

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 v1.18.0, last revised 2015/02/23.

[†]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	11
3.2.1 Basis	11
3.2.2 Content before specific <code>\pstart</code> and after <code>\pend</code>	11
3.2.3 Content before every <code>\pstart</code> and after every <code>\pend</code>	11
3.2.4 Producing automatically <code>\pstart... \pend</code>	12
3.2.5 Numbering paragraphs (<code>\pstart</code>)	12
3.2.6 Languages written in Right to Left	12
3.2.7 Memory limits	12
3.3 Lineation commands	13
3.3.1 Disabling lineation	13
3.3.2 Setting lineation start and step	13
3.3.3 Setting lineation reset	14
3.3.4 Setting line number margin	14
3.3.5 Other settings	14
3.4 Changing the line numbers	15
4 The apparatus	16
4.1 Commands	16
4.2 Disambiguation of identical words in the apparatus	18
4.3 Alternate footnote formatting	19
4.4 Display options	20
4.4.1 Control line number printing	20
4.4.2 Separator between the lemma and the note content	21
4.4.3 Font style	22
4.4.4 Font of the lemma	22
4.4.5 Styles of notes content	22
4.4.6 Arbitrary code at the beginning of notes	23
4.4.7 Options for notes in columns	23
4.4.8 Options for paragraphed footnotes	23
4.4.9 Options for block of notes	24
4.5 Page layout	25
4.6 Fonts	25
4.7 Create a new series	26

5 Verse	26
5.1 Repeating stanza indents	27
5.2 Manual stanza indent	28
5.3 Stanza breaking	28
5.4 Hanging symbol	28
5.5 Long verse and page break	29
5.6 Various tools	29
5.7 Hanging symbol	29
5.8 Text before/after verses	30
6 Grouping	30
7 Crop marks	30
8 Endnotes	31
9 Cross referencing	31
10 Side notes	33
11 Familiar footnotes	33
11.1 Position of the familiar footnotes	34
12 Indexing	34
13 Tabular material	35
14 Sectioning commands	39
15 Quotation environments	40
16 Page breaks	40
17 Miscellaneous	41
17.1 Known and suspected limitations	41
17.2 Use with other packages	42
17.3 Parallel typesetting	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	56
21.1 Choosing the system of lineation	56
21.2 List macros	60
21.3 Line-number counters and lists	61
21.4 Reading the line-list file	65
21.5 Commands within the line-list file	67
21.6 Writing to the line-list file	73
22 Marking text for notes	77
22.1 <code>\edtext</code> (and <code>\critext</code>) itself	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	105
24.1 Fonts	105
24.2 Outer-level footnote commands	105
24.3 Normal footnote formatting	107
24.4 Standard footnote definitions	113
24.5 Paragraphed footnotes	115
24.6 Insertion of the footnotes separator	120
24.7 Columnar footnotes	121
25 Familiar footnotes	126
25.1 Generality	126
25.2 Footnote formats	127
25.3 Two columns footnotes	131
25.4 Three columns footnotes	133
25.5 Paragraphed footnotes	134
26 Footnotes' width for two columns	137
27 Footnotes' order	138
28 Footnotes' rule	139

29 Footnotes' output	139
30 Endnotes	140
31 Generate series	142
31.1 Test if series is still existing	143
31.2 Create all commands to memorize display options	143
31.3 Create inserts, needed to add notes in foot	144
31.4 Create commands for critical apparatus, <code>\Xfootnote</code>	144
31.5 Create tools for familiar footnotes (<code>\footnoteX</code>)	145
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	151
32 Output routine	153
33 Cross referencing	159
34 Side notes	165
35 Minipages and such	171
36 Indexing	174
36.1 Memoir compatibility	176
36.2 Normal setting	177
36.3 Choose the right variant	178
36.4 Hyperref compatibility	179
37 Macro as environment	180
38 Verse	183
39 Arrays and tables	187
40 Section's title commands	207
40.1 Deprecated commands	207
40.2 New commands : <code>\eledxxx</code>	210
41 Page breaking or no page breaking depending of specific lines	220
42 Long verse: prevents being separated by a page break	221
43 The End	222

Appendix A Some things to do when changing version	223
Appendix A.1 Migrating from edmac	223
Appendix A.2 Migration from ledmac to eledmac	224
Appendix A.3 Migration to eledmac 1.5.1	225
Appendix A.4 Migration to eledmac 1.12.0	225
Appendix A.5 Migration to eledmac 17.1	226
References	227
Index	227
Change History	248

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 TeX 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. \LaTeX and `eledmac` will take care of the formatting and visual correlation of all the disparate types of information.

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

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

`EDNOTES` [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 Appendix A.1); the complete source code for the package, with extensive documentation (in sections 18 and following); and an Index to the source code. We do not suggest that you need to read the source code for this package in order to use it; we provide this code primarily for reference, and many of our comments on it repeat material that is also found in the earlier sections. But no documentation, however thorough, can cover every question that comes up, and many can be answered quickly by consultation of the code. On a first reading, we suggest that you should read only the general

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 `TEX` format for critical editions’, *TUGboat* 11 (1990), pp.623–643.

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

At the time of writing (1994), we are pleased to be able to say that `EDMAC` is being used for real-life book production of several interesting editions, such as the Latin texts of Euclid’s *Elements*,² an edition of the letters of Nicolaus Coperni-

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).

cus,³ Simon Bredon's *Arithmetica*,⁴ a Latin translation by Plato of Tivoli of an 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.20). These changes can trigger small problems with the old customization. That is why a new name was selected: *eledmac*. To migrate from *ledmac* to *eledmac*, please read Appendix A.2 (p.224).

1.2.3 List of works edited with (e)ledmac

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

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.

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.

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

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

3.2.1 Basis

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

```
\pstart
<paragraph of text>
\pend
```

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

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

<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>	

3.2.2 Content before specific `\pstart` and after `\pend`

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.

3.2.3 Content before every `\pstart` and after every `\pend`

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

3.2.4 Producing automatically `\pstart... \pend`

`\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

3.2.5 Numbering paragraphs (`\pstart`)

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

```
\setcounter{numberpstart}{value}
```

On each `\beginnumbering` the numbering restarts.

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

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

3.2.6 Languages written in Right to Left

If you use languages writtin in Right to Left, we Lua^AT_EX or X_Y^AT_EX, you must switch text direction `\before` the `\pstart` command.

3.2.7 Memory limits

`\pausenumbering` This paragraph is kept for history, but problem described below should not appear with **eledmac**. **eledmac** stores a lot of information about line num-

`\resumenumbering` bers and footnotes in memory as it goes through a numbered section. But at the end of such a section, it empties its memory out, so to speak. If your text has a very long numbered section it is possible that your \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:

```

\beginnumbering
\pstart
Paragraph of text.
\pend
\pausenumbering
\resumenumbering
\pstart
Another paragraph.
\pend
\endnumbering

```

- 1 Paragraph of
- 2 text.
- 3 Another paragraph.

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

```
\newcommand{\memorybreak}{\pausenumbering\resumenumbering}
```

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

3.3 Lineation commands

3.3.1 Disabling lineation

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

3.3.2 Setting lineation start and step

`\firstlinenum` By default, `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

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

difference between successive numbered lines. For example, to start printing numbers at the first line and to have every other line numbered:

```
\firstlinenum{1} \linenumincrement{2}
```

```
\firstsublinenum
\sublinenumincrement
\linenumberlist
```

There are similar commands, `\firstsublinenum{<num>}` and `\sublinenumincrement{<num>}` for controlling sub-line numbering. You can define `\linenumberlist` to specify a non-uniform distribution of printed line numbers. For example:

```
\def\linenumberlist{1,2,3,5,7,11,13,17,19,23,29}
```

to have numbers printed on prime-numbered lines only. There must be no spaces within the definition which consists of comma-separated decimal numbers. The numbers can be in any order but it is easier to read if you put them in numerical order. Either omitting the definition of `\linenumberlist` or following the vacuous definition

```
\def\linenumberlist{}
```

the standard numbering sequence is applied. The standard sequence is that specified by the combination of the `firstlinenum`, `linenumincrement`, `firstsublinenum` and `linenumincrement` counter values.

3.3.3 Setting lineation reset

```
\lineation
```

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

3.3.4 Setting line number margin

```
\linenummargin
```

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

```
\linenummargin{left}
```

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

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

3.3.5 Other settings

```
\leftlinenum
\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` 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` 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:

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

`\Afootnote` Five separate series of the footnotes are maintained; each macro taking one
`\Bfootnote` argument like `\Afootnote{<text>}`. When all five are used, the A notes appear
`\Cfootnote` in a layer just below the main text, followed by the rest in turn, down to the E
`\Dfootnote` notes at the bottom. These are the main macros that you will use to construct
`\Efootnote` 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.31)
`\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:

```
\edtext{I saw my friend
  \edtext{Smith}{\Afootnote{Jones      1 I saw my friend
    C, D.}} on Tuesday.}              2 Smith on Tuesday.
  {\lemma{I \dots\ Tuesday.}          2 Smith] Jones C, D.
  \Bfootnote{The date was              1–2 I ... Tuesday.]
    July 16, 1954.}                    The date was July 16, 1954.
}
```

`\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 number for the start of the lemma; the same three numbers for the end of the lemma;

and the font specifier for the lemma. As the argument to `\linenum`, you specify those seven parameters in that order, separated by vertical bars (the `|` character). However, you can retain the value computed by `eledmac` for any number by simply omitting it; and you can omit a sequence of vertical bars at the end of the argument. For example, `\linenum{|||23}` changes one number, the ending page number of the current lemma.

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

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

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

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`.

ous term with the `\sameword` command:

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

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

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

- In the first argument of `\edtext`, use `\sameword` with the optional argument `'[inlemma]`.
- In the content of `\lemma`, use `\sameword` with no optional argument.

Like this :

```
\edtext{\sameword[inlemma]{sw}}{\lemma{\sameword{sw} some lemma}\Afootnote{some note}}
```

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

```
\newcommand{\showwordrank}[2]{%
  #1\textsuperscript{#2}%
}
```

4.3 Alternate footnote formatting

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

`\footparagraph` 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 `\foottwocol` you can select a different format for a series of notes. `\footthreecol`

- `\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 L^AT_EX 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 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.

`\pstartinfootnoteeverytime`

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

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

have called `\numberpstarttrue`. We don't know why you would like to print the `pstart` number in the notes and not in the main text. However, if you want to do it, you can call `\pstartinfootnoteeverytime[⟨s⟩]`. In this case, the `pstart` number will be printed every time in footnote.

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

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

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

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

`\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.

<code>\afterlemmaseparator</code>	Using <code>\afterlemmaseparator[⟨s⟩]{⟨l⟩}</code> 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.
<code>\nolemmaseparator</code>	You can suppress the lemma separator, using <code>\nolemmaseparator[⟨s⟩]</code> , which is simply a alias of <code>\lemmaseparator[⟨s⟩]{}</code> .
<code>\inplaceoflemmaseparator</code>	With <code>\inplaceoflemmaseparator[⟨s⟩]{⟨l⟩}</code> you can add a space if no lemma separator is printed. The default value is 1 em.

4.4.3 Font style

<code>\Xnotenumfont</code>	<code>\Xnotenumfont[⟨s⟩]{⟨command⟩}</code> is used to change the font style for line numbers in critical footnotes ; <code>⟨command⟩</code> must be one (or more) switching command, like <code>\bfseries</code> .
<code>\Xendnotenumfont</code>	<code>\Xendnotenumfont[⟨s⟩]{⟨command⟩}</code> is used to change the font style for line numbers in critical footnotes. <code>⟨command⟩</code> must be one (or more) switching command, like <code>\bfseries</code> .
<code>\notenumfontX</code>	<code>\notenumfontX[⟨s⟩]{⟨command⟩}</code> is used to change the font style for note numbers in familiar footnotes. <code>⟨command⟩</code> must be one (or more) switching command, like <code>\bfseries</code> .
<code>\Xnotefontsize</code>	<code>\Xnotefontsize[⟨s⟩]{⟨command⟩}</code> is used to define the font size of critical footnotes of the series. The default value is <code>\footnotesize</code> . The <code>⟨command⟩</code> must not be a size in pt, but a standard L ^A T _E X size, like <code>\small</code> .
<code>\notefontsizeX</code>	<code>\notefontsizeX[⟨s⟩]{⟨command⟩}</code> is used to define the font size of critical footnotes of the series. The default value is <code>\footnotesize</code> . The <code>⟨command⟩</code> must not be a size in pt, but a standard L ^A T _E X size, like <code>\small</code> .
<code>\Xendnotefontsize</code>	<code>\Xendnotefontsize[⟨s⟩]{⟨l⟩}</code> is used to define the font size of end critical footnotes of the series. The default value is <code>\footnotesize</code> . The <code>⟨command⟩</code> must not be a size in pt, but a standard L ^A T _E X size, like <code>\small</code> .

4.4.4 Font of the lemma

<code>\Xlemmadisablefontselection</code>	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 <code>\Xlemmadisablefontselection[⟨s⟩]</code> command allows to disable it for a specific series.
<code>\Xendlemmadisablefontselection</code>	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 <code>\Xendlemmadisablefontselection[⟨s⟩]</code> to disable it for a specific series.

4.4.5 Styles of notes content

<code>\Xhangindent</code>	For critical notes NOT paragraphed you can define an indent with <code>\Xhangindent[⟨s⟩]{⟨l⟩}</code> , 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.
<code>\hangindentX</code>	For familiar notes NOT paragraphed you can define an indent with <code>\Xhangindent[⟨s⟩]{⟨l⟩}</code> ,

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}.}}
```

<code>\bhookXnote</code>	<code>\bhookXnote[⟨series⟩]{⟨code⟩}</code> is to be used at the beginning of the critical footnotes.
<code>\bhooknoteX</code>	<code>\bhooknoteX[⟨series⟩]{⟨code⟩}</code> is to be used at the beginning of the familiar footnotes.
<code>\bhookXendnote</code>	<code>\bhookXendnote[⟨series⟩]{⟨code⟩}</code> is to be used at the beginning of the end-notes.

4.4.7 Options for notes in columns

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

<code>\hsizetwocol</code>	<code>\hsizetwocol[⟨s⟩]{⟨l⟩}</code> is used to change width of a column when critical notes are displaying in two columns. Default value is <code>.45 \hsizetwocol</code> .
<code>\hsizethreecol</code>	<code>\hsizethreecol[⟨s⟩]{⟨l⟩}</code> is used to change width of a column when critical notes are displaying in three columns. Default value is <code>.3 \hsizethreecol</code> .
<code>\hsizetwocolX</code>	<code>\hsizetwocolX[⟨s⟩]{⟨l⟩}</code> is used to change width of a column when familiar notes are displaying in two columns. Default value is <code>.45 \hsizetwocolX</code> .
<code>\hsizethreecolX</code>	<code>\hsizethreecolX[⟨s⟩]{⟨l⟩}</code> is used to change width of a column when familiar notes are displaying in three columns. Default value is <code>.3 \hsizethreecolX</code> .

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 1.2em plus .6em minus .6em.</p> <p>Be careful, the standard L^AT_EX footnote rule, which is used by eledmac, 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 1.2em plus .6em minus .6em.</p> <p>Be careful, the standard L^AT_EX footnote rule, which is used by eledmac, 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 0pt.</p> <p>Be careful, the standard L^AT_EX footnote rule, which is used by eledmac, 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>\aftererruleX</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 0pt.</p> <p>Be careful, the standard L^AT_EX footnote rule, which is used by eledmac, 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 0pt. You can disable this feature by setting the length to 0pt.</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>	<p>You can want the space before the first printed (in a page) series of familiar notes not to be the same as before other series. Default value is 0pt. You can do it with <code>\prenotesX{⟨l⟩}</code>. You can disable this feature by setting the length to 0 pt.</p> <p>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 <code>\maxhXnotes[⟨s⟩]{⟨l⟩}</code>. 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 <code>\maxhXnotes{.33\textheight}</code>.</p>
<code>\maxhXnotes</code>	
<code>\maxhnotesX</code>	<p><code>\maxhnotesX[⟨s⟩]{⟨l⟩}</code> is the same as previous, but for familiar footnotes.</p> <p>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.</p>

4.5 Page layout

You should set up the page layout parameters, and in particular the `\baselineskip` of the footnotes (this is done for you if you use the standard `\notefontsetup`), before you call any of these macros because their action depends on these; too much or too little space will be allotted for the notes on the page if these macros use the wrong values.¹⁶

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

- `\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

```
\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

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.

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 `LATEX` `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.

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

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

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

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

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

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

If you want the hanging verse to be flush right, you can use `\hanginsymbol`: see p. 28.

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

5.1 Repeating stanza indents

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

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

is like

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

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

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 \TeX 's grouping conventions, so if one stanza among several has a different structure, its indentations (penalties) may be set within a group; the prior values will be restored when the group ends.

5.2 Manual stanza indent

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

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

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

5.3 Stanza breaking

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

The command

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

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

The first entry “1” is a control value. If it is zero, then no penalties are passed on to 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

`\hangingsymbol` 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

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

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

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

`\flagstanza` Putting `\flagstanza[⟨len⟩]{⟨text⟩}` at the start of a line in a stanza (or elsewhere) will typeset `⟨text⟩` at a distance `⟨len⟩` before the line. The default `⟨len⟩` is `\stanzaindentbase`.

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

```
\newcounter{stanzanum}
\setcounter{stanzanum}{0}
\newcommand*{\startstanzahook}{\refstepcounter{stanzanum}}
\newcommand{\numberit}{\flagstanza{\thestanzanum}}
...
\stanza
\numberit First line...&
    rest of stanza\&

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

5.7 Hanging symbol

`\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 L^AT_EX 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{<letter>}` closes the `.end` file that contains the text of the endnotes, if it's open, and prints one series of endnotes, as specified by a series-letter argument, e.g., `\doendnotes{A}`. `\endprint` is the macro that's called to print each note. It uses `\select@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{<num>}` 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{<lab>}`. `<lab>` can be almost anything you like, including letters, numbers, punctuation, or a combination—anything but spaces; you might say `\edlabel{toves-3}`, for example.¹⁸

`\edpageref` Elsewhere in the text, either before or after the `\edlabel`, you can refer to its location via `\edpageref{<lab>}`, or `\edlineref{<lab>}`¹⁹, `\sublineref{<lab>}`, or `\pstartref{<lab>}`. These commands will produce, respectively, the page, line, sub-line and pstart on which the `\edlabel{<lab>}` command occurred.

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

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

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

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

¹⁹Previously, the `\edlineref` command was `\lineref`. But some packages also define `\lineref`. That is why you should use `\edlineref` instead of `\lineref`. `eledmac` defines `\lineref` as equal to `\edlineref`, except if one package has also defined a `\lineref` command.

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 num-
`\ref` bered text, and operate in the familiar fashion.
`\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 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 (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 definitions are:

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

These can of course be changed to suit.

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

11 Familiar footnotes

The `footmisc` package [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}{\normalfont,}`
and can be changed if necessary.

`\footnoteA` As well as the standard L^AT_EX footnotes generated via `\footnote`, the pack-
`\footnoteB` age also provides five series of additional footnotes called `\footnoteA` through
`\footnoteC` `\footnoteE`. These have the familiar marker in the text, and the marked text at
`\footnoteD` the foot of the page can be formatted using any of the styles described for the
`\footnoteE` 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
`\footparagraphX` (e.g., B). `\footnormalX` is the typical footnote format. With `\footparagraphX`
`\foottwocolX` the series is typeset a one paragraph, with `\foottwocolX` the notes are in two
`\footthreecolX` columns, and are in three columns with `\footthreecolX`.

`\thefootnoteA` As well as using the `\foot...X` macros to specify the general footnote arrange-
`\bodyfootmarkA` ment for a series, each series uses a set of macros for styling the marks. The mark
`\footfootmarkA` 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.26).

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 & linenumber 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 & linenumber 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.

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

```

\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{C}) 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.

`\edtabcolsep` The distance between the columns is controlled by the length `\edtabcolsep`.
`\spreadmath` `\spreadmath{<math>}` typesets `{<math>}` but the `{<math>}` has no effect on
`\spreadtext` the calculation of column widths. `\spreadtext{<text>}` is the analogous command
for use in `edtabular` environments.

```

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

```

1	2	3	4
	$F + G + C$		
a	bb	ccc	dddd

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

Note that every row must have the same number of columns, even if some would not appear to be necessary.

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

```
\begin{edtabularr}
1          & & & & & \\
Q          & & & fd & h & qwertziohg \\
v          & wptz & x & y & vb & \\
g          & nnn & & \edrowfill{3}{5}{\upbracefill} & & \\
\edrowfill{1}{3}{\downbracefill} & & & pq & dgh & \\
k          & & l & co & ghweropjklmnbcxys & \\
1          & 2 & 3 & \edrowfill{4}{5}{\hrulefill} & & \\
\end{edtabularr}
```

1	2	3	4	5
Q		fd	h	qwertziohg
v	wptz	x	y	vb
g	nnn	$\underbrace{\hspace{10em}}$		
			pq	dgh
k	$\underbrace{\hspace{4em}}$		l	co ghweropjklmnbcxys
1	2	3	$\underline{\hspace{10em}}$	

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

```
\newcommand*{\upbracketfill}{%
  \vrule height 4pt depth 0pt\hrulefill\vrule height 4pt depth 0pt}
```

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

```
\begin{edarrayc}
1 & 2 & & & 3 & 4 \\
a & \edrowfill{2}{3}{\upbracketfill} & & d \\
A & B & & C & D \\
\end{edarrayc}
```

1	2	3	4
<i>a</i>	$\underbrace{\hspace{1.5em}}$		<i>d</i>
<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>

`\edatleft` `\edatleft[$\langle math \rangle$]{ $\langle symbol \rangle$ }{ $\langle halfheight \rangle$ }` typesets the math $\langle symbol \rangle$ as `\left $\langle symbol \rangle$` 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 $\langle symbol \rangle$` 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}

```

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

`\edbeforetab` `\edbeforetab{<text>}{<entry>}`, where `<entry>` is an entry in the leftmost column, typesets `<text>` left justified before the `<entry>`. Similarly `\edaftertab{<entry>}{<text>}`, where `<entry>` is an entry in the rightmost column, typesets `<text>` right justified after the `<entry>`.

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}

```

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

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

```

\edvertdots
\begin{edarrayr}
a & b & C & d & \\
v & w & x & y & \\
m & n & o & p & \\
k & & L & cvb & \edvertline{4pc}
\end{edarrayr}

```

a	b	C	d	
v	w	x	y	
m	n	o	p	
k		L	cvb	

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 \LaTeX commands. Each individual command must be called alone in a `\pstart... \pend`:

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

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

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

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. $n + 1$.

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 `eledpbsetting{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 \LaTeX 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` \LaTeX is a three-pass system, but even after a document has been processed three times, there are some tricky situations in which the page breaks decided by \TeX never settle down. At each successive run, `eledmac` may oscillate between two

different sets of page decisions. To stop this happening, should it arise, Wayne Sullivan suggested the inclusion of the quantity `\ballast`. The amount of `\ballast` will be subtracted from the penalties which apply to the page breaks calculated on the *previous* run through \TeX , thus reinforcing these breaks. So if you find your page breaks oscillating, say

```
\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. 25, and described in more detail on p. 117, really is a nuisance if that's something you need to do. There are some possible solutions, described by Michael Downes, but this area remains unsatisfactory.

\LaTeX has a reputation for putting things in the wrong margin after a page break. The `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 \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 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 \LaTeX internals and `eledmac` does things that are not normally seen in \LaTeX .

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 this²⁰ 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
\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`

²⁰Reported by Dirk-Jan Dekker in the CTT thread ‘Incompatibility of “color” package’ on 2003/08/28.

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

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}[2015/02/23 v1.18.0 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}{}%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 57 \newcommand*{\led@err@NumberingStarted}{%
\led@err@NumberingShouldHaveStarted 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@err@edtextoutsidepstart
63 \newcommand*{\led@err@edtextoutsidepstart}{%
64 \eledmac@error{\string\edtext\space outside numbered paragraph (\pstart...\pend)}{\@ehc}}

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

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

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

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

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

```



```

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

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

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

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

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

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

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

```

```

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

\led@warn@AddfootinsXobsolete
\led@warn@Addfootinsobsolete 125 \newcommand{\led@warn@AddfootinsXobsolete}{%
126   \eledmac@warning{AddfootinsX is obsolete in eledmac 1.0. Use newseries instead.}%
127 }%
128 \newcommand{\led@warn@AddfootinsObsolete}{%
129   \eledmac@warning{Addfootins is obsolete in eledmac 1.0. Use newseries instead.}%
130 }%

\led@warn@SeriesStillExist
131 \newcommand{\led@warn@SeriesStillExist}[1]{%
132   \eledmac@warning{Series #1 is still existing !}%
133 }%

\led@err@ManySidenotes
\led@err@ManyLeftnotes 134 \newcommand{\led@err@ManySidenotes}{%
\led@err@ManyRightnotes 135   \ifledRcol%
136     \eledmac@warning{\itemcount@space sidenotes on line \the\line@numRspace p. \the\pag
137   }%
138   \eledmac@warning{\itemcount@space sidenotes on line \the\line@numspace p. \the\pag
139   \fi%
140 }%
141 \newcommand{\led@err@ManyLeftnotes}{%
142   \ifledRcol%
143     \eledmac@warning{\itemcount@space leftnotes on line \the\line@numRspace p. \the\pag
144   }%
145   \eledmac@warning{\itemcount@space leftnotes on line \the\line@numspace p. \the\pag
146   \fi%
147 }%
148 \newcommand{\led@err@ManyRightnotes}{%
149   \ifledRcol%
150     \eledmac@warning{\itemcount@space rightnotes on line \the\line@numRspace p. \the\pag
151   }%
152   \eledmac@warning{\itemcount@space rightnotes on line \the\line@numspace p. \the\pag
153   \fi%
154 }%

\led@war@FalseverseDeprecated
\led@war@ledxxxDeprecated 155 \newcommand{\led@war@FalseverseDeprecated}{%
156   \eledmac@warning{\string\falseverse\space deprecated. Look at \string\newverse\space ins
157 }%
158 \newcommand{\led@war@ledxxxDeprecated}[1]{%
159   \eledmac@warning{\string\led#1\space deprecated. Look at \string\led#1 instead.}%
160 }%

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

```

```

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

```

err@EdtextWithoutFootnote

```

172 \newcommand*\led@err@EdtextWithoutFootnote}{%
173   \eledmac@error{edtext without Xfootnote. Check syntax.}{\@ehd}%
174 }%

```

error@ImakeidxAfterEledmac

```

175 \newcommand*\led@error@ImakeidxAfterEledmac}{%
176   \eledmac@error{Imakeidx must be loaded before eledmac.}{\@ehd}%
177 }%

```

or@IndextoolsAfterEledmac

```

178 \newcommand*\led@error@IndextoolsAfterEledmac}{%
179   \eledmac@error{Indextools must be loaded before eledmac.}{\@ehd}%
180 }%

```

19.5 Gobbling

\@gobblethree

```

\@gobblefour 181 \providecommand*\@gobblethree}[3]{
182 \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

```

183 \let\linenumberlist=\empty
184

```

\@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).

```

185 \newcount\@l@tempcnta \newcount\@l@tempcntb

```

20 Sectioning commands

`\section@num` You use `\beginnumbering` and `\endnumbering` 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^AT_EX will maintain and display a ‘section number’ as a count named `\section@num` that counts how many `\beginnumbering` and `\resumenumbers` commands have appeared; it needn’t be related to the logical divisions of your text.

