l@dcolwidth < \wd\hilfsbox
4175     \l@dcolwidth = \wd\hilfsbox
4176   \else \relax \fi}
4177
```

`\EDTEXT` We need to be able to modify the `\edtext` and `\critext` macros and also restore `\xedtext` their original definitions.

```
\CRITEXT 4178 \let\EDTEXT=\edtext
\xcritext 4179 \newcommand{\xedtext}[2]{\EDTEXT{#1}{#2}}
4180 \let\CRITEXT=\critext
4181 \long\def\xcritext #1#2/{\CRITEXT{#1}{#2}/}
```

`\EDLABEL` We need to be able to modify and restore the `\edlabel` macro.

```
\xedlabel 4182 \let\EDLABEL=\edlabel
4183 \newcommand*\xedlabel[1]{\EDLABEL{#1}}
```

`\EDINDEX` Macros supporting modification and restoration of `\edindex`.

```
\xedindex 4184 \let\EDINDEX=\edindex
\nulledindex 4185 \ifl@dmemoir
4186   \newcommand*\xedindex{\@bsphack%
4187     \@ifnextchar [{\l@d@index}{\l@d@index[\jobname]}}
4188   \newcommand*\nulledindex[2][\jobname]{\@bsphack\@esphack}
4189 \else
4190   \newcommand*\xedindex{\@bsphack%
4191     \doedindexlabel
4192     \begingroup
4193     \@sanitize
4194     \@wredindex}
4195   \newcommand*\nulledindex[1]{\@bsphack\@esphack}
4196 \fi
4197
```

`\@line@@num` Macro supporting restoration of `\linenum`.

```
4198 \let\@line@@num=\linenum
```

`\l@dgoobledarg` `\l@dgoobledarg` replaces its delineated argument by `\relax` (was `\verschwinden`).

`\l@dgooblearg` `\l@dgoobleoptarg[\arg]{\arg}` replaces these two arguments (first is optional) by `\relax`.

```

4199 \def\l@dgoobledarg #1/{\relax}
4200 \newcommand*{\l@dgoobleoptarg}[2][\relax]%
4201

\Relax
\NEXT 4202 \let\Relax=\relax
\@hilfs@count 4203 \let\NEXT=\next
4204 \newcount\@hilfs@count
4205

\measuremcell Measure (recursively) the width required for a math cell. (was \messen)
4206 \def\measuremcell #1{%
4207   \ifx #1\ \ifnum\l@dcolcount=0\let\NEXT\relax%
4208     \else\l@dcheckcols%
4209       \l@dcolcount=0%
4210       \let\NEXT\measuremcell%
4211     \fi%
4212   \else\setbox\hilfsbox=\hbox{$\displaystyle{#1}$}%
4213     \step1@dcolcount%
4214     \l@dsetmaxcolwidth%
4215     \let\NEXT\measuremcell%
4216   \fi\NEXT}
4217

\measuretcell Measure (recursively) the width required for a text cell. (was \messentext)
4218 \def\measuretcell #1{%
4219   \ifx #1\ \ifnum\l@dcolcount=0\let\NEXT\relax%
4220     \else\l@dcheckcols%
4221       \l@dcolcount=0%
4222       \let\NEXT\measuretcell%
4223     \fi%
4224   \else\setbox\hilfsbox=\hbox{#1}%
4225     \step1@dcolcount%
4226     \l@dsetmaxcolwidth%
4227     \let\NEXT\measuretcell%
4228   \fi\NEXT}
4229

\measuremrow Measure (recursively) the width required for a math row. (was \Messen)
4230 \def\measuremrow #1\\{%
4231   \ifx #1&\let\NEXT\relax%
4232   \else\measuremcell #1&\\&%
4233     \let\NEXT\measuremrow%
4234   \fi\NEXT}

\measuretrow Measure (recursively) the width required for a text row. (was \Messentext)
4235 \def\measuretrow #1\\{%
4236   \ifx #1&\let\NEXT\relax%
4237   \else\measuretcell #1&\\&%

```

```

4238     \let\NEXT\measuretrrow%
4239     \fi\NEXT}
4240

```

`\edtabcolsep` The length `\edtabcolsep` controls the distance between columns. (was `\abstand`)

```

4241 \newskip\edtabcolsep
4242 \global\edtabcolsep=10pt
4243

```

```

\NEXT
\Next 4244 \let\NEXT\relax
4245 \let\Next=\next

```

```

\variab
4246 \newcommand{\variab}{\relax}
4247

```

`\l@dcheckcols` Check that the number of columns is consistent. (was `\tabfehlermeldung`)

```

4248 \newcommand*{\l@dcheckcols}{%
4249   \ifnum\l@dcolcount=1\relax
4250   \else
4251     \ifnum\l@dampcount=1\relax
4252     \else
4253       \ifnum\l@dcolcount=\l@dampcount\relax
4254       \else
4255         \l@d@err@UnequalColumns
4256       \fi
4257     \fi
4258     \l@dampcount=\l@dcolcount
4259   \fi}
4260

```

`\l@dmodforcritext` Modify and restore various macros for when `\critext` is used.

```

\l@drestoreforcritext 4261 \newcommand{\l@dmodforcritext}{%
4262   \let\critext\relax%
4263   \def\do##1{\global\csletcs{##1footnote}{\l@dgoobledarg}}
4264   \dolistloop{\@series}%
4265   \let\edindex\nulledindex%
4266   \let\linenum@gobble}
4267 \newcommand{\l@drestoreforcritext}{%
4268   \def\do##1{\csdef{##1footnote}##1##2/{\csuse{##1@footnote}{##1}{##2}}}
4269   \dolistloop{\@series}%
4270   \let\edindex\xedndex}
4271

```

`\l@dmodforedtext` Modify and restore various macros for when `\edtext` is used.

```

\l@drestoreforedtext 4272 \newcommand{\l@dmodforedtext}{%
4273   \let\edtext\relax
4274   \def\do##1{\global\csletcs{##1footnote}{\l@dgoobleoptarg}}%

```



```

4275 \dolistloop{\@series}%
4276 \let\edindex\nulledindex
4277 \let\linenum@gobble}
4278 \newcommand{\l@drestoreforedtext}{%
4279 \def\do##1{\global\csletcs{##1footnote}{##1@footnote}}
4280 \dolistloop{\@series}%
4281 \let\edindex\xedndex}

```

`\l@dnulldfills` Nullify and restore some column fillers, etc.

```

\l@drestorefills 4282 \newcommand{\l@dnulldfills}{%
4283 \def\edlabel##1{}%
4284 \def\edrowfill##1##2##3{}%
4285 }
4286 \newcommand{\l@drestorefills}{%
4287 \def\edrowfill##1##2##3{\@EDROWFILL@{##1}{##2}{##3}}%
4288 }
4289

```

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

```

\def\rverteilen #1&{\def\label##1{}%
  \ifx #1! \ifnum\l@dcolcount=0%\removelastskip
    \let\Next\relax%
  \else\l@dcolcount=0%
    \let\Next=\rverteilen%
  \fi%
\else%
  \footnoteverschw%
  \stepl@dcolcount%
  \setbox\hilfsbox=\hbox{$\displaystyle{#1}$}%
  \let\critext=\xcritext\let\Dfootnote=\D@@footnote
  \let\Afootnote=\A@@footnote\let\Bfootnote=\B@@footnote
  \let\Cfootnote=\C@@footnote\let\linenum=\@line@@num%
  \hilfsskip=\Dimenzuordnung%
  \advance\hilfsskip by -\wd\hilfsbox
  \def\label##1{\xlabel{##1}}%
  \hskip\hilfsskip$\displaystyle{#1}$%
  \hskip\edtabcolsep%
  \let\Next=\rverteilen%
\fi\Next}

```

where the lines

```

\let\critext=\xcritext\let\Dfootnote=\D@@footnote
\let\Afootnote=\A@@footnote\let\Bfootnote=\B@@footnote
\let\Cfootnote=\C@@footnote\let\linenum=\@line@@num%
\hilfsskip=\Dimenzuordnung%
\advance\hilfsskip by -\wd\hilfsbox
\def\label##1{\xlabel{##1}}%

```

were common across the several **verteilen** macros, and also

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

\letsforverteilen Gathers some lets and other code that is common to the **verteilen** macros.

```
4290 \newcommand{\letsforverteilen}{%
4291   \let\critext\xcritext
4292   \let\edtext\xedtext
4293   \let\edindex\xedindex
4294   \def\do##1{\global\csletcs{##1footnote}{##1@footnote}}
4295   \dolistloop{\@series}%
4296   \let\linenum\@line@num
4297   \hilfsskip=\l@dcwidth%
4298   \advance\hilfsskip by -\wd\hilfsbox
4299   \def\edlabel##1{\xedlabel{##1}}
4300 }
```

\setmcellright Typeset (recursively) cells of display math right justified. (was *\rverteilen*)

```
4301 \def\setmcellright #1{\def\edlabel##1{%
4302   \let\edindex\nulledindex
4303   \ifx #1\ \ifnum\l@dcwidth=0%\removelastskip
4304     \let\Next\relax%
4305   \else\l@dcwidth=0%
4306     \let\Next=\setmcellright%
4307   \fi%
4308   \else%
4309     \disablel@dtabfeet%
4310     \step1@dcwidth%
4311     \disable@notes%
4312     \setbox\hilfsbox=\hbox{$\displaystyle{#1}$}%
4313     \restore@notes%
4314     \letsforverteilen%
4315     \hskip\hilfsskip$\displaystyle{#1}$%
4316     \hskip\edtabcolsep%
4317     \let\Next=\setmcellright%
4318   \fi\Next}
4319 }
```

\settcellright Typeset (recursively) cells of text right justified. (was *\rverteilentext*)

```
4320 \def\settcellright #1{\def\edlabel##1{%
4321   \let\edindex\nulledindex
4322   \ifx #1\ \ifnum\l@dcwidth=0%\removelastskip
4323     \let\Next\relax%
```

```

4324         \else\l@dcolcount=0%
4325         \let\Next=\settcellright%
4326     \fi%
4327 \else%
4328     \disablel@dtabfeet%
4329     \step1@dcolcount%
4330     \disable@notes%
4331     \setbox\hilfsbox=\hbox{#1}%
4332     \restore@notes%
4333     \letsforverteilen%
4334     \hskip\hilfsskip#1%
4335     \hskip\edtabcolsep%
4336     \let\Next=\settcellright%
4337 \fi\Next}

```

`\setmcellleft` Typeset (recursively) cells of display math left justified. (was `\lverteilen`)

```

4338 \def\setmcellleft #1&{\def\edlabel##1{}%
4339     \let\edindex\nulledindex
4340     \ifx #1\\ \ifnum\l@dcolcount=0 \let\Next\relax%
4341         \else\l@dcolcount=0%
4342         \let\Next=\setmcellleft%
4343     \fi%
4344 \else \disablel@dtabfeet%
4345     \step1@dcolcount%
4346     \disable@notes%
4347     \setbox\hilfsbox=\hbox{$\displaystyle{#1}$}%
4348     \restore@notes%
4349     \letsforverteilen%
4350     $\displaystyle{#1}$\hskip\hilfsskip\hskip\edtabcolsep%
4351     \let\Next=\setmcellleft%
4352 \fi\Next}
4353

```

`\settcellleft` Typeset (recursively) cells of text left justified. (was `\lverteilentext`)

```

4354 \def\settcellleft #1&{\def\edlabel##1{}%
4355     \let\edindex\nulledindex
4356     \ifx #1\\ \ifnum\l@dcolcount=0 \let\Next\relax%
4357         \else\l@dcolcount=0%
4358         \let\Next=\settcellleft%
4359     \fi%
4360 \else \disablel@dtabfeet%
4361     \step1@dcolcount%
4362     \disable@notes%
4363     \setbox\hilfsbox=\hbox{#1}%
4364     \restore@notes%
4365     \letsforverteilen%
4366     #1\hskip\hilfsskip\hskip\edtabcolsep%
4367     \let\Next=\settcellleft%
4368 \fi\Next}

```

`\setmcellcenter` Typeset (recursively) cells of display math centered. (was `\zverteilen`)

```

4369 \def\setmcellcenter #1&{\def\edlabel##1{}%
4370   \let\edindex\nulledindex
4371   \ifx #1\\ \ifnum\l@dc@lcount=0\let\Next\relax%
4372           \else\l@dc@lcount=0%
4373           \let\Next=\setmcellcenter%
4374           \fi%
4375   \else   \disablel@dtabfeet%
4376           \stepl@dc@lcount%
4377           \disable@notes%
4378           \setbox\hilfsbox=\hbox{$\displaystyle{#1}$}%
4379           \restore@notes%
4380           \letsforverteilen%
4381           \hskip 0.5\hilfsskip$\displaystyle{#1}$\hskip0.5\hilfsskip%
4382           \hskip\edtabcolsep%
4383           \let\Next=\setmcellcenter%
4384   \fi\Next}
4385
```

`\settc@lcenter` Typeset (recursively) cells of text centered. (new)

```

4386 \def\settc@lcenter #1&{\def\edlabel##1{}%
4387   \let\edindex\nulledindex
4388   \ifx #1\\ \ifnum\l@dc@lcount=0 \let\Next\relax%
4389           \else\l@dc@lcount=0%
4390           \let\Next=\settc@lcenter%
4391           \fi%
4392   \else   \disablel@dtabfeet%
4393           \stepl@dc@lcount%
4394           \disable@notes%
4395           \setbox\hilfsbox=\hbox{#1}%
4396           \restore@notes%
4397           \letsforverteilen%
4398           \hskip 0.5\hilfsskip #1\hskip 0.5\hilfsskip%
4399           \hskip\edtabcolsep%
4400           \let\Next=\settc@lcenter%
4401   \fi\Next}
4402
```

`\NEXT`

```

4403 \let\NEXT=\relax
4404
```

`\setmrowright` Typeset (recursively) rows of right justified math. (was `\rsetzen`)

```

4405 \def\setmrowright #1\\{%
4406   \ifx #1& \let\NEXT\relax
4407   \else \centerline{\setmcellright #1&\\&\\&}
4408   \let\NEXT=\setmrowright
4409   \fi\NEXT}

```

`\settroright` Typeset (recursively) rows of right justified text. (was `\rsetzentext`)

```
4410 \def\settroright #1\\{%
4411   \ifx #1& \let\NEXT\relax
4412   \else \centerline{\settcclright #1&\\&\\&}
4413         \let\NEXT=\settroright
4414   \fi\NEXT}
4415
```

`\setmrowleft` Typeset (recursively) rows of left justified math. (was `\lsetzen`)

```
4416 \def\setmrowleft #1\\{%
4417   \ifx #1& \let\NEXT\relax
4418   \else \centerline{\setmcclleft #1&\\&\\&}
4419         \let\NEXT=\setmrowleft
4420   \fi\NEXT}
```

`\settrorleft` Typeset (recursively) rows of left justified text. (was `\lsetzentext`)

```
4421 \def\settrorleft #1\\{%
4422   \ifx #1& \let\NEXT\relax
4423   \else \centerline{\settcclleft #1&\\&\\&}
4424         \let\NEXT=\settrorleft
4425   \fi\NEXT}
4426
```

`\setmrowcenter` Typeset (recursively) rows of centered math. (was `\zsetzen`)

```
4427 \def\setmrowcenter #1\\{%
4428   \ifx #1& \let\NEXT\relax%
4429   \else \centerline{\setmcclcenter #1&\\&\\&}
4430         \let\NEXT=\setmrowcenter
4431   \fi\NEXT}
```

`\settrorcenter` Typeset (recursively) rows of centered text. (new)

```
4432 \def\settrorcenter #1\\{%
4433   \ifx #1& \let\NEXT\relax
4434   \else \centerline{\settcclcenter #1&\\&\\&}
4435         \let\NEXT=\settrorcenter
4436   \fi\NEXT}
4437
```

`\nullsetzen` (was `\nullsetzen`)

```
4438 \newcommand{\nullsetzen}{%
4439   \step1@dcolcount%
4440   \l@dcolwidth=0pt%
4441   \ifnum\l@dcolcount=30\let\NEXT\relax%
4442         \l@dcolcount=0\relax
4443   \else\let\NEXT\nullsetzen%
4444   \fi\NEXT}
4445
```

```

\edatleft \edatleft[⟨math⟩]{⟨symbol⟩}{⟨len⟩} (combination and generalisation of original
\Seklam and \Seklamgl). Left ⟨symbol⟩, 2⟨len⟩ high with prepended ⟨math⟩
vertically centered.
4446 \newcommand{\edatleft}[3][\@empty]{%
4447   \ifx#1\@empty
4448     \vbox to 10pt{\vss\hbox{$\left#2\vrule width0pt height #3
4449       depth 0pt \right. $\hss}\vfil}
4450   \else
4451     \vbox to 4pt{\vss\hbox{$#1\left#2\vrule width0pt height #3
4452       depth 0pt \right. $\vfil}
4453   \fi}

\edatright \edatright[⟨math⟩]{⟨symbol⟩}{⟨len⟩} (combination and generalisation of origi-
nal \sekla and \sekla). Right ⟨symbol⟩, 2⟨len⟩ high with appended ⟨math⟩
vertically centered.
4454 \newcommand{\edatright}[3][\@empty]{%
4455   \ifx#1\@empty
4456     \vbox to 10pt{\vss\hbox{$\left.\vrule width0pt height #3
4457       depth 0pt \right#2 $\hss}\vfil}
4458   \else
4459     \vbox to 4pt{\vss\hbox{$\left.\vrule width0pt height #3
4460       depth 0pt \right#2 #1 $\vfil}
4461   \fi}
4462

\edvertline \edvertline{⟨len⟩} vertical line ⟨len⟩ high. (was \sestrich)
4463 \newcommand{\edvertline}[1]{\vbox to 8pt{\vss\hbox{\vrule height #1}\vfil}}
4464

\edvertdots \edvertdots{⟨len⟩} vertical dotted line ⟨len⟩ high. (was \sepunkte)
4465 \newcommand{\edvertdots}[1]{\vbox to 1pt{\vss\vbox to #1%
4466   {\cleaders\hbox{$\m@th\hbox{.}\vbox to 0.5em{ }$}\vfil}}}
4467

```

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]

`\edfilldimen` A length. (was `\klamdimen`)

```
4468 \newdimen\edfilldimen
```

```
4469 \edfilldimen=0pt
```

```
4470
```

`\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....`

```
4471 \newcounter{addcolcount}
```

```
4472 \renewcommand{\theaddcolcount}{\roman{addcolcount}}
```

`\l@dtabaddcols` `\l@dtabaddcols{<startcol>}{<endcol>}` adds the widths of the columns `<startcol>` through `<endcol>` to `\edfilldimen`. It is a LaTeX style reimplement of the original `\@add@`.

```
4473 \newcommand{\l@dtabaddcols}[2]{%
```

```
4474 \l@dccheckstartend{#1}{#2}%
```

```
4475 \ifl@dstartendok
```

```
4476 \setcounter{addcolcount}{#1}%
```

```
4477 \@whilenum \value{addcolcount}<#2\relax \do
```

```
4478 {\advance\edfilldimen by \the \csname dcol\theaddcolcount\endcsname
```

```
4479 \advance\edfilldimen by \edtabcolsep
```

```
4480 \stepcounter{addcolcount}}%
```

```
4481 \advance\edfilldimen by \the \csname dcol\theaddcolcount\endcsname
```

```
4482 \fi
```

```
4483 }
```

```
4484
```

`\ifl@dstartendok` `\l@dccheckstartend{<startcol>}{<endcol>}` checks that the values of `<startcol>` and `<endcol>` are sensible. If they are then `\ifl@dstartendok` is set TRUE, otherwise it is set FALSE.

```
4485 \newif\ifl@dstartendok
```

```
4486 \newcommand{\l@dccheckstartend}[2]{%
```

```
4487 \l@dstartendoktrue
```

```
4488 \ifnum #1<\@ne
```

```

4489 \l@dstartendokfalse
4490 \led@err@LowStartColumn
4491 \fi
4492 \ifnum #2>30\relax
4493 \l@dstartendokfalse
4494 \led@err@HighEndColumn
4495 \fi
4496 \ifnum #1>#2\relax
4497 \l@dstartendokfalse
4498 \led@err@ReverseColumns
4499 \fi
4500 }
4501

```

`\edrowfill` `\edrowfill{<startcol>}{<endcol>}` fill fills columns `<startcol>` to `<endcol>` inclusive with `<fill>` (e.g. `\hrulefill`, `\upbracefill`). This is a LaTeX style reimplementa-
`\@edrowfill@` tion and generalization of the original `\waklam`, `\Waklam`, `\waklamec`, `\wastricht`
`\@EDROWFILL@` and `\wapunktel` macros.