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

```
186 \newcount\section@num
187 \section@num=0
188 \let\extensionchars=\empty
```

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

```
189 \newif\ifnumbering
```

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

`\ifl@dpairing`

`\ifl@dpaging`

`\l@dpagingtrue`

`\l@dpagingfalse`

`\ifl@dpagingpages`

`\l@dpagingpagestrue`

`\l@dpagingpagesfalse`

`\ifl@dpagingcolumns`

`\l@dpagingcolumnstrue`

`\l@dpagingcolumnsfalse`

`\l@dpairingtrue`

`\l@dpairingfalse`

`\ifpst@rtedL`

`\pst@rtedLtrue`

`\pst@rtedLfalse`

`\l@dnumpststartL`

`\ifledRcol`

`\ifledRcol@`

```
190 \newif\ifl@dpairing
```

```
191 \l@dpairingfalse
```

```
192 \newif\ifl@dpaging%
```

```
193 \l@dpagingfalse%
```

```
194 \newif\ifl@dpagingpages%
```

```
195 \l@dpagingpagesfalse%
```

```
196 \newif\ifl@dpagingcolumns%
```

```
197 \l@dpagingcolumnsfalse%
```

```
198 \newif\ifpst@rtedL
```

```
199 \pst@rtedLfalse
```

```
200 \newcount\l@dnumpststartL
```

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

```
201 \newif\ifledRcol
```

```
202 \newif\ifledRcol@
```

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

```
203 \newif\ifnumberingR
```

`\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`.

```
204 \newcommand*{\beginnumbering}{%
205   \ifnumbering
206     \led@err@NumberingStarted
207     \endnumbering
208   \fi
209   \global\numberingtrue
210   \global\advance\section@num \@ne
211   \initnumbering@reg
212   \message{Section \the\section@num }%
213   \line@list@stuff{\jobname.\extensionchars\the\section@num}%
214   \l@dend@stuff
215   \setcounter{pstart}{1}
216   \ifl@dpairing
217     \global\l@dnumpstartsL \z@
218     \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.

```
219 \else
220   \begingroup
221   \initnumbering@sectcmd
222   \ifwidthliketwocolumns%
```

```

223     \csuse{setwidthliketwocolumns@\columns@position}%
224     \csuse{setpositionliketwocolumns@\columns@position}%
225     \fi%
226 \fi
227 \gdef\eled@sections@{}%
228 \if@noeled@sec\else%
229     \makeatletter\input{IfFileExists{\jobname.eledsec\the\section@num}{}}{\makeatother}%
230     \immediate\openout\eled@sectioning@out=\jobname.eledsec\the\section@num\relax%
231     \fi%
232 }
233 \newcommand*\initnumbering@reg{%
234     \global\pst@rtedLfalse
235     \global\l@dnumpstartsL \z@
236     \global\absline@num \z@
237     \gdef\normal@page@break{}
238     \gdef\l@prev@pb{}
239     \gdef\l@prev@nopb{}
240     \global\line@num \z@
241     \global\subline@num \z@
242     \global\@lock \z@
243     \global\sub@lock \z@
244     \global\sublines@false
245     \global\let\next@page@num=\relax
246     \global\let\sub@change=\relax
247     \resetprevline@
248     \resetprevpage@num
249 }
250

```

`\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.

```

251 \def\endnumbering{%
252     \ifnumbering
253         \global\numberingfalse
254         \normal@pars
255         \ifl@dpairing
256             \global\pst@rtedLfalse
257         \else
258             \ifx\insertlines@list\empty\else
259                 \global\noteschanged@true
260             \fi
261             \ifx\line@list\empty\else
262                 \global\noteschanged@true
263             \fi
264         \fi
265         \ifnoteschanged@
266             \led@mess@NotesChanged
267         \fi
268     \else

```

```

269 \led@err@NumberingNotStarted
270 \fi
271 \autoparfalse
272 \if@noeled@sec\else%
273 \immediate\closeout\eled@sectioning@out%
274 \fi%
275 \ifl@dpairing\else
276 \global\l@dnumpstartsL=\z@%
277 \endgroup
278 \fi
279 }

```

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

```

280 \newcommand{\pausenumbering}{%
281 \ifautopar\global\autopar@pausetrue\fi%
282 \endnumbering\global\numberingtrue}

```

The `\resumenummering` 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 `\resumenummering` to ensure that `\pausenumbering` was actually invoked.

```

283 \newcommand*{\resumenummering}{%
284 \ifnumbering
285 \ifautopar@pause\autopar\fi
286 \global\pst@rtedLtrue
287 \global\advance\section@num \@ne
288 \led@mess@SectionContinued{\the\section@num}%
289 \line@list@stuff{\jobname.\extensionchars\the\section@num}%
290 \l@dend@stuff
291 \ifl@dpairing\else%
292 \begingroup%
293 \initnumbering@sectcmd%
294 \ifwidthliketwocolumns%
295 \csuse{setwidthliketwocolumns@\columns@position}%
296 \csuse{setpositionliketwocolumns@\columns@position}%
297 \fi%
298 \fi%
299 \else
300 \led@err@NumberingShouldHaveStarted
301 \endnumbering
302 \beginnumbering
303 \fi}
304
305

```

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

21.1 Choosing the system of lineation

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.

```

306 \newif\ifbypage@
307 \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.

308 \newcommand*{\lineation}[1]{%
309   \ifnumbering
310     \led@err@LineationInNumbered
311   \else
312     \def\@tempa{#1}\def\@tempb{page}%
313     \ifx\@tempa\@tempb
314       \global\bypage@true
315       \global\bypstart@false
316       \pstartinfootnote[] [false]
317     \else
318       \def\@tempb{pstart}%
319       \ifx\@tempa\@tempb
320         \global\bypage@false
321         \global\bypstart@true
322         \pstartinfootnote[] [false]
323       \else
324         \def\@tempb{section}
325         \ifx\@tempa\@tempb
326           \global\bypage@false
327           \global\bypstart@false
328           \pstartinfootnote[] [false]
329         \else
330           \led@warn@BadLineation
331         \fi
332       \fi
333     \fi
334   \fi}}
```


`\linenummargin` You call `\linenummargin{<word>}` to specify which margin you want your line numbers in; it takes one argument, a string. You can put the line numbers in the same margin on every page using `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.

```

335 \newcount\line@margin
336 \newcommand*{\linenummargin}[1]{%
337   \l@getline@margin{#1}%
338   \ifnum\@l@tempcntb>\m@ne
339     \global\line@margin=\@l@tempcntb
340   \fi}%
341 \newcommand*{\l@getline@margin}[1]{%
342   \def\@tempa{#1}\def\@tempb{left}%
343   \ifx\@tempa\@tempb
344     \@l@tempcntb \z@
345   \else
346     \def\@tempb{right}%
347     \ifx\@tempa\@tempb
348       \@l@tempcntb \@ne
349     \else
350       \def\@tempb{outer}%
351       \ifx\@tempa\@tempb
352         \@l@tempcntb \tw@
353       \else
354         \def\@tempb{inner}%
355         \ifx\@tempa\@tempb
356           \@l@tempcntb \thr@@
357         \else
358           \led@warn@BadLinenummargin
359           \@l@tempcntb \m@ne
360         \fi
361       \fi
362     \fi
363   \fi}%
364
```

`\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.

```

365 \newcounter{firstlinenum}
366 \setcounter{firstlinenum}{5}
```

```

367 \newcounter{linenumincrement}
368 \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.

369 \newcounter{firstsublinenum}
370 \setcounter{firstsublinenum}{5}
371 \newcounter{sublinenumincrement}
372 \setcounter{sublinenumincrement}{5}
373

\firstlinenum These macros can be used to set the corresponding counters.
\linenumincrement 374 \newcommand*{\firstlinenum}[1]{\setcounter{firstlinenum}{#1}}
\firstsublinenum 375 \newcommand*{\linenumincrement}[1]{\setcounter{linenumincrement}{#1}}
\sublinenumincrement 376 \newcommand*{\firstsublinenum}[1]{\setcounter{firstsublinenum}{#1}}
377 \newcommand*{\sublinenumincrement}[1]{\setcounter{sublinenumincrement}{#1}}
378

\lockdisp When line locking is being used, the \lockdisp{<word>} macro specifies whether
\lock@disp a line number—if one is due to appear—should be printed on the first printed line
\l@getlock@disp or on the last, or by all of them. Its argument is a word, either first, last, or
Initially, it is set to first.
\lock@disp encodes the selection: 0 for first, 1 for last, 2 for all.

379 \newcount\lock@disp
380 \newcommand{\lockdisp}[1]{%
381 \l@getlock@disp{#1}%
382 \ifnum\l@dttempcntb>\m@ne
383 \global\lock@disp=\l@dttempcntb
384 \else
385 \led@warn@BadLockdisp
386 \fi}}
387 \newcommand*{\l@getlock@disp}[1]{
388 \def\@tempa{#1}\def\@tempb{first}%
389 \ifx\@tempa\@tempb
390 \l@dttempcntb \z@
391 \else
392 \def\@tempb{last}%
393 \ifx\@tempa\@tempb
394 \l@dttempcntb \@ne
395 \else
396 \def\@tempb{all}%
397 \ifx\@tempa\@tempb
398 \l@dttempcntb \tw@
399 \else
400 \l@dttempcntb \m@ne
401 \fi
402 \fi
403 \fi}
404

```

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

```
405 \newcount\sublock@disp
406 \newcommand{\sublockdisp}[1]{\%
407   \l@getlock@disp{#1}%
408   \ifnum\@l@tempcntb>\m@ne
409     \global\sublock@disp=\@l@tempcntb
410   \else
411     \led@warn@BadSublockdisp
412   \fi}}
413
```

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

`\linenumr@p` 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 set-
`\sublinenumr@p` ting the number representation (`\linenumrep` and `\sublinenumrep`) for line and
 sub-line numbers.

```
414 \newcommand*{\linenumberstyle}[1]{\%
415   \def\linenumrep##1{\@nameuse{##1}}
416 \newcommand*{\sublinenumberstyle}[1]{\%
417   \def\sublinenumrep##1{\@nameuse{##1}}}
```

Initialise the number styles to arabic.

```
418 \linenumberstyle{arabic}
419 \let\linenumr@p\linenumrep
420 \sublinenumberstyle{arabic}
421 \let\sublinenumr@p\sublinenumrep
422
```

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

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

`\ledlinenum` typesets the line (and subline) number.

The original `\numlabfont` specification is equivalent to the L^AT_EX `\scriptsize` for a 10pt document.

```
423 \newlength{\linenumsep}
```

```

424 \setlength{\linenumsep}{1pc}
425 \newcommand*{\numlabfont}{\normalfont\scriptsize}
426 \newcommand*{\ledlinenum}{%
427   \numlabfont\linenumrep{\line@num}%
428   \ifsublines@
429     \ifnum\subline@num>0\relax
430       \unskip\fullstop\sublinenumrep{\subline@num}%
431     \fi
432   \fi}
433 \newcommand*{\leftlinenum}{%
434   \ledlinenum
435   \kern\linenumsep}
436 \newcommand*{\rightlinenum}{%
437   \kern\linenumsep
438   \ledlinenum}
439

```

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.

```

440 \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`.

```

441 \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`.

```

442 \newtoks\led@toksa \newtoks\led@toksb
443 \global\led@toksa={}
444 \long\def\xright@appenditem#1\to#2{%
445   \global\led@toksb=\expandafter{#2}%
446   \xdef#2{\the\led@toksb\the\led@toksa\expandafter{#1}}%
447   \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`.

```
448 \long\def\xleft@appenditem#1\to#2{%
449   \global\led@toksb=\expandafter{#2}%
450   \xdef#2{\the\led@toksa\expandafter{#1}\the\led@toksb}%
451   \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.

```
452 \def\gl@p#1\to#2{\expandafter\gl@poff#1\gl@poff#1#2}
453 \long\def\gl@poff\#1#2\gl@poff#3#4{\gdef#4{#1}\gdef#3{#2}}
454
```

21.3 Line-number counters and lists

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

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

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

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

```
455 \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.

```
456 \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

`\sublines@false`

locking in odd ways: several pieces of a logical line might be interrupted by pieces of sub-lineated text, and those sub-line numbers should not return to zero until the next change in the major line number. This is common in the typesetting of English Renaissance verse drama, in which stage directions are given sub-line numbers: a single line of verse may be interrupted by several stage directions.

457 `\newif\ifsublines@`

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

458 `\newcount\absline@num`

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

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

459 `\newcount\@lock`

460 `\newcount\sub@lock`

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

`\insertlines@list`
`\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|0T1/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:

```

461 \list@create{\line@list}
462 \list@create{\insertlines@list}
463 \list@create{\actionlines@list}
464 \list@create{\actions@list}
465
\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
466 \newcount\page@num
467 \newcount\endpage@num
468 \newcount\endline@num
469 \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.
470 \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@
471 \newcommand*{\resetprevline@}{%
472   \def\do##1{\global\csundef{prevline##1}}%
473   \dolistloop{\@series}%
474 }
```

`\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@
475 \newcommand*{\resetprevpage@num}{%
476   \def\do##1{\ifcsdef{prevpage##1@num}{\global\csname prevpage##1@num\endcsname=0}{}}%
477   \dolistloop{\@series}%
478 }
```

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.

```
479 \newread\@inputcheck
480 \newcommand*{\read@linelist}[1]{%
481   \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 `\resumenumbers`, those things should still have the values they had when `\pausenumbers` was executed.

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

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

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

```
483 \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.

```
484 \global\page@num=\m@ne
485 \ifx\actionlines@list\empty
486   \gdef\next@actionline{1000000}%
487 \else
488   \glp\actionlines@list\to\next@actionline
489   \glp\actions@list\to\next@action
490 \fi}
491
```

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

```
492 \newcommand*{\list@clearing@reg}{%
493   \list@clear{\line@list}%
494   \list@clear{\insertlines@list}%
495   \list@clear{\actionlines@list}%
496   \list@clear{\actions@list}%
497   \list@clear{\sw@list}%
498   \list@clear{\sw@list@inedtext}%
499 }
```

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

```
500 \newcommand*{\get@linelistfile}[1]{%
501   \InputIfFileExists{#1}{%
502     \global\noteschanged@false
503     \begingroup
504       \catcode'\[=1 \catcode'\]=2
505       \makeatletter \catcode'\^M=9}%
506     \led@warn@NoLineFile{#1}%
507     \global\noteschanged@true
508     \begingroup}%
509 }
510
```

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 `\pausenumbers` and `\resumenumbers` 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.

```

511 \newcommand{\@nl}[2]{%
512   \fix@page{#1}%
513   \@nl@reg}
514 \newcommand*{\@nl@reg}{%
515   \ifx\l@dchset@num\relax \else
516     \advance\absline@num \@ne
517     \set@line@action
518     \let\l@dchset@num=\relax
519     \advance\absline@num \m@ne
520     \advance\line@num \m@ne
521   \fi

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

522   \advance\absline@num \@ne
523     \ifx\next@page@num\relax \else
524       \page@action
525       \let\next@page@num=\relax
526     \fi
527     \ifx\sub@change\relax \else
```

```

528         \ifnum\sub@change>\z@
529             \sublines@true
530         \else
531             \sublines@false
532         \fi
533         \sub@action
534         \let\sub@change=\relax
535     \fi

```

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

```

536         \ifcase\@lock
537             \or
538                 \@lock \tw@
539             \or \or
540                 \@lock \z@
541         \fi
542         \ifcase\sub@lock
543             \or
544                 \sub@lock \tw@
545             \or \or
546                 \sub@lock \z@
547         \fi

```

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

```

548         \ifsublines@
549             \ifnum\sub@lock<\tw@
550                 \advance\subline@num \@ne
551             \fi
552         \else
553             \ifnum\@lock<\tw@
554                 \advance\line@num \@ne \subline@num \z@
555             \fi
556         \fi}
557

```

`\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`.

```

558 \newcount\last@page@num
559 \last@page@num=-10000
560 \newcommand*{\fix@page}[1]{%
561     \ifnum #1=\last@page@num
562     \else
563         \ifbypage@
564             \line@num=\z@ \subline@num=\z@
565         \fi
566         \page@num=#1\relax
567         \last@page@num=#1\relax
568         \def\next@page@num{#1}%
569         \listxadd{\normal@page@break}{\the\absline@num}

```

```

570 \fi}
571

\@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

572 \newcommand*{\@pend}[1]{}
573 \newcommand*{\@pendR}[1]{}
574 \newcommand*{\@lopL}[1]{}
575 \newcommand*{\@lopR}[1]{}
576

\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 \@n1 of the necessary action.

577 \newcommand*{\sub@on}{\ifsublines@
578     \let\sub@change=\relax
579 \else
580     \def\sub@change{1}%
581 \fi}
582 \newcommand*{\sub@off}{\ifsublines@
583     \def\sub@change{-1}%
584 \else
585     \let\sub@change=\relax
586 \fi}
587

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

588 \newcommand*{\@adv}[1]{\ifsublines@
589     \advance\subline@num by #1\relax
590     \ifnum\subline@num<\z@
591         \led@warn@BadAdvancelineSubline
592         \subline@num \z@
593     \fi
594 \else
595     \advance\line@num by #1\relax
596     \ifnum\line@num<\z@
597         \led@warn@BadAdvancelineLine
598         \line@num \z@
599     \fi
600 \fi
601 \set@line@action}
602

\@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.

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

```

```

604     \subline@num=#1\relax
605   \else
606     \line@num=#1\relax
607   \fi
608   \set@line@action}
609

```

`\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.

```

610 \newcommand*{\l@d@set}[1]{%
611   \line@num=#1\relax
612   \advance\line@num \@ne
613   \def\l@dchset@num{#1}}
614 \let\l@dchset@num\relax
615

```

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

```

616 \newcommand*{\page@action}{%
617   \xright@appenditem{\the\absline@num}\to\actionlines@list
618   \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.

```

619 \newcommand*{\set@line@action}{%
620   \xright@appenditem{\the\absline@num}\to\actionlines@list
621   \ifsublines@
622     \@l@tempcnta=-\subline@num
623   \else
624     \@l@tempcnta=-\line@num
625   \fi
626   \advance\@l@tempcnta by -5000
627   \xright@appenditem{\the\@l@tempcnta}\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.

```

628 \newcommand*{\sub@action}{%
629   \xright@appenditem{\the\absline@num}\to\actionlines@list
630   \ifsublines@
631     \xright@appenditem{-1001}\to\actions@list
632   \else
633     \xright@appenditem{-1002}\to\actions@list
634   \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.

```

635 \newcommand*{\lock@on}{\futurelet\next\do@lockon}
636 \newcommand*{\do@lockon}{%
637   \ifx\next\lock@off
638     \global\let\lock@off=\skip@lockoff
639   \else
640     \do@lockonL
641   \fi}
642 \newcommand*{\do@lockonL}{%
643   \xright@appenditem{\the\absline@num}\to\actionlines@list
644   \ifsublines@
645     \xright@appenditem{-1005}\to\actions@list
646     \ifnum\sub@lock=\z@
647       \sub@lock \@ne
648     \else
649       \ifnum\sub@lock=\thr@@
650         \sub@lock \@ne
651       \fi
652     \fi
653   \else
654     \xright@appenditem{-1003}\to\actions@list
655     \ifnum\@lock=\z@
656       \@lock \@ne
657     \else
658       \ifnum\@lock=\thr@@
659         \@lock \@ne
660       \fi
661     \fi
662   \fi}
663
```

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

```

\do@lockoff 664 \newcommand*{\do@lockoffL}{%
\do@lockoffL 665   \xright@appenditem{\the\absline@num}\to\actionlines@list
\skip@lockoff 666   \ifsublines@
667     \xright@appenditem{-1006}\to\actions@list
668     \ifnum\sub@lock=\tw@
669       \sub@lock \thr@@
670     \else
671       \sub@lock \z@
672     \fi
673   \else
674     \xright@appenditem{-1004}\to\actions@list
675     \ifnum\@lock=\tw@
676       \@lock \thr@@
677     \else
```

```

678      \@lock \z@
679      \fi
680    \fi}
681 \newcommand*{\do@lockoff}{\do@lockoffL}
682 \newcommand*{\skip@lockoff}{\global\let\lock@off=\do@lockoff}
683 \global\let\lock@off=\do@lockoff
684

```

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

```

685 \newcommand*{\n@num}{\n@num@reg}
686 \newcommand*{\n@num@reg}{%
687   \xright@appenditem{\the\absline@num}\to\actionlines@list
688   \xright@appenditem{-1007}\to\actions@list}
689

```

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

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

```

690      \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.

```

691 \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.

```

692 \newcommand*{\@ref}[2]{%
693   \@ref@reg{#1}{#2}}
694 \newcommand*{\@ref@reg}[2]{%
695   \global\insert@count=#1\relax
696   \loop\ifnum\insert@count>\z@
697     \xright@appenditem{\the\absline@num}\to\insertlines@list
698     \global\advance\insert@count \m@ne
699   \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.


```

700 \begingroup
701   \let\@ref=\dummy@ref
702   \let\@lopL\@gobble
703   \let\page@action=\relax
704   \let\sub@action=\relax
705   \let\set@line@action=\relax
706   \let\@lab=\relax
707   \let\@sw\@gobbletwo%
708   #2
709   \global\endpage@num=\page@num
710   \global\endline@num=\line@num
711   \global\endsubline@num=\subline@num
712 \endgroup

```

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

```

713   \xright@appenditem%
714     {\the\page@num|\the\line@num|%
715     \ifsublines@ \the\subline@num \else 0\fi}%
716     \the\endpage@num|\the\endline@num|%
717     \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.

```

718   #2}
719

```

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`.

```
720 \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`.

```
721 \newif\iffirst@linenum@out@
```

```
722 \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.

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

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

```
724 \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.

```
725 \iffirst@linenum@out@
```

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

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

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

```
729 \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.

```
730 \if@minipage%
```

```
731 \leavevmode%
```

```
732 \fi%
```

```
733 \closeout\linenum@out%
```

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

```
735 \fi}
```

```
736
```

`\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.

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

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

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

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

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

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

```
743 }%
```

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

```
745 }%
```

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

```
747 }%
```

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

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

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

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

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

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

```
754 }%
```

```

755      {\write\linenum@out{\string\n@l[\the\c@page][\thepage]}}%
756    }%
757    {\write\linenum@out{\string\@nl[\the\c@page][\thepage]}}%
758  }%
759  {}%
760  \IfStrEqCase{\led@pb@setting}{{before}{\relax}{after}{\relax}}{\write\linenum@out{\string\@nl[\the\c@page][\thepage]}}%
761 }
762

```

\if@noneed@Footnote **\if@noneed@Footnote** is a boolean to check if we have to print a error message when a **\edtext** is called without any footnotes.

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

```

763 \newif\if@noneed@Footnote%
764
765 \newcommand*{\flag@start}{%
766   \ifledRcol%
767     \edef\next{\write\linenum@outR{%
768       \string\@ref[\the\insert@countR] []}%
769     \next%
770     \ifnum\insert@countR<1%
771       \if@noneed@Footnote\else%
772         \led@err@EdtextWithoutFootnote%
773       \fi%
774     \fi%
775   \else%
776     \edef\next{\write\linenum@out{%
777       \string\@ref[\the\insert@count] []}%
778     \next%
779     \ifnum\insert@count<1%
780       \if@noneed@Footnote\else%
781         \led@err@EdtextWithoutFootnote%
782       \fi%
783     \fi%
784   \fi}%
785

```

\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.

```
786 \newcommand*{\page@start}{\fi}
787
```

`\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.