```

4502 \newcommand*{\edrowfill}[3]{%
4503 \l@dtabaddcols{#1}{#2}%
4504 \hb@xt@ \the\l@dcolwidth{\hb@xt@ \the\edfilldimen{#3}\hss}}
4505 \let\@edrowfill@=\edrowfill
4506 \def\@EDROWFILL@#1#2#3{\@edrowfill@{#1}{#2}{#3}}
4507

```

`\edbeforetab` The macro `\edbeforetab{<text>}{<math>}` puts `<text>` at the left margin be-
`\edaftertab` fore array cell entry `<math>`. Conversely, the macro `\edaftertab{<math>}{<text>}`
puts `<text>` at the right margin after array cell entry `<math>`. `\edbeforetab` should
be in the first column and `\edaftertab` in the last column. The following macros
support these.

`\leftltab` `\leftltab{<text>}` for `\edbeforetab` in `\ltab`. (was `\linksltab`)

```

4508 \newcommand{\leftltab}[1]{%
4509 \hb@xt@ \z@{\vbox{\edtabindent%
4510 \moveleft\Hilfsskip\hbox{\ #1}}\hss}}
4511

```

`\leftrtab` `\leftrtab{<text>}{<math>}` for `\edbeforetab` in `\rtab`. (was `\linksrtab`)

```

4512 \newcommand{\leftrtab}[2]{%
4513 #2\hb@xt@ \z@{\vbox{\edtabindent%
4514 \advance\Hilfsskip by\dcoli%
4515 \moveleft\Hilfsskip\hbox{\ #1}}\hss}}
4516

```

`\leftctab` `\leftctab{<text>}{<math>}` for `\edbeforetab` in `\ctab`. (was `\linksztab`)

```

4517 \newcommand{\leftctab}[2]{%
4518 \hb@xt@ \z@{\vbox{\edtabindent\l@dcolcount=\l@dampcount%
4519 \advance\Hilfsskip by 0.5\dcoli%
4520 \setbox\hilfsbox=\hbox{\def\edlabel#1{}}%

```



```

4521      \disablel@dtabfeet$\displaystyle{#2}$}%
4522      \advance\Hilfsskip by -0.5\wd\hilfsbox%
4523      \moveleft\Hilfsskip\hbox{\ #1}}\hss}%
4524      #2}
4525

```

`\rightctab` `\rightctab{<math><math>}{<text>}` for `\edaftertab` in `\ctab`. (was `\rechtsztab`)

```

4526 \newcommand{\rightctab}[2]{%
4527     \setbox\hilfsbox=\hbox{\def\edlabel##1}%
4528     \disablel@dtabfeet#2}\l@dampcount=\l@dcolcount%
4529     #1\hb@xt@z@{\vbox{\edtabindent\l@dcolcount=\l@dampcount%
4530     \advance\Hilfsskip by 0.5\l@dcolwidth%
4531     \advance\Hilfsskip by -\wd\hilfsbox%
4532     \setbox\hilfsbox=\hbox{\def\edlabel##1}%
4533     \disablel@dtabfeet$\displaystyle{#1}$}%
4534     \advance\Hilfsskip by -0.5\wd\hilfsbox%
4535     \advance\Hilfsskip by \edtabcolsep%
4536     \moveright\Hilfsskip\hbox{ #2}}\hss}%
4537     }
4538

```

`\rightltab` `\rightltab{<math><math>}{<text>}` for `\edaftertab` in `\ltab`. (was `\rechtsltab`)

```

4539 \newcommand{\rightltab}[2]{%
4540     \setbox\hilfsbox=\hbox{\def\edlabel##1}%
4541     \disablel@dtabfeet#2}\l@dampcount=\l@dcolcount%
4542     #1\hb@xt@z@{\vbox{\edtabindent\l@dcolcount=\l@dampcount%
4543     \advance\Hilfsskip by\l@dcolwidth%
4544     \advance\Hilfsskip by-\wd\hilfsbox%
4545     \setbox\hilfsbox=\hbox{\def\edlabel##1}%
4546     \disablel@dtabfeet$\displaystyle{#1}$}%
4547     \advance\Hilfsskip by-\wd\hilfsbox%
4548     \advance\Hilfsskip by\edtabcolsep%
4549     \moveright\Hilfsskip\hbox{ #2}}\hss}%
4550     }
4551

```

`\rightrtab` `\rightrtab{<math><math>}{<text>}` for `\edaftertab` in `\rtab`. (was `\rechtsrtab`)

```

4552 \newcommand{\rightrtab}[2]{%
4553     \setbox\hilfsbox=\hbox{\def\edlabel##1}%
4554     \disablel@dtabfeet#2}%
4555     #1\hb@xt@z@{\vbox{\edtabindent%
4556     \advance\Hilfsskip by-\wd\hilfsbox%
4557     \advance\Hilfsskip by\edtabcolsep%
4558     \moveright\Hilfsskip\hbox{ #2}}\hss}%
4559     }
4560

```

`\rtab` `\rtab{<body>}` typesets `<body>` as an array with the entries right justified. (was `\edbforetab` `\rtab`) (Here and elsewhere, `\edbforetab` and `\edaftertab` were originally `\edaftertab`

`\davor` and `\danach`) The original `\rtab` and friends included a fair bit of common code which I have extracted into macros.

The process is first to measure the $\langle body \rangle$ to get the column widths, and then in a second pass to typeset the body.

```

4561 \newcommand{\rtab}[1]{%
4562   \l@dnnullfills
4563   \def\edbeforetab##1##2{\lefttab{##1}{##2}}%
4564   \def\edaftertab##1##2{\righttab{##1}{##2}}%
4565   \measurebody{#1}%
4566   \l@drestorefills
4567   \variab
4568   \setmrowright #1\\&\\%
4569   \enablel@dtabfeet}
4570
```

`\measurebody` `\measurebody{ $\langle body \rangle$ }` measures the array $\langle body \rangle$.

```

4571 \newcommand{\measurebody}[1]{%
4572   \disablel@dtabfeet%
4573   \l@dcolcount=0%
4574   \nullsetzen%
4575   \l@dcolcount=0
4576   \measuremrow #1\\&\\%
4577   \global\l@dampcount=1}
4578
```

`\rtabtext` `\rtabtext{ $\langle body \rangle$ }` typesets $\langle body \rangle$ as a tabular with the entries right justified. (was `\rtabtext`)

```

4579 \newcommand{\rtabtext}[1]{%
4580   \l@dnnullfills
4581   \measuretbody{#1}%
4582   \l@drestorefills
4583   \variab
4584   \settroright #1\\&\\%
4585   \enablel@dtabfeet}
4586
```

`\measuretbody` `\measuretbody{ $\langle body \rangle$ }` measures the tabular $\langle body \rangle$.

```

4587 \newcommand{\measuretbody}[1]{%
4588   \disable@notes%
4589   \disablel@dtabfeet%
4590   \l@dcolcount=0%
4591   \nullsetzen%
4592   \l@dcolcount=0
4593   \measuretbody #1\\&\\%
4594   \restore@notes%
4595   \global\l@dampcount=1}
4596
```

`\ltab` Array with entries left justified. (was `\ltab`)

`\edbeforetab`
`\edaftertab`

```

4597 \newcommand{\ltab}[1]{%
4598   \l@dnnullfills
4599   \def\edbeforetab##1##2{\leftltab{##1}{##2}}%
4600   \def\edaftertab##1##2{\rightltab{##1}{##2}}%
4601   \measurebody{#1}%
4602   \l@drestorefills
4603   \variab
4604   \setmrowleft #1\&\%
4605   \enablel@dtabfeet}
4606

```

`\ltabtext` Tabular with entries left justified. (was `\ltabtext`)

```

4607 \newcommand{\ltabtext}[1]{%
4608   \l@dnnullfills
4609   \measuretbody{#1}%
4610   \l@drestorefills
4611   \variab
4612   \settrrowleft #1\&\%
4613   \enablel@dtabfeet}
4614

```

`\ctab` Array with centered entries. (was `\ztab`)

```

\edbeforetab 4615 \newcommand{\ctab}[1]{%
\edaftertab 4616   \l@dnnullfills
4617   \def\edbeforetab##1##2{\leftctab{##1}{##2}}%
4618   \def\edaftertab##1##2{\rightctab{##1}{##2}}%
4619   \measurebody{#1}%
4620   \l@drestorefills
4621   \variab
4622   \setmrowcenter #1\&\%
4623   \enablel@dtabfeet}
4624

```

`\ctabtext` Tabular with entries centered. (new)

```

4625 \newcommand{\ctabtext}[1]{%
4626   \l@dnnullfills
4627   \measuretbody{#1}%
4628   \l@drestorefills
4629   \variab
4630   \settrrowcenter #1\&\%
4631   \enablel@dtabfeet}
4632

```

`\spreadtext` (was `\breitertext`)

```

4633 \newcommand{\spreadtext}[1]{%\l@dcolcount=\l@dampcount%
4634   \hb@xt@ \the\l@dcolwidth{\hbox{#1}\hss}}

```

`\spreadmath` (was `\breiter`, ‘breiter’ = ‘broadly’)

```

4635 \newcommand{\spreadmath}[1]{%

```

```

4636 \hb@xt@ \the\l@dcowidth{\hbox{$\displaystyle{#1}$}\hss}}
4637

```

I have left the remaining TABMAC alone, apart from changing some names. I'm not yet sure what they do or how they do it. Authors should not use any of these as they are likely to be mutable.

`\tabellzwischen` (was `\tabellzwischen`)

```

4638 \def\tabellzwischen #1&{%
4639     \ifx #1\ \let\NEXT\relax \l@dcowcount=0
4640     \else \step1@dcowcount%
4641         \l@dcowwidth = #1 mm
4642         \let\NEXT=\tabellzwischen
4643     \fi \NEXT }
4644

```

`\edatabell` For example `\edatabell 4 & 19 & 8 \` specifies 3 columns with widths of 4, 19, and 8mm. (was `\atabell`)

```

4645 \def\edatabell #1\{%
4646     \tabellzwischen #1&\&}

```

`\Setzen` (was `\Setzen`, 'setzen' = 'set')

```

4647 \def\Setzen #1&{%
4648     \ifx #1\relax \let\NEXT=\relax
4649     \else \step1@dcowcount%
4650         \let\tabelskip=\l@dcowwidth
4651         \EDTAB #1|
4652         \let\NEXT=\Setzen
4653     \fi\NEXT}
4654

```

`\EDATAB` (was `\ATAB`)

```

4655 \def\EDATAB #1\{%
4656     \ifx #1\Relax \centerline{\Setzen #1\relax&}
4657         \let\Next\relax
4658     \else \centerline{\Setzen #1&\relax&}
4659         \let\Next=\EDATAB
4660     \fi\Next}

```

`\edatab` (was `\atab`)

```

4661 \newcommand{\edatab}[1]{%
4662     \variab%
4663     \EDATAB #1\\\Relax\\}
4664

```

`\HILFSskip` More helpers.

```

\Hilfsskip 4665 \newskip\HILFSskip
4666 \newskip\Hilfsskip
4667

```

`\edtabindent` (was `\tabindent`)

`\EDTAB` (was `\TAB`)

`\EDTABtext` (was `\TABtext`)

`\tabhilfbox` Further helpers.

```
% That finishes tabmac
```

%%%

edarrayl The 'environment' forms for \ltab, \ctab and \rtab.

edarrayc 4705 \newenvironment{edarrayl}{\l@dcollect@body\ltab}{}

edarrayr 4706 \newenvironment{edarrayc}{\l@dcollect@body\ctab}{}

4707 \newenvironment{edarrayr}{\l@dcollect@body\rtab}{}

4708

edtabularl The 'environment' forms for \ltabtext, \ctabtext and \rtabtext.

edtabularc 4709 \newenvironment{edtabularl}{\l@dcollect@body\ltabtext}{}

edtabularr 4710 \newenvironment{edtabularc}{\l@dcollect@body\ctabtext}{}

4711 \newenvironment{edtabularr}{\l@dcollect@body\rtabtext}{}

4712

Here's the code for enabling \edtext (instead of \critext).

\usingcritext Declarations for using \critext{}.../ or using \edtext{}{} inside tabulars.
\disablel@dtabfeet The default at this point is for \edtext.

\enablel@dtabfeet 4713 \newcommand{\usingcritext}{%

\usingedtext 4714 \def\disablel@dtabfeet{\l@modforcritext}%

4715 \def\enablel@dtabfeet{\l@drestoreforcritext}}

4716 \newcommand{\usingedtext}{%

4717 \def\disablel@dtabfeet{\l@modforedtext}%

4718 \def\enablel@dtabfeet{\l@drestoreforedtext}}

4719

4720 \usingedtext

4721

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.

\ledsubsection 4722 \newcommand{\initnumbering@sectcmd}{

\ledsubsection* 4723 \newcommand{\ledsection}[2][]{%

\ledsubsubsection 4724 \led@war@ledxxxDeprecated{section}%

\ledsubsubsection* 4725 \leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\ifl@dpairing\else\skipnumb

\ledchapter 4726 \pstart%

\ledchapter* 4727 \leavevmode\ifledsecnolinenumber\skipnumbering\fi\section[##1]{##2}\leavevmode\vsp

\@patchforledchapter 4728 \vspace{-2\parskip}\vspace{-2\baselineskip}%

\quotation 4729 \ifautopar\else\pstart\fi

\endquotation 4730 }

\quote 4731 \WithSuffix\newcommand\ledsection*[1]{%

\quote 4732 \led@war@ledxxxDeprecated{section}%

\endquote 4733 \leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\ifl@dpairing\else\skipnumb

```

4734     \pstart%
4735     \leavevmode\ifledsecnolinenumber\skipnumbering\fi\section*{##1}\leavevmode\vspace{2.3ex \@plus
4736     \vspace{-2\parskip}\vspace{-2\baselineskip}%
4737     \ifautopar\else\pstart\fi
4738 }
4739 \newcommand{\ledsubsection}[2][]{%
4740     \led@war@ledxxxDeprecated{subsection}%
4741     \leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\ifl@dpairing\else\skipnumbering\fi%
4742     \pstart%
4743     \leavevmode\ifledsecnolinenumber\skipnumbering\fi\subsection[##1]{##2}\leavevmode\vspace{1.5ex \@plus
4744     \vspace{-2\parskip}\vspace{-2\baselineskip}%
4745     \ifautopar\else\pstart\fi
4746 }
4747 \WithSuffix\newcommand\ledsubsection*[1]{%
4748     \led@war@ledxxxDeprecated{subsection}%
4749     \leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\ifl@dpairing\else\skipnumbering\fi%
4750     \pstart%
4751     \leavevmode\ifledsecnolinenumber\skipnumbering\fi\subsection*{##1}\leavevmode\vspace{1.5ex \@plus
4752     \vspace{-2\parskip}\vspace{-2\baselineskip}%
4753     \ifautopar\else\pstart\fi
4754 }
4755 \newcommand{\ledsubsubsection}[2][]{%
4756     \led@war@ledxxxDeprecated{subsubsection}%
4757     \leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\ifl@dpairing\else\skipnumbering\fi%
4758     \pstart%
4759     \leavevmode\ifledsecnolinenumber\skipnumbering\fi\subsubsection[##1]{##2}\leavevmode\vspace{1.5ex \@plus
4760     \vspace{-2\parskip}\vspace{-2\baselineskip}%
4761     \ifautopar\else\pstart\fi
4762 }
4763 \WithSuffix\newcommand\ledsubsubsection*[1]{%
4764     \led@war@ledxxxDeprecated{subsubsection}%
4765     \leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\ifl@dpairing\else\skipnumbering\fi%
4766     \pstart%
4767     \leavevmode\ifledsecnolinenumber\skipnumbering\fi\subsubsection*{##1}\leavevmode\vspace{1.5ex \@plus
4768     \vspace{-2\parskip}\vspace{-2\baselineskip}%
4769     \ifautopar\else\pstart\fi
4770 }
4771 \newcommand\ledchapter[2][]{%
4772     \led@war@ledxxxDeprecated{chapter}%
4773     \ifl@dmemor%
4774     \gdef\ch@pt@c{##1}%
4775     \fi%
4776     ~\pend\skipnumbering%
4777     \pstart%
4778     \@patchforledchapter\chapter[##1]{##2}%
4779     \pend\pstart}
4780 \WithSuffix\newcommand\ledchapter*[1]{%
4781     \led@war@ledxxxDeprecated{chapter}%
4782     ~\pend\skipnumbering%
4783     \pstart%

```

```

4784     \@patchforledchapter\chapter*{##1}\pend%
4785     \pstart}
4786 \def\@patchforledchapter{
4787     \patchcmd{\@makeschapterhead}{1\par}{1}{-}{-}
4788     \pretocmd{\@makeschapterhead}{\par}{-}{-}
4789     \apptocmd{\@makeschapterhead}{\par}{-}{-}
4790     \patchcmd{\@makeschapterhead}{\vskip 40\p@}{-}{-}{-}
4791     \patchcmd{\@makechapterhead}{1\par}{1}{-}{-}
4792     \pretocmd{\@makechapterhead}{\par}{-}{-}
4793     \apptocmd{\@makechapterhead}{\par}{-}{-}
4794     \patchcmd{\@makechapterhead}{\vskip 40\p@}{-}{-}{-}
4795     \apptocmd{\@chapter}{\par\leavevmode\vspace{40 \p@}\skipnumbering}{-}{-}
4796     \apptocmd{\@schapter}{\par\leavevmode\vspace{40 \p@}\skipnumbering}{-}{-}
4797     \newcommand\beforeledchapter{\pend\cleardoublepage\pstart}
4798     \patchcmd{\chapter}{\cleardoublepage}{\relax}{-}{-}
4799     \patchcmd{\chapter}{\clearpage}{\relax}{-}{-}
4800 }
4801 \ifnoquotation@else
4802 \renewcommand{\quotation}{\par\leavevmode%
4803     \parindent=1.5em%
4804     \skipnumbering%
4805     \ifautopar%
4806         \vskip-\parskip%
4807     \else%
4808         \vskip\topsep%
4809     \fi%
4810     \global\leftskip=\leftmargin%
4811     \global\rightskip=\leftmargin%
4812 }
4813 \renewcommand{\endquotation}{\par%
4814     \global\leftskip=0pt%
4815     \global\rightskip=0pt%
4816     \leavevmode%
4817     \skipnumbering%
4818     \ifautopar%
4819         \vskip-\parskip%
4820     \else%
4821         \vskip\topsep%
4822     \fi%
4823 }
4824 \renewcommand{\quote}{\par\leavevmode%
4825     \parindent=0pt%
4826     \skipnumbering%
4827     \ifautopar%
4828         \vskip-\parskip%
4829     \else%
4830         \vskip\topsep%
4831     \fi%
4832     \global\leftskip=\leftmargin%
4833     \global\rightskip=\leftmargin%

```



```

4834 }
4835 \renewcommand{\endquote}{\par%
4836     \global\leftskip=0pt%
4837     \global\rightskip=0pt%
4838     \leavevmode%
4839     \skipnumbering%
4840     \ifautopar%
4841         \vskip-\parskip%
4842     \else%
4843         \vskip\topsep%
4844     \fi%
4845 }
4846 \fi
4847 }

```

`\ledsectnotoc` The `\ledsectnotoc` only disables the `\addcontentsline` macro.

```

4848 \newcommand{\ledsectnotoc}{\let\addcontentsline\@gobblethree}

```

`\ledsectnomark` The `\ledsectnomark` only disables the `\chaptermark`, `\sectionmark` and `\subsectionmark` macros.

```

4849 \newcommand{\ledsectnomark}{%
4850     \let\chaptermark\@gobble%
4851     \let\sectionmark\@gobble%
4852     \let\subsectionmark\@gobble%
4853 }

```

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.

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

```
4854 \ifl@dmemoir
4855   \newcommand\beforeeledchapter{\clearforchapter}
4856 \else
4857   \newcommand\beforeeledchapter{\if@openright\cleardoublepage\else\clearpage\fi}
4858 \fi
```

`\if@eled@sectioning` The boolean `\if@eled@sectioning` is set to true when a sectioning command is called by a `\eledxxx` command, and set to false after. It is used to enable/disable line number printing.

```
4859 \newif\if@eled@sectioning
```

`\print@leftmargin@eledsection` `\print@leftmargin@eledsection` and `\print@rightmargin@eledsection` are added by `eledmac` inside the code of sectioning command, in order to affix line numbers. They include tests for RTL languages.

```
4860 \def\print@rightmargin@eledsection{%
4861   \if@eled@sectioning%
4862     \begingroup%
4863     \if@RTL%
4864       \let\llap\rlap%
4865       \let\leftlinenum\rightlinenum%
4866       \let\leftlinenumR\rightlinenumR%
4867       \let\l@dld@ta\l@dld@ta%
4868       \let\l@dlsn@te\l@dlsn@te%
4869     \fi%
4870     \hfill\l@dld@ta \csuse{LR}{\l@dlsn@te}%
4871     \endgroup%
4872   \fi%
4873 }%
4874
4875 \def\print@leftmargin@eledsection{%
4876   \if@eled@sectioning%
4877     \leavevmode%
4878     \begingroup%
4879     \if@RTL%
4880       \let\rlap\llap%
4881       \let\rightlinenum\leftlinenum%
4882       \let\rightlinenumR\leftlinenumR%
4883       \let\l@dld@ta\l@dld@ta%
4884       \let\l@dlsn@te\l@dlsn@te%
4885     \fi%
4886     \l@dld@ta\csuse{LR}{\l@dlsn@te}%
4887     \endgroup%
4888   \fi%
4889 }%
4890
```

`\chapter` We have to patch L^AT_EX, `book` and `memoir` sectioning commands in order to:

`\M@sect`

- Disable `\edtext` inside.

`\@mem@old@ssect`

`\@makechapterhead`

- Disable page breaking (for `\chapter`).

`\@makechapterhead`

`\@makeschapterhead`

- Add line numbers and sidenotes.

`\@sect`

`\@ssect`

Unfortunately, Maïeul Rouquette was not able to try if `memoir` is loaded. That is why `eledmac` tries to define for both standard class and `memoir` class.

```

4891 \catcode'\#12 % Space NEEDS by \catcode
4892 \AtBeginDocument{%
4893 \patchcmd{\chapter}{\clearforchapter}{%
4894   \if@eled@sectioning\else%
4895     \clearforchapter
4896   \fi%
4897 }
4898 {}
4899 {}
4900
4901 \pretocmd{\M@sect}
4902   {\let\old@edtext=\edtext%
4903    \let\edtext=\dummy@edtext@showlemma%
4904   }
4905 {}
4906 {}
4907
4908 \apptocmd{\M@sect}
4909   {\let\edtext=\old@edtext}
4910 {}
4911 {}
4912
4913 \patchcmd{\M@sect}
4914   { #9}
4915   { #9%
4916     \print@rightmargin@eledsection%
4917   }
4918 {}
4919 {}
4920
4921 \patchcmd{\M@sect}
4922   {\hskip #3\relax}
4923   {\hskip #3\relax%
4924     \print@leftmargin@eledsection%
4925   }
4926 {}
4927 {}
4928
4929
4930

```

```

4931 \patchcmd{\@mem@old@ssect}
4932   {#5}
4933   {#5%
4934   \print@leftmargin@eledsection%
4935   }
4936   {}
4937   {}
4938
4939 \patchcmd{\@mem@old@ssect}
4940   {\hskip #1}
4941   {\hskip #1%
4942   \print@rightmargin@eledsection%
4943   }
4944   {}
4945   {}
4946
4947 \patchcmd{\chapter}{\if@openright\cleardoublepage\else\clearpage\fi}{%
4948   \if@eled@sectioning\else%
4949     \if@openright\cleardoublepage\else\clearpage\fi%No clearpage inside a \eledsection : v
4950   \fi%
4951   }%
4952   {}%
4953   {}%
4954
4955 \patchcmd{\@makechapterhead}
4956   {#1}
4957   {\print@leftmargin@eledsection%
4958     #1%
4959   \print@rightmargin@eledsection%
4960   }
4961   {}
4962   {}
4963
4964 \patchcmd{\@makechapterhead}% For BIDI
4965   {\if@RTL\raggedleft\else\raggedright\fi}%
4966   {\if@eled@sectioning\else%
4967     \if@RTL\raggedleft\else\raggedright\fi%
4968   \fi%
4969   }%
4970   {}%
4971   {}%
4972
4973 \patchcmd{\@makeschapterhead}
4974   {#1}
4975   {\print@leftmargin@eledsection%
4976     #1%
4977   \print@rightmargin@eledsection%
4978   }
4979   {}
4980   {}

```

```

4981
4982 \pretocmd{\@sect}
4983   {\let\old@edtext=\edtext
4984    \let\edtext=\dummy@edtext@showlemma%
4985   }
4986   {}
4987   {}
4988
4989 \apptocmd{\@sect}
4990   {\let\edtext=\old@edtext}
4991   {}
4992   {}
4993
4994 \pretocmd{\@ssect}
4995   {\let\old@edtext=\edtext%
4996    \let\edtext=\dummy@edtext@showlemma%
4997   }
4998   {}
4999   {}
5000
5001 \apptocmd{\@ssect}
5002   {\let\edtext=\old@edtext}
5003   {}
5004   {}
5005

```

hyperref 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.

```

5006 \@ifpackageloaded{nameref}{
5007
5008   \patchcmd{\NR@sect}
5009     {#8}
5010     {#8%
5011      \print@rightmargin@eledsection%
5012     }
5013     {}
5014     {}
5015
5016   \patchcmd{\NR@sect}
5017     {\hskip #3\relax}
5018     {\hskip #3\relax%
5019      \print@leftmargin@eledsection%
5020     }
5021     {}
5022     {}
5023
5024   \patchcmd{\NR@ssect}
5025     {#5}
5026     {#5%

```

```

5027 \print@rightmargin@eledsection%
5028 }
5029 {}
5030 {}
5031
5032 \patchcmd{\NR@ssect}
5033 {\hskip #1}
5034 {\hskip #1%
5035 \print@leftmargin@eledsection%
5036 }
5037 {}
5038 {}
5039 }%
5040 {
5041 \patchcmd{\@ssect}
5042 {#8}
5043 {#8%
5044 \print@rightmargin@eledsection%
5045 }
5046 {}
5047 {}
5048
5049 \patchcmd{\@ssect}
5050 {\hskip #3\relax}
5051 {\hskip #3\relax%
5052 \print@leftmargin@eledsection%
5053 }
5054 {}
5055 {}
5056
5057 \patchcmd{\@ssect}
5058 {#5}
5059 {#5%
5060 \print@rightmargin@eledsection%
5061 }
5062 {}
5063 {}
5064
5065 \patchcmd{\@ssect}
5066 {\hskip #1}
5067 {\hskip #1%
5068 \print@leftmargin@eledsection%
5069 }
5070 {}
5071 {}
5072 }%
5073 }
5074 \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.

```
5075 \newwrite\eled@sectioning@out
```

\noeledsec The boolean \if@noeled@sec is set to true when \noeledsec is called. It is used
 \if@noeled@sec to disable external file creation.

```
5076 \newif\if@noeled@sec%
```

```
5077 \newcommand{\noeledsec}{\global\@noeled@sectrue}%
```

\eledchapter And now, the user sectioning commands, which write to the file, and also add
 \eledsection content as a "normal" line.

```
\eledsubsection 5078 \newcommand{\eledchapter}[2] [] {%
\eledsubsubsection 5079 #2%
\eledchapter* 5080 \ifledRcol%
\eledsection* 5081 \immediate\write\eled@sectioningR@out{%
\eledsubsection* 5082 \string\eled@chapter{#1}{\unexpanded{#2}}{\the\l@dumpstartsR}{R}
\eledsubsubsection* 5083 }%
5084 \else%
5085 \immediate\write\eled@sectioning@out{%
5086 \string\eled@chapter{#1}{\unexpanded{#2}}{\the\l@dumpstartsL}{L}
5087 }%
5088 \fi%
5089 }
5090
5091 \newcommand{\eledsection}[2] [] {%
5092 #2%
5093 \ifledRcol%
5094 \immediate\write\eled@sectioningR@out{%
5095 \string\eled@section{#1}{\unexpanded{#2}}{\the\l@dumpstartsR}{R}
5096 }%
5097 \else%
5098 \immediate\write\eled@sectioning@out{%
5099 \string\eled@section{#1}{\unexpanded{#2}}{\the\l@dumpstartsL}{L}
5100 }%
5101 \fi%
5102 }
5103
5104 \newcommand{\eledsubsection}[2] [] {%
5105 #2%
5106 \ifledRcol%
5107 \immediate\write\eled@sectioningR@out{%
5108 \string\eled@subsection{#1}{\unexpanded{#2}}{\the\l@dumpstartsR}{R}
5109 }%
5110 \else%
5111 \immediate\write\eled@sectioning@out{%
5112 \string\eled@subsection{#1}{\unexpanded{#2}}{\the\l@dumpstartsL}{L}
5113 }%
5114 \fi%
5115 }
5116 \newcommand{\eledsubsubsection}[2] [] {%
```

```

5117 #2%
5118 \ifledRcol%
5119 \immediate\write\eled@sectioningR@out{%
5120 \string\eled@subsubsection{#1}{\unexpanded{#2}}{\the\l@dnumpst
5121 }%
5122 \else%
5123 \immediate\write\eled@sectioning@out{%
5124 \string\eled@subsubsection{#1}{\unexpanded{#2}}{\the\l@dnumps
5125 }%
5126 \fi%
5127 }
5128
5129
5130 \WithSuffix\newcommand\eledchapter*[2] [] {%
5131 #2%
5132 \ifledRcol%
5133 \immediate\write\eled@sectioningR@out{%
5134 \string\eled@chapter{#1}{\unexpanded{#2}}{\the\l@dnumpst
5135 }%
5136 \else%
5137 \immediate\write\eled@sectioning@out{%
5138 \string\eled@chapter{#1}{\unexpanded{#2}}{\the\l@dnumps
5139 }%
5140 \fi%
5141 }
5142
5143 \WithSuffix\newcommand\eledsection*[2] [] {%
5144 #2%
5145 \ifledRcol%
5146 \immediate\write\eled@sectioningR@out{%
5147 \string\eled@section{#1}{\unexpanded{#2}}{\the\l@dnumpst
5148 }%
5149 \else%
5150 \immediate\write\eled@sectioning@out{%
5151 \string\eled@section{#1}{\unexpanded{#2}}{\the\l@dnumps
5152 }%
5153 \fi%
5154 }
5155
5156 \WithSuffix\newcommand\eledsubsection*[2] [] {%
5157 #2%
5158 \ifledRcol%
5159 \immediate\write\eled@sectioningR@out{%
5160 \string\eled@subsection{#1}{\unexpanded{#2}}{\the\l@dnumps
5161 }%
5162 \else%
5163 \immediate\write\eled@sectioning@out{%
5164 \string\eled@subsection{#1}{\unexpanded{#2}}{\the\l@dnumps
5165 }%
5166 \fi%

```



```

5167 }
5168
5169 \WithSuffix\newcommand\eledsubsubsection*[2][]{%
5170   #2%
5171   \ifledRcol%
5172     \immediate\write\eled@sectioningR@out{%
5173       \string\eled@subsubsection{#1}{\unexpanded{#2}}{\the\l@dumpstartsR}{*}{R}
5174     }%
5175   \else%
5176     \immediate\write\eled@sectioning@out{%
5177       \string\eled@subsubsection{#1}{\unexpanded{#2}}{\the\l@dumpstartsL}{*}{L}
5178     }%
5179   \fi%
5180 }

```

\eled@chapter The sectioning macros, called in the auxiliary file. They have five arguments:

- | | |
|--|---|
| \eled@section
\eled@subsection
\eled@subsubsection | <ol style="list-style-type: none"> 1. Optional arguments of L^AT_EX sectioning command. 2. Mandatory arguments of L^AT_EX sectioning command. 3. Pstart number. 4. Side: R if right, nothing if left. 5. Starred or not. |
|--|---|

```

5181 \def\eled@chapter#1#2#3#4#5{%
5182   \ifstrempy{#4}%
5183   {%
5184     \ifstrempy{#1}%
5185     {%
5186       \global\csdef{eled@sectioning@#3#5}{\let\edtext=\dummy@edtext@showlemma\chapter{#2}}%
5187       \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext{}\chaptermark{#2}}%
5188       }%Need for \pairs, because of using parbox.
5189     {%
5190       \global\csdef{eled@sectioning@#3#5}{\let\edtext=\dummy@edtext@showlemma\chapter[#1]{#2}}%
5191       \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext{}\chaptermark{#2}}%Need for \pairs
5192     }%
5193   }%
5194   {%
5195     \ifstrempy{#1}%
5196     {%\global\csdef{eled@sectioning@#3#5}{\let\edtext=\dummy@edtext@showlemma\chapter*{#2}}}%
5197     {%\global\csdef{eled@sectioning@#3#5}{\let\edtext=\dummy@edtext@showlemma\chapter*{#1}{#2}}}%B
5198   }%
5199   \listcsadd{eled@sections#5@@}{#3}%
5200 }
5201 \def\eled@section#1#2#3#4#5{%
5202   \ifstrempy{#4}%
5203   {\ifstrempy{#1}%
5204     {%

```

```

5205     \global\csdef{eled@sectioning@#3#5}{\section{#2}}%
5206     \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext{}\sectionmark{#2}}%Ne
5207     }%
5208     {%
5209     \global\csdef{eled@sectioning@#3#5}{\section[#1]{#2}}%
5210     \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext{}\sectionmark{#1}}%Ne
5211     }%
5212     }%
5213     {\ifstrepty{#1}%
5214     {\global\csdef{eled@sectioning@#3#5}{\section*{#2}}}%
5215     {\global\csdef{eled@sectioning@#3#5}{\section*{#1}{#2}}}%Bug in LaTeX!
5216     }
5217     \listcsadd{eled@sections#5@@}{#3}%
5218     }
5219 \def\eled@subsection#1#2#3#4#5{%
5220     \ifstrepty{#4}%
5221     {\ifstrepty{#1}%
5222     {%
5223     \global\csdef{eled@sectioning@#3#5}{\subsection{#2}}%
5224     \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext{}\csuse{subsectionmar
5225     }%
5226     {%
5227     \global\csdef{eled@sectioning@#3#5}{\subsection[#1]{#2}}%
5228     \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext{}\csuse{subsectionmar
5229     }%
5230     }%
5231     {\ifstrepty{#1}%
5232     {\global\csdef{eled@sectioning@#3#5}{\subsection*{#2}}}%
5233     {\global\csdef{eled@sectioning@#3#5}{\subsection*{#1}{#2}}}%Bug in LaTeX!
5234     }
5235     \listcsadd{eled@sections#5@@}{#3}%
5236     }
5237 \def\eled@subsubsection#1#2#3#4#5{%
5238     \ifstrepty{#4}%
5239     {\ifstrepty{#1}%
5240     {\global\csdef{eled@sectioning@#3#5}{\subsubsection{#2}}}%
5241     {\global\csdef{eled@sectioning@#3#5}{\subsubsection[#1]{#2}}}%
5242     }%
5243     {\ifstrepty{#1}%
5244     {\global\csdef{eled@sectioning@#3#5}{\subsubsection*{#2}}}%
5245     {\global\csdef{eled@sectioning@#3#5}{\subsubsection*{#1}{#2}}}%Bug in LaTeX!
5246     }
5247     \listcsadd{eled@sections#5@@}{#3}%
5248     }
5249

```

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  \normal@page@break is an etoolbox list which contains the absolute line number
                    of the last line, for each page.
5250 \def\normal@page@break{}

\prev@pb  The \l@prev@pb macro is a etoolbox list, which contains the lines in which page
\prev@nopb 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.
5251 \def\l@prev@pb{}
5252 \def\l@prev@nopb{}

\ledpb  The \ledpb macro writes the call to \led@pb in line-list file. The \ledpbnum
\ledpbnum macro writes the call to \led@pbnum in line-list file. The \lednopb macro writes
\lednopb the call to \led@nopb in line-list file. The \lednopbnum macro writes the call to
\lednopbnum \led@nopbnum in line-list file.
5253 \newcommand{\ledpb}{\write\linenum@out{\string\led@pb}}
5254 \newcommand{\ledpbnum}[1]{\write\linenum@out{\string\led@pbnum{#1}}}
5255 \newcommand{\lednopb}{\write\linenum@out{\string\led@nopb}}
5256 \newcommand{\lednopbnum}[1]{\write\linenum@out{\string\led@nopbnum{#1}}}

\led@pb  The \led@pb adds the absolute line number in the \prev@pb list. The \led@pbnum
\led@pbnum adds the argument in the \prev@pb list. The \led@nopb adds the absolute line
\led@nopb number in the \prev@nopb list. The \led@nopbnum adds the argument in the
\led@nopbnum \prev@nopb list.
5257 \newcommand{\led@pb}{\listxadd{\l@prev@pb}{\the\absline@num}}
5258 \newcommand{\led@pbnum}[1]{\listxadd{\l@prev@pb}{#1}}
5259 \newcommand{\led@nopb}{\listxadd{\l@prev@nopb}{\the\absline@num}}
5260 \newcommand{\led@nopbnum}[1]{\listxadd{\l@prev@nopb}{#1}}

\ledpbsetting  The \ledpbsetting macro only changes the value of \led@pb@macro, for which
\led@pb@setting the default value is before.
5261 \def\led@pb@setting{before}
5262 \newcommand{\ledpbsetting}[1]{\gdef\led@pb@setting{#1}}

\led@check@pb  The \led@check@pb and \led@check@nopb are called before or after each line.
\led@check@nopb They check if a page break must occur, depending on the current line and on the
                content of \l@pb.
5263 \newcommand{\led@check@pb}{\xifinlist{\the\absline@num}{\l@prev@pb}{\pagebreak[4]}}}
```

```

5264 \newcommand{\led@check@nopb}{%
5265   \IfStrEq{\led@pb@setting}{before}{%
5266     \xifinlist{\the\absline@num}{\l@prev@nopb}%
5267     {\numdef{\abs@prevline}{\the\absline@num-1}%
5268     \xifinlist{\abs@prevline}{\normal@page@break}%
5269     {\nopagebreak[4]\enlargethispage{\baselineskip}}%
5270     {}}%
5271   }%
5272 }%
5273 }%
5274 \IfStrEq{\led@pb@setting}{after}{%
5275   \xifinlist{\the\absline@num}{\l@prev@nopb}{%
5276     \xifinlist{\the\absline@num}{\normal@page@break}%
5277     {\nopagebreak[4]\enlargethispage{\baselineskip}}%
5278     {}%
5279 }%
5280   }%
5281 }%
5282 }%
5283 }

```

42 Long verse: prevents being separated by a page break

`\iflednopbinverse` The `\lednopbinverse` boolean is set to false by default. If set to true, `eledmac` will automatically prevent page breaks inside verse. The declaration is made at the beginning of the file, because it is used as a package option.

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

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

```

5284 \newcommand{\check@pb@in@verse}{%
5285   \ifinstanza\iflednopbinverse\ifinserthangingsymbol% Using stanzas and enabling page br
5286     \ifnum\page@num=\last@page@num\else%If we have change page
5287       \IfStrEq{\led@pb@setting}{before}{%
5288         \numgdef{\abs@line@verse}{\the\absline@num-1}%
5289         \ledpbnum{\abs@line@verse}%
5290       }%
5291       \IfStrEq{\led@pb@setting}{after}{%
5292         \numgdef{\abs@line@verse}{\the\absline@num-1}%
5293         \lednopbnum{\abs@line@verse}%
5294       }%
5295     \fi%
5296   \fi\fi\fi%
5297 }

```

43 The End

i/code

Appendix A Some things to do when changing version

Appendix A.1 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 4.7 p.25). You must delete your `\Xfootnote` command.

If you customized the `\XXXXXfmt` command, you should see if commands for display options (4.4 p.19) and options in `\Xfootnote` (4.1 p.16) can't do the same things. If not, you can add a new ticket in Github to request a new function it³⁰.

If for some reason you don't want to make the modifications to use eledmac new functions, you can continue to use your own `\XXXXXfmt` command, but you must replace:

```
\renewcommand*{XXXXfmt}[3]
```

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.2 Migration to eledmac 1.5.1

The version 1.5.1 corrects a bug with `stanzaindentrepetition` (cf. p. 26). This bug had two consequences:

1. `stanzaindentrepetition` didn't work when its value was greater than 2.
2. `stanzaindentrepetition` worked wrong when its value was equal to 2.

So, if you used `stanzaindentrepetition` with value equal to 2, you must change your `\setstanzaindent`. Explanation:

```
\setcounter{stanzaindentrepetition}{2}
\setstanzaindent{5,1,0}
```

³⁰<https://github.com/maieul/ledmac/issues>

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{stanzaindentrepetition}{2}
\setstanzaindent{5,1,0}
```

by:

```
\setcounter{stanzaindentrepetition}{2}
\setstanzaindent{5,0,1}
```

Appendix A.3 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, 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. 38).

References

- [Bre96] Herbert Breger. **TABMAC**. October 1996. (Available from CTAN in `macros/plain/contrib/tabmac`)
- [Bur01] John Burt. ‘Typesetting critical editions of poetry’. *TUGboat*, **22**, 4, pp 353–361, December 2001. (Code available from CTAN in `macros/latex/contrib/poemscol`)
- [Eck03] Matthias Eckermann. *The Parallel-Package*. April 2003. (Available from CTAN in `macros/latex/contrib/parallel`)
- [Fai03] Robin Fairbairns. *footmisc — a portmanteau package for customising footnotes in L^AT_EX*. February 2003. (Available from CTAN in `macros/latex/contrib/footmisc`)
- [LW90] John Lavagnino and Dominik Wujastyk. ‘An overview of EDMAC: a PLAIN TeX format for critical editions’. *TUGboat*, **11**, 4, pp. 623–643, November 1990. (Code available from CTAN in `macros/plain/contrib/edmac`)
- [Lüc03] Uwe Lück. ‘ednotes — critical edition typesetting with LaTeX’. *TUGboat*, **24**, 2, pp. 224–236, 2003. (Code available from CTAN in `macros/latex/contrib/ednotes`)
- [Sul92] Wayne G. Sullivan. *The file edstanza.doc*. June 1992. (Available from CTAN in `macros/plain/contrib/edmac`)
- [Wil02] Peter Wilson. *The memoir class for configurable typesetting*. November 2002. (Available from CTAN in `macros/latex/contrib/memoir`)
- [Wil04] Peter Wilson and Maïeul Rouquette. *Parallel typesetting for critical editions: the eledpar package*. December 2004. (Available from CTAN in `macros/latex/contrib/ledmmac`)

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		3954, 3955, 4017, 4032, 4053, 4055
<code>\#</code> 4891, 5074	<code>\@@ledleftnote</code> 4095, 4103
<code>\&</code> 26, 3950,	<code>\@@ledrightnote</code> 4094, 4102