```
788 \newcommand*{\startsub}{\dimen0\lastskip
789   \ifdim\dimen0>0pt \unskip \fi
790   \write\linenum@out{\string\sub@on}%
791   \ifdim\dimen0>0pt \hskip\dimen0 \fi}
792 \def\endsub{\dimen0\lastskip
793   \ifdim\dimen0>0pt \unskip \fi
794   \write\linenum@out{\string\sub@off}%
795   \ifdim\dimen0>0pt \hskip\dimen0 \fi}
796
```

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

```
797 \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.

```
798 \newcommand*{\setline}[1]{%
799   \ifnum#1<\z@
800     \led@warn@BadSetline
801   \else
802     \write\linenum@out{\string\@set[#1]}%
803   \fi}
804
```

`\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.

```
805 \newcommand*{\setlinenum}[1]{%
806   \ifnum#1<\z@
807     \led@warn@BadSetlinenum
808   \else
809     \write\linenum@out{\string\l@d@set[#1]}%
810   \fi}
811
```

```

\startlock  You can use \startlock or \endlock in running text to start or end line number
\endlock    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.

812 \newcommand*\startlock{\write\linenum@out{\string\lock@on}}
813 \def\endlock{\write\linenum@out{\string\lock@off}}
814

\ifl@dskipnumber  In numbered text \skipnumbering will suspend the numbering for that particular
\l@dskipnumbertrue line.
\l@dskipnumberfalse
\skipnumbering 815 \newif\ifl@dskipnumber
\skipnumbering@reg 816 \l@dskipnumberfalse
817 \newcommand*\skipnumbering{\skipnumbering@reg}
818 \newcommand*\skipnumbering@reg{%
819   \write\linenum@out{\string\n@num}%
820   \advanceline{-1}}
821

```

22 Marking text for notes

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
823 \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.

```
824 \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`.

```
825 \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 `\edtext` itself, if it appears within its own argument, do nothing but copy its first argument.

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

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

```

826 \newcommand*{\no@expands}{%
827   \let\select@lemfont=0%
828   \let\startsub=\relax \let\endsub=\relax
829   \let\startlock=\relax \let\endlock=\relax
830   \let\edlabel=\@gobble
831   \let\setline=\@gobble \let\advanceline=\@gobble
832   \let\critext=\dummy@text
833   \let\sameword\sameword@inedtext%
834   \let\edtext=\dummy@edtext
835   \l@dtabnoexpands
836   \morenoexpands}
837 \let\morenoexpands=\relax
838
```

`\@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.

```

839 \newcommand{\@tag}{}
840 % \end{macrocode}
841 % \end{macro}
842 % \begin{macro}{\if@edtext@}
```



```

843 % \changes{v1.15.0}{2015/01/12}{New boolean \cs{if@edtext@}.}
844 % This boolean is set to TRUE inside a \cs{edtext} (or \cs{critext}).
845 % That is useful for some commands which can have a different behavior if called inside or outside of
846 % \begin{macrocode}
847 \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.

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

```

848 \long\def\critext#1#2/{\edtext{#1}{#2}}%

```

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

```

849 \newcommand{\edtext}[2]{\leavevmode%

```

Then, check if we are in a numbered paragraph (`\pstart... \pend`).

```

850 \ifnumberedpar%

```

We switch the `\@edtext@` flag, to let know to other commands that we are in a `\edtext`.

```

851 \set@edtexttrue%

```

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

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

```

852 \begingroup%
853 \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`.

```

854 \set@line%

```

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

```

855 \ifledRcol \global\insert@countR \z@%
856 \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.

```
857 \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.

```
858 \ifundefined{xpg@main@language}{%if not polyglossia
859 \flag@start}%
860 {\if@RTL\flag@end\else\flag@start\fi% With polyglossia, you must track whether
861 }%
862 \endgroup%
863 \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.

```
864 \ifx\end@lemmas\empty \else%
865 \glp\end@lemmas\to\x@lemma%
866 \x@lemma%
867 \global\let\x@lemma=\relax%
868 \fi%
869 \ifundefined{xpg@main@language}{%if not polyglossia
870 \flag@end}%
871 {\if@RTL\flag@start\else\flag@end\fi% With polyglossia, you must track whether
872 }%
```

We switch to false some flag.

- The one that checks having footnotes inside a `\edtext`.
- The one that says we are inside a `\edtext`.
- The one that says we are inside a `\@lemma`.

```
873 \global\@noneed@Footnotefalse%
874 \@edtext@false%
875 \global\@lemmacommand@false%
```

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

```
876 \else%
877 \showlemma{#1} (\textbf{\textsc{Edtext outside numbered paragraph}})\led@err@edtextouts.
878 \fi%
879 }%
880
881 \newcommand*{\flag@end}{%
882 \ifledRcol%
```

```

883 \write\linenum@outR{}}%
884 \else%
885 \write\linenum@out{}}%
886 \fi}%
887

```

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

```

888 \newif\ifnumberline
889 \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.

```

890 \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.

```

891 \ifx\line@list\empty
892 \global\noteschanged>true
893 \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
894 \else
895 \gl@p\line@list\to\@tempb
896 \xdef\l@d@nums{\@tempb|\edfont@info}%
897 \global\let\@tempb=\undefined
898 \fi}
899

```

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

```

900 \newcommand*{\edfont@info}{\f@encoding/\f@family/\f@series/\f@shape}
901

```

22.2 Substitute lemma

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

```

902 \newcommand*{\lemma}[1]{%
903 \global\@lemmacommand>true%
904 \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.

```

905 \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.

```
906 \newcommand*{\linenum}[1]{%
907   \xdef\@tempa{#1|\\|\\|\\|\\|\\|\\|\\noexpand\\l@d@nums}%
908   \global\let\l@d@nums=\empty
909   \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.

```
910 \def\line@set#1|#2|\\#3|#4|\\{%
911   \gdef\@tempb{#1}%
912   \ifx\@tempb\empty
913     \l@d@add{#3}%
914   \else
915     \l@d@add{#1}%
916   \fi
917   \gdef\@tempb{#4}%
918   \ifx\@tempb\empty\else
919     \l@d@add{|}\line@set#2|\\#4|\\%
920   \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`.

```
921 \newcommand{\l@d@add}[1]{\xdef\l@d@nums{\l@d@nums#1}}
922
```

22.4 Lemma disambiguation

The mechanism which counts the occurrence of a same word in a same line is quite complex, because, when \LaTeX 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 923 \list@create{\sw@list}%
                  924 \list@create{\sw@list@inedtext}%

\sameword   The high level macro \sameword, used by the editor.
925 \newcommandx{\sameword}[2][1,usedefault]{\leavevmode%
First, increament the counter corresponding to the argument.
926     \unless\ifledRcol%
927     \csnumgdef{sw@#2}{\csuse{sw@#2}+1}%
Then, write its value to the numbered file
928     \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.
929     \unless\ifx\sw@list\empty%
930     \gl@p\sw@list\to\@tempb%
931     \if@edtext%
932     \unless\if@lemmacommand%
```

```

933      \xright@appenditem{\@tempb}\to\sw@list@inedtext%
934      \else%
935      \ifstrequal{#1}{inlemma}{\xright@appenditem{\@tempb}\to\sw@list@inedtext}{}%
936      \fi%
937      \fi%
938      \global\let\@tempb=\undefined%
939      \fi%
940 % \end{macrocode}
941 % Do the same thing if we are in the right columns.
942 % \begin{macrocode}
943 \else%
944   \csnumgdef{sw@#2@R}{\csuse{sw@#2@R}+1}%
945   \protected@write\linenum@outR{}{\string\@sw{#2}{\csuse{sw@#2@R}}}%
946   \unless\ifx\sw@listR\empty%
947     \glp\sw@listR\to\@tempb%
948     \if@edtext@%
949       \unless\if@lemmacommand@%
950         \xright@appenditem{\@tempb}\to\sw@list@inedtextR%
951       \else%
952         \ifstrequal{#1}{inlemma}{\xright@appenditem{\@tempb}\to\sw@list@inedtextR}{}%
953       \fi%
954     \fi%
955     \global\let\@tempb=\undefined%
956   \fi%
957 \fi%
958 % \end{macrocode}
959 % In any case, print the word.
960 % \begin{macrocode}
961 #2%
962 }%

```

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

```

963 \newcommand{\@sw}[2]{%
964   \unless\ifledRcol%

```

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

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

```

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

```

966   \numdef{\prev@line}{\the\absline@num-1}%
967   \ifcsundef{sw@#1@prev@line @the\section@num}{%
968     \csnumgdef{sw@#1@prev@line @the\section@num}{#2-1}%
969   }{}%

```

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

```

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

```

And do the same thing for the right side.

```
972 \else%

```

```

973 \csxdef{sw@#1@the\absline@numR @the\section@numR @R}{#2}%
974 \numdef{\prev@line}{\the\absline@numR-1}%
975 \ifcsundef{sw@#1@prev@line @the\section@numR @R}{%
976 \csnumgdef{sw@#1@prev@line @the\section@numR @R}{#2-1}%
977 }{}%
978 \numdef{\the@sw}{#2-\csuse{sw@#1@prev@line @the\section@numR @R}}%
979 \xright@appenditem{\the@sw}\to\sw@listR%
980 \fi%
981 }%

```

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

```

982 \def\sameword@inedtext#1{%
983 \unless\ifledRcol@%

```

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

```

984 \ifcsdef{sw@#1@the\absline@num @the\section@num}{%
985 \numdef{\prev@line}{\the\absline@num-1}%
986 \numdef{\sw@atthisline}{\csuse{sw@#1@the\absline@num @the\section@num}-\csuse{sw@#1@prev@line @the\section@num}}%
987 }%
988 {\numdef{\sw@atthisline}{0}}%

```

Just a precaution.

```

989 \ifx\sw@list@inedtext\empty%
990 \def\the@sw{999}%
991 \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.

```

992 \gl@p\sw@list@inedtext\to\the@sw%
993 \fi%

```

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

```

994 \ifnumgreater{\sw@atthisline}{1}%
995 {\showwordrank{#1}{\the@sw}}%
996 {#1}%

```

And the same for right side.

```

997 \else%
998 \ifcsdef{sw@#1@the\absline@numR @the\section@numR @R}{%
999 \numdef{\prev@line}{\the\absline@numR-1}%
1000 \numdef{\sw@atthisline}{\csuse{sw@#1@the\absline@numR @the\section@numR @R}-\csuse{sw@#1@prev@line @the\section@numR @R}}%
1001 }%
1002 {\numdef{\sw@atthisline}{0}}%
1003 \ifx\sw@list@inedtextR\empty%
1004 \def\the@sw{999}%
1005 \else%
1006 \gl@p\sw@list@inedtextR\to\the@sw%
1007 \fi%
1008 \ifnumgreater{\sw@atthisline}{1}%
1009 {\showwordrank{#1}{\the@sw}}%

```

```

1010      {#1}%
1011    \fi%
1012 }%

\showwordrank

1013 % Finally, the way the rank will be printed.
1014 \newcommand{\showwordrank}[2]{%
1015   #1\textsuperscript{#2}%
1016 }%

```

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.

1017 \newbox\raw@text
1018 \newif\ifnumberedpar@
1019 \newcount\num@lines
1020 \newbox\one@line
1021 \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 Beware: everything that occurs between \pstart and \pend is happening
\thepstart within a group; definitions must be global if you want them to survive past the
end of the paragraph.

1022
1023 \newcommand{\AtEveryPstart}[1]{\xdef\at@every@pstart{\noindent\unexpanded{#1}}}%
1024 \xdef\at@every@pstart{}%

```



```

1025
1026 \newcounter{pstart}
1027 \renewcommand{\thepstart}{\bfseries\@arabic{c@pstart}. }
1028 \newif\ifnumberpstart
1029 \numberpstartfalse
1030 \newif\iflabelpstart
1031 \labelpstartfalse
1032 \newcommandx*{\pstart}[1][1]{%
1033   \ifstrepty{#1}{\at@every@pstart}{\noindent#1}%
1034   \ifluatex%
1035     \edef\l@luatexttextdir@L{\the\luatexttextdir}%
1036   \fi%
1037   \if@nobreak%
1038     \let\@oldnobreak\@nobreaktrue%
1039   \else%
1040     \let\@oldnobreak\@nobreakfalse%
1041   \fi%
1042   \@nobreaktrue%
1043   \ifnumbering \else%
1044     \led@err@PstartNotNumbered%
1045     \beginnumbering%
1046   \fi%
1047   \ifnumberedpar@%
1048     \led@err@PstartInPstart%
1049   \pend%
1050 \fi%
1051 \list@clear{\inserts@list}%
1052 \global\let\next@insert=\empty%
1053 \begingroup\normal@pars%
1054 \global\advance \l@dnumpstartsL\@ne
1055 \global\setbox\raw@text=\vbox\bgroup%
1056   \ifautopar\else%
1057     \ifnumberpstart%
1058       \ifinstanza\else%
1059         \ifsidepstartnum\else%
1060           \thepstart%
1061         \fi%
1062       \fi%
1063     \fi%
1064   \fi%
1065   \numberedpar@true%
1066   \iflabelpstart\protected@edef\@currentlabel%
1067     {\p@pstart\thepstart}%
1068   \fi%
1069   \l@dzeropenalties%
1070 }

```

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

```

1071 \newcommandx*{\pend}[1][1]{\ifnumbering \else%
1072   \led@err@PendNotNumbered%

```

```

1073 \fi%
1074 \ifnumberedpar@ \else%
1075     \led@err@PendNoPstart%
1076 \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.

```

1077 \l@dzeropenalties%
1078 \endgraf\global\num@lines=\prevgraf\egroup%
1079 \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`.

```

1080 \csnumdef{pstartline}{0}%
1081 \loop\ifvbox\raw@text%
1082     \csnumdef{pstartline}{\pstartline+1}%
1083     \do@line%
1084     \ifbypstart@%
1085         \ifnumequal{pstartline}{1}{\setline{1}\resetprevline@}{}%
1086     \fi%
1087 \repeat%

```

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

```

1088 \flush@notes%
1089 \endgroup%
1090 \ignorespaces%
1091 \ifnumberpstart%
1092 \pstartnumtrue%
1093 \fi%
1094 \@oldnobreak%
1095 \addtocounter{pstart}{1}%
1096 \ifstrempy{#1}{\at@every@pend}{\noindent#1}%
1097 }
1098

```

```

\AtEveryPend
\at@every@pend

```

```

1100 \newcommand{\AtEveryPend}[1]{\xdef\at@every@pend{\noindent\unexpanded{#1}}}%
1101 \xdef\at@every@pend{}%
1102

```

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

```

1103 \newcommand*{\l@dzeropenalties}{%

```

```

1104 \brokenpenalty \z@ \clubpenalty \z@
1105 \displaywidowpenalty \z@ \interlinepenalty \z@ \predisplaypenalty \z@
1106 \postdisplaypenalty \z@ \widowpenalty \z@}
1107

```

`\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.

```

1108 \newif\ifautopar
1109 \autoparfalse
1110 \newcommand*{\autopar}{
1111   \ifledRcol
1112     \ifnumberingR \else
1113       \led@err@AutoparNotNumbered
1114       \beginnumberingR
1115       \fi
1116     \else
1117       \ifnumbering \else
1118         \led@err@AutoparNotNumbered
1119         \beginnumbering
1120         \fi
1121       \fi
1122     \autopartrue
1123     \everypar{\setbox0=\lastbox
1124       \endgraf \vskip-\parskip
1125       \pstart \noindent \kern\wd0 \ifnumberpstart\ifinstanza\else\thepstart\fi\fi
1126       \let\par=\pend}%
1127     \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.

```
1128 \newcommand*{\normal@pars}{\everypar{}\let\par\endgraf}
1129
```

`\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.

```
1130 \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.

```
\l@dunhbox@line
1131 \newcommand*{\l@dunhbox@line}[1]{\unhbox #1}
1132 \newcommand*{\do@line}{%
1133   {\vbadness=10000
1134     \splittopskip=\z@
1135     \do@linehook
1136 \l@emptyd@ta
1137   \global\setbox\one@line=\vsplit\raw@text to\baselineskip}%
1138   \unvbox\one@line \global\setbox\one@line=\lastbox
1139   \getline@num
1140   \IfStrEq{\led@pb@setting}{before}{\led@check@pb\led@check@nopb}{\}
1141   \ifnum\@lock>\@ne
1142     \inserthangingsymboltrue
1143   \else
1144     \inserthangingsymbolfalse
1145   \fi
1146   \check@pb@in@verse
1147   \affixline@num

Depending wether a sectioning command is called at this pstart or not we print
sectioning command or normal line,
1148   \xifinlist{\the\l@dnumpstartsL}{\eled@sections@@}%
1149     {\print@eledsection}%
1150     {\print@line}%
1151   \IfStrEq{\led@pb@setting}{after}{\led@check@pb\led@check@nopb}{\}
1152   }%
```

`\print@line` `\print@line` is for normal line, i. e. line without sectioning command.

```
1153 \def\print@line{
```

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

```
1154   \affixpstart@num%
```

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

```
1155   \hb@xt@ \linewidth{%
```

User hook.

```
1156 \do@insidelinehook%
```

Left line number

```
1157 \l@dld@ta%
```

Restore marginal and footnotes.

```
1158 \add@inserts\affixside@note%
```

Print left notes.

```
1159 \l@dlsn@te
```

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

```
1160 {\ledllfill\hb@xt@ \wd\one@line{\new@line%
```

If we use Lua^LA^TE^X then restore the direction.

```
1161 \ifluatex%
```

```
1162 \luatextextdir\l@luatextextdir@L%
```

```
1163 \fi%
```

Insert, if needed, the hanging symbol.

```
1164 \inserthangingsymbol %Space kept for backward compatibility
```

And so, print the line.

```
1165 \l@dunhbox@line{\one@line}}%
```

Right line number

```
1166 \ledrlfill\l@drd@ta%
```

Print right notes.

```
1167 \l@drsn@te
```

```
1168 }}%
```

And reinsert penalties (for page breaking)...

```
1169 \add@penalties%
```

```
1170 }
```

`\print@eledsection` `\print@eledsection` to print sectioning command with line number. It sets the correct spacing, depending whether a sectioning command was called at previous `\pstart`, calls the sectioning command, prints the normal line outside of the paper, to be able to have critical footnotes. Because of how this prints, a vertical spacing correction is added.

```
1171 \def\print@eledsection{%
```

```
1172 \add@inserts\affixside@note%
```

```
1173 \numdef{\temp@}{\l@dnumpstartsL-1}%
```

```
1174 \xifinlist{\temp@}{\eled@sections@@}{\@nobreaktrue}{\@nobreakfalse}%
```

```
1175 \@eled@sectioningtrue%
```

```
1176 \csuse{eled@sectioning@the\l@dnumpstartsL}%
```

```
1177 \@eled@sectioningfalse%
```

```
1178 \global\csundef{eled@sectioning@the\l@dnumpstartsL}%
```

```
1179 \if@RTL%
```

```
1180 \hspace{-3\paperwidth}%
```

```
1181 {\hbox{\l@dunhbox@line{\one@line}} \new@line}%
```

```

1182 \else%
1183 \hspace{3\paperwidth}%
1184 {\new@line \hbox{\l@dunhbox@line{\one@line}}}%
1185 \fi%
1186 \vskip-\baselineskip%
1187 }

```

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

```

1188 \newcommand*\dolinehook[1]{\gdef\do@linehook{#1}}%
1189 \newcommand*\doinsidelinehook[1]{\gdef\do@insidelinehook{#1}}%
1190

```

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

```

1191 \newcommand*\do@linehook{}
1192 \newcommand*\do@insidelinehook{}

```

`\l@emptyd@ta` Nulls the `\...d@ta`, which may later hold line numbers. Similarly for `\l@dcsnotetext`,

`\l@dld@ta` `\l@dcsnotetext@l`, `\l@dcsnotetext@r` for the texts of the sidenotes, left and

`\l@drd@ta` right notes.

```

\l@dcsnotetext 1193 \newcommand*\l@emptyd@ta{}%
\l@dcsnotetext@l 1194 \gdef\l@dld@ta{}%
\l@dcsnotetext@r 1195 \gdef\l@drd@ta{}%
1196 \gdef\l@dcsnotetext@l{}%
1197 \gdef\l@dcsnotetext@r{}%
1198 \gdef\l@dcsnotetext{}%
1199

```

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

```

\l@drsn@te 1200 \newcommand*\l@dlsn@te{}%
1201 \hb@xt@ \z@{\hss\box\l@dldp@rbox\kern\ledlsnotesep}}
1202 \newcommand*\l@drsn@te{}%
1203 \hb@xt@ \z@{\kern\ledrsnotesep\box\l@drp@rbox\hss}}
1204

```

`\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`.

```

1205 \newcommand*\ledllfill{\hfil}
1206 \newcommand*\ledrlfill{}
1207

```

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.

```

1208 \newcommand*{\getline@num}{%
1209     \global\advance\absline@num \@ne
1210     \do@actions
1211     \do@ballast
1212     \ifnumberline
1213     \ifsublines@
1214         \ifnum\sub@lock<\tw@
1215             \global\advance\subline@num \@ne
1216         \fi
1217     \else
1218         \ifnum\@lock<\tw@
1219             \global\advance\line@num \@ne
1220             \global\subline@num \z@
1221         \fi
1222     \fi
1223 \fi
1224 }
```

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

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

```

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

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

```

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

`\do@actions` The `\do@actions` macro looks at the list of actions to take at particular absolute
`\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 `\do@actions@next` that is always the last thing that `\do@actions` does. If there could be more actions to

process for this line, `\do@actions@next` is set equal to `\do@actions`; otherwise it's just `\relax`.

```
1237 \newcommand*{\do@actions}{%
1238   \global\let\do@actions@next=\relax
1239   \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.

```
1240   \ifnum\next@action>-1001
1241     \global\page@num=\next@action
1242     \ifbypage@
1243       \global\line@num=\z@ \global\subline@num=\z@
1244       \resetprevline@
1245     \fi
```

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

```
1246   \else
1247     \ifnum\next@action<-4999
1248       \@l@tempcnta=-\next@action
1249       \advance\@l@tempcnta by -5001
1250       \ifsublines@
1251         \global\subline@num=\@l@tempcnta
1252       \else
1253         \global\line@num=\@l@tempcnta
1254     \fi
```

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

```
1255   \else
1256     \@l@tempcnta=-\next@action
1257     \advance\@l@tempcnta by -1000
1258     \do@actions@fixedcode
1259   \fi
1260 \fi
```

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

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

```
1261   \ifx\actionlines@list\empty
1262     \gdef\next@actionline{1000000}%
1263   \else
1264     \glp\actionlines@list\to\next@actionline
1265     \glp\actions@list\to\next@action
1266     \global\let\do@actions@next=\do@actions
1267   \fi
1268 \fi
```


Make the recursive call, if necessary.

```
1269 \do@actions@next}
1270
```

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

```
1271 \newcommand*{\do@actions@fixedcode}{%
1272   \ifcase\l@dttempcnta
1273   \or% % 1001
1274     \global\sublines@true
1275   \or% % 1002
1276     \global\sublines@false
1277   \or% % 1003
1278     \global\@lock=\@ne
1279   \or% % 1004
1280     \ifnum\@lock=\tw@
1281       \global\@lock=\thr@@
1282     \else
1283       \global\@lock=\z@
1284     \fi
1285   \or% % 1005
1286     \global\sub@lock=\@ne
1287   \or% % 1006
1288     \ifnum\sub@lock=\tw@
1289       \global\sub@lock=\thr@@
1290     \else
1291       \global\sub@lock=\z@
1292     \fi
1293   \or% % 1007
1294     \l@dskipnumbertrue
1295   \else
1296     \led@warn@BadAction
1297   \fi}
1298
1299
```

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:

$$\begin{aligned}
 n &= \text{int}((\text{linenum} - \text{firstlinenum}) / \text{linenumincrement}) \\
 m &= \text{firstlinenum} + (n \times \text{linenumincrement})
 \end{aligned}$$

(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 $\backslash\text{line@num} \leq \backslash\text{firstlinenum}$, we compare the two directly instead of making these calculations.

We compute, in the scratch counter $\backslash\text{@l@tempcnta}$, 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 $\backslash\text{@l@tempcntb}$ for comparison.

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

```
1300 \newcommand*{\affixline@num}{%
```

No number is attached if $\backslash\text{ifl@dskipnumber}$ is TRUE (and then it is set to its normal FALSE value). No number is attached if $\backslash\text{ifnumberline}$ is FALSE (the normal value is TRUE).

```
1301 \ifl@dskipnumber\else\ifnumberline
1302 \ifl@dskipnumber
1303   \global\l@dskipnumberfalse
1304 \else
1305   \ifsublines@
1306     \l@tempcntb=\subline@num
1307     \ifnum\subline@num>\c@firstsublinenum
1308       \l@tempcnta=\subline@num
1309       \advance\l@tempcnta by-\c@firstsublinenum
1310       \divide\l@tempcnta by\c@sublinenumincrement
1311       \multiply\l@tempcnta by\c@sublinenumincrement
1312       \advance\l@tempcnta by\c@firstsublinenum
1313     \else
1314       \l@tempcnta=\c@firstsublinenum
1315     \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.

```
1316   \ch@cksub@l@ck
```

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

```
1317 \else
1318   \l@tempcntb=\line@num
1319   Check on the \linenumberlist If it's \empty use the standard algorithm.
1320   \ifnum\line@num>\c@firstlinenum
1321     \l@tempcnta=\line@num
1322     \advance\l@tempcnta by-\c@firstlinenum
1323     \divide\l@tempcnta by\c@linenumincrement
1324     \multiply\l@tempcnta by\c@linenumincrement
1325     \advance\l@tempcnta by\c@firstlinenum
1326   \else
1327     \l@tempcnta=\c@firstlinenum
```

```

1328     \fi
1329     \else
    The \linenumberlist wasn't \empty, so here's Wayne's numbering mechanism.
    This takes place in TeX's mouth.
1330     \@l@tempcnta=\line@num
1331     \edef\rem@inder{\,\linenumberlist,\number\line@num,}%
1332     \edef\sc@n@list{\def\noexpand\sc@n@list
1333         ###1,\number\@l@tempcnta,###2|{\def\noexpand\rem@inder{###2}}}%
1334     \sc@n@list\expandafter\sc@n@list\rem@inder|%
1335     \ifx\rem@inder\empty\advance\@l@tempcnta\@ne\fi
1336     \fi

```

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

```

1337     \ch@ck@l@ck
1338     \fi

```

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

```

1339     \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 1340 \if@twocolumn
1341     \if@firstcolumn
1342         \gdef\l@dld@ta{\llap{\leftlinenum}}}%
1343     \else
1344         \gdef\l@drd@ta{\rlap{\rightlinenum}}}%
1345     \fi
1346 \else

```

Continuing the original code ...