- \@ledsidenote 4096, 4104
- \@eline 1933
- \@wrindexm@m ... 3777, 3779, 3782,
3789, 3791, 3794, 3799, 3801, 3804
- \@EDROWFILL@ 4287, 4502
- \@M 1933, 4010, 4022
- \@MM 1584
- \@adv 582, 791
- \@arabic 1042
- \@auxout .. 3255, 3776, 3778, 3781,
3788, 3790, 3793, 3798, 3800, 3803
- \@botlist 3208, 3210
- \@cclv 3113, 3117, 3118, 3206, 3207, 3235
- \@chapter 4795
- \@checkend 3931
- \@colht 3094, 3211, 3223
- \@colroom 3211
- \@combinefloats 3089
- \@currentlabel
.... 1078, 2168, 2292, 2359, 2476
- \@currenvir 3915, 3918, 3919
- \@currlist 3212, 3215
- \@dbldeferlist 3221, 3226, 3228
- \@dblfloatplacement 3225
- \@dbltoplist 3221, 3222
- \@deferlist 3208, 3217, 3218
- \@docleapage 3195
- \@edindex@fornote@true 3734
- \@edindex@hyperref 3822, 3874
- \@edrowfill@ 4502
- \@edtext@false 865, 892
- \@edtext@true 843, 869
- \@ehb 3214
- \@ehd 171, 174, 177
- \@eled@sectioningfalse 1182
- \@eled@sectioningtrue 1180
- \@emptytoks 3906, 3916
- \@fnpos 2565, 3131, 3134
- \@footnotemark 2097
- \@footnotetext
2096, 2110, 2125, 3635, 3671, 3698
- \@freelist 3087
- \@gobble . 696, 824, 825, 2624, 3928,
4097–4099, 4266, 4277, 4850–4852
- \@gobblefour 179, 2797
- \@gobblethree 179, 4848
- \@gobbletwo 701
- \@h 1931
- \@hilfs@count 4202
- \@idxfile . 3766, 3777, 3779, 3782,
3789, 3791, 3794, 3799, 3801, 3804
- \@ifclassloaded
.... 44, 2095, 3162, 3187, 3851
- \@ifnextchar 3756, 4187
- \@ifpackageloaded 47, 49,
3109, 3713, 3716, 3852, 3855, 5006
- \@iiiminipage 3624
- \@iiiparbox 3651
- \@index@command
. 3738, 3740–3742, 3745–3747,
3818, 3819, 3828, 3831, 3886, 3890
- \@index@command@ 3741, 3742, 3746, 3747
- \@index@parenthesis
.... 3739, 3743, 3748, 3885, 3889
- \@index@txt 3737,
3818, 3819, 3822, 3828, 3831, 3903
- \@indexfile 3827, 3830, 3834
- \@inputcheck 473
- \@insert 1470–1472, 1506–1508
- \@k 1931
- \@kludgeins 3091, 3159
- \@l@dttempcnta .. 183, 616, 618, 620,
621, 1253, 1254, 1256, 1258,
1261, 1262, 1277, 1313–1317,
1319, 1326–1330, 1332, 1335,
1338, 1340, 1344, 1375, 1379,
1383, 1390, 1394, 1398, 1479,
1483, 1487, 1490, 1493, 1496, 1497
- \@l@dttempcntb
. 183, 332, 333, 338, 342, 346,
350, 353, 376, 377, 384, 388,
392, 394, 402, 403, 1311, 1323,
1344, 1352–1354, 1356, 1375,
1379, 1383, 1390, 1394, 1398,
1428–1430, 1432, 1485, 1486,
3359, 3361, 3363, 3369, 3373,
3377, 3381, 3384, 3471–3473,
3476–3478, 3481, 3489–3491,
3494–3496, 3499, 3560–3562, 3564
- \@lab 700, 3246, 3287
- \@latexerr 3214
- \@led@extranofeet .. 3184, 3193, 3201
- \@led@nofootfalse 3197–3199
- \@led@nofoottrue 3196
- \@led@testifnofoot 3195
- \@lemmacommand@false 866, 893
- \@lemmacommand@true 918
- \@line@num 4198, 4296
- \@listdepth 3637

- \@lock . 236, 453, 530, 532, 534, 547,
649, 650, 652, 653, 669, 670,
672, 1153, 1223, 1283, 1285,
1286, 1288, 1387, 1402, 1404, 1406
- \@lopL 566, 696
- \@lopR 566
- \@makechapterhead .. 4791–4794, 4891
- \@makecol 3166
- \@makefcolumn .. 3217, 3218, 3226, 3228
- \@makeschapterhead .. 4787–4790, 4891
- \@makespecialcolbox 3092
- \@maxdepth 3107, 3116
- \@mem@extranofeet 3188
- \@mem@nofootfalse 3189, 3190
- \@mem@old@ssect 4891
- \@midlist 3087, 3088
- \@minipagefalse 3648
- \@minipagerestore 3638
- \@minus 2717, 2718,
4725, 4733, 4741, 4749, 4757, 4765
- \@mpargs 3628, 3651
- \@mpfn 3634, 3670, 3697
- \@mpfnpos 2565, 3584, 3587
- \@mpfootins 3644,
3654, 3660, 3662, 3666, 3676, 3703
- \@mpfootnotetext ... 3635, 3671, 3698
- \@mplistdepth 3637
- \@nameuse 409, 411, 1589,
1590, 1727, 1834, 1835, 1880,
1988, 2056, 2140, 2144, 2146,
2155, 2158, 2159, 2165, 2169,
2173, 2177, 2200, 2205, 2225,
2237, 2243, 2289, 2293, 2303,
2311, 2356, 2360, 2370, 2378,
2440, 2454, 2462, 2463, 2468,
2477, 2481, 2494, 2599, 2600,
2602, 2603, 2608, 3174, 3175,
3177, 3178, 3180, 3182, 3608, 3610
- \@next@page 735, 736
- \@nl .. 505, 736, 738, 740, 747, 751, 754
- \@nl@reg 505
- \@nobreakfalse 1052, 1179
- \@nobreaktrue 1050, 1054, 1179
- \@noeled@ssectrue 5077
- \@noneed@Footnotefalse 891
- \@noneed@Footnotetrue 2790
- \@nowrindex 3765
- \@oldnbreak 1050, 1052, 1106
- \@opcol 3218, 3236
- \@opxtrafeetii 3143, 3144, 3173
- \@outputbox
. 2582, 2583, 2597, 2598, 3094,
3096, 3097, 3113, 3115, 3141, 3142
- \@outputpage 3227
- \@parboxrestore 1594, 2163, 3633
- \@patchforledchapter 4722
- \@pboxswfalse 3626
- \@pend 566
- \@pendR 566
- \@plus 1605,
2299, 2366, 2717, 2718, 4725,
4727, 4733, 4735, 4741, 4743,
4749, 4751, 4757, 4759, 4765, 4767
- \@ref 684, 762, 771
- \@ref@reg 686
- \@reinserts 3167
- \@schapter 4796
- \@sect 4891
- \@series 467, 471,
2585, 2590, 2674, 2676, 2805,
2814, 2815, 2826, 2828, 2852,
2866, 3146, 3153, 3192, 3200,
3594, 3615, 3620, 3623, 4084,
4090, 4264, 4269, 4275, 4280, 4295
- \@set 597, 796
- \@setminipage 3639
- \@showidx 3773
- \@ssect 4891
- \@startstanza 4012
- \@stopstanza 4012
- \@sw 701, 943, 960, 978
- \@tag 833, 844, 871,
919, 2118, 2119, 2746, 2759, 2794
- \@tempboxa 3206, 3207, 3629, 3651
- \@tempdima 3117, 3627, 3631
- \@templ@d 3550, 3552
- \@templ@n 3551, 3552
- \@textbottom 3099
- \@texttop 3095
- \@toplist 3208, 3209
- \@whilenum 4477
- \@whilesw 3218, 3227
- \@wredindex 3812, 4194
- \@x@sf 2089, 2092, 2100, 2106, 2130, 2136
- \@xloop 1504, 1511
- \@xympar 3347
- ^ 499
- _ 3878, 3880, 4510, 4515, 4523

A

`\abs@line@verse` 5288, 5289, 5292, 5293
`\abs@prevline` 5267, 5268
`\absline@num` 230,
 452, 510, 513, 516, 563, 611,
 614, 623, 637, 659, 681, 691,
 733, 734, 743, 745, 980, 981,
 999–1001, 1214, 1235, 1236,
 1244, 1469, 5257, 5259, 5263,
 5266, 5267, 5275, 5276, 5288, 5292
`\absline@numR` ... 988, 989, 1013–1015
 Abu Kamil Shuja' b. Aslam 9
`\actionlines@list`
 455, 479, 482, 489, 611,
 614, 623, 637, 659, 681, 1266, 1269
`\actions@list`
 . 455, 483, 490, 612, 621, 625,
 627, 639, 648, 661, 668, 682, 1270
`\add@inserts` 1169, 1177, 1458
`\add@inserts@next` 1458
`\add@penalties` 1174, 1479
`\addcontentsline` 4848
`\addfootins` 3170
`\addfootinsX` 2593
`\addtocounter` 1107
`\addtol@den@body` ... 3910, 3932, 3934
 Adelard II 8
`\advancelabel@refs` 3252, 3260
`\advanceline` . 15, 87, 90, 791, 814, 825
`\advancepageno` 3082
`\Aendnote` 17
`\affixline@num` 1159, 1305
`\affixpstart@num` 1167, 1417
`\affixside@note` 1169, 1177, 3528, 3537
`\Afootnote` 16
`\afterlemmaseparator` 21
`\afternote` 23
`\afternumberinfootnote` 20
`\afterruleX` 23
`\aftersymmlinenumber` 20
`\afterXrule` 23
`\allowbreak` 1981, 2045, 2304, 2371
`\ampersand` 28, 3950, 4055
`\appto` 3543, 3544
`\apptocmd` 2096, 2115, 4789,
 4793, 4795, 4796, 4908, 4989, 5001
`\at@every@pend` 1108, 1111
`\at@every@pstart` ... 1038, 1039, 1048
`\AtBeginDocument` 3109, 3283, 3711, 4892
`\AtEveryPend` 11, 1111

`\AtEveryPstart` 11, 1037
`\autopar` 11, 108, 279, 1120
`\autopar@pausetrue` 275
`\autoparfalse` 265, 1121
`\autopartrue` 1134

B

`\ballast` 41
`\ballast@count` . 1230, 1233, 1238, 1479
 Beeton, Barbara Ann Neuhaus Friend 12
`\beforeelectedchapter` 4854
`\beforeledchapter` 4797
`\beforelemmaseparator` 21
`\beforenotesX` 23
`\beforenumberinfootnote` 20
`\beforesymmlinenumber` 20
`\beforeXnotes` 23
`\beginnumbering` 10, 198, 296, 1057, 1131
`\beginnumberingR` 1126
`\Bendnote` 17
`\Bfootnote` 16
`\bfseries` 1042
`\bhooknoteX` 22
`\bhookXendnote` 22
`\bhookXnote` 22
`\body` 1512, 1513, 3952, 4054
`\bodyfootmarkA` 33
`\box` .. 1206, 1208, 1829, 1844, 1913,
 1932, 2458, 2472, 3113, 3207, 3235
`\boxlinenum` 20
`\boxmaxdepth` 3116
`\boxsymmlinenumber` 21
 Bredon, Simon 8
 Breger, Herbert 6, 9, 186
 Brey, Gerhard 8, 9
`\brokenpenalty` 1116
 Burt, John 7
 Busard, Hubert L. L. 8
`\bypage@false` 300, 314, 320
`\bypage@true` 300, 308
`\bypstart@false` 300, 309, 321
`\bypstart@true` 300, 315

C

`\c@addcolcount` 4471
`\c@ballast` 1230, 1238
`\c@firstlinenum`
 359, 1325, 1327, 1330, 1332
`\c@firstsublinenum`
 363, 1312, 1314, 1317, 1319

- `\c@labidx` 3719
- `\c@linenumincrement` .. 359, 1328, 1329
- `\c@mpfootnote` 3634, 3670, 3697
- `\c@page` 738,
740, 746, 749, 751, 754, 3405, 3413
- `\c@pstart` 1042, 3256
- `\c@sublinenumincrement` 363, 1315, 1316
- `\Cendnote` 17
- `\centerline` 4407, 4412,
4418, 4423, 4429, 4434, 4656, 4658
- `\Cfootnote` 16
- `\ch@ck@l@ck` 1342, 1371
- `\ch@cksub@l@ck` 1321, 1371
- `\ch@pt@c` 4774
- `\changes` .. 837, 2833, 2836, 2837, 3535
- `\chapter` 4778, 4784, 4798,
4799, 4891, 5186, 5190, 5196, 5197
- `\chaptermark` 4850, 5187, 5191
- `\char` 3950
- `\chardef` 2669, 3952, 3954
- `\check@pb@in@verse` 1158, 5284
- Chester, Robert of 8
- Claassens, Geert H. M. 9
- class 1 feet 139, 155
- class 2 feet 155, 156
- `\cleaders` 4466
- `\cleardoublepage`
.... 4797, 4798, 4857, 4947, 4949
- `\clearforchapter` ... 4855, 4893, 4895
- `\closeout` 267, 720, 727, 2613
- `\clubpenalty` 1116, 1483
- `\color@begingroup`
1595, 1840, 2164, 2468, 3120, 3630
- `\color@endgroup`
1596, 1840, 2165, 2468, 3124, 3649
- `\columns@position` 217, 218,
289, 290, 2532, 2545, 2555, 2561
- `\columnwidth` 1593, 1800,
2162, 2424, 2537, 2550, 3632, 3685
- `\content` 2734, 2746, 2759, 2774, 2778,
2791, 2794, 3426, 3429, 3433,
3441, 3444, 3448, 3456, 3459, 3463
- Copernicus, Nicolaus 8
- `\count` 1768, 1773, 1785, 1791, 1956,
1959, 2025, 2053, 2257, 2262,
2278, 2281, 2346, 2349, 2410, 2415
- `\countdef` 3082
- `\cr` 1934, 1937
- `\create@edindex@for@memoir` 3753, 3857
- `\create@edindex@notfor@memoir` ..
..... 3811, 3853, 3856, 3860
- `\CRITEXT` 4178
- `\critext` 43, 826, 842, 4180, 4262, 4291
- `\cs` 837, 838, 1165, 2802,
2820, 2836–2838, 2960, 3531, 3533
- `\csdef` 4268, 5186,
5187, 5190, 5191, 5196, 5197,
5205, 5206, 5209, 5210, 5214,
5215, 5223, 5224, 5227, 5228,
5232, 5233, 5240, 5241, 5244, 5245
- `\csgdef` ... 1761, 1779, 1945, 2014,
2247, 2268, 2336, 2403, 2680–
2698, 2703, 2705, 2706, 2708–
2710, 2712–2723, 2769, 2786, 2860
- `\cslet` 2711, 2797
- `\csletcs` 2801, 3593, 3619,
4078, 4088, 4263, 4274, 4279, 4294
- `\csnumdef` 1092, 1094
- `\csnumgdef` 942, 959, 983, 991
- `\csundef` 466, 1183
- `\csuse` 217, 218, 289, 290, 942,
943, 959, 960, 985, 993, 1001,
1015, 1181, 1573, 1574, 1591,
1592, 1603, 1609, 1612–1614,
1620, 1706, 1717, 1721, 1739,
1745, 1748, 1769, 1770, 1774,
1775, 1786, 1787, 1792, 1793,
1796, 1809, 1816, 1821, 1822,
1836, 1837, 1855, 1862–1864,
1872, 1873, 1892, 1898, 1904,
1905, 1920, 1922–1924, 1926,
1964, 1969, 1973, 1977–1979,
1983, 2000, 2006, 2008, 2028,
2033, 2037, 2041–2043, 2046,
2047, 2068, 2074, 2076, 2147,
2148, 2155, 2160, 2161, 2172,
2173, 2183, 2195, 2217, 2223,
2229, 2258, 2259, 2263, 2264,
2286, 2296, 2297, 2303, 2306,
2323, 2329, 2354, 2362, 2364,
2370, 2373, 2390, 2396, 2411,
2412, 2416, 2417, 2420, 2433,
2444, 2450, 2451, 2464, 2465,
2481, 2490, 2507, 2513, 2520,
2532, 2545, 2555, 2561, 2571,
2577, 2584, 2589, 2621, 2708,
2709, 2745, 2758, 2764, 2776–
2778, 2846, 2847, 2862, 2863,
3009, 3015, 3026, 3028–3032,

- 3035, 3036, 3040, 3045, 3049,
3050, 3054, 3059, 3063, 3064,
3069, 3074, 3145, 3152, 3189,
3190, 3198, 3199, 3605, 3613,
3622, 3751, 3752, 3864, 4080,
4081, 4268, 4870, 4886, 5224, 5228
`\csxdef` 980, 988, 1535,
1537, 1541, 1543, 1550, 1553,
1556, 1561, 1564, 1567, 1801, 3076
`\ctab` 4060, 4615, 4706
`\ctabtext` 4064, 4625, 4710
- D**
- `\dcolerr` 4154, 4166
`\dcoli` ... 4124, 4156, 4157, 4514, 4519
`\dcolii` 4125, 4157
`\dcoliii` 4126, 4157
`\dcoliv` 4127, 4158
`\dcolix` 4132, 4159
`\dcolv` 4128, 4158
`\dcolvi` 4129, 4158
`\dcolvii` 4130, 4159
`\dcolviii` 4131, 4159
`\dcolx` 4133, 4159
`\dcolxi` 4134, 4160
`\dcolxii` 4135, 4160
`\dcolxiii` 4136, 4160
`\dcolxiv` 4137, 4161
`\dcolxix` 4142, 4162
`\dcolxv` 4138, 4161
`\dcolxvi` 4139, 4161
`\dcolxvii` 4140, 4162
`\dcolxviii` 4141, 4162
`\dcolxx` 4143, 4162
`\dcolxxi` 4144, 4163
`\dcolxxii` 4145, 4163
`\dcolxxiii` 4146, 4163
`\dcolxxiv` 4147, 4164
`\dcolxxix` 4152, 4165
`\dcolxxv` 4148, 4164
`\dcolxxvi` 4149, 4164
`\dcolxxvii` 4150, 4165
`\dcolxxviii` 4151, 4165
`\dcolxxx` 4153, 4165
`\DeclareOption` 12–18
Dekker, Dirk-Jan 7, 42
`\Dendnote` 17
`\Dfootnote` 16
`\dimen` 782, 783,
785–787, 789, 1769, 1774, 1786,
1792, 1798–1800, 1802, 1939–
1941, 1957, 1960, 2026, 2054,
2258, 2263, 2279, 2282, 2347,
2350, 2411, 2416, 2422–2424, 2427
`\dimen@` 3096, 3098
`\dimexpr` 3984
`\dingdef` 3608, 3610, 3660, 3662
`\disable@familiarnotes` .. 4076, 4108
`\disable@notes` 4106, 4311,
4330, 4346, 4362, 4377, 4394, 4588
`\disable@sidenotes` 4093, 4107
`\disablel@dtabfeet`
..... 4309, 4328, 4344, 4360,
4375, 4392, 4521, 4528, 4533,
4541, 4546, 4554, 4572, 4589, 4713
`\displaystyle` 4212, 4312,
4315, 4347, 4350, 4378, 4381,
4521, 4533, 4546, 4636, 4688, 4689
`\displaywidowpenalty` 1117
`\divide` .. 1315, 1328, 1800, 1940, 2424
`\do@actions` 1215, 1242
`\do@actions@fixedcode` ... 1263, 1276
`\do@actions@next` 1242
`\do@ballast` 1216, 1230
`\do@insidelinehook` .. 1168, 1194, 1196
`\do@line` 1095, 1143
`\do@linehook` 1147, 1193, 1196
`\do@lockoff` 658
`\do@lockoffL` 658
`\do@lockon` 629
`\do@lockonL` 629
`\docsvlist` 1531, 2672, 2855, 2869
`\doedindexlabel` 3724, 3767, 3844, 4191
`\doendnotes` 30, 2662
`\doinsidelinehook` 1193
`\dolinehook` 1193
`\dolistloop`
.. 467, 471, 2585, 2590, 2814,
2826, 2852, 2866, 3146, 3153,
3192, 3200, 3548, 3568, 3575,
3594, 3615, 3620, 3623, 4084,
4090, 4264, 4269, 4275, 4280, 4295
`\doreinextrafeeti` ... 2581, 2601, 3148
`\doreinextrafeetii` .. 3149, 3151, 3176
`\dosplits` 1931
Downes, Michael 41, 116, 118
`\doxtrafeet` 3130
`\doxtrafeeti`
..... 2581, 2596, 3132, 3135, 3136
`\doxtrafeetii` .. 3132, 3135, 3136, 3140

- \dp 1585,
1827, 1842, 2456, 2470, 3096, 3117
- \dummy@edtext 818, 828,
5187, 5191, 5206, 5210, 5224, 5228
- \dummy@edtext@showlemma 819, 4903,
4984, 4996, 5186, 5190, 5196, 5197
- \dummy@ref 685, 695
- \dummy@text 817, 826
- E**
- \edaftertab
.. 37, 200, 4066, 4561, 4597, 4615
- edarrayc (environment) 35, 4705
- edarrayl (environment) 35, 4705
- edarrayr (environment) 35, 4705
- \EDATAB 4655, 4663
- \edatab 4067, 4661
- \edatabell 4068, 4645
- \edatleft 37, 4069, 4446
- \edatright 37, 4070, 4454
- \edbeforetab
.. 37, 200, 4065, 4561, 4597, 4615
- \edfilldimen
.... 4468, 4478, 4479, 4481, 4504
- \edfont@info 908, 911, 915
- \EDINDEX 4184
- \edindex 34, 3753, 4184,
4265, 4270, 4276, 4281, 4293,
4302, 4321, 4339, 4355, 4370, 4387
- \edindexlab 34, 3719,
3725, 3728, 3887, 3891, 3896, 3898
- \EDLABEL 4182
- \edlabel 30, 824, 3245,
3725, 4182, 4283, 4299, 4301,
4320, 4338, 4354, 4369, 4386,
4520, 4527, 4532, 4540, 4545, 4553
- \edmakelabel 32, 3345
- \edpageref 30, 3296
- \edrowfill . 36, 4073, 4284, 4287, 4502
- \EDTAB 4651, 4687
- \edtabcolsep 36, 4241,
4316, 4335, 4350, 4366, 4382,
4399, 4479, 4535, 4548, 4557, 4673
- \EDTABINDENT 4668, 4681
- \edtabindent 4509,
4513, 4518, 4529, 4542, 4555, 4677
- \EDTABtext 4695
- edtabularc (environment) 35, 4709
- edtabularl (environment) 35, 4709
- edtabularr (environment) 35, 4709
- \EDTEXT 4178
- \edtext 15, 828, 868, 2112,
2241, 3401, 3402, 3420, 4178,
4273, 4292, 4902, 4903, 4909,
4983, 4984, 4990, 4995, 4996,
5002, 5186, 5187, 5190, 5191,
5196, 5197, 5206, 5210, 5224, 5228
- \edvertdots 37, 4072, 4465
- \edvertline 37, 4071, 4463
- \Eendnote 17
- \Efootnote 16
- \eled@chapter
.... 5082, 5086, 5134, 5138, 5181
- \eled@section
.... 5095, 5099, 5147, 5151, 5181
- \eled@sectioning@out
.. 224, 267, 5075, 5085, 5098,
5111, 5123, 5137, 5150, 5163, 5176
- \eled@sectioningR@out 5081, 5094,
5107, 5119, 5133, 5146, 5159, 5172
- \eled@sections@@ 221, 1160, 1179
- \eled@subsection
.... 5108, 5112, 5160, 5164, 5181
- \eled@subsubsection
.... 5120, 5124, 5173, 5177, 5181
- \eledchapter 5078
- \eledchapter* 5078
- \Eledmac 2799
- \eledmac@error 56, 58, 60, 62,
74, 97, 100, 103, 106, 108, 160,
162, 165, 167, 169, 171, 174, 177
- \eledmac@warning
.... 55, 77, 79, 81, 83, 85, 87,
90, 93, 95, 111, 113, 115, 118,
120, 122, 124, 127, 130, 134,
136, 141, 143, 148, 150, 154, 157
- \eledsection 4949, 5078
- \eledsection* 5078
- \eledsubsection 5078
- \eledsubsection* 5078
- \eledsubsubsection 5078
- \eledsubsubsection* 5078
- \emph 3751, 3752
- \empty 181, 186, 252, 255,
434, 435, 479, 856, 882, 906,
923, 927, 933, 944, 961, 1004,
1018, 1064, 1266, 1324, 1340,
1460–1462, 1473, 1505, 3247, 3965
- \enablel@dtabfeet 4569,
4585, 4605, 4613, 4623, 4631, 4713

- \end@lemmas . . . 816, 856, 857, 882, 883
 - \endashchar 25, 1617, 1697, 2654
 - \endgraf 1090, 1136, 1140
 - \endline@num 460, 704, 710
 - \endlock 14, 806, 823, 4021, 4038, 4046
 - \endminipage 3641
 - \endnumbering . . 10, 201, 245, 276, 295
 - \endpage@num 460, 703, 710
 - \endprint 30, 2621, 2665
 - \endquotation 4722
 - \endquote 4722
 - \endstanzaextra 28, 4012
 - \endsub 14, 782, 822
 - \endsubline@num 460, 705, 711
 - \enlargethispage 5269, 5277
 - \enskip 2622
 - \enspace . 2173, 2303, 2370, 2481, 2622
 - environments:
 - edarrayc 35, 4705
 - edarrayl 35, 4705
 - edarrayr 35, 4705
 - edtabularc 35, 4709
 - edtabularl 35, 4709
 - edtabularr 35, 4709
 - ledgroup 29, 3668
 - ledgroupsize 29, 3682
 - minipage 29
 - Euclid 8
 - \ExecuteOptions 19
 - \expandonce 1556, 1567, 2119, 2230,
 - 2746, 2759, 2778, 2794, 3429,
 - 3433, 3444, 3448, 3459, 3463, 3812
 - \extensionchars 40, 184, 207, 283
 - \extractendline@ 3000, 3003
 - \extractendsubline@ 3001, 3003
 - \extractline@ 2998, 3003, 3006
 - \extractsubline@ . . . 2999, 3003, 3006
- F**
- \f@encoding 915
 - \f@family 915
 - \f@series 915
 - \f@shape 915
 - \f@x@l@cks 1365, 1371
 - Fairbairns, Robin 33
 - \falseverse 154, 4012
 - \first@linenum@out@false . . . 715, 721
 - \first@linenum@out@true 715
 - \firstlinenum 12, 14, 368
 - \firstseries 2809
 - \firstsublinenum 12, 14, 368
 - \fix@page 506, 552
 - \flag@end 757,
 - 852, 862, 863, 878, 888, 889, 896
 - \flag@start
 - 757, 851, 852, 863, 877, 878, 889
 - \flagstanza 28, 4050
 - \floatingpenalty 1584
 - \flush@notes 1100, 1503, 3601
 - \flush@notesR 3599
 - \fnpos 33, 2565
 - Folkerts, Menso 8
 - \fontencoding 1517
 - \fontfamily 1517
 - \fontseries 1517
 - \fontshape 1517
 - \footfootmarkA 33
 - \footfudgefiddle 41, 1795, 1800, 2424
 - \footins . 3112, 3119, 3123, 3157, 3197
 - \footnormal 1751, 2770, 3172
 - \footnormalX 33, 2246, 2595, 2787
 - \footnote@luatexpardir 1601
 - \footnote@luatextextdir . 1600, 1622
 - \footnoteA 33
 - \footnoteB 33
 - \footnoteC 33
 - \footnoteD 33
 - \footnoteE 33
 - \footnotelang@lua . . 1533, 2738, 2751
 - \footnotelang@poly . . 1547, 2742, 2755
 - \footnoteoptions@
 - 1520, 2743, 2747, 2756, 2761
 - \footnoterule . . 1719, 2197, 3122, 3656
 - \footnotesize 2985, 3398, 3399
 - \footparagraph 19, 1778
 - \footparagraphX 33, 2402
 - \footsplitskips
 - 1575, 1582, 1823, 1838, 1965,
 - 2029, 2149, 2287, 2355, 2452, 2466
 - \footthreecol 19, 1944
 - \footthreecolX 33, 2335
 - \foottwocol 19, 2013
 - \foottwocolX 33, 2267
 - \foottwocolX 2267
 - \fullstop 25, 424, 1617,
 - 1694, 1696, 1698, 1700, 2653, 2657
- G**
- \g@addto@macro 2596,
 - 2601, 2604, 2607, 3163, 3164,

3173, 3176, 3179, 3181, 3188, 3754
 Gädeke, Nora 9
`\get@edindex@hyperref` ... 3821, 3874
`\get@index@command`
 ... 3736, 3817, 3826, 3883, 3894
`\get@linelistfile` 476, 494
`\getline@num` 1151, 1213
`\gl@p` 446, 482, 483, 857,
 883, 910, 945, 962, 1007, 1021,
 1269, 1270, 1466, 1470, 1506, 3250
`\gl@poff` 446, 447

H

`\hangafter` 4008
`\hangindentX` 22
`\hangingsymbol` 28, 29, 3939, 3946
`\hb@xt@` 1168, 1171,
 1206, 1208, 4504, 4509, 4513,
 4518, 4529, 4542, 4555, 4634, 4636
`\hfilneg` 1933
`\Hilfsbox` 4119
`\hilfsbox` 4119, 4174, 4175,
 4212, 4224, 4298, 4312, 4331,
 4347, 4363, 4378, 4395, 4520,
 4522, 4527, 4531, 4532, 4534,
 4540, 4544, 4545, 4547, 4553, 4556
`\hilfscount` 4119, 4672–4674, 4680
`\HILFSskip` 4665
`\Hilfsskip` 4510,
 4514, 4515, 4519, 4522, 4523,
 4530, 4531, 4534–4536, 4543,
 4544, 4547–4549, 4556–4558,
 4665, 4671, 4673, 4679, 4683, 4684
`\hilfsskip`
 . 4119, 4297, 4298, 4315, 4334,
 4350, 4366, 4381, 4398, 4682–4684
`\hsize@fornote` 2528, 2533,
 2536, 2537, 2541, 2546, 2549, 2550
`\hsizethreecol` 22
`\hsizethreecolX` 22
`\hsizetwocol` 22
`\hsizetwocolX` 22
`\Hy@temp@A` 3785, 3786
`\HyInd@ParenLeft` 3786
`\hyperlink`
 3292, 3293, 3863, 3864, 3867, 3875
`\hyperlinkformat` 3861, 3871
`\hyperlinkformatR` 3870
`\hyperlinkR` 3866
`\hyperpage` 3752

`\hypertarget` 3257

I

`\if@edindex@fornote@`
 3731, 3775, 3787, 3797, 3816, 3825
`\if@edindex@fornote@true` 3731
`\if@edtext@` 836, 841, 946, 963
`\if@eled@sectioning`
 4859, 4861, 4876, 4894, 4948, 4966
`\if@fcolmade` 3218, 3227
`\if@firstcolumn` 1346, 1422, 3220, 3554
`\if@led@nofoot` 3184, 3205
`\if@lemmacommand@` 920, 947, 964
`\if@nobreak` 1049
`\if@noeled@sec` 222, 266, 5076
`\if@noneed@Footnote` 757
`\if@openright` 4857, 4947, 4949
`\if@RTL` 29, 54, 852,
 863, 878, 889, 1184, 1549, 1560,
 1829, 2155, 4863, 4879, 4965, 4967
`\ifautopar` 275, 1068,
 1120, 4729, 4737, 4745, 4753,
 4761, 4769, 4805, 4818, 4827, 4840
`\ifautopar@pause` 279, 1142
`\IfBeginWith` 3740, 3745
`\ifbypage@` 300, 557, 1247, 1668
`\ifbypstart@` 300, 1096
`\ifcsdef` ... 470, 999, 1013, 1921, 3020
`\ifcsempy`
 1612, 1862, 1923, 1977, 2041, 3024
`\ifcsequal` 3022
`\ifcsstring` 1869, 1870, 1901, 1902,
 2486, 2487, 2516, 2517, 2813, 2825
`\ifcsundef` 982, 990
`\ifdef` 29, 54, 3257, 3292, 3875
`\ifdefstring` 1622
`\ifdim` 783, 785, 787, 789, 2088, 4174, 4672
`\ifdimequal` 1703, 1806,
 2180, 2430, 3029, 3036, 3050, 3064
`\iffirst@linenum@out@` 715, 719
`\ifFN@bottom` 3109, 3119
`\ifhbox` 1912, 1917
`\ifhmode` 2099, 2106, 2129, 2136
`\ifinserthangingsymbol` .. 3942, 5285
`\ifinstanza` 1070, 1137, 3939, 3945, 5285
`\ifl@d@dash` 1640, 1697, 2654
`\ifl@d@elin` 1640,
 1686, 1699, 1700, 2644, 2656, 2657
`\ifl@d@esl` 1640, 1700, 2657

- \ifl@d@pnum 1640, 1674, 1694, 1698, 2632, 2655
- \ifl@d@ssub 1640, 1696, 2653
- \ifl@dend@ 2610, 2616
- \ifl@dmemoir 43, 4185, 4773, 4854
- \ifl@dpaging 188, 1576, 2150
- \ifl@dpairing .. 188, 210, 249, 269, 285, 1576, 1630, 1711, 1728, 1734, 1881, 1887, 1989, 1995, 2057, 2063, 2150, 2189, 2206, 2212, 2312, 2318, 2379, 2385, 2495, 2502, 3597, 3606, 3658, 4725, 4733, 4741, 4749, 4757, 4765
- \ifl@dskipnumber 809, 1307
- \ifl@dstartendok 4475, 4485
- \ifl@imakeidx 46, 3712, 3815
- \ifl@indextools 48, 3715
- \iflabelpstart 1045, 1078
- \ifledfinal 4, 30, 40
- \ifledgroupnotesL@ 1306, 3709
- \ifledgroupnotesR@ 3709
- \iflednopbinverse 4, 5284, 5285
- \ifledplinenum 2984
- \ifledRcol 188, 760, 847, 873, 897, 941, 979, 1123, 1631, 2736, 3360, 3428, 3443, 3458, 3598, 3607, 3659, 3884, 3895, 5080, 5093, 5106, 5118, 5132, 5145, 5158, 5171
- \ifledRcol@ 133, 140, 147, 188, 998, 3470, 3488
- \ifledsecnolinenum 4, 4727, 4735, 4743, 4751, 4759, 4767
- \ifleftnoteup 3510, 3523
- \ifluatex 1599, 1621, 2737, 2750
- \ifnoquotation@ 4, 4801
- \ifnoteschanged@ 259, 464
- \ifnumberedpar@ . 1032, 1059, 1086, 2111, 2117, 2228, 2240, 2735, 2793, 2794, 3349, 3427, 3442, 3457
- \ifnumbering 187, 199, 246, 278, 303, 1055, 1083, 1129
- \ifnumberingR 188, 1124
- \ifnumberline 903, 1217, 1306
- \ifnumberpstart 1043, 1069, 1103, 1137
- \ifnumequal 1097, 1920, 1922, 3541
- \ifnumgreater 1009, 1023, 3549, 3569, 3576
- \ifodd 1356, 1432, 3405, 3413, 3481, 3499, 3564
- \ifparapparatus@ 4
- \ifparledgroup 4, 1728, 1733, 1881, 1886, 1989, 1994, 2057, 2062, 2206, 2211, 2312, 2317, 2379, 2384, 2495, 2501, 3606, 3657
- \ifpst@rtedL 188
- \ifpstartnum 1443, 1446, 1451
- \ifreportnoidxfile 3760
- \ifrightnoteup 3421, 3518
- \ifshowindexmark 3773
- \ifsidepstartnum 1071, 1417
- \ifstrempty 1048, 1108, 2851, 2865, 3862, 5182, 5184, 5195, 5202, 5203, 5213, 5220, 5221, 5231, 5238, 5239, 5243
- \IfStrEq 732, 742, 1152, 1163, 3131, 3134, 3584, 3587, 5265, 5274, 5287, 5291
- \IfStrEqCase 754
- \ifstrequal 950, 967, 1522, 1534, 1548, 2844, 2861
- \ifsublines@ 422, 451, 542, 571, 576, 582, 597, 615, 624, 638, 660, 709, 711, 1218, 1255, 1310, 3265, 3289
- \IfSubStr 3817, 3826, 3882
- \iftoggle . 1612, 1705, 1808, 1862, 1977, 2041, 2182, 2432, 2530, 2543, 2554, 2560, 3002, 3008, 3013, 3018, 3037, 3041, 3042, 3045, 3051, 3052, 3055, 3056, 3059, 3065, 3066, 3070, 3071, 3074
- \ifvbox 1093, 3091, 3159
- \ifvmode 3251
- \ifvoid 2584, 2589, 2599, 2602, 2608, 3112, 3145, 3152, 3157, 3174, 3177, 3182, 3189, 3190, 3197–3199, 3605, 3622, 3644, 3676, 3703
- \ifwidthliketwocolumns .. 4, 216, 288
- \imki@wrindexentry 52, 53, 3818, 3819, 3822
- \indexentry 3828, 3831, 3835
- \indtl@wrindexentry 52, 53
- \initnumbering@reg 198
- \initnumbering@sectcmd 215, 287, 4722
- \inplaceoflemmaseparator 21
- \inplaceofnumber 20
- \InputIfFileExists 223, 495

- \insert 1572, 1819, 1963,
2027, 2146, 2285, 2353, 2448,
2589, 2603, 3152, 3157, 3159, 3178
 - \insert@count 683, 684,
771, 773, 848, 874, 1524, 1536,
1538, 1551, 1554, 1557, 2121,
2232, 2760, 3435, 3450, 3465
 - \insert@countR 762, 764, 847,
873, 1528, 1542, 1544, 1562,
1565, 1568, 2748, 3431, 3446, 3461
 - \inserthangingsymbol 1171, 3943
 - \inserthangingsymbolfalse 1156
 - \inserthangingsymboltrue 1154
 - \inserthangingymbol 3942
 - \insertlines@list
... 252, 455, 488, 691, 1462, 1466
 - \insertparafootsep .. 1858, 1919, 2479
 - \inserts@list 1063, 1457,
1460, 1470, 1505, 1506, 1523,
1535, 1537, 1550, 1553, 1556,
2120, 2231, 2759, 3434, 3449, 3464
 - \inserts@listR
..... 1527, 1541, 1543, 1561,
1564, 1567, 2746, 3430, 3445, 3460
 - \instanzafalse 3941, 4041
 - \instanzatrue 4015
 - \interfootnotelinepenalty 1583
 - \interlinepenalty
..... 1117, 1490, 1583, 4021
 - \interparanoteglue 2992
 - \ipn@skip 2992
 - \itemcount@ 134, 136,
141, 143, 148, 150, 3539, 3541,
3546, 3549, 3567, 3569, 3574, 3576
- J**
- Jayaditya 9
- K**
- Kabelschacht, Alois 104
- Krukov, Alexej 75
- L**
- \l@d@w@rindexhyp 3771
 - \l@d@add 928, 930, 934, 936
 - \l@d@dashfalse 1649, 1667, 2627
 - \l@d@dashtrue
1671, 1677, 1689, 2630, 2635, 2647
 - \l@d@elinfalse 1645, 1674, 2632
 - \l@d@elintrue .. 1674, 1676, 2632, 2634
 - \l@d@end
2610, 2612, 2613, 2619, 2669, 2792
 - \l@d@err@UnequalColumns 4255
 - \l@d@eslfalse
.... 1647, 1683, 1686, 2641, 2644
 - \l@d@esltrue ... 1686, 1688, 2644, 2646
 - \l@d@index 3756, 3758, 4187
 - \l@d@makecol 3103, 3166, 3236
 - \l@d@nums . 846, 908, 911, 922, 923,
936, 2744, 2746, 2757, 2759, 2793
 - \l@d@pnumfalse 1641, 1667, 2627
 - \l@d@pnumtrue 1670, 2629
 - \l@d@reinserts 3156, 3167
 - \l@d@section 2619, 2621
 - \l@d@set 604, 803
 - \l@d@ssubfalse 1643, 1679, 2637
 - \l@d@ssubtrue 1681, 2639
 - \l@d@wrindexm@m 3770, 3771
 - \l@dampcount 4114,
4251, 4253, 4258, 4518, 4528,
4529, 4541, 4542, 4577, 4595, 4633
 - \l@dbegin@stack 3916, 3927–3929
 - \l@dbfnote 2112, 2116
 - \l@dcheckcols 4208, 4220, 4248
 - \l@dcheckstartend 4474, 4485
 - \l@dchset@num 509, 512, 604
 - \l@dcolcount 4114, 4156,
4168, 4169, 4207, 4209, 4219,
4221, 4249, 4253, 4258, 4303,
4305, 4322, 4324, 4340, 4341,
4356, 4357, 4371, 4372, 4388,
4389, 4441, 4442, 4518, 4528,
4529, 4541, 4542, 4573, 4575,
4590, 4592, 4633, 4639, 4669, 4678
 - \l@dcollect@@body 3918, 3926
 - \l@dcollect@body
.... 3913, 4705–4707, 4709–4711
 - \l@dcolwidth 4156, 4174, 4175,
4297, 4440, 4504, 4530, 4543,
4634, 4636, 4641, 4650, 4671, 4672
 - \l@dcsnote 3420, 3421
 - \l@dcsnotetext
1198, 3484, 3500, 3505, 3548, 3551
 - \l@dcsnotetext@l 1198, 3482, 3551, 3568
 - \l@dcsnotetext@r 1198, 3502, 3551, 3575
 - \l@ddodoreinxtrafeet 3147, 3158, 3164
 - \l@ddofootinsert 3104, 3110
 - \l@ddoxtrafeet 3127, 3130, 3163
 - \l@dedbeginmini 3179, 3582, 3592

- \l@dedendmini 3181, 3585, 3588, 3589, 3592
- \l@emptyd@ta 1148, 1198
- \l@dend@close 2612, 2662
- \l@dend@false 2610, 2613
- \l@dend@open 2612, 2617
- \l@dend@stuff ... 208, 284, 2615, 2668
- \l@dend@true 2610, 2612
- \l@denvbody 3908, 3911, 3914–3916
- \l@dfambeginmini ... 2604, 3582, 3618
- \l@dfamendmini 2607, 3585, 3588, 3589, 3618
- \l@dfeetbeginmini 3582, 3636, 3672, 3699
- \l@dfeetendmini 3582, 3647, 3679, 3706
- \l@dgetline@margin 329
- \l@dgetlock@disp 373, 401
- \l@dgetref@num 3296, 3297, 3299, 3300, 3302, 3303, 3305, 3306, 3313, 3335, 3340
- \l@dgetsidenote@margin 3356
- \l@dobblearg 4199
- \l@dobbledarg 4199
- \l@dobbleoptarg 4200
- \l@dlabel@parse 3319, 3322
- \l@dld@ta 1168, 1198, 1345, 1423, 1435, 4867, 4883, 4886
- \l@dldp@rbox 1206, 3391, 3509
- \l@dlsn@te 1170, 1205, 4868, 4884, 4886
- \l@dlsnote 3401, 3421
- \l@dmake@labels 3256, 3275, 3284
- \l@dmemoirfalse 44
- \l@dmemoirtrue 44
- \l@dmodforcritext 4261, 4714
- \l@dmodforedtext 4272, 4717
- \l@dnullfills 4282, 4562, 4580, 4598, 4608, 4616, 4626
- \l@dnumpstartsL 188, 211, 229, 270, 1066, 1160, 1178, 1181, 1183, 5086, 5099, 5112, 5124, 5138, 5151, 5164, 5177
- \l@dnumpstartsR 5082, 5095, 5108, 5120, 5134, 5147, 5160, 5173
- \l@dold@xympar 3347
- \l@doldold@footnotetext 2110
- \l@dp@rsefootspec 1650, 3733
- \l@dpagingfalse 188
- \l@dpagingtrue 188
- \l@dpairingfalse 188
- \l@dpairingtrue 188
- \l@dparsedendline 1650, 3730
- \l@dparsedendpage 1650, 3730
- \l@dparsedendsub 1650
- \l@dparsedstartline 1650, 3729
- \l@dparsedstartpage 1650, 3729
- \l@dparsedstartsub 1650
- \l@dparsefootspec 1650
- \l@dpush@begins 3923, 3927
- \l@drd@ta 1171, 1198, 1345, 1425, 1433, 4867, 4870, 4883
- \l@dref@undefined 3296, 3299, 3302, 3305, 3308
- \l@drestorefills 4282, 4566, 4582, 4602, 4610, 4620, 4628
- \l@drestoreforcritext ... 4261, 4715
- \l@dstoreforedtext 4272, 4718
- \l@drp@rbox 1208, 3391, 3517
- \l@drsn@te 1172, 1205, 4868, 4870, 4884
- \l@drsnote 3402, 3421
- \l@dsetmaxcolwidth .. 4173, 4214, 4226
- \l@dskipnumberfalse 809, 1308
- \l@dskipnumbertrue 809, 1299
- \l@dstartendokfalse . 4489, 4493, 4497
- \l@dstartendoktrue 4487
- \l@dtabaddcols 4473, 4503
- \l@dtabnoexpands 829, 4058
- \l@dunboxmpfoot 3645, 3653, 3677, 3704
- \l@dunhbox@line 1143, 1171, 1186, 1189
- \l@dzeropenalties .. 1081, 1089, 1115
- \l@imakeidxtrue 47, 51
- \l@indextoolstrue 50
- \l@prev@nopb 233, 733, 744, 5252, 5259, 5260, 5266, 5275
- \l@prev@pb 232, 5251, 5257, 5258, 5263
- Lück, Uwe 7
- \label 32
- \label@refs 3248, 3250, 3256, 3263, 3266, 3268, 3270
- \labelpstartfalse 1037
- \labelpstarttrue 1037
- \labelref@list . 3241, 3247, 3250, 3289
- \labelrefsparseline 3260
- \labelrefsparsesubline 3260
- \language name 1556, 1567
- \last@page@num 552, 5286
- \lastbox . 1135, 1150, 1850, 1911, 1916
- \lastkern 2088
- \lastskip 782, 786
- Lavagnino, John 6, 8
- Leal, Jeronimo@Leal, Jerónimo 7