```

1347     \@l@tempcntb=\line@margin
1348     \ifnum\@l@tempcntb>\@ne
1349         \advance\@l@tempcntb \page@num
1350     \fi

```

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

```

1351     \ifodd\@l@tempcntb
1352         \gdef\l@drd@ta{\rlap{\rightlinenum}}}%
1353     \else

```

```

1354     \gdef\l@dld@ta{\llap{\leftlinenum}}}%
1355     \fi
1356 \fi
1357 \else

```

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

```

1358 %%     #1%
1359 \fi

```

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

```

1360 \f@x@l@cks
1361 \fi
1362 \fi
1363 \fi
1364 }
1365

```

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

```

1366 \newcommand*{\ch@cksub@l@ck}{%
1367     \ifcase\sub@lock
1368     \or
1369         \ifnum\sublock@disp=\@ne
1370             \l@dttempcntb=\z@ \l@dttempcnta=\@ne
1371         \fi
1372     \or
1373         \ifnum\sublock@disp=\tw@ \else
1374             \l@dttempcntb=\z@ \l@dttempcnta=\@ne
1375         \fi
1376     \or
1377         \ifnum\sublock@disp=\z@
1378             \l@dttempcntb=\z@ \l@dttempcnta=\@ne
1379         \fi
1380     \fi}

```

Similarly for line numbers.

```

1381 \newcommand*{\ch@ck@l@ck}{%
1382     \ifcase\@lock
1383     \or
1384         \ifnum\lock@disp=\@ne
1385             \l@dttempcntb=\z@ \l@dttempcnta=\@ne
1386         \fi
1387     \or
1388         \ifnum\lock@disp=\tw@ \else
1389             \l@dttempcntb=\z@ \l@dttempcnta=\@ne
1390         \fi
1391     \or
1392         \ifnum\lock@disp=\z@

```

```

1393         \@l@dttempcntb=\z@ \@l@dttempcnta=\@ne
1394     \fi
1395 \fi}

```

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

```

1396 \newcommand*{\f@x@l@cks}{%
1397 \ifcase\@lock
1398 \or
1399 \global\@lock=\tw@
1400 \or \or
1401 \global\@lock=\z@
1402 \fi
1403 \ifcase\sub@lock
1404 \or
1405 \global\sub@lock=\tw@
1406 \or \or
1407 \global\sub@lock=\z@
1408 \fi}
1409

```

`\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.

```

1410 \newcommand{\pageparbreak}{\pend\newpage\pstart\noindent}
1411

```

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 1412
\ifsidepstartnum 1413 \newif\ifsidepstartnum
1414 \newcommand*{\affixpstart@num}{%
1415 \ifsidepstartnum

```

```

1416     \if@twocolumn
1417         \if@firstcolumn
1418             \gdef\l@dld@ta{\llap{\leftstartnum}}}%
1419         \else
1420             \gdef\l@drd@ta{\rlap{\rightstartnum}}}%
1421         \fi
1422     \else
1423         \l@dttempcntb=\line@margin
1424         \ifnum\l@dttempcntb>\@ne
1425             \advance\l@dttempcntb \page@num
1426         \fi
1427         \ifodd\l@dttempcntb
1428             \gdef\l@drd@ta{\rlap{\rightstartnum}}}%
1429         \else
1430             \gdef\l@dld@ta{\llap{\leftstartnum}}}%
1431         \fi
1432     \fi
1433 \fi
1434
1435 }
1436 %
1437
1438 \newif\ifpstartnum
1439 \pstartnumtrue
1440 \newcommand*{\leftstartnum}{
1441     \ifpstartnum\thepstart
1442     \kern\linenumsep\fi
1443     \global\pstartnumfalse
1444 }
1445 \newcommand*{\rightstartnum}{
1446     \ifpstartnum
1447     \kern\linenumsep
1448     \thepstart
1449     \fi
1450     \global\pstartnumfalse
1451 }

```

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.

```
1452 \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.

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

process for this line, `\add@inserts@next` is set equal to `\add@inserts`; otherwise it's just `\relax`.

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

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

```
1455   \ifx\inserts@list\empty \else
```

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

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

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

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

Make the recursive call, if necessary.

```
1472 \add@inserts@next}
1473
```

23.7 Penalties

`\add@penalties` `\add@penalties` is the last macro used by `\do@line`. It adds up the club, widow, and interline penalties, and puts a single penalty of the appropriate size back into the paragraph; these penalties get removed by the `\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.95). Finally, the penalty is checked to see that it doesn't go below -10000 .

```

1474 \newcommand*{\add@penalties}{\@l@tempcnta=\ballast@count
1475   \ifnum\num@lines>\@ne
1476     \global\advance\par@line \@ne
1477     \ifnum\par@line=\@ne
1478       \advance\@l@tempcnta \clubpenalty
1479     \fi
1480     \@l@tempcntb=\par@line \advance\@l@tempcntb \@ne
1481     \ifnum\@l@tempcntb=\num@lines
1482       \advance\@l@tempcnta \widowpenalty
1483     \fi
1484     \ifnum\par@line<\num@lines
1485       \advance\@l@tempcnta \interlinepenalty
1486     \fi
1487   \fi
1488   \ifnum\@l@tempcnta=\z@
1489     \relax
1490   \else
1491     \ifnum\@l@tempcnta>-10000
1492       \penalty\@l@tempcnta
1493     \else
1494       \penalty -10000
1495     \fi
1496   \fi}
1497
```

23.8 Printing leftover notes

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

```

1498 \newcommand*{\flush@notes}{%
1499   \@xloop
1500   \ifx\inserts@list\empty \else
1501     \glp\inserts@list\to\@insert
1502     \@insert
1503     \global\let\@insert=\undefined
1504   \repeat}
1505
```

`\@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.

```
1506 \def\@xloop#1\repeat{%
1507   \def\body{#1\expandafter\body\fi}%
1508   \body}
1509
```

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.

```
1510 \def\select@lemmafont#1|#2|#3|#4|#5|#6|#7|{\select@@lemmafont#7|}
1511 \def\select@@lemmafont#1/#2/#3/#4|{%
1512   {\fontencoding{#1}\fontfamily{#2}\fontseries{#3}\fontshape{#4}%
1513   \selectfont}
1514
```

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.

```
1515 \newcommand*{\footnoteoptions@}[3][1=L,usedefault]{%
1516   \def\do##1{%
1517     \ifstrequal{#1}{L}{% In Leftside
1518       \xright@appenditem{\global\noexpand\settogle{##1@}{#3}}\to\inserts@list% Switch toggle, in
1519       \global\advance\insert@count \@ne% Increment the left insert counter.
```

```

1520     }%
1521     {%
1522         \xright@appenditem{\global\noexpand\settoggle{##1@}{#3}}\to\inserts@listR% Switch
1523         \global\advance\insert@countR \@ne% Increment the right insert counter insert.
1524     }%
1525 }%
1526 \notblank{#2}{\docsvlist{#2}}{ }% Parsing all options
1527 }

```

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

```

1528 \newcommand*{\footnotelang@lua}[1][1=L,usedefault]{%
1529     \ifstrequal{#1}{L}{%
1530         \xright@appenditem{\csxdef{footnote@luatextextdir}{\the\luatextextdir}}\to\inserts@listR%
1531         \global\advance\insert@count \@ne%
1532         \xright@appenditem{\csxdef{footnote@luatexpardir}{\the\luatexpardir}}\to\inserts@listR%
1533         \global\advance\insert@count \@ne%
1534     }%
1535     {%
1536         \xright@appenditem{\csxdef{footnote@luatextextdir}{\the\luatextextdir}}\to\inserts@listR%
1537         \global\advance\insert@countR \@ne%
1538         \xright@appenditem{\csxdef{footnote@luatexpardir}{\the\luatexpardir}}\to\inserts@listR%
1539         \global\advance\insert@countR \@ne%
1540     }%
1541 }

```

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

```

1542 \newcommand*{\footnotelang@poly}[1][1=L,usedefault]{%
1543     \ifstrequal{#1}{L}{%
1544         \if@RTL%
1545             \xright@appenditem{\csxdef{footnote@dir}{@RTLtrue}}\to\inserts@listR%Know the language
1546             \global\advance\insert@count \@ne%
1547         \else
1548             \xright@appenditem{\csxdef{footnote@dir}{@RTLfalse}}\to\inserts@listR%Know the language
1549             \global\advance\insert@count \@ne%
1550         \fi%
1551         \xright@appenditem{\csxdef{footnote@lang}{\expandonce\language\language}}\to\inserts@listR%
1552         \global\advance\insert@count \@ne%
1553     }%
1554     {%
1555         \if@RTL
1556             \xright@appenditem{\csxdef{footnote@dir}{@RTLtrue}}\to\inserts@listR%Know the language
1557             \global\advance\insert@countR \@ne%
1558         \else
1559             \xright@appenditem{\csxdef{footnote@dir}{@RTLfalse}}\to\inserts@listR%Know the language
1560             \global\advance\insert@countR \@ne%
1561         \fi
1562         \xright@appenditem{\csxdef{footnote@lang}{\expandonce\language\language}}\to\inserts@listR%

```

```

1563 \global\advance\insert@countR \@ne%
1564 }%
1565 }

```

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.

```

1566 \notbool{parapparatus@}{\newcommand*}{\newcommand}{\normalvfootnote}[2]{%
1567 \insert\csname #1footins\endcsname\bgroup
1568 \csuse{bhookXnote@#1}
1569 \csuse{Xnotefontsize@#1}
1570 \footsplitskips
1571 \ifl@dpairing\ifl@dpaging\else%
1572 \setXnoteswidthliketwocolumns@{#1}%
1573 \fi\fi%
1574 \setXnotespositionliketwocolumns@{#1}%
1575 \spaceskip=\z@skip \xspaceskip=\z@skip
1576 \csname #1footfmt\endcsname #2[#1]\egroup}

```

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

```

1577 \newcommand*{\footsplitskips}{%
1578 \interlinepenalty=\interfootnotelinepenalty
1579 \floatingpenalty=\@MM
1580 \splittopskip=\ht\strutbox \splitmaxdepth=\dp\strutbox
1581 \leftskip=\z@skip \rightskip=\z@skip}
1582

```

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

```

1583 \notbool{parapparatus@}{\newcommand*}{\newcommand}{\mpnormalvfootnote}[2]{%

```

```

1584 \global\setbox\@nameuse{mp#1footins}\vbox{%
1585   \unvbox\@nameuse{mp#1footins}
1586   \csuse{bhookXnote@#1}
1587   \csuse{Xnotefontsize@#1}
1588   \hsize\columnwidth
1589   \@parboxrestore
1590   \color@begingroup
1591   \csname #1footfmt\endcsname #2[#1]\color@endgroup}}
1592

```

`\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).

```

1593 \newcommand*{\ledsetnormalparstuff}{%
1594   \ifluatex%
1595     \luatextextdir\footnote@luatextextdir%
1596     \luatexpardir\footnote@luatexpardir%
1597   \fi%
1598   \csuse{\csuse{footnote@dir}}%
1599   \normal@pars%
1600   \noindent \parfillskip \z@ \@plus 1fil}
1601
1602 \notbool{parapparatus@}{\newcommandx*}{\newcommandx}{\normalfootfmt}[4][4=Z]{% 4th arg is
1603   \ledsetnormalparstuff%
1604   \hangindent=\csuse{Xhangindent@#4}
1605   \strut{\printlinefootnote{#1}{#4}}%
1606   {\nottoggle{Xlemmadisablefontselection@#4}{\select@lemmafnt#1|#2}{#2}}%
1607   \iftoggle{nosep@}{\hskip\csuse{inplaceoflemmaseparator@#4}}{\ifcsemtyp{lemmaseparator@#4}
1608     {\hskip\csuse{inplaceoflemmaseparator@#4}}%
1609     {\nobreak\hskip\csuse{beforelemmaseparator@#4}\csuse{lemmaseparator@#4}\hskip\csuse{a
1610     }}%
1611   #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.

```

1612 \def\endashchar{\textnormal{--}}
1613 \newcommand*{\fullstop}{\textnormal{.}}
1614 \newcommand*{\rbracket}{\textnormal{}}
1615 \csuse{text}\csuse{footnote@lang}}{%
1616     \ifluatex%
1617         \ifdefstring{\footnote@luatextextdir}{TRT}{\thinspace[]{\thinspace}}%
1618         \else%
1619             \thinspace}%
1620     \fi}%
1621 }%
1622 }
1623

```

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

```

1624 \newcommand{\printpstart}[0]{%
1625     \ifboolexpr{bool{!@dpairing} or bool{!@dprintingpages} or bool{!@dprintingcolumns}}{%
1626         \ifledRcol%
1627             \thepstartR%
1628         \else%
1629             \thepstartL%
1630         \fi%
1631     }{%
1632         \thepstart%
1633     }%
1634 }

```

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 1635 \newif\ifl@d@pnum
\ifl@d@elin 1636 \l@d@pnumfalse
\ifl@d@esl 1637 \newif\ifl@d@ssub
\ifl@d@dash 1638 \l@d@ssubfalse
               1639 \newif\ifl@d@elin
               1640 \l@d@elinfalse
               1641 \newif\ifl@d@esl
               1642 \l@d@eslfalse
               1643 \newif\ifl@d@dash
               1644 \l@d@dashfalse

\l@dp@rsefootsec \l@dp@rsefootsec{<spec>}{<lemma>}{<text>} parses a footnote specification.
\l@dp@rsefootsec <lemma> and <text> are the lemma and text respectively. <spec> is the line and
\l@dp@rsefootsec page number and lemma font specifier in \l@d@nums style format. The real work
\l@dp@rsefootsec is done by \l@dp@rsefootsec which defines macros holding the numeric values.
\l@dp@rsefootsec 1645 \newcommand*{\l@dp@rsefootsec}[3]{\l@dp@rsefootsec#1|}
\l@dp@rsefootsec 1646 \def\l@dp@rsefootsec#1|#2|#3|#4|#5|#6|#7|{%
\l@dp@rsefootsec 1647 \gdef\l@dp@rsefootsec#1|#2|#3|#4|#5|#6|#7|{%
\l@dp@rsefootsec 1648 \gdef\l@dp@rsefootsec#1|#2|#3|#4|#5|#6|#7|{%
               1649 \gdef\l@dp@rsefootsec#1|#2|#3|#4|#5|#6|#7|{%
               1650 \gdef\l@dp@rsefootsec#1|#2|#3|#4|#5|#6|#7|{%
               1651 \gdef\l@dp@rsefootsec#1|#2|#3|#4|#5|#6|#7|{%
               1652 \gdef\l@dp@rsefootsec#1|#2|#3|#4|#5|#6|#7|{%
               1653 }

    Initialise the several number value macros.
1654 \def\l@dp@rsefootsec#1|#2|#3|#4|#5|#6|#7|{%
1655 \def\l@dp@rsefootsec#1|#2|#3|#4|#5|#6|#7|{%
1656 \def\l@dp@rsefootsec#1|#2|#3|#4|#5|#6|#7|{%
1657 \def\l@dp@rsefootsec#1|#2|#3|#4|#5|#6|#7|{%
1658 \def\l@dp@rsefootsec#1|#2|#3|#4|#5|#6|#7|{%
1659 \def\l@dp@rsefootsec#1|#2|#3|#4|#5|#6|#7|{%
1660

\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.
1661 \newcommand*{\setprintlines}[6]{%
1662 \l@d@pnumfalse \l@d@dashfalse
1663 \ifbypage@

```

```

1664 \ifnum#4=#1 \else
1665 \l@d@pnumtrue
1666 \l@d@dashtrue
1667 \fi
1668 \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.

```

1669 \ifl@d@pnum \l@d@elintrue \else \l@d@elinfalse \fi
1670 \ifnum#2=#5 \else
1671 \l@d@elintrue
1672 \l@d@dashtrue
1673 \fi

```

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

```

1674 \l@d@ssubfalse
1675 \ifnum#3=0 \else
1676 \l@d@ssubtrue
1677 \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.

```

1678 \l@d@eslfalse
1679 \ifnum#6=0 \else
1680 \ifnum#6=#3
1681 \ifl@d@elin \l@d@esltrue \else \l@d@eslfalse \fi
1682 \else
1683 \l@d@esltrue
1684 \l@d@dashtrue
1685 \fi
1686 \fi}

```

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

```

1687 \def\printlines#1|#2|#3|#4|#5|#6|#7|{\begingroup
1688 \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).

```

1689 \ifl@d@pnum #1\fullstop\fi
1690 \linenumrep{#2}

1691 \ifl@d@ssub \fullstop \sublinenumrep{#3}\fi
1692 \ifl@d@dash \endashchar\fi
1693 \ifl@d@pnum #4\fullstop\fi
1694 \ifl@d@elin \linenumrep{#5}\fi
1695 \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumrep{#6}\fi
1696 \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`.

```

1697 \newcommand*{\normalfootstart}[1]{%
1698     \ifdimequal{0pt}{\preXnotes@}{}%
1699     {%
1700         \iftoggle{preXnotes@}{%
1701             \togglefalse{preXnotes@}\skip\csname #1footins\endcsname=\csuse{preXnotes@}}%
1702         }%
1703     }%
1704     \vskip\skip\csname #1footins\endcsname%
1705     \leftskip0pt \rightskip0pt
1706     \ifl@dpairing\else%
1707         \hsize=\old@hsize%
1708     \fi%
1709     \setXnoteswidthliketwocolumns@{#1}%
1710     \setXnotespositionliketwocolumns@{#1}%
1711     \print@Xfootnoterule{#1}%
1712     \vskip\csuse{afterXrule@#1}%
1713     \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.

```

1714 \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.

```

1715 \newcommand*{\normalfootgroup}[1]{%
1716     {\csuse{Xnotefontsize@#1}\noindent\csuse{txtbeforeXnotes@#1}}%
1717     \unvbox\csname #1footins\endcsname%
1718     \hsize=\old@hsize%
1719     }%
1720

```

`\mpnormalfootgroup` A somewhat different version for minipages.


```

1721 \newcommand*{\mpnormalfootgroup}[1]{\{
1722   \vskip\skip\@nameuse{mp#1footins}
1723   \ifl@dpairing\ifparledgroup%
1724     \leavevmode\marks\parledgroup@{begin}%
1725     \marks\parledgroup@series{#1}%
1726     \marks\parledgroup@type{Xfootnote}%
1727   \fi\fi\normalcolor%
1728   \ifparledgroup%
1729     \ifl@dpairing%
1730     \else%
1731       \setXnoteswidthliketwocolumns@{#1}%
1732       \setXnotespositionliketwocolumns@{#1}%
1733       \print@Xfootnoterule{#1}%
1734       \vskip\csuse{afterXrule@#1}%
1735     \fi%
1736   \else%
1737     \setXnoteswidthliketwocolumns@{#1}%
1738     \setXnotespositionliketwocolumns@{#1}%
1739     \print@Xfootnoterule{#1}%
1740     \vskip\csuse{afterXrule@#1}%
1741   \fi%
1742   \setlength{\parindent}{0pt}
1743   {\csuse{Xnotefontsize@#1}\csuse{txtbeforeXnotes@#1}}
1744   \unvbox\csname mp#1footins\endcsname}}
1745

```

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 `\baselineskip`, 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\vsizel
\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
1746 \newcommand*{\ledfootinsdim}{0.8\vsizex} % kept for backward compatibility, should'nt be us

\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.

1747 \newtoggle{preXnotes@}
1748 \toggletrue{preXnotes@}
1749 \newcommand{\preXnotes@}{0pt}
1750 \newcommand*{\preXnotes}[1]{\renewcommand{\preXnotes@}{#1}}
```

The same, but for familiar footnotes.

```
\preXnotes
\preXnotes@ 1751 \newtoggle{prenotesX@}
1752 \toggletrue{prenotesX@}
1753 \newcommand{\prenotesX@}{0pt}
1754 \newcommand*{\prenotesX}[1]{\renewcommand{\prenotesX@}{#1}}
```

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

```
1755 \newcommand*{\footnormal}[1]{%
1756   \csgdef{series@display#1}{normal}
1757   \expandafter\let\csname #1footstart\endcsname=\normalfootstart
1758   \expandafter\let\csname v#1footnote\endcsname=\normalvfootnote
1759   \expandafter\let\csname #1footfmt\endcsname=\normalfootfmt
1760   \expandafter\let\csname #1footgroup\endcsname=\normalfootgroup
1761   \expandafter\let\csname #1footnoterule\endcsname=%
1762                                     \normalfootnoterule
1763   \count\csname #1footins\endcsname=1000
1764   \dimen\csname #1footins\endcsname=\csuse{maxhXnotes@#1}
1765   \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).

```
1766   \expandafter\let\csname mpv#1footnote\endcsname=\mpnormalvfootnote
1767   \expandafter\let\csname mp#1footgroup\endcsname=\mpnormalfootgroup
1768   \count\csname mp#1footins\endcsname=1000
1769   \dimen\csname mp#1footins\endcsname=\csuse{maxhXnotes@#1}
1770   \skip\csname mp#1footins\endcsname=\csuse{beforeXnotes@#1}
1771 }
1772
```

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

24.5 Paraphed footnotes

The paraphed-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 TeXbook*, pp.398–400, with alterations for our environment. This algorithm uses a considerable amount of save-stack space: a TeX of ordinary size may not be able to handle more than about 100 notes of this kind on a page.

\footparagraph The `\footparagraph` macro sets up everything for one series of the footnotes so that they'll be paraphed; 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 paraphed 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 TeX will try to put too many or too few of these notes on each page. If you need to change the `\hsize` within the document, call `\footparagraph` again afterwards to take account of the new value. The argument of `\footparagraph` is the letter (A–E) denoting the series of notes to be paraphed.

```

1773 \newcommand*{\footparagraph}[1]{%
1774   \csgdef{series@display#1}{paragraph}
1775   \expandafter\newcount\csname prevpage#1@num\endcsname
1776   \expandafter\let\csname #1footstart\endcsname=\parafootstart
1777   \expandafter\let\csname v#1footnote\endcsname=\para@vfootnote
1778   \expandafter\let\csname #1footfmt\endcsname=\parafootfmt
1779   \expandafter\let\csname #1footgroup\endcsname=\para@footgroup
1780   \count\csname #1footins\endcsname=1000
1781   \dimen\csname #1footins\endcsname=\csuse{maxhXnotes@#1}
1782   \skip\csname #1footins\endcsname=\csuse{beforeXnotes@#1}
1783   \para@footsetup{#1}

```

And the extra setup for minipages.

```

1784   \expandafter\let\csname mpv#1footnote\endcsname=\mppara@vfootnote
1785   \expandafter\let\csname mp#1footgroup\endcsname=\mppara@footgroup
1786   \count\csname mp#1footins\endcsname=1000
1787   \dimen\csname mp#1footins\endcsname=\csuse{maxhXnotes@#1}
1788   \skip\csname mp#1footins\endcsname=\csuse{beforeXnotes@#1}
1789 }

```

\footfudgefiddle For paraphed 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.

```

1790 \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 L^AT_EX not `\hsize`. I've also included `\footfudgefiddle`.

```

1791 \newcommand*{\para@footsetup}[1]{\csuse{Xnotefontsize@#1}
1792   \setXnoteswidthliketwocolumns@{#1}%
1793   \dimen0=\baselineskip
1794   \multiply\dimen0 by 1024
1795   \divide \dimen0 by \columnwidth \multiply\dimen0 by \footfudgefiddle\relax
1796   \csxdef{#1footfudgefactor}{%
1797     \expandafter\strip@pt\dimen0 }}
1798
```

EDMAC defines `\en@number` which does the same as the L^AT_EX 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.

```

1799 \newcommand*{\parafootstart}[1]{%
1800   \rightskip=0pt \leftskip=0pt \parindent=0pt
1801   \ifdimequal{0pt}{\preXnotes@}{}%
1802   {%
1803     \iftoggle{preXnotes@}{%
1804       \togglefalse{preXnotes@}\skip\csname #1footins\endcsname=\csuse{preXnotes@}}%
1805     }%
1806   }%
1807   \vskip\skip\csname #1footins\endcsname%
1808   \setXnoteswidthliketwocolumns@{#1}%
1809   \setXnotespositionliketwocolumns@{#1}%
1810   \print@Xfootnoterule{#1}%
1811   \vskip\csuse{afterXrule@#1}%
1812   \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.

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. 112 above). We need to do this, since \footfudgefactor is calculated on the assumption that the notes are \hsize wide.

So, finally, here is the modified foot-paragraph code, which sets the footnote in vertical mode so that language and discretionary nodes are included.

```
1813 \newcommand*{\para@vfootnote}[2]{%
1814   \insert\csname #1footins\endcsname
1815   \bgroup
1816   \csuse{bhookXnote@#1}
```

²³Michael Downes, 'Line Breaking in \unboxed 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.

²⁶'Line Breaking', p. 610.

```

1817 \csuse{Xnotefontsize@#1}
1818 \footsplitskips
1819 \setbox0=\vbox{\hsize=\maxdimen
1820 \noindent\csname #1footfmt\endcsname#2[#1]}%
1821 \setbox0=\hbox{\unvxh0[#1]}%
1822 \dp0=0pt
1823 \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.

```

1824 \if@RTL\noindent \leavevmode\fi\box0%
1825 \penalty0
1826 \egroup}
1827

```

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 \TeX book*, 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 `\makehboxofhboxes`. So it does not affect how the footnote paragraphs are typeset (the notes still have a penalty of -10 between them, which is added by `\parafootfmt`).