- \led 157
- \led@check@nopb 1152, 1163, 5263
- \led@check@pb 1152, 1163, 5263
- \led@err@AutoparNotNumbered
..... 96, 1125, 1130
- \led@err@EdtextWithoutFootnote .
..... 170, 766, 775
- \led@err@HighEndColumn ... 159, 4494
- \led@err@LineationInNumbered 73, 304
- \led@err@LowStartColumn .. 159, 4490
- \led@err@ManyLeftnotes ... 132, 3569
- \led@err@ManyRightnotes .. 132, 3576
- \led@err@ManySidenotes ... 132, 3549
- \led@err@NumberingNotStarted 57, 263
- \led@err@NumberingShouldHaveStarted
..... 57, 294
- \led@err@NumberingStarted ... 57, 200
- \led@err@PendNoPstart 96, 1087
- \led@err@PendNotNumbered ... 96, 1084
- \led@err@PstartInPstart ... 96, 1060
- \led@err@PstartNotNumbered . 96, 1056
- \led@err@ReverseColumns .. 159, 4498
- \led@err@TooManyColumns .. 159, 4170
- \led@err@UnequalColumns 159
- \led@error@ImakeidxAfterEledmac .
..... 173, 3713
- \led@error@IndextoolsAfterEledmac
..... 176
- \led@error@indextoolsAfterEledmac
..... 3716
- \led@mess@NotesChanged 63, 260
- \led@mess@SectionContinued .. 71, 282
- \led@nopb 5255, 5257
- \led@nopbnum 5256, 5257
- \led@pb 5253, 5257
- \led@pb@setting 732, 742, 754, 1152,
1163, 5261, 5265, 5274, 5287, 5291
- \led@pbnum 5254, 5257
- \led@toksa 436, 444
- \led@toksb 436, 443–445
- \led@war@FalseverseDeprecated ..
..... 153, 4026
- \led@war@ledxxxDeprecated
..... 153, 4724, 4732,
4740, 4748, 4756, 4764, 4772, 4781
- \led@warn@AddfootinsObsolete ...
..... 126, 3171
- \led@warn@Addfootinsobsolete ... 123
- \led@warn@AddfootinsXObsolete ..
..... 123, 2594
- \led@warn@AddfootinsXobsolete .. 123
- \led@warn@BadAction 110, 1301
- \led@warn@BadAdvancelineLine 86, 591
- \led@warn@BadAdvancelineSubline .
..... 86, 585
- \led@warn@BadLineation 76, 324
- \led@warn@BadLinenummargin .. 76, 352
- \led@warn@BadLockdisp 76, 379
- \led@warn@BadSetline 92, 794
- \led@warn@BadSetlinenum 92, 801
- \led@warn@BadSidenotemargin 119, 3383
- \led@warn@BadSublockdisp 76, 405
- \led@warn@DuplicateLabel .. 112, 3278
- \led@warn@NoIndexFile 121, 3761
- \led@warn@NoLineFile 84, 500
- \led@warn@NoMarginpars ... 117, 3350
- \led@warn@RefUndefined ... 112, 3310
- \led@warn@SeriesStillExist 129, 2676
- \ledchapter 4722
- \ledchapter* 4722
- \ledfinalfalse 14
- \ledfinaltrue 13
- \ledfootinsdim 1751, 2722, 2723
- ledgroup (environment) 29, 3668
- ledgroupsized (environment) . 29, 3682
- \ledinnernote 32, 3401
- \ledinnote 3751
- \ledinnotehyperpage 3751
- \ledleftnote 32, 3401, 4095, 4098, 4103
- \ledlinenum 417
- \ledllfill 1171, 1210, 3686, 3690
- \ledlsnotefontsetup ... 32, 3394, 3508
- \ledlsnotesep 32, 1206, 3394
- \ledlsnotewidth 32, 3394, 3508
- \lednopb 39, 5253
- \lednopbinversetrue 16, 40
- \lednopbnum 5253, 5293
- \ledouternote 32, 3412
- \ledouterote 3401
- \ledpb 39, 5253
- \ledpbnum 5253, 5289
- \ledpbsetting 40, 5261
- \ledplinenumtrue 2987
- \ledrightnote 32, 3401, 4094, 4097, 4102
- \ledrlfill 1171, 1210, 3687, 3694
- \ledrsnotefontsetup ... 32, 3394, 3516
- \ledrsnotesep 32, 1208, 3394
- \ledrsnotewidth 32, 3394, 3516
- \ledsecnolinenumbertrue 17
- \ledsection 4722

- `\ledsection*` [4722](#)
 - `\ledsectnomark` [4849](#)
 - `\ledsectnotoc` [4848](#)
 - `\ledsetnormalparstuff`
..... [1598](#), [1859](#), [2171](#), [2480](#)
 - `\ledsidenote` [32](#), [3401](#), [4096](#), [4099](#), [4104](#)
 - `\ledsubsection` [4722](#)
 - `\ledsubsection*` [4722](#)
 - `\ledsubsubsection` [4722](#)
 - `\ledsubsubsection*` [4722](#)
 - `\left` [4448](#), [4451](#), [4456](#), [4459](#)
 - `\leftctab` [4517](#), [4617](#)
 - `\leftlinenum`
.. [14](#), [417](#), [1347](#), [1359](#), [4865](#), [4881](#)
 - `\leftlinenumR` [4866](#), [4882](#)
 - `\leftltab` [4508](#), [4599](#)
 - `\leftmargin` [4810](#), [4811](#), [4832](#), [4833](#)
 - `\leftnoteupfalse` [32](#)
 - `\leftnoteuptrue` [3524](#)
 - `\leftpstartnum` [1417](#)
 - `\leftrtab` [4512](#), [4563](#)
 - `Leibniz` [9](#)
 - `\lemma` [17](#), [917](#)
 - `\lemmaseparator` [21](#), [2991](#)
 - `\letsforverteilen` [4290](#),
[4314](#), [4333](#), [4349](#), [4365](#), [4380](#), [4397](#)
 - `\line@list` [255](#), [455](#), [487](#), [711](#), [906](#), [910](#)
 - `\line@list@stuff` [207](#), [283](#), [717](#)
 - `\line@margin` [329](#), [1352](#), [1428](#)
 - `\line@num` . [136](#), [143](#), [150](#), [234](#), [421](#),
[449](#), [514](#), [548](#), [558](#), [589](#), [590](#),
[592](#), [600](#), [605](#), [606](#), [618](#), [704](#),
[708](#), [1224](#), [1248](#), [1258](#), [1323](#),
[1325](#), [1326](#), [1335](#), [1336](#), [1922](#), [3288](#)
 - `\line@numR` [134](#), [141](#), [148](#)
 - `\line@set` [924](#), [925](#)
 - `\lineation` [13](#), [74](#), [77](#), [302](#)
 - `\lineinfo@` [3003](#), [3006](#), [3076](#)
 - `\linenum` [17](#),
[921](#), [3342](#), [4198](#), [4266](#), [4277](#), [4296](#)
 - `\linenum@out` [714](#),
[720](#), [722](#), [727](#), [728](#), [736](#), [738](#),
[740](#), [747](#), [749](#), [751](#), [754](#), [770](#),
[784](#), [788](#), [791](#), [796](#), [803](#), [806](#),
[807](#), [813](#), [900](#), [943](#), [3246](#), [5253](#)–[5256](#)
 - `\linenum@outR` [761](#), [898](#), [960](#)
 - `\linenumberlist` .. [14](#), [181](#), [1324](#), [1336](#)
 - `\linenumberstyle` [15](#), [408](#)
 - `\linenumincrement` [12](#), [14](#), [368](#)
 - `\linenummargin` [13](#), [79](#), [329](#)
 - `\linenumr@p` [408](#)
 - `\linenumrep` [408](#),
[421](#), [1695](#), [1699](#), [2652](#), [2656](#), [3288](#)
 - `\linenumsep`
.. [14](#), [417](#), [1447](#), [1452](#), [3396](#), [3397](#)
 - `\lineref` [30](#), [3299](#)
 - `\linewidth` [1168](#), [2525](#), [2528](#), [2529](#), [2542](#)
 - `\list@clear` [435](#), [487](#)–[492](#), [1063](#)
 - `\list@clearing@reg` [475](#), [486](#)
 - `\list@create` [434](#),
[455](#)–[458](#), [816](#), [938](#), [939](#), [1457](#), [3241](#)
 - `\listadd` [2813](#), [2825](#)
 - `\listcsgadd` [5199](#), [5217](#), [5235](#), [5247](#)
 - `\listead` [2812](#), [2827](#)
 - `\listgadd` [3482](#), [3484](#), [3500](#), [3502](#), [3505](#)
 - `\listxadd` [563](#), [2805](#), [5257](#)–[5260](#)
 - `\lock@disp` [373](#), [1389](#), [1393](#), [1397](#)
 - `\lock@off` [631](#), [632](#), [658](#), [807](#)
 - `\lock@on` [629](#), [806](#)
 - `\lockdisp` [15](#), [81](#), [373](#)
 - `Lorch, Richard` [9](#)
 - `\Lpack` [3849](#)
 - `\ltab` [4061](#), [4597](#), [4705](#)
 - `\ltabtext` [4063](#), [4607](#), [4709](#)
 - `\luatexpardir` [1537](#), [1543](#), [1601](#)
 - `\luatextextdir` [1535](#), [1541](#), [1600](#)
 - `Luecking, Dan` [46](#)
- M**
- `\m@m@makecolfloats` [3086](#), [3105](#)
 - `\m@m@makecolintro` [3086](#)
 - `\m@m@makecoltext` [3086](#), [3106](#)
 - `\m@m@dodoreinextrafeet` [3164](#)
 - `\m@m@doextrafeet` [3163](#)
 - `\m@m@mf@check` [2087](#), [2101](#), [2131](#)
 - `\m@m@mf@prepare`
.... [2084](#), [2096](#), [2105](#), [2135](#), [2778](#)
 - `\M@ssect` [4891](#)
 - `\m@th` [4466](#)
 - `\makehboxofhboxes`
.... [1871](#), [1903](#), [1908](#), [2488](#), [2518](#)
 - `\makememindexhook` [3754](#)
 - `\managestanza@modulo` [3971](#), [4002](#)
 - `\marg` [839](#), [2838](#)
 - `\marginparwidth` [3394](#), [3395](#)
 - `\marks` [1729](#)–[1731](#), [1882](#)–[1884](#), [1990](#)–
[1992](#), [2058](#)–[2060](#), [2207](#)–[2209](#),
[2313](#)–[2315](#), [2380](#)–[2382](#), [2497](#)–[2499](#)
 - `\mathchardef` [3966](#)
 - `\maxdepth` [3107](#)

- \maxdimen 1824, 1839, 2453, 2467
 - \maxhnotesX 24
 - \maxhXnotes 24
 - Mayer, Gyula 9
 - \measurebody 4565, 4571, 4601, 4619
 - \measurecell 4206, 4232
 - \measuremrow 4230, 4576
 - \measuretbody 4581, 4587, 4609, 4627
 - \measuretcell 4218, 4237
 - \measuretrow 4235, 4593
 - \message 72, 206
 - Middleton, Thomas 9, 62
 - minipage (environment) 29
 - Mittelbach, Frank 8
 - \morenoexpands 42, 820
 - \moveleft 2571, 2577, 4510, 4515, 4523
 - \moveright 4536, 4549, 4558
 - \mpfnpos 33, 2565
 - \mpnormalfootgroup 1726, 1772
 - \mpnormalfootgroupX 2204, 2261
 - \mpnormalvfootnote
 - 1588, 1771, 1950, 2019
 - \mpnormalvfootnoteX
 - 2157, 2260, 2273, 2341
 - \mppara@footgroup 1790, 1878
 - \mppara@footgroupX 2414, 2484
 - \mppara@vfootnote 1789, 1833
 - \mppara@vfootnoteX 2413, 2447
 - \mpthreecolfootgroup 1951, 1987
 - \mpthreecolfootgroupX 2342, 2373
 - \mpthreecolfootsetup 1952, 1958
 - \mpthreecolfootsetupX 2343, 2345
 - \mptwocolfootgroup 2020, 2052
 - \mptwocolfootgroupX 2274, 2306
 - \mptwocolfootsetup 2021, 2052
 - \mptwocolfootsetupX 2275, 2277
 - \multfootsep 33, 2081, 2091
 - \multiplefootnotemarker
 - 2081, 2085, 2086, 2088
- N**
- \n@l 749
 - \n@num 679, 813
 - \n@num@reg 679
 - \nc@page 746, 747
 - \NeedsTeXFormat 2
 - \new 2811–
 - 2813, 2815, 2824, 2825, 2827, 2828
 - \new@line 731, 1171, 1186, 1189
 - \newbox 1032, 1035,
 - 3391, 3392, 4119, 4121, 4702, 4703
 - \newcommand
 - 940, 1047, 1083, 1520, 1533,
 - 1547, 1607, 1848, 1857, 1967,
 - 2031, 2841, 2858, 2963, 2976,
 - 2991, 3812, 3995, 4035, 4037, 4045
 - \newcounter
 - 359, 361, 363, 365, 1041, 1231,
 - 2781, 3260, 3261, 3721, 3974, 4471
 - \newhookcommand@series 2872
 - \newhookcommand@series@reload 2945
 - \newhooktoggle@series
 - 2959, 2962, 2967–2974, 2975
 - \newhooktoggle@series@reload
 - 2975, 2981, 2982
 - \newif 4–10, 29, 43,
 - 46, 48, 54, 187, 188, 190, 192,
 - 195–197, 300, 301, 451, 464,
 - 715, 757, 809, 841, 903, 920,
 - 1033, 1043, 1045, 1120, 1142,
 - 1418, 1443, 1640, 1642, 1644,
 - 1646, 1648, 2611, 2986, 3109,
 - 3184, 3421, 3523, 3709, 3710,
 - 3731, 3940, 3942, 4485, 4859, 5076
 - \newinsert 2727–2730
 - \newlength 417, 3958
 - \newlinechar 2789
 - \newread 473
 - \newrobustcmd 3291
 - \newseries 2670, 2807, 2808
 - \newseries@ 2671, 2675
 - \newtoggle 1752,
 - 1756, 2678, 2679, 2699–2702,
 - 2704, 2707, 2724, 2725, 2989, 2990
 - \newverse 154, 4012
 - \newwrite 714, 2610, 5075
 - \NEXT 4202,
 - 4207, 4210, 4215, 4216, 4219,
 - 4222, 4227, 4228, 4231, 4233,
 - 4234, 4236, 4238, 4239, 4244,
 - 4403, 4406, 4408, 4409, 4411,
 - 4413, 4414, 4417, 4419, 4420,
 - 4422, 4424, 4425, 4428, 4430,
 - 4431, 4433, 4435, 4436, 4441,
 - 4443, 4444, 4639, 4642, 4643,
 - 4648, 4652, 4653, 4669, 4675, 4676
 - \Next 4244, 4304,
 - 4306, 4317, 4318, 4323, 4325,
 - 4336, 4337, 4340, 4342, 4351,

- 4352, 4356, 4358, 4367, 4368,
 4371, 4373, 4383, 4384, 4388,
 4390, 4400, 4401, 4657, 4659, 4660
 \next@absline 743, 744
 \next@action 111, 483, 1237,
 1245, 1246, 1252, 1253, 1261, 1270
 \next@actionline
 . 480, 482, 1236, 1244, 1267, 1269
 \next@insert
 1064, 1461, 1464, 1466, 1469, 1473
 \next@page@num 239, 517, 519, 562, 612
 \no@expands 820, 845, 871, 919
 \noalign 1936
 \noeledsec 39, 5076
 \noendnotes 30, 2668
 \noindent 1038, 1048, 1108,
 1112, 1137, 1415, 1605, 1718,
 1721, 1817, 1825, 1829, 1840,
 1874, 1906, 1983, 2008, 2047,
 2076, 2155, 2454, 2468, 2491, 2521
 \nointerlineskip 2570, 2572, 2576, 2578
 \nolemmaseparator 21, 2991
 \nonbreakableafternumber 20
 \nonum@ 2989
 \nonumberinfootnote 20
 \noquotation@true 12
 \normal@footnotemarkX ... 2138, 2249
 \normal@page@break
 231, 563, 734, 745, 5250, 5268, 5276
 \normal@pars 248, 1065,
1140, 1604, 1968, 2032, 2295, 2363
 \normalbfnoteX 2227, 2241
 \normalbodyfootmarkX ... 2143, 2250
 \normalcolor 1732, 1885, 1993, 2061,
 2210, 2316, 2383, 2500, 3121, 3655
 \normalfont 419, 2082, 2144, 2984
 \normalfootfmt 1598, 1764
 \normalfootfmtX 2167, 2253
 \normalfootfootmarkX ... 2176, 2254
 \normalfootgroup 1720, 1765
 \normalfootgroupX 2199, 2255
 \normalfootnoterule 1719, 1767
 \normalfootnoteruleX 2197, 2256, 2409
 \normalfootstart 1702, 1762
 \normalfootstartX 2179, 2248
 \normalvfootnote 1571, 1763
 \normalvfootnoteX 2145, 2251
 \nosep@ 2990
 \notblank 1531
 \notbool 1571, 1588,
 1607, 1962, 1967, 2027, 2031, 2732
 \notfontsetup
 1982, 2691–2693, 2984, 2994
 \notfontsizeX 21
 \notenumfont 2688–2690, 2984
 \notenumfontX 21
 \noteschanged@false 464, 496
 \noteschanged@true
 253, 256, 464, 501, 907, 1463
 \notesXwidthliketwocolumns 24
 \nottoggle 1611, 1861, 1976, 2040, 2622
 \NR@ssect 5008, 5016
 \NR@sssect 5024, 5032
 \nulledindex 4184, 4265, 4276,
 4302, 4321, 4339, 4355, 4370, 4387
 \nullsetzen 4438, 4574, 4591
 \num@lines 1032, 1090, 1480, 1486, 1489
 \numberedpar@false 1032
 \numberedpar@true 1032, 1077
 \numberingfalse 187, 247
 \numberingtrue 187, 203, 276
 \numberlinefalse 13
 \numberlinetrue 13, 904
 \numberonlyfirstinline 19
 \numberonlyfirstintwolines 20
 \numberpstartfalse 13, 1037
 \numberpstarttrue 12, 13, 1037
 \numdef 743,
 981, 985, 989, 993, 1000, 1001,
 1003, 1014, 1015, 1017, 1178, 5267
 \numgdef 735, 746,
 3539, 3546, 3567, 3574, 5288, 5292
 \numlabfont 24, 417
- ## O
- \old@edtext
 4902, 4909, 4983, 4990, 4995, 5002
 \old@hsize
 1712, 1723, 1875, 2190, 2201, 2523
 \old@Rlineflag 3813, 3840
 \one@line
1032, 1149, 1150, 1171, 1186, 1189
 \onlypstartinfootnote 20
 \openout 224, 722, 728, 2612
- ## P
- \p@pstart 1079
 \PackageError 56
 \PackageWarning 55