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

```

1828 \newcommand*{\mppara@vfootnote}[2]{%
1829 \global\setbox\@nameuse{mp#1footins}\vbox{%
1830 \unvbox\@nameuse{mp#1footins}%
1831 \csuse{bhookXnote@#1}
1832 \csuse{Xnotefontsize@#1}
1833 \footsplitskips
1834 \setbox0=\vbox{\hsize=\maxdimen
1835 \noindent\color@begingroup\csname #1footfmt\endcsname #2[#1]\color@endgroup}%
1836 \setbox0=\hbox{\unvxh0[#1]}%
1837 \dp0=\z@
1838 \ht0=\csname #1footfudgefactor\endcsname\wd0
1839 \box0
1840 \penalty0
1841 }}
1842

```

`\unvxh` Here is (modified) Michael’s definition of `\unvxh`, used above. Michael’s macro also takes care to remove some unwanted penalties and glue that \TeX automatically attaches to the end of paragraphs. When \TeX finishes a paragraph, it throws away any remaining glue, and then tacks on the following items: a `\penalty` of 10000,

a `\parfillskip` and a `\rightskip` (*The TeXbook*, pp. 99–100). `\unvvh` cancels these unwanted paragraph-final items using `\unskip` and `\unpenalty`.

```

1843 \newcommand*{\unvvh}[2][2=Z]{% 2th is optional for retro-compatibility
1844   \setbox0=\vbox{\unvvhbox#1%
1845     \global\setbox1=\lastbox}%
1846   \unhbox1
1847   \unskip           % remove \rightskip,
1848   \unskip           % remove \parfillskip,
1849   \unpenalty        % remove \penalty of 10000,
1850   \hskip\csuse{afternote@#2}} % but add the glue to go between the notes
1851

```

`\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).

```

1852 \newcommand*{\parafootfmt}[4][4=Z]{%
1853   \insertparafootsep{#4}%
1854   \ledsetnormalparstuff%
1855   \printlinefootnote{#1}{#4}%
1856   {\nottoggle{Xlemmadisablefontselection@#4}{\select@lemmafont#1|#2}{#2}}%
1857   {\iftoggle{nosep@}{\hskip\csuse{inplaceoflemmaseparator@#4}}{\ifcsemtylemmaseparator@#4}%
1858     {\hskip\csuse{inplaceoflemmaseparator@#4}}}%
1859     {\nobreak\hskip\csuse{beforelemmaseparator@#4}\csuse{lemmaseparator@#4}\hskip\csuse{afterlemmasep
1860     }}%
1861   #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.

```

1862 \newcommand*{\para@footgroup}[1]{%
1863   \unvvh\csname #1footins\endcsname
1864   \ifcsstring{Xragged@#1}{L}{\RaggedLeft}{}%
1865   \ifcsstring{Xragged@#1}{R}{\RaggedRight}{}%
1866   \makehboxofhboxes
1867   \setbox0=\hbox{\csuse{Xnotefontsize@#1}\csuse{txtbeforeXnotes@#1}}\unhbox0 \removehboxes}%
1868   \csuse{Xnotefontsize@#1}
1869   \noindent\unhbox0\par%
1870   \global\hsize=\old@hsize%
1871   }%
1872

```

`\mppara@footgroup` The minipage version.

```

1873 \newcommand*{\mppara@footgroup}[1]{%
1874   \setXnoteswidthliketwocolumns@{#1}%
1875   \vskip\skip\@nameuse{mp#1footins}
1876   \ifl@dpairing\ifparledgroup%
1877     \leavevmode\marks\parledgroup@{begin}%
1878     \marks\parledgroup@series{#1}%
1879     \marks\parledgroup@type{Xfootnote}%
1880   \fi\fi\normalcolor
1881   \ifparledgroup%
1882     \ifl@dpairing%
1883     \else%
1884       \setXnoteswidthliketwocolumns@{#1}%
1885       \setXnotespositionliketwocolumns@{#1}%
1886       \print@Xfootnoterule{#1}%%
1887       \vskip\csuse{afterXrule@#1}%
1888     \fi%
1889   \else%
1890     \setXnoteswidthliketwocolumns@{#1}%
1891     \setXnotespositionliketwocolumns@{#1}%
1892     \print@Xfootnoterule{#1}%%
1893     \vskip\csuse{afterXrule@#1}%
1894   \fi%
1895   \unvbox\csname mp#1footins\endcsname
1896   \ifcsstring{Xragged@#1}{L}{\RaggedLeft}{}%
1897   \ifcsstring{Xragged@#1}{R}{\RaggedRight}{}%
1898   \makehboxofhboxes
1899   \setbox0=\hbox{\csuse{Xnotefontsize@#1}\csuse{txtbeforeXnotes@#1}}\unhbox0 \removehboxes
1900   \csuse{Xnotefontsize@#1}
1901   \noindent\unhbox0\par}}
1902
```

`\makehboxofhboxes`

```

\removehboxes 1903 \newcommand*{\makehboxofhboxes}{\setbox0=\hbox{}}%
1904   \loop
1905     \unpenalty
1906     \setbox2=\lastbox
1907     \ifhbox2
1908       \setbox0=\hbox{\box2\unhbox0}%
1909     \repeat}
1910
1911 \newcommand*{\removehboxes}{\setbox0=\lastbox
1912   \ifhbox0{\removehboxes}\unhbox0 \fi}
1913
```

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 1914 \newcommand{\insertparafootsep}[1]{%
1915   \ifnumequal{\csuse{prevpage#1@num}}{\page@num}%
1916     {\ifcscdef{prevline#1}% Be sur \prevline#1 exists.
1917       {\ifnumequal{\csuse{prevline#1}}{\line@num}%
1918         {\ifcseempty{sympnenum}{\csuse{parafootsep@#1}}{}}}%
1919       {\csuse{parafootsep@#1}}}%
1920   }%
1921   {\csuse{parafootsep@#1}}}%
1922 }%
1923 {}%
1924 \global\csname prevpage#1@num\endcsname=\page@num%
1925 }

```

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 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 L^AT_EX **\line** macro has no relationship to the TeX **\line**. The L^AT_EX equivalent is **\@@line**.

```

1926 \newcount\@k \newdimen\@h
1927 \newcommand*\rigidbalance[3]{\setbox0=\box#1 \@k=#2 \@h=#3
1928   \@@line{\splittopskip=\@h \vbadness=\@M \hfilneg
1929     \valign{##\vfil\cr\dosplits}}}
1930
1931 \newcommand*\dosplits{\ifnum\@k>0 \noalign{\hfil}\splitoff
1932   \global\advance\@k-1\cr\dosplits\fi}
1933
1934 \newcommand*\splitoff{\dimen0=\ht0
1935   \divide\dimen0 by\@k \advance\dimen0 by\@h
1936   \setbox2 \vsplit0 to \dimen0
1937   \unvbox2 }
1938

```

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.

```

1939 \newcommand*\footthreecol[1]{%
1940   \csgdef{series@display#1}{threecol}
1941   \expandafter\let\csname v#1footnote\endcsname=\threecolvfootnote
1942   \expandafter\let\csname #1footfmt\endcsname=\threecolfootfmt

```

```

1943 \expandafter\let\csname #1footgroup\endcsname=\threecolfootgroup
1944 \threecolfootsetup{#1}

```

The additional setup for minipages.

```

1945 \expandafter\let\csname mpv#1footnote\endcsname=\mpnormalvfootnote
1946 \expandafter\let\csname mp#1footgroup\endcsname=\mpthreecolfootgroup
1947 \mpthreecolfootsetup{#1}
1948 }
1949

```

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

```

1950 \newcommand*{\threecolfootsetup}[1]{%
1951   \count\csname #1footins\endcsname 333
1952   \multiply\dimen\csname #1footins\endcsname \thr@@}

```

`\mpthreecolfootsetup` The setup for minipages.

```

1953 \newcommand*{\mpthreecolfootsetup}[1]{%
1954   \count\csname mp#1footins\endcsname 333
1955   \multiply\dimen\csname mp#1footins\endcsname \thr@@}
1956

```

`\threecolvfootnote` `\threecolvfootnote` is the `\vfootnote` command for three-column notes. The call to `\notefontsetup` ensures that the `\splittopskip` and `\splitmaxdepth` take their values from the right `\strutbox`: the one used in a footnotes. Note especially the importance of temporarily reducing the `\hsize` to 0.3 of its normal value. This determines the widths of the individual columns. So if the normal `\hsize` is, say, 10 cm, then each column will be $0.3 \times 10 = 3$ cm wide, leaving a gap of 1 cm spread equally between columns (i.e., .5 cm between each).

The arguments are 1) the note series letter and 2) the full text of the note (including numbers, lemma and text).

```

1957 \notbool{parapparatus@}\newcommand*{\threecolvfootnote}[2]{%
1958   \insert\csname #1footins\endcsname\bgroup
1959   \csuse{Xnotefontsize@#1}

```

```

1960 \footsplitskips
1961 \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.

```

1962 \notbool{parapparatus@}\newcommand*{\newcommandx}{\threecolfootfmt}[4][4=Z]{%
1963   \normal@pars
1964   \hsize \csuse{hsizethreecol@#4}
1965   \parindent=0pt
1966   \tolerance=5000
1967   \raggedright
1968   \hangindent=\csuse{Xhangindent@#4}
1969   \leavevmode
1970   \strut{\printlinefootnote{#1}{#4}}%
1971   {\nottoggle{Xlemmadisablefontselection@#4}{\select@lemfont#1|#2}{#2}}%
1972   \iftoggle{nosep@}{\hskip\csuse{inplaceoflemmaseparator@#4}}{\ifcempty{lemmaseparator@#4}%
1973     {\hskip\csuse{inplaceoflemmaseparator@#4}}%
1974     {\nobreak\hskip\csuse{beforelemmaseparator@#4}\csuse{lemmaseparator@#4}\hskip\csuse{afterlemmasep
1975     }}%
1976   #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.

```

1977 \newcommand*{\threecolfootgroup}[1]{\notefontsetup
1978   {\csuse{Xnotefontsize@#1}\noindent\csuse{txtbeforeXnotes@#1}}\par
1979   \splittopskip=\ht\strutbox
1980   \expandafter
1981   \rigidbalance\csname #1footins\endcsname \thr@@ \splittopskip}

```

`\mpthreecolfootgroup` The setup for minipages.

```

1982 \newcommand*{\mpthreecolfootgroup}[1]{%
1983   \vskip\skip\@nameuse{mp#1footins}
1984   \ifl@dpairing\ifparledgroup%
1985     \leavevmode\marks\parledgroup@{begin}%
1986     \marks\parledgroup@series{#1}%
1987     \marks\parledgroup@type{Xfootnote}%
1988   \fi\fi\normalcolor

```

```

1989 \ifparledgroup%
1990   \ifl@dpairing%
1991   \else%
1992     \setXnoteswidthliketwocolumns@{#1}%
1993     \setXnotespositionliketwocolumns@{#1}%
1994     \print@Xfootnoterule{#1}%%
1995     \vskip\csuse{afterXrule@#1}%
1996   \fi%
1997 \else%
1998   \setXnoteswidthliketwocolumns@{#1}%
1999   \setXnotespositionliketwocolumns@{#1}%
2000   \print@Xfootnoterule{#1}%%
2001   \vskip\csuse{afterXrule@#1}%
2002 \fi%
2003 {\csuse{Xnotefontsize@#1}\noindent\csuse{txtbeforeXnotes@#1}}\par
2004 \splittopskip=\ht\strutbox
2005 \expandafter
2006 \rigidbalance\csname mp#1footins\endcsname \thr@@ \splittopskip}}
2007

```

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.

```

2008 \newcommand*{\foottwocol}[1]{%
2009   \csgdef{series@display#1}{twocol}
2010   \expandafter\let\csname v#1footnote\endcsname=\twocolvfootnote
2011   \expandafter\let\csname #1footfmt\endcsname=\twocolfootfmt
2012   \expandafter\let\csname #1footgroup\endcsname=\twocolfootgroup
2013   \twocolfootsetup{#1}

```

The additional setup for minipages.

```

2014   \expandafter\let\csname mpv#1footnote\endcsname=\mpnormalvfootnote
2015   \expandafter\let\csname mp#1footgroup\endcsname=\mptwocolfootgroup
2016   \mptwocolfootsetup{#1}
2017 }
2018

```

`\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`

```

2019 \newcommand*{\twocolfootsetup}[1]{%
2020   \count\csname #1footins\endcsname 500
2021   \multiply\dimen\csname #1footins\endcsname \tw@
2022 \notbool{parapparatus@}{\newcommand*}{\newcommand}{\twocolvfootnote}[2]{\insert\csname #1
2023   \csuse{Xnotefontsize@#1}
2024   \footsplitskips
2025   \csname #1footfmt\endcsname #2[#1]\egroup}

```

```

2026 \notbool{parapparatus@}{\newcommand*}{\newcommand*}{\twocolfootfmt}[4][4=Z]{% 4th arg is optional, f
2027   \normal@pars
2028   \hsize \csuse{hsizeetwocol@#4}
2029   \parindent=0pt
2030   \tolerance=5000
2031   \raggedright
2032   \hangindent=\csuse{Xhangindent@#4}
2033   \leavevmode
2034   \strut{\printlinefootnote{#1}{#4}}%
2035   {\nottoggle{Xlemmadisablefontselection@#4}{\select@lemmafnt#1|#2}{#2}}%
2036   \iftoggle{nosep@}{\hskip\csuse{inplaceoflemmaseparator@#4}}{\ifcsemt{lemmaseparator@#4}}%
2037     {\hskip\csuse{inplaceoflemmaseparator@#4}}%
2038     {\nobreak\hskip\csuse{beforelemmaseparator@#4}\csuse{lemmaseparator@#4}\hskip\csuse{afterlemmasep
2039     }}%
2040   #3\strut\par\allowbreak}

2041 \newcommand*{\twocolfootgroup}[1]{\csuse{Xnotefontsize@#1}
2042   {\csuse{Xnotefontsize@#1}\noindent\csuse{txtbeforeXnotes@#1}}\par
2043   \splittopskip=\ht\strutbox
2044   \expandafter
2045   \rigidbalance\csname #1footins\endcsname \tw@ \splittopskip}}
2046

```

`\mptwocolfootsetup` The versions for minipages.

```

\mptwocolfootgroup 2047 \newcommand*{\mptwocolfootsetup}[1]{%
2048   \count\csname mp#1footins\endcsname 500
2049   \multiply\dimen\csname mp#1footins\endcsname \tw@}

2050 \newcommand*{\mptwocolfootgroup}[1]{%
2051   \vskip\skip\@nameuse{mp#1footins}
2052   \ifl@dpairing\ifparledgroup%
2053     \leavevmode\marks\parledgroup@{begin}%
2054     \marks\parledgroup@series{#1}%
2055     \marks\parledgroup@type{Xfootnote}%
2056   \fi\fi\normalcolor
2057   \ifparledgroup%
2058     \ifl@dpairing%
2059     \else%
2060       \setXnoteswidthliketwocolumns@{#1}%
2061       \setXnotespositionliketwocolumns@{#1}%
2062       \print@Xfootnoterule{#1}%
2063       \vskip\csuse{afterXrule@#1}%
2064     \fi%
2065   \else%
2066     \setXnoteswidthliketwocolumns@{#1}%
2067     \setXnotespositionliketwocolumns@{#1}%
2068     \print@Xfootnoterule{#1}%
2069     \vskip\csuse{afterXrule@#1}%
2070   \fi%
2071   {\csuse{Xnotefontsize@#1}\noindent\csuse{txtbeforeXnotes@#1}}\par
2072   \splittopskip=\ht\strutbox

```

```

2073 \expandafter
2074 \rigidbalance\csname mp#1footins\endcsname \tw@ \splittopskip}}
2075

```

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.

```

2076 \providecommand*\multiplefootnotemarker}{3sp}
2077 \providecommand*\multfootsep}{\textsuperscript{\normalfont,}}
2078

```

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

```

2079 \providecommand*\m@mmf@prepare}{%
2080 \kern-\multiplefootnotemarker
2081 \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`.

```

2082 \providecommand*\m@mmf@check}{%
2083 \ifdim\lastkern=\multiplefootnotemarker\relax
2084 \edef\x@sf{\the\spacefactor}%
2085 \unkern
2086 \multfootsep
2087 \spacefactor\x@sf\relax
2088 \fi}
2089

```

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

```

2090 \@ifclassloaded{memoir}{-}{%

```

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

```

2091 \apptocmd{\@footnotetext}{\m@mmf@prepare}{-}{-}

```

`\@footnotemark` Modify `\@footnotemark` to cater for adjacent `\footnotes`.

```

2092 \renewcommand*\@footnotemark}{%

```

```

2093 \leavevmode
2094 \ifhmode
2095   \edef\x@sf{\the\spacefactor}%
2096   \m@mmf@check
2097   \nobreak
2098 \fi
2099 \@makefnmark
2100 \m@mmf@prepare
2101 \ifhmode\spacefactor\x@sf\fi
2102 \relax}

```

Finished the modifications for the non-memoir case.

```

2103 }
2104

```

`\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.

```

2105 \pretocmd{\@footnotetext}{%
2106   \ifnumberedpar@
2107     \edtext{}{\l@dbfnote{#1}}%
2108   \else
2109     {}{}
2110 \apptocmd{\@footnotetext}{\fi}{}{}%

```

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

```

2111 \newcommand{\l@dbfnote}[1]{%
2112   \ifnumberedpar@
2113   \gdef\@tag{#1\relax}%
2114   \xright@appenditem{\noexpand\l@dbfnote{\@tag}}{\@thefnmark}}%
2115   \to\inserts@list
2116   \global\advance\insert@count \@ne
2117 \fi\ignorespaces}
2118 \newcommand{\vl@dbfnote}[2]{%
2119   \def\@thefnmark{#2}%
2120   \@footnotetext{#1}%
2121 }%

```

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 2122 \newcommand*\prebodyfootmark{%
2123   \leavevmode
2124   \ifhmode

```

```

2125 \edef\@x@sf{\the\spacefactor}%
2126 \m@mmf@check
2127 \nobreak
2128 \fi}
2129 \newcommand{\postbodyfootmark}{%
2130 \m@mmf@prepare
2131 \ifhmode\spacefactor\@x@sf\fi\relax}
2132

```

`\normal@footnotemarkX` `\normal@footnotemarkX{<series>}` sets up the typesetting of the marker at the point where the footnote is called for.

```

2133 \newcommand*{\normal@footnotemarkX}[1]{%
2134 \prebodyfootmark
2135 \@nameuse{bodyfootmark#1}%
2136 \postbodyfootmark}
2137

```

`\normalbodyfootmarkX` The `\normalbodyfootmarkX{<series>}` *really* typesets the in-text marker. The style is the normal superscript.

```

2138 \newcommand*{\normalbodyfootmarkX}[1]{%
2139 \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>`.

```

2140 \newcommand*{\normalvfootnoteX}[2]{%
2141 \insert\@nameuse{footins#1}\bgroup
2142 \csuse{bhooknoteX@#1}
2143 \csuse{notefontsizeX@#1}
2144 \footplitskips
2145 \ifl@dpairing\ifl@dpaging\else%
2146 \setnotesXwidthliketwocolumns@{#1}%
2147 \fi\fi%
2148 \setnotesXpositionliketwocolumns@{#1}%
2149 \spaceskip=\z@skip \xspaceskip=\z@skip
2150 \csuse{\csuse{footnote@dir}}\if@RTL\else\noindent\leavevmode\fi\@nameuse{footfmt#1}{#1}%
2151

```

`\mpnormalvfootnoteX` The minipage version.

```

2152 \newcommand*{\mpnormalvfootnoteX}[2]{%
2153 \global\setbox\@nameuse{mpfootins#1}\vbox{%
2154 \unvbox\@nameuse{mpfootins#1}
2155 \csuse{bhooknoteX@#1}
2156 \csuse{notefontsizeX@#1}
2157 \hsize\columnwidth
2158 \@parboxrestore
2159 \color@begingroup
2160 \@nameuse{footfmt#1}{#1}{#2}\color@endgroup}}
2161

```


`\normalfootfmtX` `\normalfootfmtX{<series>}{<text>}` typesets the footnote text, prepended by the marker.

```
2162 \newcommand*{\normalfootfmtX}[2]{%
2163   \protected@edef\@currentlabel{%
2164     \@nameuse{@thefnmark#1}%
2165   }%
2166   \ledsetnormalparstuff
2167   \hangindent=\csuse{hangindentX@#1}%
2168   {\csuse{notenumfontX@#1}\@nameuse{footfootmark#1}}\strut%\enspace
2169   #2\strut\par}}
2170
```

`\normalfootfootmarkX` `\normalfootfootmarkX{<series>}` is called by `\normalfootfmtX` to typeset the footnote marker in the footer before the footnote text.

```
2171 \newcommand*{\normalfootfootmarkX}[1]{%
2172   \textsuperscript{\@nameuse{@thefnmark#1}}}
2173
```

`\normalfootstartX` `\normalfootstartX{<series>}` is the `<series>` footnote starting macro used in the output routine.

```
2174 \newcommand*{\normalfootstartX}[1]{%
2175   \ifdimequal{0pt}{\prenotesX@}{}%
2176   {%
2177     \iftoggle{prenotesX@}{%
2178       \togglefalse{prenotesX@} \skip\csname footins#1\endcsname=\csuse{prenotesX@}}%
2179     }%
2180   }%
2181   \vskip\skip\csname footins#1\endcsname%
2182   \leftskip=\z@
2183   \rightskip=\z@
2184   \ifl@dpairing\else%
2185     \hspace=\old@hsize%
2186   \fi%
2187   \setnotesXwidthliketwocolumns@{#1}%
2188   \setnotesXpositionliketwocolumns@{#1}%
2189   \print@footnoteXrule{#1}%
2190   \vskip\csuse{afterruleX@#1}}
2191
```

`\normalfootnoteruleX` The rule drawn before the footnote series group.

```
2192 \let\normalfootnoteruleX=\footnoterule
2193
```

`\normalfootgroupX` `\normalfootgroupX{<series>}` sends the contents of the `<series>` insert box to the output page without alteration.

```
2194 \newcommand*{\normalfootgroupX}[1]{%
2195   \unvbox\@nameuse{footins#1}%
2196   \hspace=\old@hsize%
2197   }%
```

2198

\mpnormalfootgroupX The minipage version.

```

2199 \newcommand*{\mpnormalfootgroupX}[1]{%
2200   \vskip\skip\@nameuse{mpfootins#1}
2201   \ifl@dpairing\ifparledgroup%
2202     \leavevmode\marks\parledgroup@{begin}%
2203     \marks\parledgroup@series{#1}%
2204     \marks\parledgroup@type{footnoteX}%
2205   \fi\fi\normalcolor
2206   \ifparledgroup%
2207     \ifl@dpairing%
2208     \else%
2209       \setnotesXwidthliketwocolumns@{#1}%
2210       \setnotesXpositionliketwocolumns@{#1}%
2211       \print@footnoteXrule{#1}%
2212       \vskip\csuse{afterruleX@#1}%
2213     \fi%
2214   \else%
2215     \setnotesXwidthliketwocolumns@{#1}%
2216     \setnotesXpositionliketwocolumns@{#1}%
2217     \print@footnoteXrule{#1}%
2218     \vskip\csuse{afterruleX@#1}%
2219   \fi%
2220   \unvbox\@nameuse{mpfootins#1}}
2221
```

\normalbfnoteX

```

2222 \newcommand{\normalbfnoteX}[2]{%
2223   \ifnumberedpar@
2224     \protected@xdef\thisfootnote{\csuse{thefootnote#1}}%
2225     \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}}{\expandonce\thisfootnote}}%
2226     \to\inserts@list
2227     \global\advance\insert@count \@ne
2228   \fi\ignorespaces}
2229
```

\vbfnoteX

```

2230 \newcommand{\vbfnoteX}[3]{%
2231   \@namedef{@thefnmark#1}{#3}%
2232   \@nameuse{regvfootnote#1}{#1}{#2}}
2233
```

\vnumfootnoteX

```

2234 \newcommand{\vnumfootnoteX}[2]{%
2235   \ifnumberedpar@
2236     \edtext{}{\normalbfnoteX{#1}{#2}}%
2237   \else
2238     \@nameuse{regvfootnote#1}{#1}{#2}%

```

```

2239 \fi}
2240

```

`\footnormalX` `\footnormalX{<series>}` initialises the settings for the `<series>` footnotes. This should always be called for each series.

```

2241 \newcommand*{\footnormalX}[1]{%
2242   \csgdef{series@displayX#1}{normalX}
2243   \expandafter\let\csname footstart#1\endcsname=\normalfootstartX
2244   \@namedef{@footnotemark#1}{\normal@footnotemarkX{#1}}
2245   \@namedef{bodyfootmark#1}{\normalbodyfootmarkX{#1}}
2246   \expandafter\let\csname regvfootnote#1\endcsname=\normalvfootnoteX
2247   \expandafter\let\csname vfootnote#1\endcsname=\vnumfootnoteX
2248   \expandafter\let\csname footfmt#1\endcsname=\normalfootfmtX
2249   \@namedef{footfootmark#1}{\normalfootfootmarkX{#1}}
2250   \expandafter\let\csname footgroup#1\endcsname=\normalfootgroupX
2251   \expandafter\let\csname footnoterule#1\endcsname=\normalfootnoteruleX
2252   \count\csname footins#1\endcsname=1000
2253   \dimen\csname footins#1\endcsname=\csuse{maxhnotesX@#1}
2254   \skip\csname footins#1\endcsname=\csuse{beforenotesX@#1}

```

Additions for minipages.

```

2255   \expandafter\let\csname mpvfootnote#1\endcsname=\mpnormalvfootnoteX
2256   \expandafter\let\csname mpfootgroup#1\endcsname=\mpnormalfootgroupX
2257   \count\csname mpfootins#1\endcsname=1000
2258   \dimen\csname mpfootins#1\endcsname=\csuse{maxhnotesX@#1}
2259   \skip\csname mpfootins#1\endcsname=\csuse{beforenotesX@#1}
2260 }
2261

```

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>}`

```

2262 \newcommand*{\foottwocolX}[1]{%
2263   \csgdef{series@displayX#1}{twocol}
2264   \expandafter\let\csname regvfootnote#1\endcsname=\twocolvfootnoteX
2265   \expandafter\let\csname footfmt#1\endcsname=\twocolfootfmtX
2266   \expandafter\let\csname footgroup#1\endcsname=\twocolfootgroupX
2267   \twocolfootsetupX{#1}
2268   \expandafter\let\csname mpvfootnote#1\endcsname=\mpnormalvfootnoteX
2269   \expandafter\let\csname mpfootgroup#1\endcsname=\mptwocolfootgroupX
2270   \mptwocolfootsetupX{#1}}
2271

```

```

\twocolfootsetupX \twocolfootsetupX{<series>}
\mptwocolfootsetupX 2272 \newcommand*{\twocolfootsetupX}[1]{%
2273   \count\csname footins#1\endcsname 500

```

```

2274 \multiply\dimen\csname footins#1\endcsname by \tw@}
2275 \newcommand*{\mptwocolfootsetupX}[1]{%
2276 \count\csname mpfootins#1\endcsname 500
2277 \multiply\dimen\csname mpfootins#1\endcsname by \tw@}
2278

```

```
\twocolvfootnoteX \twocolvfootnoteX{<series>}
```

```

2279 \newcommand*{\twocolvfootnoteX}[2]{%
2280 \insert\csname footins#1\endcsname\bgroup
2281 \csuse{notefontsizeX@#1}
2282 \footsplitskips
2283 \spaceskip=\z@skip \xspaceskip=\z@skip
2284 \@nameuse{footfmt#1}{#1}{#2}\egroup}
2285

```

```
\twocolfootfmtX \twocolfootfmtX{<series>}
```

```

2286 \newcommand*{\twocolfootfmtX}[2]{%
2287 \protected@edef\@currentlabel{%
2288 \@nameuse{thefnmark#1}%
2289 }%
2290 \normal@pars
2291 \hangindent=\csuse{hangindentX@#1}%
2292 \hsize \csuse{hsizetwocolX@#1}
2293 \parindent=\z@
2294 %%% \parfillskip=0pt \@plus 1fil
2295 \tolerance=5000\relax
2296 \raggedright
2297 \leavevmode
2298 {\csuse{notenumfontX@#1}\@nameuse{footfootmark#1}\strut%\enspace
2299 #2\strut\par}\allowbreak}
2300

```

```
\twocolfootgroupX \twocolfootgroupX{<series>}
```

```

\mptwocolfootgroupX 2301 \newcommand*{\twocolfootgroupX}[1]{\csuse{notefontsizeX@#1}
2302 \splittopskip=\ht\strutbox
2303 \expandafter
2304 \rigidbalance\csname footins#1\endcsname \tw@ \splittopskip}}
2305 \newcommand*{\mptwocolfootgroupX}[1]{%
2306 \vskip\skip\@nameuse{mpfootins#1}
2307 \ifl@dpairing\ifparledgroup%
2308 \leavevmode\marks\parledgroup@{begin}%
2309 \marks\parledgroup@series{#1}%
2310 \marks\parledgroup@type{footnoteX}%
2311 \fi\fi\normalcolor
2312 \ifparledgroup%
2313 \ifl@dpairing%
2314 \else%
2315 \setnotesXwidthliketwocolumns@{#1}%
2316 \setnotesXpositionliketwocolumns@{#1}%

```

```

2317     \print@footnoteXrule{#1}%
2318     \vskip\csuse{afterruleX@#1}%
2319     \fi%
2320 \else%
2321     \setnotesXwidthliketwocolumns@{#1}%
2322     \setnotesXpositionliketwocolumns@{#1}%
2323     \print@footnoteXrule{#1}%
2324     \vskip\csuse{afterruleX@#1}%
2325     \fi%
2326     \splittopskip=\ht\strutbox
2327     \expandafter
2328     \rigidbalance\csname mpfootins#1\endcsname \tw@ \splittopskip}}
2329

```

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>}

2330 \newcommand*{\footthreecolX}[1]{%
2331   \csgdef{series@displayX#1}{threecol}
2332   \expandafter\let\csname regvfootnote#1\endcsname=\threecolvfootnoteX
2333   \expandafter\let\csname footfmt#1\endcsname=\threecolfootfmtX
2334   \expandafter\let\csname footgroup#1\endcsname=\threecolfootgroupX
2335   \threecolfootsetupX{#1}
2336   \expandafter\let\csname mpvfootnote#1\endcsname=\mpnormalvfootnoteX
2337   \expandafter\let\csname mpfootgroup#1\endcsname=\mpthreecolfootgroupX
2338   \mpthreecolfootsetupX{#1}}
2339

\threecolfootsetupX \threecolfootsetupX{<series>}
\mpthreecolfootsetupX 2340 \newcommand*{\threecolfootsetupX}[1]{%
2341   \count\csname footins#1\endcsname 333
2342   \multiply\dimen\csname footins#1\endcsname by \thr@@}
2343 \newcommand*{\mpthreecolfootsetupX}[1]{%
2344   \count\csname mpfootins#1\endcsname 333
2345   \multiply\dimen\csname mpfootins#1\endcsname by \thr@@}
2346

\threecolvfootnoteX \threecolvfootnoteX{<series>}{<text>}
2347 \newcommand*{\threecolvfootnoteX}[2]{%
2348   \insert\csname footins#1\endcsname\bgroup
2349   \csuse{notefontsizeX@#1}
2350   \footplitskips
2351   \@nameuse{footfmt#1}{#1}{#2}\egroup}
2352

\threecolfootfmtX \threecolfootfmtX{<series>}

```

```

2353 \newcommand*{\threecolfootfmtX}[2]{%
2354   \protected@edef\@currentlabel{%
2355     \@nameuse{@thefnmark#1}%
2356   }%
2357   \hangindent=\csuse{hangindentX@#1}%
2358   \normal@pars
2359   \hsize \csuse{hsizethreecolX@#1}
2360   \parindent=\z@
2361   %%% \parfillskip=0pt \@plus 1fil
2362   \tolerance=5000\relax
2363   \raggedright
2364   \leavevmode
2365   {\csuse{notenumfontX@#1}\@nameuse{footfootmark#1}\strut%\enspace
2366     #2\strut\par}\allowbreak}
2367
\threecolfootgroupX \threecolfootgroupX{\series}
\mpthreecolfootgroupX 2368 \newcommand*{\threecolfootgroupX}[1]{\csuse{notefontsizeX@#1}
2369   \splittopskip=\ht\strutbox
2370   \expandafter
2371   \rigidbalance\csname footins#1\endcsname \thr@@ \splittopskip}}
2372 \newcommand*{\mpthreecolfootgroupX}[1]{\csuse{notefontsizeX@#1}
2373   \vskip\skip\@nameuse{mpfootins#1}
2374   \ifl@dpairing\ifparledgroup
2375     \leavevmode\marks\parledgroup@{begin}%
2376     \marks\parledgroup@series{#1}%
2377     \marks\parledgroup@type{footnoteX}%
2378   \fi\fi\normalcolor
2379   \ifparledgroup%
2380     \ifl@dpairing%
2381     \else%
2382       \setnotesXwidthliketwocolumns@{#1}%
2383       \setnotesXpositionliketwocolumns@{#1}%
2384       \print@footnoteXrule{#1}%
2385       \vskip\csuse{afterruleX@#1}%
2386     \fi%
2387   \else%
2388     \setnotesXwidthliketwocolumns@{#1}%
2389     \setnotesXpositionliketwocolumns@{#1}%
2390     \print@footnoteXrule{#1}%
2391     \vskip\csuse{afterruleX@#1}%
2392   \fi%
2393   \splittopskip=\ht\strutbox
2394   \expandafter
2395   \rigidbalance\csname mpfootins#1\endcsname \thr@@ \splittopskip}}
2396

```

25.5 Paragraphed footnotes

The following macros set footnotes as one paragraph.

```

\footparagraphX \footparagraphX{<series>}
2397 \newcommand*{\footparagraphX}[1]{%
2398 \csgdef{series@displayX#1}{\paragraphX}%
2399 \expandafter\newcount\csname prevpage#1@num\endcsname
2400 \expandafter\let\csname footstart#1\endcsname=\parafootstartX
2401 \expandafter\let\csname regvfootnote#1\endcsname=\para@vfootnoteX
2402 \expandafter\let\csname footfmt#1\endcsname=\parafootfmtX
2403 \expandafter\let\csname footgroup#1\endcsname=\para@footgroupX
2404 \expandafter\let\csname footnoterule#1\endcsname=\normalfootnoteruleX
2405 \count\csname footins#1\endcsname=1000
2406 \dimen\csname footins#1\endcsname=\csuse{maxhnotesX@#1}
2407 \skip\csname footins#1\endcsname=\csuse{beforenotesX@#1}
2408 \expandafter\let\csname mpvfootnote#1\endcsname=\mppara@vfootnoteX
2409 \expandafter\let\csname mpfootgroup#1\endcsname=\mppara@footgroupX
2410 \count\csname mpfootins#1\endcsname=1000
2411 \dimen\csname mpfootins#1\endcsname=\csuse{maxhnotesX@#1}
2412 \skip\csname mpfootins#1\endcsname=\csuse{beforenotesX@#1}
2413 \para@footsetupX{#1}}
2414

\para@footsetupX \para@footsetupX{<series>}
2415 \newcommand*{\para@footsetupX}[1]{\csuse{notefontsizeX@#1}
2416 \setnotesXwidthliketwocolumns@{#1}%
2417 \dimen0=\baselineskip
2418 \multiply\dimen0 by 1024
2419 \divide\dimen0 by \columnwidth \multiply\dimen0 by \footfudgefiddle\relax%
2420 \expandafter
2421 \xdef\csname footfudgefactor#1\endcsname{%
2422 \expandafter\strip@pt\dimen0 }}
2423

\parafootstartX \parafootstartX{<series>}
2424 \newcommand*{\parafootstartX}[1]{%
2425 \ifdimequal{0pt}{\prenotesX@}{}%
2426 {%
2427 \iftoggle{prenotesX@}{%
2428 \togglefalse{prenotesX@}\skip\csname footins#1\endcsname=\csuse{prenotesX@}}%
2429 }%
2430 }%
2431 \vskip\skip\csname footins#1\endcsname%
2432 \leftskip=\z@
2433 \rightskip=\z@
2434 \parindent=\z@
2435 \vskip\skip\@nameuse{footins#1}%
2436 \setnotesXwidthliketwocolumns@{#1}%
2437 \setnotesXpositionliketwocolumns@{#1}%
2438 \print@footnoterule{#1}%
2439 \vskip\csuse{afterterruleX@#1}%
2440 }
2441

```

```

\para@vfootnoteX \para@vfootnoteX{<series>}{<text>}
\mppara@vfootnoteX 2442 \newcommand*{\para@vfootnoteX}[2]{%
2443   \insert\csname footins#1\endcsname
2444   \bgroup
2445     \csuse{bhooknoteX@#1}
2446     \csuse{notefontsizeX@#1}
2447     \footsplitskips
2448     \setbox0=\vbox{\hsize=\maxdimen
2449       \noindent\@nameuse{footfmt#1}{#1}{#2}}%
2450     \setbox0=\hbox{\unvbox0[#1]}%
2451     \dp0=\z@
2452     \ht0=\csname footfudgefactor#1\endcsname\wd0
2453     \box0
2454     \penalty0
2455   \egroup}
2456 \newcommand*{\mppara@vfootnoteX}[2]{%
2457   \global\setbox\@nameuse{mpfootins#1}\vbox{%
2458     \unvbox\@nameuse{mpfootins#1}
2459     \csuse{bhooknoteX@#1}
2460     \csuse{notefontsizeX@#1}
2461     \footsplitskips
2462     \setbox0=\vbox{\hsize=\maxdimen
2463       \noindent\color@begingroup\@nameuse{footfmt#1}{#1}{#2}\color@endgroup}%
2464     \setbox0=\hbox{\unvbox0[#1]}%
2465     \dp0=\z@
2466     \ht0=\csname footfudgefactor#1\endcsname\wd0
2467     \box0
2468     \penalty0}}
2469

\parafootfmtX \parafootfmtX{<series>}
2470 \newcommand*{\parafootfmtX}[2]{%
2471   \protected@edef\@currentlabel{%
2472     \@nameuse{@thefnmark#1}%
2473   }%
2474   \insertparafootsep{#1}%
2475   \ledsetnormalparstuff
2476   {\csuse{notenunfontX@#1}\csuse{notenunfontX@#1}\@nameuse{footfootmark#1}\strut%\enspace
2477     #2\penalty-10}}
2478

\para@footgroupX \para@footgroupX{<series>}
\mppara@footgroupX 2479 \newcommand*{\para@footgroupX}[1]{%
2480   \unvbox\csname footins#1\endcsname
2481   \ifcsstring{raggedX@#1}{L}{\RaggedLeft}{}%
2482   \ifcsstring{raggedX@#1}{R}{\RaggedRight}{}%
2483   \makehboxofhboxes
2484   \setbox0=\hbox{\unhbox0 \removehboxes}%
2485   \csuse{notefontsizeX@#1}

```



```

2486 \noindent\unhbox0\par}
2487 \newcommand*\mppara@footgroupX}[1]{%
2488 \setnotesXwidthliketwocolumns@{#1}%
2489 \vskip\skip\@nameuse{mpfootins#1}
2490 \ifl@dpairing\ifparledgroup
2491 \leavevmode%
2492 \leavevmode\marks\parledgroup@{begin}%
2493 \marks\parledgroup@series{#1}%
2494 \marks\parledgroup@type{footnoteX}%
2495 \fi\fi\normalcolor
2496 \ifparledgroup%
2497 \ifl@dpairing%
2498 \else%
2499 \setnotesXwidthliketwocolumns@{#1}%
2500 \setnotesXpositionliketwocolumns@{#1}%
2501 \print@footnoteXrule{#1}%
2502 \vskip\csuse{afterruleX@#1}%
2503 \fi%
2504 \else%
2505 \setnotesXwidthliketwocolumns@{#1}%
2506 \setnotesXpositionliketwocolumns@{#1}%
2507 \print@footnoteXrule{#1}%
2508 \vskip\csuse{afterruleX@#1}%
2509 \fi%
2510 \unvbox\csname mpfootins#1\endcsname
2511 \ifcsstring{raggedX@#1}{L}{\RaggedLeft}{}%
2512 \ifcsstring{raggedX@#1}{R}{\RaggedRight}{}%
2513 \makehboxofhboxes
2514 \setbox0=\hbox{\unhbox0 \removehboxes}%
2515 \csuse{notefontsizeX@#1}
2516 \noindent\unhbox0\par}}
2517

```

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 at the on the setup for paragraphed notes.

```

2518
2519 \newdimen\old@hsize%
2520 \old@hsize=\linewidth%
2521
2522 \newcommand{\setXnoteswidthliketwocolumns@}[1]{%
2523 \global\let\hsize@fornote=\linewidth%

```

```

2524 \global\old@hsize=\linewidth%
2525 \iftoggle{Xnoteswidthliketwocolumns@#1}{%
2526   {%
2527     \csuse{setwidthliketwocolumns@\columns@position}%
2528     \global\let\hsize@fornote=\hsize%
2529   }%
2530   {%
2531     \let\hsize=\hsize@fornote%
2532     \let\columnwidth=\hsize@fornote%
2533   }%
2534
2535 \newcommand{\setnotesXwidthliketwocolumns@}[1]{%
2536   \global\let\hsize@fornote=\hsize%
2537   \global\old@hsize=\linewidth%
2538   \iftoggle{notesXwidthliketwocolumns@#1}{%
2539     {%
2540       \csuse{setwidthliketwocolumns@\columns@position}%
2541       \global\let\hsize@fornote=\hsize%
2542     }%
2543     {%
2544       \let\hsize=\hsize@fornote%
2545       \let\columnwidth=\hsize@fornote%
2546     }%
2547

```

`\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`.

```

2548 \newcommand{\setXnotespositionliketwocolumns@}[1]{%
2549   \iftoggle{Xnoteswidthliketwocolumns@#1}{%
2550     \csuse{setnotespositionliketwocolumns@\columns@position}%
2551   }{}%
2552 }%
2553
2554 \newcommand{\setnotesXpositionliketwocolumns@}[1]{%
2555   \iftoggle{notesXwidthliketwocolumns@#1}{%
2556     \csuse{setnotespositionliketwocolumns@\columns@position}%
2557   }{}%
2558 }%
2559

```

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 2560 \def\@fnpos{familiar-critical}
\@mpfnpos 2561 \def\@mpfnpos{critical-familiar}

```

```

2562 \newcommand{\fnpos}[1]{\xdef\@fnpos{#1}}
2563 \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
2564 \newcommand{\print@Xfootnoterule}[1]{%
2565   \nointerlineskip%
2566   \moveleft-\leftskip\vbox{\csuse{#1footnoterule}}%
2567   \nointerlineskip%
2568 }%
2569
2570 \newcommand{\print@footnoteXrule}[1]{%
2571   \nointerlineskip%
2572   \moveleft-\leftskip\vbox{\csuse{footnoterule#1}}%
2573   \nointerlineskip%
2574 }%
2575

```

29 Footnotes' output

`\doxtrafeeti` We have to add all the new kinds of familiar footnotes to the output routine.
`\doreinextrafeeti` These are the class 1 feet.

```

2576 \newcommand*\doxtrafeeti{%
2577   \setbox\@outputbox \vbox{%
2578     \unvbox\@outputbox
2579     \def\do##1{\ifvoid\csuse{footins##1}\else\csuse{footstart##1}{##1}\csuse{footgroup##1}{##1}\fi}
2580     \dolistloop{\@series}%
2581   }}
2582
2583 \newcommand{\doreinextrafeeti}{%
2584   \def\do##1{\ifvoid\csuse{footins##1}\else\insert\csuse{footins##1}{\unvbox\csuse{footins##1}}\fi}%
2585   \dolistloop{\@series}%
2586 }
2587

```

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

```

2588 \newcommand*\addfootinsX[1]{%
2589   \led@warn@AddfootinsX@obsolete%
2590   \footnormalX{#1}%
2591   \g@addto@macro{\doxtrafeeti}{%
2592     \setbox\@outputbox \vbox{%
2593       \unvbox\@outputbox
2594       \ifvoid\@nameuse{footins#1}\else

```

```

2595      \@nameuse{footstart#1}{#1}\@nameuse{footgroup#1}{#1}\fi}}%as
2596      \g@addto@macro{\doreinextrafeeti}{%
2597          \ifvoid\@nameuse{footins#1}\else
2598              \insert\@nameuse{footins#1}{\unvbox\@nameuse{footins#1}}\fi}%
2599      \g@addto@macro{\l@dfambeginmini}{%
2600          \expandafter\expandafter\expandafter\let\expandafter\expandafter
2601              \csname footnote#1\endcsname \csname mpfootnote#1\endcsname}%
2602      \g@addto@macro{\l@dfamendmini}{%
2603          \ifvoid\@nameuse{mpfootins#1}\else\@nameuse{mpfootgroup#1}{#1}\fi}%
2604  }

```

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` 2605 `\newwrite\l@d@end`
 2606 `\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.
 2607 `\newcommand{\l@dend@open}[1]{\global\l@dend@true\immediate\openout\l@d@end=#1\relax}`
 2608 `\newcommand{\l@dend@close}{\global\l@dend@false\immediate\closeout\l@d@end}`
 2609

`\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.
 2610 `\newcommand{\l@dend@stuff}{%`
 2611 `\ifl@dend@\relax\else`
 2612 `\l@dend@open{<jobname>.end}%`
 2613 `\fi`
 2614 `\immediate\write\l@d@end{\string\l@d@section{\the\section@num}}`
 2615

`\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.

```

2616 \global\notbool{parapparatus@}{\long}\def\endprint#1#2#3#4{\csuse{bhookXendnote@#4}\csu
2617     \enspace{\nottoggle{Xendlemmadisablefontselection@#4}{\select@lemmafnt#1|#2}{#2}}\
2618
2619 \let\l@d@section=\@gobble
2620

```

`\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.

```
2621 \newcommand*{\setprintendlines}[6]{%
2622   \l@dpnumfalse \l@ddashfalse
2623   \ifnum#4=#1 \else
2624     \l@dpnumtrue
2625     \l@ddashtrue
2626   \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.

```
2627   \ifl@dpnum \l@d@elintrue \else \l@d@elinfalse \fi
2628   \ifnum#2=#5 \else
2629     \l@d@elintrue
2630     \l@ddashtrue
2631   \fi
```

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

```
2632   \l@d@ssubfalse
2633   \ifnum#3=0 \else
2634     \l@d@ssubtrue
2635   \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.

```
2636   \l@d@eslfalse
2637   \ifnum#6=0 \else
2638     \ifnum#6=#3
2639       \ifl@d@elin \l@d@esltrue \else \l@d@eslfalse \fi
2640     \else
2641       \l@d@esltrue
2642       \l@ddashtrue
2643     \fi
2644   \fi}
```

`\printendlines` Now we're ready to print it all.

```
2645 \def\printendlines#1|#2|#3|#4|#5|#6|#7|{\begingroup
2646   \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).

```

2647 \printnpnum{#1} \linenumrep{#2}%
2648 \ifl@d@ssub \fullstop \sublinenumrep{#3}\fi
2649 \ifl@d@dash \endashchar\fi
2650 \ifl@d@pnum \printnpnum{#4}\fi
2651 \ifl@d@elin \linenumrep{#5}\fi
2652 \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumrep{#6}\fi
2653 \endgroup}
2654

```

\printnpnum A macro to print a page number in an endnote.

```

2655 \newcommand*{\printnpnum}[1]{p.#1} }
2656

```

\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.

```

2657 \newcommand*{\doendnotes}[1]{\l@dend@close
2658   \begingroup
2659     \makeatletter
2660     \expandafter\let\csname #1end\endcsname=\endprint
2661     \input\jobname.end
2662   \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.

```

2663 \newcommand*{\noendnotes}{\global\let\l@dend@stuff=\relax
2664   \global\chardef\l@dend=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@**.

```

2665 \newcommand{\newseries}[1]{%
2666   \def\do##1{\newseries@{##1}}%
2667   \docsvlist{#1}
2668 }

```

\@series The **\series@** macro is an etoolbox list, which contains the name of all series.

```

2669 \newcommand{\@series}{}

```

The command **\newseries@\series** creates a new series of the footnote.

\newseries@

```

2670 \newcommand{\newseries@}[1]{

```

31.1 Test if series is still existing

```
2671 \xifinlist{#1}{\@series}{\led@warn@SeriesStillExist{#1}}%
2672 {%
```

31.2 Create all commands to memorize display options

```
2673 \newtoggle{Xlemmadisablefontselection@#1}
2674 \newtoggle{Xendlemmadisablefontselection@#1}
2675 \csgdef{Xhangindent@#1}{0pt}%
2676 \csgdef{hangindentX@#1}{0pt}%
2677 \csgdef{Xragged@#1}{}%
2678 \csgdef{raggedX@#1}{}%
2679 \csgdef{hsizetwocol@#1}{0.45 \hsize}%
2680 \csgdef{hsizetwocolX@#1}{0.45 \hsize}%
2681 \csgdef{hsizethreecol@#1}{.3 \hsize}%
2682 \csgdef{hsizethreecolX@#1}{.3 \hsize}%
2683 \csgdef{Xnotenumfont@#1}{\notenumfont}%
2684 \csgdef{Xendnotenumfont@#1}{\notenumfont}%
2685 \csgdef{notenumfontX@#1}{\notenumfont}%
2686 \csgdef{Xnotefontsize@#1}{\notefontsetup}%
2687 \csgdef{notefontsizeX@#1}{\notefontsetup}%
2688 \csgdef{Xendnotefontsize@#1}{\notefontsetup}%
2689 \csgdef{bhooknoteX@#1}{}%
2690 \csgdef{bhookXnote@#1}{}%
2691 \csgdef{bhookXendnote@#1}{}%
2692 \csgdef{boxlinenum@#1}{0pt}%
2693 \csgdef{boxsymlinenum@#1}{0pt}%
2694 \newtoggle{numberonlyfirstinline@#1}%
2695 \newtoggle{numberonlyfirstintwolines@#1}%
2696 \newtoggle{onlypstartinfootnote@#1}%
2697 \newtoggle{pstartinfootnoteeverytime@#1}%
2698 \newtoggle{pstartinfootnote@#1}%
2699 \csgdef{symlinenum@#1}{\symlinenum}%
2700 \newtoggle{nonnumberinfootnote@#1}%
2701 \csgdef{beforenumberinfootnote@#1}{0pt}%
2702 \csgdef{afternumberinfootnote@#1}{0.5em}%
2703 \newtoggle{nonbreakableafternumber@#1}%
2704 \csgdef{beforesymlinenum@#1}{\csuse{beforenumberinfootnote@#1}}%
2705 \csgdef{aftersymlinenum@#1}{\csuse{afternumberinfootnote@#1}}%
2706 \csgdef{inplaceofnumber@#1}{1em}%
2707 \global\cslet{lemmaseparator@#1}{\rbracket}%
2708 \csgdef{beforelemmaseparator@#1}{0em}%
2709 \csgdef{afterlemmaseparator@#1}{0.5em}%
2710 \csgdef{inplaceoflemmaseparator@#1}{1em}%
2711 \csgdef{afternote@#1}{1em plus.4em minus.4em}%
2712 \csgdef{parafootsep@#1}{\parafootftmsep}%
2713 \csgdef{beforeXnotes@#1}{1.2em \@plus .6em \@minus .6em}%
2714 \csgdef{beforeXnotesX@#1}{1.2em \@plus .6em \@minus .6em}%
2715 \csgdef{afterXrule@#1}{0pt}%
2716 \csgdef{afterruleX@#1}{0pt}
```

```

2717 \csgdef{txtbeforeXnotes@#1}{%
2718 \csgdef{maxhnotesX@#1}{\ledfootinsdim}%
2719 \csgdef{maxhXnotes@#1}{\ledfootinsdim}
2720 \newtoggle{Xnoteswidthliketwocolumns@#1}%
2721 \newtoggle{notesXwidthliketwocolumns@#1}%

```

31.3 Create inserts, needed to add notes in foot

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

```

2722
2723 \expandafter\newinsert\csname mpfootins#1\endcsname
2724 \expandafter\newinsert\csname footins#1\endcsname
2725 \expandafter\newinsert\csname #1footins\endcsname
2726 \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.