- \page@action 518, 610, 697
- \page@num 136, 143,
150, 460, 478, 560, 703, 708,
1246, 1354, 1430, 1920, 1929,
3405, 3413, 3478, 3496, 3562, 5286
- \page@numR .. 134, 141, 148, 3473, 3491
- \page@start 780, 3111
- \pagebreak 5263
- \pagelinesep 34, 3719, 3728–3730
- \pageno 113, 115, 3082
- \pageparbreak 41, 1415
- \pageref 32
- \pairs 5188, 5191, 5206, 5210, 5224, 5228
- \paperwidth 1185, 1188
- \par@line
1032, 1091, 1481, 1482, 1485, 1489
- \para@footgroup 1784, 1867
- \para@footgroupX 2408, 2484
- \para@footsetup 1788, 1796
- \para@footsetupX 2418, 2420
- \para@vfootnote 1782, 1818
- \para@vfootnoteX 2406, 2447
- \parafootfmt 1783, 1857
- \parafootfmtX 2407, 2475
- \parafootftmsep 2716, 2996
- \parafootsep 23
- \parafootstart 1781, 1804
- \parafootstartX 2405, 2429
- \parapparatus@false 11
- \parapparatus@true 15
- \parledgroup@ 1729, 1882,
1990, 2058, 2207, 2313, 2380, 2497
- \parledgroup@beforenotesL 3610, 3662
- \parledgroup@beforenotesR 3608, 3660
- \parledgroup@series .. 1730, 1883,
1991, 2059, 2208, 2314, 2381, 2498
- \parledgroup@type ... 1731, 1884,
1992, 2060, 2209, 2315, 2382, 2499
- \patchcmd 4787, 4790,
4791, 4794, 4798, 4799, 4893,
4913, 4921, 4931, 4939, 4947,
4955, 4964, 4973, 5008, 5016,
5024, 5032, 5041, 5049, 5057, 5065
- \pausenumbering 12, 274
- \pend 10, 103, 106, 1061, 1083,
1138, 1415, 4039, 4046, 4725,
4727, 4733, 4735, 4741, 4743,
4749, 4751, 4757, 4759, 4765,
4767, 4776, 4779, 4782, 4784, 4797
- Plato of Tivoli 8
- \postbodyfootmark 2127, 2141
- \postdisplaypenalty 1118
- \prebodyfootmark 2127, 2139
- \predisplaypenalty 1117
- \prenotesX 23, 1759
- \prenotesX@ 1758, 1759, 2180, 2430
- \prepare@edindex@fornote
..... 2744, 2757, 3732
- \pretocmd 2110,
3842, 4788, 4792, 4901, 4982, 4994
- \prev@line 981–983, 985, 989–
991, 993, 1000, 1001, 1014, 1015
- \prev@nopb 5251
- \prev@pb 5251
- \prevgraf 1090
- \prevline 1921, 3021
- \prevpage@num 1919
- \preXnotes 23, 1752, 1756
- \preXnotes@ 1703, 1752, 1756, 1806
- \print@eledsection 1161, 1176
- \print@footnoteXrule
. 2194, 2216, 2222, 2322, 2328,
2389, 2395, 2443, 2506, 2512, 2569
- \print@leftmargin@eledsection ..
..... 4860, 4924, 4934,
4957, 4975, 5019, 5035, 5052, 5068
- \print@line 1162, 1165
- \print@rightmargin@eledsection .
..... 4860, 4916, 4942,
4959, 4977, 5011, 5027, 5044, 5060
- \print@Xfootnoterule
. 1716, 1738, 1744, 1815, 1891,
1897, 1999, 2005, 2067, 2073, 2569
- \printendlines 2621, 2650
- \printlinefootnote
.... 1610, 1860, 1975, 2039, 2997
- \printlines 1692,
3038, 3042, 3052, 3056, 3066, 3071
- \printnpnum 30, 2652, 2655, 2660
- \printpstart 1629,
3037, 3041, 3051, 3055, 3065, 3070
- \processl@denvbody
..... 3915, 3919, 3920, 3936
- \ProcessOptions 20
- \protected@csxdef 2776, 4080
- \protected@edef
.... 1078, 2168, 2292, 2359, 2476
- \protected@write
..... 943, 960, 3255, 3776,

3778, 3781, 3788, 3790, 3793,
3798, 3800, 3803, 3827, 3830, 3834
`\protected@xdef` 2229
`\ProvidesPackage` 3
`\pst@rtedLfalse` ... 188, 212, 228, 250
`\pst@rtedLtrue` 188, 280
`\pstart` 10, 97, 100, 101, 106,
1037, 1137, 1415, 4004, 4726,
4729, 4734, 4737, 4742, 4745,
4750, 4753, 4758, 4761, 4766,
4769, 4777, 4779, 4783, 4785, 4797
`\pstarteref` 3305
`\pstartinfootnote` .. 20, 310, 316, 322
`\pstartline` 1094, 1097
`\pstartnum` 1417
`\pstartnumfalse` 1448, 1455
`\pstartnumtrue` 1104, 1444
`\pstartref` 30, 3305
`\rightctab` 4526, 4618
`\rightlinenum`
... 14, 417, 1349, 1357, 4865, 4881
`\rightlinenumR` 4866, 4882
`\rightltab` 4539, 4600
`\rightnoteupfalse` 32
`\rightnoteuptrue` 3422
`\rightpstartnum` 1425, 1433, 1450
`\rightrtab` 4552, 4564
`\rightstartnum` 1417
`\rigidbalance` ... 1931, 1986, 2011,
2050, 2079, 2309, 2333, 2376, 2400
`\rlap` 1349, 1357, 1425, 1433, 4864, 4880
`\Rlineflag` 3813, 3814, 3840, 3867, 3871
Robinson, Peter 7
`\roman` 4472
`\rtab` 4059, 4561, 4707
`\rtabtext` 4062, 4579, 4711

Q

`\quotation` 4722
`\quote` 4722

R

`\RaggedLeft` 1869, 1901, 2486, 2516
`\RaggedRight` ... 1870, 1902, 2487, 2517
`\raggedright` 1972,
2036, 2301, 2368, 3399, 4965, 4967
`\raggedX` 23
`\raw@text` 1032, 1067, 1093, 1149
`\rbracket` 25, 1617, 2711
`\read@linelist` 473, 718
`\ref` 32
`\Relax` 4202, 4656, 4663
`\rem@inder` 1336, 1338–1340
`\removehboxes`
.... 1872, 1904, 1908, 2489, 2519
`\removelastskip` 4303, 4322
`\RequirePackage`
.... 22, 23, 25–28, 36, 37, 39–42
`\reserveinserts` 24, 38
`\resetprevline@` 65, 241, 465, 1097, 1249
`\resetprevpage@` 469
`\resetprevpage@num` . 65, 242, 469, 3669
`\restore@familiarnotes` .. 4076, 4112
`\restore@notes` 4106, 4313,
4332, 4348, 4364, 4379, 4396, 4594
`\restore@sidenotes` 4093, 4111
`\resumenumbering` 12, 274
`\right` 4449, 4452, 4457, 4460

S

Sacrobosco 9
`\sameword` 18, 827, 940
`\sameword@inedtext` 827, 997
`\sc@n@list` 1337, 1339
Schöpf, Rainer 8
`\section@num` 184, 204,
206, 207, 223, 224, 281–283,
980, 982, 983, 985, 999, 1001, 2619
`\section@numR`
... 988, 990, 991, 993, 1013, 1015
`\sectionmark` 4851, 5206, 5210
`\select@lemmafont` 821, 1515
`\select@lemmafont` 25,
1515, 1611, 1861, 1976, 2040, 2622
`\series` .. 2670, 2810, 2812, 2823, 2827
`\seriesatbegin` 2809
`\seriesatend` 2819, 2822
`\set@line` 846, 872, 905
`\set@line@action` 511, 595, 602, 613, 699
`\setcommand@series` .. 2858, 2874, 2947
`\setl@dlp@rbox` . 3507, 3555, 3570, 3572
`\setl@drp@rbox` . 3515, 3557, 3565, 3577
`\setl@drpr@box` 3507
`\setline` 15, 93, 792, 825, 1097
`\setlinenum` 15, 95, 799
`\setmcellcenter` 4369, 4429
`\setmcellleft` 4338, 4418
`\setmcellright` 4301, 4407
`\setmrowcenter` 4427, 4622
`\setmrowleft` 4416, 4604

- `\setmrowright` 4405, 4568
- `\setnotesXpositionliketwocolumns@`
..... 2153,
2193, 2215, 2221, 2321, 2327,
2388, 2394, 2442, 2505, 2511, 2553
- `\setnotesXwidthliketwocolumns@` .
..... 2151, 2192, 2214,
2220, 2320, 2326, 2387, 2393,
2421, 2441, 2493, 2504, 2510, 2523
- `\setprintendlines` 2626, 2651
- `\setprintlines` 1666, 1693
- `\setstanzaindents` 26, 3971
- `\setstanzapenalties` 27, 3971
- `\setstanzavalues` 3961, 3971, 3972, 4056
- `\settcclcenter` 4386, 4434
- `\settcclleft` 4354, 4423
- `\settcclright` 4320, 4412
- `\settoggle` 1523, 1527, 2843
- `\settoggle@series`
..... 2835, 2841, 2964, 2977
- `\setthrowcenter` 4432, 4630
- `\setthrowleft` 4421, 4612
- `\setthrowright` 4410, 4584
- `\setXnotespositionliketwocolumns@`
..... 1579,
1715, 1737, 1743, 1814, 1890,
1896, 1998, 2004, 2066, 2072, 2553
- `\setXnoteswidthliketwocolumns@` .
..... 1577, 1714, 1736,
1742, 1797, 1813, 1879, 1889,
1895, 1997, 2003, 2065, 2071, 2523
- `\Setzen` 4647, 4656, 4658
- `\showlemma` 30, 40, 819, 855, 881
- `\showwordrank` ... 18, 1010, 1024, 1028
- `\sidenote@margin` 3356, 3476, 3494, 3560
- `\sidenote@marginR` .. 3361, 3471, 3489
- `\sidenotecontent@`
. 3538, 3543, 3544, 3555, 3557,
3565, 3566, 3570, 3572, 3573, 3577
- `\sidenotemargin` 32, 3356
- `\sidenotemmargint` 120
- `\sidenotesep` 33, 3525
- `\skip` . 1706, 1709, 1727, 1770, 1775,
1787, 1793, 1809, 1812, 1880,
1988, 2056, 2183, 2186, 2205,
2259, 2264, 2311, 2378, 2412,
2417, 2433, 2436, 2440, 2494,
3119, 3608, 3610, 3654, 3660, 3662
- `\skip@lockoff` 632, 658
- `\skipnumbering` 15, 809,
4029, 4725, 4727, 4733, 4735,
4741, 4743, 4749, 4751, 4757,
4759, 4765, 4767, 4776, 4782,
4795, 4796, 4804, 4817, 4826, 4839
- `\skipnumbering@reg` 809
- `\spacefactor`
2089, 2092, 2100, 2106, 2130, 2136
- `\spaceskip` 1580, 2154, 2288
- `\splitmaxdepth` 1585
- `\splitoff` 1931
- `\splittopskip` ... 1146, 1585, 1933,
1984, 1986, 2009, 2011, 2048,
2050, 2077, 2079, 2307, 2309,
2331, 2333, 2374, 2376, 2398, 2400
- `\spreadmath` 36, 4635
- `\spreadtext` 36, 4633
- `\stanza` 26, 4012
- `\stanza@count` ... 3952, 3963, 3966,
3968, 3997, 4009, 4018, 4028, 4047
- `\stanza@chang` 3995, 4020
- `\stanza@line` 3995, 4033, 4047
- `\stanza@modulo` 3975, 3978–
3980, 3989–3991, 4000, 4018, 4027
- `\stanzaindent` 27, 3983
- `\stanzaindent*` 27, 3983
- `\stanzaindentbase` 26,
3952, 3984, 3998, 4001, 4007, 4050
- `\startlock` 14, 806, 823, 4005
- `\startstanzahook` 28, 4012
- `\startsub` 14, 782, 822
- `\stepcounter`
.... 2775, 3264, 3267, 3724, 4480
- `\stepl@dcollcount` 4168, 4213,
4225, 4310, 4329, 4345, 4361,
4376, 4393, 4439, 4640, 4649, 4670
- `\StrGobbleLeft` 3741, 3746
- `\strip@pt` 1802, 2427
- `\strip@szacnt` 3961
- `\sub@action` 527, 622, 698
- `\sub@change` 240,
521, 522, 528, 572, 574, 577, 579
- `\sub@lock` 237, 453, 536, 538,
540, 543, 640, 641, 643, 644,
662, 663, 665, 1219, 1291, 1293,
1294, 1296, 1372, 1408, 1410, 1412
- `\sub@off` 571, 788
- `\sub@on` 571, 784
- `\subline@num` 235, 423, 424,
450, 544, 548, 558, 583, 584,

586, 598, 616, 705, 709, 1220,
1225, 1248, 1256, 1311–1313, 3289
`\sublinenumberstyle` 15, 408
`\sublinenumincrement` 12, 14, 368
`\sublinenumr@p` 408
`\sublinenumrep` 408,
424, 1696, 1700, 2653, 2657, 3289
`\sublineref` 30, 3302
`\sublines@false` .. 238, 451, 525, 1281
`\sublines@true` 451, 523, 1279
`\sublock@disp` .. 399, 1374, 1378, 1382
`\sublockdisp` 83, 399
`\subsection` 2832,
4743, 4751, 5223, 5227, 5232, 5233
`\subsectionmark` 4852, 5224, 5228
`\subsubsection` 2834,
4759, 4767, 5240, 5241, 5244, 5245
Sullivan, Wayne 8,
9, 26, 41, 51, 55, 116, 117, 159, 182
`\sw@atthisline`
1001, 1003, 1009, 1015, 1017, 1023
`\sw@list` 491, 938, 944, 945, 986
`\sw@list@inedtext`
... 492, 938, 948, 950, 1004, 1007
`\sw@list@inedtextR` 965, 967, 1018, 1021
`\sw@listR` 961, 962, 994
`\symlinenum` 20
`\symplinum` 2703, 2984
`\sza@penalty` 3995, 4024, 4046

T

`\tabellzwischen` 4638, 4646
`\tabelskip` 4650, 4690–4692, 4698–4700
`\tabHilfbox` 4689,
4691, 4693, 4697, 4699, 4701, 4702
`\tabhilfbox` 4688,
4690, 4692, 4696, 4698, 4700, 4702
Tapp, Christian 7
`\temp@` 1178, 1179, 3877, 3882
`\textheight` 3223
`\textnormal` 1617–1619
`\textsuperscript` 1030, 2082, 2144, 2177
`\textwidth` 3632, 3684
`\the@sw` 985, 986, 993, 994,
1005, 1007, 1010, 1019, 1021, 1024
`\theadcolcount` 4471, 4478, 4481
`\theendpageline`
3729, 3779, 3791, 3801, 3819, 3831
`\thefootnoteA` 33

`\thelabidx`
3725, 3728, 3887, 3891, 3896, 3898
`\theline` 3268, 3270
`\thempfn` 3634, 3670, 3697
`\thempfootnote` 3634, 3670, 3697
Theodosius 9
`\thepage` 735,
738, 740, 749, 751, 754, 3256, 3728
`\thepageline`
3727, 3782, 3794, 3804, 3822, 3835
`\thepstart`
. 13, 1037, 1137, 1446, 1453, 1637
`\thepstartL` 1634
`\thepstartR` 1632
`\thestartpageline`
3729, 3777, 3789, 3799, 3818, 3828
`\thesubline` 3268
`\thinspace` 1622, 1624
`\thisfootnote` 2229, 2230
`\thr@@` 350, 643, 652, 663, 670,
1286, 1294, 1957, 1960, 1986,
2011, 2347, 2350, 2376, 2400, 3381
`\threecolfootfmt` 1947, 1967
`\threecolfootfmtX` 2338, 2358
`\threecolfootgroup` 1948, 1982
`\threecolfootgroupX` 2339, 2373
`\threecolfootsetup` 1949, 1955
`\threecolfootsetupX` 2340, 2345
`\threecolvfootnote` 1946, 1962
`\threecolvfootnoteX` 2337, 2352
`\togglefalse` ... 1706, 1809, 2183, 2433
`\toggletrue` 1753, 1757
`\tolerance` 1971, 2035, 2300, 2367
`\twocolfootfmt` 2016, 2024
`\twocolfootfmtX` 2270, 2291
`\twocolfootgroup` 2017, 2024
`\twocolfootgroupX` 2271, 2306
`\twocolfootsetup` 2018, 2024
`\twocolfootsetupX` 2272, 2277
`\twocolvfootnote` 2015, 2024
`\twocolvfootnoteX` 2269, 2284
`\txtbeforeXnotes` 23

U

`\unexpanded` 1038, 1112,
5082, 5086, 5095, 5099, 5108,
5112, 5120, 5124, 5134, 5138,
5147, 5151, 5160, 5164, 5173, 5177

`\unhbox` 1143, 1851, 1872, 1874, 1904,
 1906, 1913, 1917, 2489, 2491,
 2519, 2521, 4692, 4693, 4700, 4701
`\unkern` 2090
`\unless` 941, 944,
 947, 961, 964, 979, 998, 3712, 3715
`\unpenalty` 1854, 1910
`\unvbox` ... 1150, 1590, 1722, 1749,
 1835, 1849, 1868, 1900, 1942,
 2159, 2200, 2225, 2463, 2485,
 2515, 2583, 2589, 2598, 2603,
 3097, 3118, 3123, 3142, 3152,
 3157, 3159, 3178, 3206, 3651, 3666
`\unvxh` ... 1826, 1841, 1848, 2455, 2469
`\usingcritext` 4713
`\usingedtext` 4713

V

`\valign` 1934
`\value` 3979, 3991, 3996, 4477
`Vamana` 9
`\variab` 4246, 4567,
 4583, 4603, 4611, 4621, 4629, 4662
`\vbadness` 1145, 1933
`\vbfnoteX` 2230, 2235
`\vbox` 1067,
 1589, 1824, 1834, 1839, 1849,
 2158, 2453, 2462, 2467, 2571,
 2577, 2582, 2597, 3094, 3115,
 3141, 3231, 3235, 3511, 3513,
 3519, 3521, 3629, 4448, 4451,
 4456, 4459, 4463, 4465, 4466,
 4509, 4513, 4518, 4529, 4542, 4555
`\vfil` 1934, 3235,
 4449, 4452, 4457, 4460, 4463, 4466
`\vfill` 3119
`\vl@dbfnote` 2116
`\vl@dcsnote` 3459, 3463, 3469
`\vl@dlsnote` 3429, 3433, 3469
`\vl@drsnote` 3444, 3448, 3469
`\vnumfootnoteX` 2239, 2252
`\vrule` ... 4448, 4451, 4456, 4459, 4463
`\vsize` 1751
`\vsplit` 1149, 1941, 3206

W

`\wd` 1137, 1171, 1828,
 1843, 2457, 2471, 4174, 4175,

4298, 4522, 4531, 4534, 4544,
 4547, 4556, 4690, 4691, 4698, 4699
 Whitney, Ron 8
`\widowpenalty` 1118, 1487
`\widthliketwocolumnstrue` 18
`\WithSuffix` 3987, 4731, 4747,
 4763, 4780, 5130, 5143, 5156, 5169
`\wrap@edcrossref`
 3291, 3296, 3299, 3302, 3305
 Wujastyk, Dominik 6, 8

X

`\x@lemma` 857–859, 883–885
`\xcritext` 4178, 4291
`\xedindex` 4184, 4270, 4281, 4293
`\xedlabel` 4182, 4299
`\xedtext` 4178, 4292
`\Xendlemmadisablefontselection` . 22
`\Xendnotefontsize` 21
`\Xendnotenumfont` 21
`\Xhangindent` 22
`\xifinlist`
 733, 734, 744, 745, 1160, 1179,
 2676, 5263, 5266, 5268, 5275, 5276
`\xleft@appenditem` 442
`\Xlemmadisablefontselection` 22
`\xlineref` 31, 3299, 3728
`\Xnotefontsize` 21
`\Xnotenumfont` 21
`\Xnoteswidthliketwocolumns` 24
`\xpageref` 31, 3296
`\xpstartref` 31, 3305
`\Xragged` 23
`\xright@appenditem` 436, 611,
 612, 614, 621, 623, 625, 627,
 637, 639, 648, 659, 661, 668,
 681, 682, 691, 707, 948, 950,
 965, 967, 986, 994, 1523, 1527,
 1535, 1537, 1541, 1543, 1550,
 1553, 1556, 1561, 1564, 1567,
 2119, 2230, 2744, 2757, 3287,
 3429, 3433, 3444, 3448, 3459, 3463
`\xspaceskip` 1580, 2154, 2288
`\xsublineref` 31, 3302
`\xxref` 31, 3330