```

2727
2728 \global\newcommand{\parapparatus@}{\expandafter\newcommand\expandafter *}{\expandafter\new
2729 \beginingroup%
2730 \newcommand{\content}{##2}%
2731 \ifnumberedpar@
2732 \ifledRcol%
2733 \ifluatex%
2734 \footnotelang@lua[R]%
2735 \fi%
2736 \@ifundefined{xpg@main@language}{%if polyglossia
2737 {}%
2738 {\footnotelang@poly[R]}%
2739 \footnoteoptions@[R]{##1}{true}%
2740 \xright@appenditem{\noexpand\prepare@edindex@fornote{\l@d@nums}%
2741 \noexpand\csuse{v#1footnote}{#1}%
2742 {\l@d@nums}{\expandonce\@tag}{\expandonce\content}}}\to\inserts@listR
2743 \footnoteoptions@[R]{##1}{false}%
2744 \global\advance\insert@countR \@ne%
2745 \else%
2746 \ifluatex%
2747 \footnotelang@lua%
2748 \fi%
2749 \@ifundefined{xpg@main@language}{%if polyglossia
2750 {}%
2751 {\footnotelang@poly}%
2752 \footnoteoptions@{##1}{true}%
2753 \xright@appenditem{\noexpand\prepare@edindex@fornote{\l@d@nums}%
2754 \noexpand\csuse{v#1footnote}{#1}%
2755 {\l@d@nums}{\expandonce\@tag}{\expandonce\content}}}\to\inserts@list
2756 \global\advance\insert@count \@ne%
2757 \footnoteoptions@{##1}{false}%

```



```

2758         \fi
2759     \else
2760         \csuse{v#1footnote}{#1}{\{0|0|0|0|0|0|0\}{\{##1\}}%
2761     \fi%
2762     \ignorespaces%
2763     \endgroup
2764 }

```

Set standard display and remember the display.

```

2765 \csgdef{series@display#1}{\}
2766 \footnormal{#1}

```

31.5 Create tools for familiar footnotes (`\footnoteX`)

First, create the `\footnoteX` command.

```

2767
2768 \global\expandafter\newcommand\csname footnote#1\endcsname[1]{%
2769     \begingroup%
2770     \newcommand{\content}{##1}%
2771     \stepcounter{footnote#1}%
2772     \protected@csxdef{\thefnmark#1}{\csuse{thefootnote#1}}%
2773     \csuse{\footnotemark#1}%
2774     \csuse{vfootnote#1}{#1}{\expandonce\content}\m@mmf@prepare%
2775     \endgroup%
2776 }

```

The counters.

```

2777 \newcounter{footnote#1}
2778 \global\expandafter\renewcommand\csname thefootnote#1\endcsname{\arabic{footnote#1}}
2779 % \end{macrocode}
2780 % Don't forget to initialize series
2781 % \begin{macrocode}
2782 \csgdef{series@displayX#1}{\}
2783 \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.

```

2784
2785 \global\expandafter\newcommand\csname #1endnote\endcsname[2]{\{\newlinechar='40
2786     \global\@noneed@Footnotetrue%
2787     \newcommand{\content}{##1}%
2788     \immediate\write\l@d@end{\expandafter\string\csname #1end\endcsname%
2789     {\ifnumberedpar@\l@d@nums\fi}%
2790     {\ifnumberedpar@\expandonce\tag\fi}{\expandonce\content}{#1}}\ignorespaces%
2791 }

```

`\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

```

2792
2793   \global\cslet{#1end}{\@gobblefour}
2794 %\end{macrocode}
2795 % We need to be able to modify \Eledmac's footnote macros and restore their
2796
2797   \global\csletcs{#1@footnote}{#1footnote}
2798 % \cs{Stock series in \cs{@series}
2799 %   \begin{macrocode}
2800
2801   \listxadd{\@series}{#1}
2802 }
2803 }% End of \newseries

```

31.7 Init standards series (A,B,C,D,E,Z)

```
2804 \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.

```

2805 \newcommand{\seriesatbegin}[1]{
2806   \edef\series{#1}
2807   \def\new{}
2808   \listead{\new}{\series}
2809   \def\do##1{\ifcsstring{series}{##1}{-}{\listadd{\new}{##1}}}
2810   \dolistloop{\@series}
2811   \xdef\@series{\new}
2812 }

```

`\seriesatend` And `\seriesatend` moves the series to the end of the list.

```

2813 \newcommand{\seriesatend}[1]{
2814   \edef\series{#1}
2815   \def\new{}
2816   \def\do##1{\ifcsstring{series}{##1}{-}{\listadd{\new}{##1}}}
2817   \dolistloop{\@series}
2818   \listead{\new}{\series}
2819   \xdef\@series{\new}
2820 }
2821 % \end{macrocode}
2822 % \end{macro}

```

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.

```

2823 % \subsection{Display}
2824 % \changes{v1.0}{2012/09/15}{New generic commands to customize footnote display.}
2825 % \subsubsection{Options}
2826 % \begin{macro}{\settoggle@series}
2827 % \changes{v1.1}{2012/09/25}{\cs{settoggle@series} switch the global value of the toggle, not only th
2828 % \changes{v1.13.0}{2014/09/16}{\cs{settoggle@series} can take an optional arguments to reload series
2829 % \cs{settoggle@series}\marg{series}\marg{toggle}\marg{value} is a generic command to switch toggles
2830
2831 % \begin{macrocode}
2832 \newcommandx{\settoggle@series}[4][4]{%
2833   \def\do##1{%
2834     \global\settoggle{#2@##1}{#3}%
2835     \ifstrequal{#4}{reload}%
2836       {%
2837         \csuse{foot\csuse{series@display##1}}{##1}%
2838         \csuse{foot\csuse{series@displayX##1}}{##1}%
2839       }%
2840     }%
2841   }%
2842   \ifstreempty{#1}{%
2843     \dolistloop{\@series}%
2844   }%
2845   {%
2846     \docsvlist{#1}%
2847   }%
2848 }

```

`\setcommand@series` `\setcommand@series{<series>}{<command>}{<value>}` is a generic command to change commands for some series.

```

2849 \newcommandx{\setcommand@series}[4][4]{%
2850   \def\do##1{
2851     \csgdef{#2@##1}{#3}
2852     \ifstrequal{#4}{reload}{
2853       \csuse{foot\csuse{series@display##1}}{##1}
2854       \csuse{foot\csuse{series@displayX##1}}{##1}
2855     }{}
2856     \ifstreempty{#1}{%
2857       \dolistloop{\@series}%
2858     }%
2859     {%
2860       \docsvlist{#1}%
2861     }%
2862 }%

```

`\newhookcommand@series` `\newhookcommand@series\command` names is a generic command to add new commands for hooks, like `\hsizetwocol`.

```

2863 \newcommand{\newhookcommand@series}[1]{%
2864   \global\expandafter\newcommand\expandafter*\csname #1\endcsname[2][2]{%
2865     \setcommand@series{##1}{#1}{##2}%

```

```

2866 }%
2867 }
2868 \newhookcommand@series{Xhangindent}
2869
2870 \newhookcommand@series{hangindentX}
2871
2872 \newhookcommand@series{Xragged}
2873
2874 \newhookcommand@series{raggedX}
2875
2876 \newhookcommand@series{hsizetwocol}
2877
2878 \newhookcommand@series{hsizethreecol}
2879
2880 \newhookcommand@series{hsizetwocolX}
2881
2882 \newhookcommand@series{hsizethreecolX}
2883
2884 \newhookcommand@series{Xnotenumfont}
2885
2886 \newhookcommand@series{notenumfontX}
2887
2888 \newhookcommand@series{Xendnotenumfont}
2889
2890 \newhookcommand@series{bhooknoteX}
2891
2892 \newhookcommand@series{bhookXnote}
2893
2894 \newhookcommand@series{bhookXendnote}
2895
2896 \newhookcommand@series{Xnotefontsize}
2897
2898 \newhookcommand@series{notefontsizeX}
2899
2900 \newhookcommand@series{Xendnotefontsize}
2901
2902 \newhookcommand@series{boxlinenum}
2903
2904 \newhookcommand@series{boxsymlinenum}
2905
2906 \newhookcommand@series{parafootsep}
2907
2908 \newhookcommand@series{symlinenum}
2909
2910 \newhookcommand@series{beforenumberinfootnote}
2911
2912 \newhookcommand@series{afternumberinfootnote}
2913
2914 \newhookcommand@series{beforesymlinenum}
2915

```

```

2916 \newhookcommand@series{aftersymmlinenum}
2917
2918 \newhookcommand@series{inplaceofnumber}
2919
2920 \newhookcommand@series{lemmaseparator}
2921
2922 \newhookcommand@series{beforelemmaseparator}
2923
2924 \newhookcommand@series{afterlemmaseparator}
2925
2926 \newhookcommand@series{inplaceoflemmaseparator}
2927
2928 \newhookcommand@series{afternote}
2929
2930 \newhookcommand@series{txtbeforeXnotes}
2931
2932 \newhookcommand@series{afterruleX}
2933
2934 \newhookcommand@series{afterXrule}
2935

```

`\newhookcommand@series@reload` `\newhookcommand@series@reload` does the same thing as `\newhookcommand@series` but the commands created by this macro also reload the series displaying (normal, paragraph, twocol, threecol).

```

2936 \newcommand{\newhookcommand@series@reload}[1]{%
2937   \global\expandafter\newcommand\expandafter*\csname #1\endcsname[2][1]{%
2938     \setcommand@series{##1}{#1}{##2}[reload]%
2939   }%
2940 }
2941 \newhookcommand@series@reload{beforeXnotes}
2942
2943 \newhookcommand@series@reload{beforenotesX}
2944
2945 \newhookcommand@series@reload{maxhnotesX}
2946
2947 \newhookcommand@series@reload{maxhXnotes}
2948 % \end{macrocode}
2949 % \end{macro}
2950 % \begin{macro}{\newhooktoggle@series}
2951 %\cs{newhooktoggle@series}\cs{command names} is a generic command to add new commands for new toggle
2952 %   \begin{macrocode}
2953 \newcommand{\newhooktoggle@series}[1]{%
2954   \global\expandafter\newcommand\expandafter*\csname #1\endcsname[2][1,2={true},usedefault]{%
2955     \settoggle@series{##1}{#1}{##2}%
2956   }%
2957 }
2958 \newhooktoggle@series{numberonlyfirstinline}
2959 \newhooktoggle@series{numberonlyfirstintwolines}
2960 \newhooktoggle@series{nonumberinfootnote}
2961 \newhooktoggle@series{pstartinfootnote}

```

```

2962 \newhooktoggle@series{pstartinfootnoteeverytime}%
2963 \newhooktoggle@series{onlypstartinfootnote}
2964 \newhooktoggle@series{nonbreakableafternumber}
2965 \newhooktoggle@series{Xlemmadisablefontselection}
2966 \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).

```

2967 \newcommand{\newhooktoggle@series@reload}[1]{%
2968   \global\expandafter\newcommandx\expandafter*\csname #1\endcsname[2][1,2={true},usedefault]
2969   \settoggle@series{##1}{#1}{##2}[reload]%
2970   }%
2971 }%
2972
2973 \newhooktoggle@series@reload{Xnoteswidthliketwocolumns}%
2974 \newhooktoggle@series@reload{notesXwidthliketwocolumns}%
2975

```

31.9 Old commands, kept for backward compatibility

The next commands are kept for ascendant compatibility, but should't be used anymore.

```

\notenumfont
\notefontsetup 2976 \newcommand*{\notenumfont}{\normalfont}
\ifledplinenum 2977 \newcommand*{\notefontsetup}{\footnotesize}
\symplinenum 2978 \newif\ifledplinenum
2979 \ledplinenumtrue
2980 \newcommand*{\symplinenum}{\ifledplinenum}

```

31.10 Hooks for a particular footnote

`\nonum@` `\nonum@` toggle is used to disable line number printing in a particular footnote.

```
2981 \newtoggle{nonum@}
```

`\nosep@` `\nonum@` toggle is used to disable the lemma separator in a particular footnote.

```
2982 \newtoggle{nosep@}
```

31.11 Alias

`\nolemmaseparator` `\nolemmaseparator[series]` is just an alias for `\lemmaseparator[series]{}`.

```
2983 \newcommandx*\nolemmaseparator[1][1]{\lemmaseparator[#1]{}}
```

`\interparanoteglue` The `\ipn@skip` skip and `\interparanoteglue` command are kept for backward compatibility, but should not be used anymore.

`\ipn@skip`

```

2984 \newskip\ipn@skip
2985 \newcommand*\interparanoteglue[1]{%

```

```

2986          {\notefontsetup\global\ipn@skip=#1 \relax}}
2987 \interparanoteglue{1em plus.4em minus.4em}

```

`\parafootftmsep` The `\parafootftmsep` macro is kept for backward compatibility. It is default value of `\parafootsep@series`.

```

2988 \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. The printing of the line number is shared in `\printlinefootnotenumbers`.

```

2989 \newcommand{\printlinefootnote}[2]{%
2990   \def\extractline@##1|##2|##3|##4|##5|##6|##7|{##2}%
2991   \def\extractsubline@##1|##2|##3|##4|##5|##6|##7|{##3}%
2992   \def\extractendline@##1|##2|##3|##4|##5|##6|##7|{##5}%
2993   \def\extractendsubline@##1|##2|##3|##4|##5|##6|##7|{##6}%
2994   \iftoggle{numberonlyfirstintwelines@#2}{%
2995     \edef\lineinfo@{\extractline@ #1| - \extractsubline@ #1| - \extractendline@ #1| - \extractends
2996     }%
2997     {%
2998       \edef\lineinfo@{\extractline@ #1| - \extractsubline@ #1|}%
2999     }%
3000   \iftoggle{nonum@}{%Try if the line number must printed for this specific not (by default, yes)
3001     \hspace{\csuse{inplaceofnumber@#2}}}%
3002   }%
3003   {%
3004     {%
3005       \iftoggle{nonumberinfootnote@#2}{%Try if the line number must printed (by default, yes)
3006       }%
3007       \hspace{\csuse{inplaceofnumber@#2}}}%
3008     }%
3009     {%
3010       {\iftoggle{numberonlyfirstinline@#2}% If for this series the line number must be printed
3011       }%
3012       \ifcsdef{prevline#2}%
3013       {%Be sure the \prevline exists.
3014       \ifcsequal{prevline#2}{\lineinfo@}%Try it
3015       }%
3016       \ifcsequal{symlinenum@#2}% Try if a symbol is define
3017       {%
3018         \hspace{\csuse{inplaceofnumber@#2}}}%
3019       }%
3020       {\hspace{\csuse{beforesymlinenum@#2}}\csuse{Xnotenumfont@#2}%
3021       \ifdimequal{\csuse{boxsymlinenum@#2}}{0pt}%
3022       {\csuse{symlinenum@#2}}%
3023       {\hbox to \csuse{boxsymlinenum@#2}{\csuse{symlinenum@#2}\hfill}}%

```

```

3024             \hspace{\csuse{aftersymlinenum@#2}}}%
3025         }%
3026         {%
3027             \printlinefootnotearea{#1}{#2}%
3028         }%
3029     }%
3030     {%
3031         \printlinefootnotearea{#1}{#2}%
3032     }%
3033 }%
3034 {%
3035     \printlinefootnotearea{#1}{#2}%
3036 }%
3037 \csxdef{prevline#2}{\lineinfo@}%
3038 }%
3039 }%
3040 }%
3041 }%
3042 }

```

`\printlinefootnotearea` This macro prints the space before the line number, changes the font, then prints the line number and the space after it. It is called by `\printlinefootnote` depending of the options about repeating line numbers. The first argument is line information, the second is the notes series (A, B, C. etc.)

```

3043 \newcommand{\printlinefootnotearea}[2]{%
3044     \printbeforenumberinfootnote{#2}%
3045     \csuse{Xnotenumfont@#2}%
3046     \boxfootnotenumbers{#1}{#2}%
3047     \printafternumberinfootnote{#2}%
3048 }%
3049 %     \end{macrocode}
3050 % \end{macro}
3051 % \begin{macro}{\boxfootnotenumbers}
3052 % Depending on the user settings, this macro will box line numbers (or not).
3053 % The first argument is line information, the second is the notes series (A, B, C. etc.).
3054 % The previous \cs{printlinefootnotearea} calls it.
3055 %     \begin{macrocode}
3056 \newcommand{\boxfootnotenumbers}[2]{%
3057     \ifdimequal{\csuse{boxlinenum@#2}}{0pt}{%
3058         \printlinefootnotenumbers{#1}{#2}%
3059     }%
3060     {%
3061         \hbox to \csuse{boxlinenum@#2}{%
3062             \printlinefootnotenumbers{#1}{#2}%
3063             \hfill}%
3064     }%
3065 }%
3066 %     \end{macrocode}
3067 % \end{macro}
3068 % \begin{macro}{\printlinefootnotenumbers}

```



```

3069 % This macro prints, if needed, the pstart number and the line number.
3070 % The first argument is line information, the second is the notes series (A, B, C. etc.)
3071 % The previous \cs{boxlinefootnote} calls it.
3072 % \begin{macrocode}
3073 \newcommand{\printlinefootnotenumbers}[2]{%
3074   \ifboolexpr{%
3075     (togl{pstartinfofootnote@#2} and bool{numberpstart}})%
3076     or togl{pstartinfofootnoteeverytime@#2}}}%
3077   {\printpstart}{}%
3078   \iftoggle{onlypstartinfofootnote@#2}{\printlines#1|}%
3079 }%

```

`\printbeforenumberinfofootnote` This macro prints a space (before the line number) in footnote. It is called by `\printlinefootnotearea`. Its only argument is the series

```

3080 \newcommand{\printbeforenumberinfofootnote}[1]{%
3081   \hspace{\csuse{beforenumberinfofootnote@#1}}%
3082 }%

```

`\printafternumberinfofootnote` This macro prints the space, adding eventually a `\nobreak`, after the line number, in footnote. It is called by `\printlinefootnotearea`. Its only argument is the series

```

3083 \newcommand{\printafternumberinfofootnote}[1]{%
3084   \iftoggle{nonbreakableafternumber@#1}{\nobreak}{}%
3085   \hspace{\csuse{afternumberinfofootnote@#1}}%
3086 }%

```

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.

```

3087 \countdef\pageno=0 \pageno=1
3088 \newcommand*{\advancepageno}{\ifnum\pageno<\z@ \global\advance\pageno\m@ne
3089   \else\global\advance\pageno\@ne\fi}
3090

```

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 T_EX; 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
\ifrggedbottom \kern-\dimen@ \vfil \fi}

```

\do@feet ships out all the footnotes. Standard EDMAC has only five feet, but there is nothing in principal to prevent you from creating an arachnoid or centipedal edition; straightforward modifications of EDMAC are all that's required. However, the myriapedal edition is ruled out by eTeX limitations: the number of insertion classes is limited to 2^{16} .

With luck we might only have to change \@makecol and \@reinserts. The kernel definition of these, and perhaps some other things, is:

```

\gdef \@makecol {%
\ifvoid\footins
\setbox\@outputbox \box\@cclv
\else
\setbox\@outputbox \vbox {%
\boxmaxdepth \@maxdepth
\@tempdima\dp\@cclv
\unvbox \@cclv
\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 3091 \providecommand{\m@m@makecolfloats}{%
3092   \xdef\@freelist{\@freelist\@midlist}%
3093   \global \let \@midlist \@empty
3094   \@combinefloats}
3095 \providecommand{\m@m@makecoltext}{%
3096   \ifvbox\@kludgeins
3097     \@makespecialcolbox
3098   \else
3099     \setbox\@outputbox \vbox to\@colht {%
3100       \@texttop
3101       \dimen@ \dp\@outputbox
3102       \unvbox\@outputbox
3103       \vskip -\dimen@
3104       \@textbottom}%
3105   \fi}
3106 \providecommand{\m@m@makecolintro}{%
3107

```

`\l@d@makecol` This is a partitioned version of the ‘standard’ `\@makecol`, with the initial code put into another macro.

```

3108 \gdef\l@d@makecol{%
3109   \l@ddofootinsert
3110   \m@m@makecolfloats
3111   \m@m@makecoltext
3112   \global \maxdepth \@maxdepth}
3113

```

`\ifFN@bottom` The `\ifFN@bottom` macro is defined by the `footmisc` package. If this package is not loaded, we define it.

```

3114 \AtBeginDocument{\@ifpackageloaded{footmisc}{\newif\ifFN@bottom}}

```

`\l@ddofootinsert` This macro essentially holds the initial portion of the kernel `\@makecol` code.

```

3115 \newcommand*\l@ddofootinsert{%
3116   %% \page@start
3117   \ifvoid\footins

```

```

3118     \setbox\@outputbox \box\@cclv
3119   \else
3120     \setbox\@outputbox \vbox {%
3121       \boxmaxdepth \@maxdepth
3122       \@tempdima\dp\@cclv
3123       \unvbox \@cclv
3124       \ifFN@bottom\vfill\fi\vskip \skip\footins%% If the option bottom of loadmisc packa
3125       \color@begingroup
3126       \normalcolor
3127       \footnoterule
3128       \unvbox \footins
3129       \color@endgroup
3130     }%
3131   \fi

```

That's the end of the copy of the kernel code. We finally call a macro to handle all the additional EDMAC feet.

```

3132   \l@ddoxtrafeet
3133 }
3134

```

`\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).

```

3135 \newcommand*{\l@ddoxtrafeet}{%
3136   \IfStrEq{familiar-critical}{\@fnpos}
3137   {\doxtrafeeti\doxtrafeetii}%
3138   {%
3139     \IfStrEq{critical-familiar}{\@fnpos}%
3140     {\doxtrafeetii\doxtrafeeti}%
3141     {\doxtrafeeti\doxtrafeetii}%
3142   }%
3143 }%
3144

```

`\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'.

```

3145 \newcommand*{\doxtrafeetii}{%
3146   \setbox\@outputbox \vbox{%
3147     \unvbox\@outputbox
3148     \@opxtrafeetii}}

```

`\@opxtrafeetii` The extra critical feet to be aded to the output.

```

3149 \newcommand*{\@opxtrafeetii}{%
3150   \def\do##1{\ifvoid\csuse{##1footins}\else\csuse{##1footstart}{##1}\csuse{##1footgroup}{%
3151     \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`.

```
3152 \newcommand*{\l@ddodoreinextrafeet}{%
3153   \doreinextrafeeti
3154   \doreinextrafeetii}
3155
```

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

```
3156 \newcommand*{\doreinextrafeetii}{%
3157   \def\do##1{\ifvoid\csuse{##1footins}\else\insert\csuse{##1ootins}{\unvbox\csuse{##1footins}}\fi}
3158   \dolistloop{\@series}
3159 }
3160
```

`\l@d@reinserts` And here is the modified version of `\@reinserts`.

```
3161 \gdef \l@d@reinserts{%
3162   \ifvoid\footins\else\insert\footins{\unvbox\footins}\fi
3163   \l@ddodoreinextrafeet
3164   \ifvbox\@kludgeins\insert\@kludgeins{\unvbox\@kludgeins}\fi
3165 }
3166
```

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 memoir is used. (It can be awkward dealing with `\if` code within `\if` code, so don’t use `\ifl@dmemoir` here.)

```
3167 \ifclassloaded{memoir}{%
  memoir is loaded so we use memoir’s built in hooks.
3168   \g@addto@macro{\m@mdoextrafeet}{\l@ddoxtrafeet}%
3169   \g@addto@macro{\m@mdodoreinextrafeet}{\l@ddodoreinextrafeet}%
3170 }{%
  memoir has not been loaded, so redefine @makecol and @reinserts.
3171   \gdef\@makecol{\l@d@makecol}%
3172   \gdef\@reinserts{\l@d@reinserts}%
3173 }
3174
```

`\addfootins` `\addfootins` is for backward compatibility, but should’nt be used anymore.

```
3175 \newcommand*{\addfootins}[1]{%
3176   \led@warn@AddfootinsObsolete%
3177   \footnormal{#1}
3178   \g@addto@macro{\opxtrafeetii}{%
3179     \ifvoid\@nameuse{#1footins}\else
3180       \@nameuse{#1footstart{#1}}\@nameuse{#1footgroup}{#1}\fi}
3181   \g@addto@macro{\doreinextrafeetii}{%
```

```

3182 \ifvoid\@nameuse{#1footins}\else
3183 \insert\@nameuse{#1footins}{\unvbox\@nameuse{#1footins}}\fi}
3184 \g@addto@macro{\l@dedbeginmini}{%
3185 \expandafter\let\csname #1footnote\endcsname = \@nameuse{mp#1footnote}}
3186 \g@addto@macro{\l@dedendmini}{%
3187 \ifvoid\@nameuse{mp#1footins}\else\@nameuse{mpfootgroup#1{#1}}\fi}
3188 }

```

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.