Z

`\z@skip` 1580, 1586, 2154, 2288
`\zz@@@` 3242, 3248, 3333, 3338

Change History

v0.1.	v0.2.2.
General: First public release 1	General: Improved paragraph foot-
v0.2.	notes 1
General: Added tabmac code, and	New Dekker example 1
extended indexing 1	Used <code>\providecommand</code> for
<code>\eledmac@error</code> : Added <code>\eledmac@error</code>	<code>\@gobblethree</code> to avoid clash
and replaced error messages . . 48	with the <code>amsfonts</code> package . . . 51
<code>\ifl@dmemoir</code> : Added <code>\ifl@dmemoir</code>	<code>\footfudgefiddle</code> : Added
for memoir class having been	<code>\footfudgefiddle</code> 115
used 47	<code>\line@list@stuff</code> : Added ini-
<code>\morenoexpands</code> : Added <code>\l@dtabnoexpands</code>	tial write of page number in
to <code>\no@expands</code> 80	<code>\line@list@stuff</code> 74
v0.2.1.	<code>\para@footsetup</code> : Added
<code>\@lab</code> : Removed page setting from	<code>\footfudgefiddle</code> to
<code>\@lab</code> 161	<code>\para@footsetup</code> 115
General: Added text about normal	<code>\para@footsetupX</code> : Added
labeling 32	<code>\footfudgefiddle</code> to
Bug fixes and match with mem-	<code>\para@footsetupX</code> 134
patch v1.8 1	v0.3.
Major changes to insert code	<code>\@lab</code> : Replaced <code>\the\line@num</code>
when memoir is loaded . . . 156	by <code>\linenumr@p\line@num</code> in
<code>\doxtrafeet</code> : Renamed <code>\doxtrafeet</code>	<code>\@lab</code> , and similar for sub-lines 161
to <code>\l@ddoxtrafeet</code> 155	<code>\@nl@reg</code> : Added a bunch of code to
<code>\edlabel</code> : Tweaked <code>\edlabel</code> to	<code>\@nl</code> for handling <code>\setlinenum</code> 67
get correct page numbers . . 159	General: Includes <code>edstanza</code> and
<code>\l@d@makecol</code> : Rewrote <code>\@makecol</code> ,	more 1
calling it <code>\l@d@makecol</code> . . . 154	<code>\ledlinenum</code> : Added <code>\linenumr@p</code>
<code>\l@ddodoreinextrafeet</code> : Re-	and <code>\sublinenumr@p</code>
named <code>\dodoreinextrafeet</code> to	to <code>\leftlinenum</code> and
<code>\l@ddodoreinextrafeet</code> . . . 156	<code>\rightlinenum</code> 59
<code>\l@ddofootinsert</code> : Renamed	<code>\linenumberlist</code> : Added
<code>\dofootinsert</code> as <code>\l@ddofootinsert</code>	<code>\linenumberlist</code> mechanism . 51
. 155	<code>\printendlines</code> : Added <code>\linenumr@p</code>
<code>\m@m@makecolintro</code> : Added	and <code>\sublinenumr@p</code> to
<code>\m@m@makecolfloats</code> , <code>\m@m@makecoltext</code>	<code>\printendlines</code> 141
and <code>\m@m@makecolintro</code> . . . 154	<code>\printlines</code> : Added <code>\linenumr@p</code>
<code>\morenoexpands</code> : Removed some	and <code>\sublinenumr@p</code> to
<code>\lets</code> from <code>\no@expands</code> . These	<code>\printlines</code> 111
were in EDMAC but I feel that	<code>\sublinenumr@p</code> : Added <code>\linenumberstyle</code>
they should not have been as	and <code>\sublinenumberstyle</code> . . 59
they disabled page/line refs in	v0.3.1.
a footnotes 80	General: Not released. Added re-
<code>\zz@@@</code> : Minor change to <code>\zz@@@</code> . 159	marks about the parallel pack-

age	1	v0.5.1.	
v0.4.		General: Added moveable side note	164
\@iiiminipage: Modified kernel		Fixed right line numbers killed in	
\@iiiminipage and		v0.5	1
\endminipage to cater for critical		\affixline@num: Changed	
footnotes	170	\affixline@num to cater for	
General: Added \showlemma to		sidenotes	97
\edtext (and \critext)	82	ledgroupsize: Only change	
Added minipage, etc., support .	1	\hsize in ledgroupsize environment	
ledgroup: Added ledgroup environment		otherwise page number	
.	171	can be in wrong place	172
ledgroupsize: Added ledgroup-		\l@dgetsidenote@margin: Added	
sized environment	172	\sidenotemargin and	
\footnormal: Added minpage foot-		\sidenote@margin	164
note setup to \footnormal .	114	v0.6.	
\ifledsecnolinenum: Added final/		\@lopR: Added \@pend,\@pendR,	
draft options	46	\@lopL and \@lopR in anticipation	
\l@dfeetendmini: Added		of parallel processing	69
\l@dfeetbeginmini,\l@dfeetendmini		\@nl@reg: Added \fix@page to	
and all their supporting code	169	\@nl	67
\mpnormalfootgroup: Added		Extended \@nl to include the	
\mpnormalfootgroup	112	page number	67
\mpnormalvfootnote: Added		General: Fixed long paragraphs	
\mpnormalvfootnote	107	looping	1
\showlemma: Added \showlemma .	46	Fixed minor typos	1
v0.4.1.		Prepared for eldpar package .	1
\@opxtrafeetii: Added \@opxtrafeetii		\fix@page: Added \last@page@num	
.	155	and \fix@page	68
General: Added code for changing		\new@line: Extended \new@line to	
\@docleapage	157	output page numbers	74
Not released. Minor editorial im-		\page@start: Made \page@start a	
provements and code tweaks .	1	no-op	75
Only change \@footnotetext		\v1@dbfnote: Changed \l@dbfnote	
and \@footnotemark if memoir		and \v1@dbfnote as originals	
not used	126	could give incorrect markers in	
\doxtrafeetii: Changed		the footnotes	127
\doxtrafeetii code for easier		v0.7.	
extensions	155	\@nl@reg: Added \@nl@reg	67
\edindex: Let eldmac take advan-		\@ref@reg: Added \@ref@reg . . .	72
tage of memoir's indexing .	174	General: eldmac having been avail-	
v0.5.		able for 2 years, deleted the	
\@footnotetext: Enabled regular		commented out original edmac	
\footnote in numbered text	126	texts	1
\@xympar: Eliminated \marginpar		Maïeul Rouquette new main-	
disturbance	163	tainer	1
General: Added left and right side		Made macros of all messages .	47
notes	164	Replaced all \interAfootnotelinepenalty,	
Added sidenotes, familiar foot-		etc., by just \interfootnotelinepenalty	
notes in numbered text	1	1

Tidying up foreledpar and ledarab packages	1		
\affixline@num: Added skipnum- mering to \affixline@num . .	97		
\do@actions@fixedcode: Added \do@actions@fixedcode	96		
\do@actions@next: Added number skipping to \do@actions	95		
\do@insidelinehook: Added \do@linehook for use in \do@line	93		
\endnumbering: Changed \endnumbering foreledpar . .	54		
\f@x@l@cks: Added \ch@cksub@l@ck, \ch@ck@l@ck and \f@x@l@cks . .	99		
\footsplitskips: Added \footsplitskips for use in many footnote styles	107		
\get@linelistfile: Added \get@linelistfile	66		
\ifledRcol@: Added \l@dnumstartsl, \ifl@dpairing and \ifpst@rted for/fromeledpar	52		
\initnumbering@reg: Added \initnumbering@reg	53		
\l@dcstotext@r: Added \l@demptyd@ta	93		
\l@ddofootinsert: Deleted \page@start from \l@ddofootinsert	155		
\l@dgetline@margin: Added \l@dgetline@margin	57		
\l@dgetlock@disp: Added \l@dgetlock@disp	58		
\l@dgetsidenote@margin: Added \l@dgetsidenote@margin . .	164		
\l@drsn@te: Added \l@dlsn@te and \l@drsn@te for use in \do@line	94		
\l@dunboxmpfoot: Added \l@dunboxmpfoot containing some common code	171		
\l@dzeropenalties: Added \l@dzeropenalties	91		
\ledlinenum: Added \ledlinenum for use by \leftlinenum and \rightlinenum	59		
\line@list@stuff: Deleted \page@start from \line@list@stuff	74		
\list@clearing@reg: Added \list@clearing@reg	66		
\n@num@reg: Added \n@num	72		
\normalbfnoteX: Removed extraneous space from \normalbfnoteX	130		
\resumenummering: Changed \resumenummering foreledpar . .	55		
\setprintendlines: Added \setprintendlines for use by \printendlines	140		
\setprintlines: Added \setprintlines for use by \printlines	110		
\skipnumbering@reg: Added \skipnumbering and supports . .	77		
\sublinenumincrement: Added \firstlinenum, \linenumincrement, \firstsublinenum and \linenumincrement	58		
\sublinenumr@p: Using \linenumrep instead of \linenumr@p	59		
Using \sublinenumrep instead of \sublinenumr@p	59		
\vnumfootnoteX: Removed extraneous space from \vnumfootnoteX	130		
	v0.8.	General: Bug on endnotes fixed: in a // text, all endnotes will print and be placed at the ends of columns ()	1
	v0.8.1.	General: Bug on \edtext ; \critex ; \lemma fixed: we can now us not switching commands	1
	v0.9.	General: No more ledpatch. All patches are now in the main file.	1
	v0.9.1.	General: Fix some bugs linked to in- tegrating ledpatch on the main file.	1
	v0.10.	General: Corrections to \section and other titles in numbered sections	1
	v0.11.	General: Makes it possible to add a symbol on each verse's hang-	

- ing, as in French typography. Redefines the command `\hangingsymbol` to define the character. 1
- v0.12.
- General: For compatibility with `eledpar`, possibility to use `\autopar` on the right side. . . . 1
 - Possibility to number `\pstart`. 13
 - Possibility to number the `pstart` with the commands `\numberpstarttrue`. 1
 - `\ifledRcol@`: Added `\ifledRcol` and `\ifnumberingR` for/from `eledpar` 52
- v0.12.1.
- General: Don't number `\pstarts` of stanza. 1
 - The numbering of `\pstarts` restarts on each `\beginnumbering`. 1
- v0.13.
- General: New `stanzaindentsrepetition` counter to repeat stanza indents every n verses. 26
 - New `stanzaindentsrepetition` counter: to repeat stanza indents every n verses. 1
 - `\managestanza@modulo`: New `stanzaindentsrepetition` counter to repeat stanza indents every n verses. 184
- v0.13.1.
- General: `\thepstartL` and `\thepstartR` use now `\bfseries` and not `\bf`, which is deprecated and makes conflicts with memoir class. 1
- v0.14.
- General: Tweaked `\edlabel` to get correct line number if the command is first element of a paragraph. 1
 - `\edlabel`: Tweaked `\edlabel` to get correct line number if the command is first element of a paragraph. 159
- v0.15.
- General: Line numbering can be reset at each `pstart`. 55
- Possibility to print `\pstart` number in side. 13
 - `\affixline@num`: Line numbering can be disabled. 97
 - `\ifinserthangingsymbol`: New management of `hangingsymbol` insertion, preventing undesirable insertions. 182
 - `\printlines`: Line numbering can be reset at each `pstart`. . . . 111
- v0.17.
- `\ifinserthangingsymbol`: New new management of `hangingsymbol` insertion, preventing undesirable insertions. 182
- v1.0.
- General: `\lemma` can contain commands. 17
 - Debug in lineation command . 13
 - New generic commands to customize footnote display. 19
 - Options `nonum` and `nosep` in `\Xfootnote`. 16
 - Options of `\Xfootnotes`. . . . 105
 - Possibility to have commands in sidenotes. 32
 - Some compatibility break with `eledmac`. Change of name: `eledmac`. 1
 - `\morenoexpands`: Change to be compatible with new features 80
- v1.0.1.
- General: Correction on `\numberonlyfirstinline` with lineation by `pstart` or by page. 19
- v1.1.
- General: Add `\labelpstarttrue`. 13
 - Add `\numberonlyfirstintwolines` 20
 - Add `\pstartinfootnote` and `\onlypstartinfootnote` 20
 - New hook to add arbitrary code at the beginning of the notes . 22
 - New options for block of notes. 23
 - New package option: `parapparat`. 1
 - New tools to change order of series 146
 - `\ledfootinsdim`: Deprecated
 - `\ledfootinsdim` 113

<code>\preXnotes</code> : New skip <code>\preXnotes@</code>	113	v1.4.6.	General: Bug with memoir class introduced by 1.4.5.	1
v1.1.0.		v1.4.7.	<code>\endquote</code> : Compatibility of sectioning commands with <code>\autopar</code>	206
General: Sectioning commands.	38	v1.4.8.	General: Corrects a bug with parallel texts introduced by 1.1. . . .	1
v1.2.		v1.4.9.	<code>\normalbfnoteX</code> : Allow to redefine <code>\thefootnoteX</code> with <code>\alph</code> when some packages are loaded. . . .	130
<code>\endquote</code> : Compatibility of <code>\ledchapter</code> with the <i>memoir</i> class.	206	v1.5.	General: Correct indexing when the call is made in critical notes. . . .	173
<code>\preXnotes</code> : Debug in familiar footnotes (but introduced by v1.1). . . .	113	<code>\do@insidelinehook</code> : Added <code>\do@insidelinehook</code> for use in <code>\do@line</code>	93	
v1.3.		<code>\edindex</code> : Compatibility with <code>imakeidx</code> package, and possibility to use multiple index with <code>\edindex</code>	174	
<code>\endquote</code> : <i>Quotation</i> and quote environment inside the numbering sections.	206	<code>\ifFN@bottom</code> : Use the bottom option of <code>footmisc</code> package.	154	
v1.4.		v1.5.1.	<code>\managestanza@modulo</code> : Correct <code>stanzaindentsrepetition</code> counter	184
General: Compatibility of <code>\edtext</code> (and <code>\critext</code>) with the right-to-left direction (with <code>Polyglossia</code>).	82	<code>\normalvfootnoteX</code> : Fix bug with normal familiar footnotes when mixing RTL and LTR text. . . .	127	
Compatibility with LuaTeX of RTL notes.	46, 47	v1.6.0.	<code>\falseverse</code> : Add <code>\falseverse</code> macro.	185
<code>\newseries@</code> : Remembers the language of the lemma, in order to create a correct direction for the footnote separator.	143	v1.6.1.	General: Corrects a false hanging verse when a verse is exactly the length of a line.	1
<code>\normalfootfmt</code> : Direction of footnotes with <code>polyglossia</code>	107	<code>\AtEveryPstart</code> : Spurious space in <code>\pstart</code>	88	
<code>\rbracket</code> : Switch the right bracket to a left bracket when the lemma is RTL (needs <code>polyglossia</code> or <code>LuaTeX</code>).	108	<code>\ifinserthangingsymbol</code> : Hang verse is now not automatically flush right.	182	
v1.4.1.		<code>\l@dunhbox@line</code> : Move the call to <code>\inserthangingsymbol</code> to allow use <code>\hfill</code> inside.	92	
<code>\endquote</code> : New option <i>noquotation</i>	206	<code>\pend</code> : Spurious space in <code>\pend</code>	89	
<code>\labelrefsparsesubline</code> : Fix bug with <code>\edlabel</code>	159			
v1.4.2.				
General: Debug with some special classes.	1			
v1.4.3.				
General: Add <code>\nonbreakableafternumber</code>	20			
Spurious space after familiar footnotes.	1			
v1.4.4.				
General: Label inside familiar footnotes.	1			
v1.4.5.				
General: Bug with <code>komasscript</code> + <code>eledpar</code> + <code>chapter</code>	1			

v1.7.0.	General: New features for managing page breaks.	39	v1.10.1.	General: Compatibility with <code>cleveref</code>	1
v1.8.0.	General: Compatibility with <code>parledgroup</code> option of <code>eledpar</code> package.	1	v1.10.2.	General: Compatibility of stanza with v1.8a of <code>babel-greek</code>	1
	If <code>imakeidx</code> and <code>hyperref</code> are loaded, adds <code>hyperref</code> in the index.	173	v1.10.3.	General: Debug of cross-referencing.	1
	<code>\endquote</code> : Correction of sectioning commands in parallel texts. . .	206	v1.10.4.	General: Debug of critical notes in <code>edtabular</code> environment.	1
	<code>\get@index@command</code> : Debug <code>\get@index@command</code> and compatibility with <code>hyperref</code> package.	174	v1.10.5.	General: Debug of <code>\pausenumbering</code>	1
	<code>\newhookcommand@series@reload</code> : Debug <code>\beforenotesX</code> and <code>\maxhnotesX</code> which didn't work.	149		Debug of <code>\xxref</code>	1
	<code>\prevpage@num</code> : Correct <code>\parafootsep</code> when using with <code>ledgroup</code> . . .	120	v1.10.6.	General: Debug of interaction between <code>\autopar</code> and <code>\pausenumbering</code>	1
v1.8.1.	General: Debug endnotes when more than one series is used (change the position where tools for endnotes are defined). . .	139	v1.11.0.	General: Add hooks to disable the font selection for lemma in footnote.	22
v1.8.2.	General: Debug compatibility problem with <code>hebrew</code> option of <code>babel</code> package.	1	v1.11.1.	General: Correct a bug when a critical note starts with plus or minus.	1
v1.8.3.	General: Fixes spurious spaces added by v1.7.0.	1	v1.12.0.	<code>\@nl@reg</code> : To ensure compatibility with <code>\musixtex</code> , <code>\@l</code> becomes <code>\@l</code> . Consequently, <code>\@l@reg</code> becomes <code>\@nl@reg</code>	67
v1.8.5.	General: Debug indexing in right column, with <code>eledpar</code>	173		General: Add <code>\ledinnernote</code> and <code>\ledouternote</code> commands. . .	32
v1.9.0.	<code>\doxtrafeet</code> : Add <code>\fnpos</code> to choice the order of footnotes.	155		Add hyperlink to <code>crossref</code> (needs <code>hyperref</code> package).	30
	<code>\l@dfteetendmini</code> : Add <code>\mpfnpos</code> to choice the order of footnotes in <code>minipage</code> / <code>ledgroup</code>	169		Compatibility with <code>musixtex</code> . . .	1
v1.10.0.	General: Add <code>\pstartref</code> and <code>\xpstartref</code> to refer to a <code>pstart</code> number (extension of <code>\edlabel</code>).	1		Debug <code>eledmac</code> sectioning command after using <code>\resumenumbering</code>	1
	<code>\endquote</code> : Correction of sectioning commands in parallel texts. . .	206		Ensure that <code>imakeidx</code> is loaded before <code>eledmac</code>	173
				New hooks: <code>\afterXrule</code> and <code>\aftererruleX</code>	23
				New options for ragged-paragraph notes	23
				New sectioning commands. . . .	38
				Optional arguments for <code>\pstart</code> and <code>\pend</code>	11

<code>\AtEveryPstart</code> : New optional argument for <code>\pstart</code> , to execute code before it.	88		
<code>\edindex</code> : Use correctly default index when <code>imakeidx</code> is loaded.	174		
<code>\endquote</code> : <code>\ledxxx</code> sectioning commands are deprecated and replaced by <code>\eledxxx</code> commands.	206		
<code>\ifledRcol@</code> : Add <code>\ifledRcol@</code> for <code>eledpar</code>	52		
<code>\initnumbering@reg</code> : <code>\beginnumbering</code> is defined only on <code>eledmac</code> , not on <code>eledpar</code>	53		
<code>\l@dcnote</code> : <code>\l@dlnote</code> , <code>\l@drnote</code> and <code>\l@dcnote</code> defined only one time, in <code>eledmac</code> , including needs for <code>eledpar</code> case.	165		
<code>\l@dgetsidenote@margin</code> : <code>\sidenotemargin</code> is now directly defined in <code>eledmac</code> to be able to manage <code>eledpar</code>	164		
<code>\l@dunhbox@line</code> : <code>\do@line</code> is split in more little commands.	92		
<code>\newhookcommand@series@reload</code> : Debug <code>\beforenotesX</code> and <code>\maxhnotesX</code> which didn't work when called after <code>\footparagraphX</code>	149		
Debug <code>\beforeXnotes</code> and <code>\maxhXnotes</code> which didn't work when called after <code>\footparagraph</code>	149		
<code>\pend</code> : New optional argument for <code>\pend</code> , to execute code after it.	89		
<code>\stanza</code> : &can have an optional argument: content to be printed after.	185		
<code>\Stanza</code> : can have an optional argument: content to be printed before.	185		
Add <code>\newverse</code> macro, <code>\falseverse</code> deprecated.	185		
v1.12.1.			
<code>\wrap@edcrossref</code> : Fix spurious spaces.	161		
v1.12.2.			
<code>\l@dunhbox@line</code> : Fix a bug with critical notes at the tops of pages (added by v12.0.0)	92		
		v1.12.3.	
		General: Add macros for new messages since v0.7	47
		Correct bug with side and familiar notes in tabular environments.	1
		Debug <code>\eledxxx</code> with some paper size	1
		Debug <code>\ledinnernote</code> and <code>\ledouternote</code> commands in the top of pages.	32
		Debug left and right notes (bugs added by 1.12.0)	1
		Underline lemma in <code>\eledxxx</code> when using draft mode.	1
		<code>\eledmac@error</code> : Replaced error messages	48
		<code>\flag@end</code> : <code>\flag@start</code> and <code>\flag@end</code> are now defined only one time for <code>eledmac</code> and <code>eledpar</code>	75
		<code>\flag@start</code> send a error message when a <code>\edtext</code> is done without insert (note)	75
		v1.12.4.	
		General: Debug spurious page breaks before <code>\chapter</code> (bug added by 1.12.0)	1
		v1.12.5.	
		<code>\@edindex@hyperref</code> : Debug <code>\edindex</code> when <code>hyperref</code> is not loaded	178
		<code>\@ssect</code> : Debug <code>\eledchapter</code> in parallel with <code>memoir</code>	211
		<code>\doinsidelinehook</code> : Added <code>\dolinehook</code> and <code>\doinsidelinehook</code>	93
		<code>\endnumbering</code> : Allow to mix parallel columns and normal text when using <code>\pausenumbering</code>	54
		<code>\l@dgobblearg</code> : <code>\l@dgobblearg</code> becomes <code>\l@dgobbleoptarg</code>	190
		<code>\l@drestoreforedtext</code> : Debug optional arguments of <code>\Xfootnote</code> in tabular context	192
		<code>\resumenumbering</code> : Debug <code>\resumenumbering</code>	55
		v1.12.6.	
		<code>\noeledsec</code> : Add <code>\noeledsec</code> macro.	215

v1.12.7.	ginning of a document when hyperref is not loaded.	179
\wrap@edcrossref: \wrap@edcrossref is now robust	Moves gobbling command	51
v1.12.8.	Provide \@gobblefour	51
\flag@end: \flag@start don't send a error message when a \edtext is done without insert (note) but have a endnote	\edindex: Let eledmac take ad- vantage of imakeidx even when memoir class is used	174
v1.13.0.	v1.14.1. \essect: Debug sectioning com- mands when using both handout and hyperref package.	213
General: Add \Xnoteswidthliketwocolumns and \notesXwidthliketwocolumns	v1.14.2. \essect: Debug \edtext after starred sectioning commands when using memoir class.	211
\ifledsecnolinenumber: Added widthliketwocolumns option		
\newhooktoggle@series: Add \newhookcommand@toggle@reload	v1.15.0. General: Fix bug with footnotes layout when using some options of the geometry package (bug add by v1.13.0).	1
\para@footsetupX: In \para@footsetupX, use \columnwidth instead of \hsize	New commands \AtEveryPstart and \AtEveryPend.	11
v1.13.1.	New tools to prevent ambiguous references in lemma	18
General: Coming back of page and line breaking penalties's man- agement, deleted by error in v0.17.	\endsub: Restore subline feature (disabled by mistake in v1.8.0).	76
Debug quotation environment in- side of a \pstart preceded by a sectioning command.	\footparagraphX: Correct bug with paragraphed familiar foot- notes setting.	134
\thepstart: Add \l@dzero penalties in \pstart	\if@lemmacommand@: New boolean \iflemmacommand@.	84
v1.13.2.	v1.15.1. \line@list@stuff: Revert modifi- cation of 1.5.2 which makes bug with numbering. Leave verti- cal mode to solve spurious space before minipage.	74
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.	v1.16.0. General: Compatibility of standard footnotes with some biblatex styles.	1
v1.13.4.	New \stanzaindent command.	1
General: Fix bug with index when memoir class is used without hyperref	\critext: \critext and \edtext are now defined only in eled- mac, not in eledpar. Debug wrong numbering when using \sameword + eledpar + \tag command.	81
v1.14.0.		
General: Debug spurious characters before endnotes.		
Delete previous override of \l@d@wrindexhyp at the be-		