```

3189 \newif\if@led@nofoot
3190 \newcommand*{\@led@extranofeet}{}
3191

```

```

3192 \ifclassloaded{memoir}{%

```

If the memoir class is loaded we hook into its modified `\@doclearpage`.

`\@mem@extranofeet`

```

3193 \g@addto@macro{\@mem@extranofeet}{%
3194 \def\do#1{\ifvoid\csuse{#1footins}\else\@mem@nofootfalse\fi%
3195 \ifvoid\csuse{footins#1}\else\@mem@nofootfalse\fi%
3196 }
3197 \dolistloop{\@series}%
3198 \@led@extranofeet}
3199 }{%

```

As memoir is not loaded we have to do it all here.

`\@led@testifnofoot`

```

\@doclearpage 3200 \newcommand*{\@led@testifnofoot}{%
3201 \if@led@nofoottrue
3202 \ifvoid\footins\else\@led@nofootfalse\fi
3203 \def\do##1{\ifvoid\csuse{##1footins}\else\@led@nofootfalse\fi%
3204 \ifvoid\csuse{footins##1}\else\@led@nofootfalse\fi}%
3205 \dolistloop{\@series}
3206 \@led@extranofeet}
3207
3208 \renewcommand{\@doclearpage}{%
3209 \@led@testifnofoot
3210 \if@led@nofoot
3211 \setbox\@tempboxa\vsplit\@cclv to\z@ \unvbox\@tempboxa
3212 \setbox\@tempboxa\box\@cclv
3213 \xdef\@deferlist{\@toplist\@botlist\@deferlist}%
3214 \global \let \@toplist \@empty
3215 \global \let \@botlist \@empty
3216 \global \@colroom \@colht
3217 \ifx \@currlist\@empty
3218 \else

```

```

3219     \latexerr{Float(s) lost}\@ehb
3220     \global \let \@currlist \@empty
3221   \fi
3222   \@makefcolumn\@deferlist
3223   \@whiles\if@fcolmade \fi{\@opcol\@makefcolumn\@deferlist}%
3224   \if@twocolumn
3225     \if@firstcolumn
3226       \xdef\@dbldeferlist{\@dbltoplist\@dbldeferlist}%
3227       \global \let \@dbltoplist \@empty
3228       \global \@colht \textheight
3229       \begingroup
3230         \dblfloatplacement
3231         \@makefcolumn\@dbldeferlist
3232         \@whiles\if@fcolmade \fi{\@outputpage
3233           \@makefcolumn\@dbldeferlist}%
3234       \endgroup
3235     \else
3236       \vbox{}\clearpage
3237     \fi
3238   \fi
3239 \else
3240   \setbox\@cclv\vbox{\box\@cclv\vfil}%
3241   \l@d@makecol\@opcol
3242   \clearpage
3243 \fi}
3244 }
3245

```

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 \edlineref commands will refer to the latest occurrence of \label{foo}.

\labelref@list Set up a new list, \labelref@list, to hold the page, line and sub-line numbers

for each label.

```
3246 \list@create{\labelref@list}
```

`\zz@@@` A convenience macro to zero two labeling counters in one go.

```
3247 %% \newcommand*\zz@@@{000|000|000} % set three counters to zero in one go
```

```
3248 \newcommand*\zz@@@{000|000} % set two counters to zero in one go
```

```
3249
```

`\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.

```
3250 \newcommand*\edlabel[1]{%
3251   \ifl@dpairing\ifautopar%
3252     \strut%
3253   \fi\fi%
3254   \@bsphack%
3255   \ifledRcol%
3256     \write\linenum@outR{\string\@lab}%
3257     \ifx\labelref@listR\empty%
3258       \xdef\label@refs{\zz@@@}%
3259     \else%
3260       \glp\labelref@listR\to\label@refs%
3261     \fi%
3262     \ifvmode%
3263       \advancelabel@refs%
3264     \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.

```
3265   \protected@write\@auxout{%
3266     {\string\l@dmake@labelsR\space\thepage|\label@refs|\the\c@pstartR|{#1}}%
3267   \ifdef\hypertarget{\hypertarget{#1}{}}{}%
3268   \else%
3269     \write\linenum@out{\string\@lab}%
3270     \ifx\labelref@listR\empty%
3271       \xdef\label@refs{\zz@@@}%
3272     \else%
```

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.


```

3273     \gl@p\labelref@list\to\label@refs%
3274     \fi%
3275     \ifvmode%
3276         \advancelabel@refs%
3277     \fi%
3278     \protected@write\@auxout{}%
3279     {\string\l@dmake@labels\space\thepage|\label@refs|\the\c@pstart|{#1}}%
3280     \ifdef{\hypertarget}{\hypertarget{#1}}{}{}%
3281     \fi%
3282 \@@esphack}%
3283

```

`\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. 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.

```

3284 \newcounter{line}%
3285 \newcounter{subline}%
3286 \newcommand{\advancelabel@refs}{%
3287     \setcounter{line}{\expandafter\labelrefsparseline\label@refs}%
3288     \stepcounter{line}%
3289     \ifsublines%
3290         \setcounter{subline}{\expandafter\labelrefsparsesubline\label@refs}%
3291         \stepcounter{subline}{1}%
3292         \def\label@refs{\theline|\thesubline}%
3293     \else%
3294         \def\label@refs{\theline|0}%
3295     \fi%
3296 }
3297 \def\labelrefsparseline#1|#2{#1}
3298 \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).

```

3299 \newcommand*{\l@dmake@labels}{%
3300 \def\l@dmake@labels#1|#2|#3|#4|#5{%
3301     \expandafter\ifx\csname the@label#5\endcsname \relax\else
3302         \led@warn@DuplicateLabel{#5}%
3303     \fi
3304     \expandafter\gdef\csname the@label#5\endcsname{#1|#2|#3|#4}%
3305     \ignorespaces}
3306

```

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.

```
3307 \AtBeginDocument{%
3308   \def\l@dmake@labels#1|#2|#3|#4|#5{%
3309 }
3310
```

`\@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`.

```
3311 \newcommand*{\@lab}{\xright@appenditem
3312   {\linenumrep{\line@num}}|}%
3313   \ifsublines@ \sublinenumrep{\subline@num}\else 0\fi}\to\labelref@list}
3314
```

`\wrap@edcrossref` `\wrap@edcrossref` is called around all `eledmac` crossref commands, except those which start with `x`. It adds the hyperlink.

```
3315 \newrobustcmd{\wrap@edcrossref}[2]{%
3316   \ifdef{\hyperlink}%
3317     {\hyperlink{#1}{#2}}%
3318     {#2}%
3319 }
```

`\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`.

```
3320 \newcommand*{\edpageref}[1]{\l@dref@undefined{#1}\wrap@edcrossref{#1}{\l@dgetref@num{1}{#1}}}%
3321 \newcommand*{\xpageref}[1]{\l@dgetref@num{1}{#1}}
3322
```

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

```
\lineref 3323 \newcommand*{\edlineref}[1]{\l@dref@undefined{#1}\wrap@edcrossref{#1}{\l@dgetref@num{2}{#1}}}%
\xlineref 3324 \AtBeginDocument{%
3325   \ifdef\lineref{\let\lineref\edlineref}%
3326 }%
3327 \newcommand*{\xlineref}[1]{\l@dgetref@num{2}{#1}}%
3328
```

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

```
\xsublineref 3329 \newcommand*{\sublineref}[1]{\l@dref@undefined{#1}\wrap@edcrossref{#1}{\l@dgetref@num{3}{#1}}}
3330 \newcommand*{\xsublineref}[1]{\l@dgetref@num{3}{#1}}
3331
```

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

```
\xpstarteref 3332 \newcommand*{\pstarteref}[1]{\l@dref@undefined{#1}\wrap@edcrossref{#1}{\l@dgetref@num{4}{#1}}}
3333 \newcommand*{\xpstarteref}[1]{\l@dgetref@num{4}{#1}}
3334
```

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.

```
3335 \newcommand*{\l@dref@undefined}[1]{%
3336   \expandafter\ifx\csname the@label#1\endcsname\relax
3337     \led@warn@RefUndefined{#1}%
3338   \fi}
3339
```

`\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.

```
3340 \newcommand*{\l@dgetref@num}[2]{%
3341   \expandafter
3342   \ifx\csname the@label#2\endcsname \relax
3343     000%
3344   \else
3345     \expandafter\expandafter\expandafter
3346     \l@dlabel@parse\csname the@label#2\endcsname|#1%
3347   \fi}
3348
```

`\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`.)

```
3349 \newcommand*{\l@dlabel@parse}{%
3350 \def\l@dlabel@parse#1|#2|#3|#4|#5{%
3351   \ifcase #5%
```

```

3352 \or #1%
3353 \or #2%
3354 \or #3%
3355 \or #4%
3356 \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.

```

3357 \newcommand*{\xxref}[2]{%
3358   {%
3359     \expandafter\ifx\csname the@label#1\endcsname \relax%
3360       \expandafter\let\csname the@@label#1\endcsname\zz@@%
3361     \else%
3362       \expandafter\def\csname the@@label#1\endcsname{\l@getref@num{1}{#1}|\l@getref@num{2}{#1}}
3363     \fi%
3364     \expandafter\ifx\csname the@label#2\endcsname \relax%
3365       \expandafter\let\csname the@@label#2\endcsname\zz@@%
3366     \else%
3367       \expandafter\def\csname the@@label#2\endcsname{\l@getref@num{1}{#2}|\l@getref@num{2}{#2}}
3368     \fi%
3369     \linenum{\csname the@@label#1\endcsname}%
3370     \csname the@@label#2\endcsname}}
3371

```

`\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`.

```

3372 \newcommand*{\edmakelabel}[2]{\expandafter\xdef\csname the@label#1\endcsname{#2}}
3373

```

(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 `\marginpar` 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 `\marginpar` in numbered text
`\@xympar` do not disturb the flow.

```
3374 \let\l@dold@xympar\@xympar
3375 \renewcommand{\@xympar}{%
3376   \ifnumberedpar@
3377     \led@warn@NoMarginpars
3378     \esphack
3379   \else
3380     \l@dold@xympar
3381   \fi}
3382
```

We provide side notes as replacement for `\marginpar` in numbered text.

`\sidenote@margin` These are the sidenote equivalents to `\line@margin` and `\linenummargin` for
`\sidenotemargin` specifying which margin. The default is the right margin (opposite to the default
`\l@dgetsidenote@margin` for line numbers). `\l@dgetsidenote@margin` returns the number associated to
 side note margin:

left : 0

right : 1

outer : 2

inner : 3

```
3383 \newcount\sidenote@margin
3384 \newcommand*{\sidenotemargin}[1]{%
3385   \l@dgetsidenote@margin{#1}%
3386   \ifnum\@l@tempcntb>\m@ne
3387     \ifledRcol
3388       \global\sidenote@marginR=\@l@tempcntb
3389     \else
3390       \global\sidenote@margin=\@l@tempcntb
3391     \fi
3392   \fi}}
3393 \newcommand*{\l@dgetsidenote@margin}[1]{%
3394   \def\@tempa{#1}\def\@tempb{left}%
3395   \ifx\@tempa\@tempb
3396     \@l@tempcntb \z@
3397   \else
3398     \def\@tempb{right}%
3399     \ifx\@tempa\@tempb
3400       \@l@tempcntb \@ne
```

```

3401 \else
3402 \def\@tempb{outer}%
3403 \ifx\@tempa\@tempb
3404 \l@dttempcntb \tw@
3405 \else
3406 \def\@tempb{inner}%
3407 \ifx\@tempa\@tempb
3408 \l@dttempcntb \thr@@
3409 \else
3410 \led@warn@BadSidenotemargin
3411 \l@dttempcntb \m@ne
3412 \fi
3413 \fi
3414 \fi
3415 \fi}
3416 \sidenotemargin{right}
3417

```

\l@dlp@rbox We need two boxes to store sidenote texts.

```

\l@drp@rbox 3418 \newbox\l@dlp@rbox
3419 \newbox\l@drp@rbox
3420

```

\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 3421 \newdimen\ledlsnotewidth \ledlsnotewidth=\marginparwidth
\ledrsnotesep 3422 \newdimen\ledrsnotewidth \ledrsnotewidth=\marginparwidth
\ledlsnotefontsetup 3423 \newdimen\ledlsnotesep \ledlsnotesep=\linenumsep
\ledrsnotefontsetup 3424 \newdimen\ledrsnotesep \ledrsnotesep=\linenumsep
3425 \newcommand*{\ledlsnotefontsetup}{\raggedleft\footnotesize}
3426 \newcommand*{\ledrsnotefontsetup}{\raggedright\footnotesize}
3427

```

\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 3428 \newcommand*{\ledleftnote}[1]{\edtext{\l@dlsnote{#1}}}
3429 \newcommand*{\ledrightnote}[1]{\edtext{\l@drsnote{#1}}}
3430
3431 \newcommand*{\ledinnernote}[1]{%
3432 \ifodd\c@page% Do not use \page@num, because it is not yet calculated when command is called
3433 \ledleftnote{#1}%
3434 \else%
3435 \ledrightnote{#1}%
3436 \fi%
3437 }
3438
3439 \newcommand*{\ledouternote}[1]{%

```

```

3440 \ifodd\c@page% Do not use \page@num, because it is not yet calculated when command is called
3441 \ledrightnote{#1}%
3442 \else%
3443 \ledleftnote{#1}%
3444 \fi%
3445 }
3446
3447 \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 3448 \newif\ifrightrightnoteup
3449 \rightnoteuptrue
3450
3451 \newcommand*{\l@dlsnote}[1]{%
3452 \begingroup%
3453 \newcommand{\content}{#1}%
3454 \ifnumberedpar@
3455 \ifledRcol%
3456 \xright@appenditem{\noexpand\l@dlsnote{\expandonce\content}}%
3457 \to\inserts@listR
3458 \global\advance\insert@countR \@ne%
3459 \else%
3460 \xright@appenditem{\noexpand\l@dlsnote{\expandonce\content}}%
3461 \to\inserts@list
3462 \global\advance\insert@count \@ne%
3463 \fi
3464 \fi\ignorespaces\endgroup}
3465
3466 \newcommand*{\l@drsnote}[1]{%
3467 \begingroup%
3468 \newcommand{\content}{#1}%
3469 \ifnumberedpar@
3470 \ifledRcol%
3471 \xright@appenditem{\noexpand\l@drsnote{\expandonce\content}}%
3472 \to\inserts@listR
3473 \global\advance\insert@countR \@ne%
3474 \else%
3475 \xright@appenditem{\noexpand\l@drsnote{\expandonce\content}}%
3476 \to\inserts@list
3477 \global\advance\insert@count \@ne%
3478 \fi
3479 \fi\ignorespaces\endgroup}
3480
3481 \newcommand*{\l@dcsnote}[1]{%
3482 \begingroup%
3483 \newcommand{\content}{#1}%
3484 \ifnumberedpar@
3485 \ifledRcol%
3486 \xright@appenditem{\noexpand\l@dcsnote{\expandonce\content}}%

```

```

3487             \to\inserts@listR
3488             \global\advance\insert@countR \@ne%
3489         \else%
3490             \xright@appenditem{\noexpand\l@dcsnote{\expandonce\content}}}%
3491             \to\inserts@list
3492             \global\advance\insert@count \@ne%
3493         \fi
3494     \fi\ignorespaces\endgroup}
3495

```

`\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.

```

3496 \newcommand*{\vl@dlsnote}[1]{%
3497     \ifledRcol%
3498         \@l@tempcntb=\sidenote@marginR%
3499         \ifnum\@l@tempcntb>\@ne%
3500             \advance\@l@tempcntb by\page@numR%
3501         \fi%
3502     \else%
3503         \@l@tempcntb=\sidenote@margin%
3504         \ifnum\@l@tempcntb>\@ne%
3505             \advance\@l@tempcntb by\page@num%
3506         \fi%
3507     \fi%
3508     \ifodd\@l@tempcntb%
3509         \listgadd{\l@dcsnotetext@l}{#1}%
3510     \else%
3511         \listgadd{\l@dcsnotetext}{#1}%
3512     \fi
3513 }
3514 \newcommand*{\vl@drsnote}[1]{%
3515     \ifledRcol%
3516         \@l@tempcntb=\sidenote@marginR%
3517         \ifnum\@l@tempcntb>\@ne%

```



```

3518     \advance\@l@dttempcntb by\page@numR%
3519     \fi%
3520   \else%
3521     \@l@dttempcntb=\sidenote@margin%
3522     \ifnum\@l@dttempcntb>\@ne%
3523       \advance\@l@dttempcntb by\page@num%
3524     \fi%
3525   \fi%
3526   \ifodd\@l@dttempcntb%
3527     \listgadd{\l@dcstetext}{#1}%
3528   \else%
3529     \listgadd{\l@dcstetext@r}{#1}%
3530   \fi%
3531 }
3532 \newcommand*{\vl@dcstetext}[1]{\listgadd{\l@dcstetext}{#1}}
3533

```

`\setl@dlp@rbox` `\setl@dlp@rbox{<lednums>}{<tag>}{<text>}` puts `<text>` into the `\l@dlp@rbox` box.
`\setl@drpr@rbox` And similarly for the right side box. It is these boxes that finally get displayed in the margins.

```

3534 \newcommand*{\setl@dlp@rbox}[1]{%
3535   {\parindent\z@\hspace=\ledl@notewidth\ledl@notefontsetup
3536     \global\setbox\l@dlp@rbox
3537     \ifleftnoteup
3538       =\vbox to\z@{\vss #1}%
3539     \else
3540       =\vbox to 0.70\baselineskip{\strut#1\vss}%
3541     \fi}}
3542 \newcommand*{\setl@drp@rbox}[1]{%
3543   {\parindent\z@\hspace=\ledr@notewidth\ledr@notefontsetup
3544     \global\setbox\l@drp@rbox
3545     \ifrightrightnoteup
3546       =\vbox to\z@{\vss#1}%
3547     \else
3548       =\vbox to0.7\baselineskip{\strut#1\vss}%
3549     \fi}}
3550 \newif\ifleftnoteup
3551 \leftnoteuptrue

```

`\sidenotesep` This macro is used to separate sidenotes of the same line.

```

3552 \newcommand{\sidenotesep}{, }

```

`\affixside@note` This macro puts any moveable sidenote text into the left or right sidenote box, depending on which margin it is meant to go in. It's a very much stripped down version of `\affixlin@num`.

Before do it, we concatenate all moveable sidenotes of the line, using `\sidenotesep` as separator. It's the result that we put on the sidenote.

```

3553 \newcommand*{\affixside@note}{%
3554   \def\sidenotecontent@{}%

```

```

3555 \numgdef{\itemcount@}{0}%
3556 \def\do##1{%
3557   \ifnumequal{\itemcount@}{0}%
3558     {%
3559       \appto\sidenotecontent@{##1}}% Not print not separator before the 1st note
3560     {\appto\sidenotecontent@{\sidenotesep ##1}%
3561     }%
3562     \numgdef{\itemcount@}{\itemcount@+1}%
3563   }%
3564   \dolistloop{\l@dcnotetext}%
3565   \ifnumgreater{\itemcount@}{1}{\led@err@ManySidenotes}{}%

```

And we do the same for left and right notes (not movable).

```

3566 \gdef\@templ@d{%
3567 \gdef\@templ@n{\l@dcnotetext\l@dcnotetext@1\l@dcnotetext@r}%
3568 \ifx\@templ@d\@templ@n \else%
3569   \if@twocolumn%
3570     \if@firstcolumn%
3571       \setl@dlp@rbox{##1}{\sidenotecontent@}%
3572     \else%
3573       \setl@drp@rbox{\sidenotecontent@}%
3574     \fi%
3575   \else%
3576     \l@dttempcntb=\sidenote@margin%
3577     \ifnum\l@dttempcntb>\@ne%
3578       \advance\l@dttempcntb by\page@num%
3579     \fi%
3580     \ifodd\l@dttempcntb%
3581       \setl@drp@rbox{\sidenotecontent@}%
3582       \gdef\sidenotecontent@{}%
3583       \numgdef{\itemcount@}{0}%
3584       \dolistloop{\l@dcnotetext@1}%
3585       \ifnumgreater{\itemcount@}{1}{\led@err@ManyLeftnotes}{}%
3586       \setl@dlp@rbox{\sidenotecontent@}%
3587     \else%
3588       \setl@dlp@rbox{\sidenotecontent@}%
3589       \gdef\sidenotecontent@{}%
3590       \numgdef{\itemcount@}{0}%
3591       \dolistloop{\l@dcnotetext@r}%
3592       \ifnumgreater{\itemcount@}{1}{\led@err@ManyRightnotes}{}%
3593       \setl@drp@rbox{\sidenotecontent@}%
3594     \fi%
3595   \fi%
3596 \fi%
3597 }

```

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@dfteetbeginmini` These will be the hooks in `\@iiminipage` and `\endminipage` They can be extended
`\l@dfteetendmini` to handle other things if necessary.

```

3598 \newcommand*{\l@dfteetbeginmini}{\l@dedbeginmini\l@dfambeginmini}
3599 \newcommand*{\l@dfteetendmini}{%
3600     \IfStrEq{critical-familiar}{\@mpfnpos}%
3601         {\l@dedendmini\l@dfamendmini}%
3602     {%
3603         \IfStrEq{familiar-critical}{\@mpfnpos}%
3604             {\l@dfamendmini\l@dedendmini}%
3605             {\l@dedendmini\l@dfamendmini}%
3606     }%
3607 }%
```

`\l@dedbeginmini` These handle the initiation and closure of critical footnotes in a minipage envi-
`\l@dedendmini` ronment.

```

3608 \newcommand*{\l@dedbeginmini}{%
3609     \def\do##1{\csletcs{v##1footnote}{mpv##1footnote}}%
3610     \dolistloop{\@series}%
3611 }
3612 \newcommand*{\l@dedendmini}{%
3613     \ifl@dpairing
3614         \ifledRcol
3615             \flush@notesR
3616         \else
3617             \flush@notes
3618         \fi
3619     \fi
3620     \def\do##1{
3621         \ifvoid\csuse{mp##1footins}\else%
3622             \ifl@dpairing\ifparledgroup%
3623                 \ifledRcol%
3624                     \dimgdef{\parledgroup@beforenotesR}{\parledgroup@beforenotesR+\skip\@nameuse{mp##1footins}}
3625                 \else%
3626                     \dimgdef{\parledgroup@beforenotesL}{\parledgroup@beforenotesL+\skip\@nameuse{mp##1footins}}
3627                 \fi%
3628             \fi\fi%
3629             \csuse{mp##1footgroup}{##1}%
3630         \fi}%
3631     \dolistloop{\@series}%
3632 }
3633
```

`\l@dfambeginmini` These handle the initiation and closure of familiar footnotes in a minipage environment.
`\l@dfamendmini`

```

3634 \newcommand*{\l@dfambeginmini}{%
3635   \def\do##1{\csletcs{vfootnote##1}{mpvfootnote##1}}%
3636   \dolistloop{\@series}}
3637 \newcommand*{\l@dfamendmini}{%
3638   \def\do##1{\ifvoid\csuse{mpfootins##1}\else\csuse{mpfootgroup##1}{##1}\fi}%
3639   \dolistloop{\@series}}
```

`\@iiiminipage` This is our extended form of the kernel `\@iiiminipage` defined in `ltboxes.dtx`.

```

3640 \def\@iiiminipage#1#2[#3]#4{%
3641   \leavevmode
3642   \@pboxswfalse
3643   \setlength\@tempdima{#4}%
3644   \def\@mpargs{{#1}{#2}[#3]{#4}}%
3645   \setbox\@tempboxa\vbox\bgroup
3646     \color@begingroup
3647     \hsize\@tempdima
3648     \textwidth\hsize \columnwidth\hsize
3649     \@parboxrestore
3650     \def\@mpfn{mpfootnote}\def\thempfn{\thempfootnote}\c@mpfootnote\z@
3651     \let\@footnotetext\@mpfootnotetext
```

The next line is our addition to the original.

```

3652     \l@dfeetbeginmini%          added
3653     \let\@listdepth\@mplistdepth \@mplistdepth\z@
3654     \@minipagerestore
3655     \@setminipage}
3656
```

`\endminipage` This is our extended form of the kernel `\endminipage` defined in `ltboxes.dtx`.

```

3657 \def\endminipage{%
3658   \par
3659   \unskip
3660   \ifvoid\@mpfootins\else
3661     \l@dunboxmpfoot
3662   \fi
```

The next line is our addition to the original.

```

3663   \l@dfeetendmini%          added
3664   \@minipagefalse
3665   \color@endgroup
3666   \egroup
3667   \expandafter\@iiiparbox\@mpargs{\unvbox\@tempboxa}}
3668
```

`\l@dunboxmpfoot`

```

3669 \newcommand*{\l@dunboxmpfoot}{%
3670   \vskip\skip\@mpfootins
```

```

3671 \normalcolor
3672 \footnoterule
3673 \ifparledgroup
3674   \ifl@dpairing
3675     \ifledRcol
3676       \dimgdef{\parledgroup@beforenotesR}{\parledgroup@beforenotesR+\skip\@mpfootins}
3677     \else
3678       \dimgdef{\parledgroup@beforenotesL}{\parledgroup@beforenotesL+\skip\@mpfootins}
3679     \fi
3680   \fi
3681 \fi
3682 \unvbox\@mpfootins}
3683

```

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.

```

3684 \newenvironment{ledgroup}{%
3685   \resetprevpage@num%
3686   \def\@mpfn{mpfootnote}\def\thempfn{\thempfootnote}\c@mpfootnote\z@%
3687   \let\@footnotetext\@mpfootnotetext
3688   \l@dfeetbeginmini%
3689 }{%
3690   \par
3691   \unskip
3692   \ifvoid\@mpfootins\else
3693     \l@dunboxmpfoot
3694   \fi
3695   \l@dfeetendmini%
3696 }
3697

```

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.

```

3698 \newenvironment{ledgroupsize}[2][1]{%

```

Set the various text measures.

```

3699   \hsize #2\relax
3700 %%   \textwidth #2\relax
3701 %%   \columnwidth #2\relax

```

Initialize fills for centering.

```

3702   \let\ledllfill\hfil
3703   \let\ledrlfill\hfil
3704   \def\@tempa{#1}\def\@tempb{1}%

```

Left adjusted numbered lines

```

3705     \ifx\@tempa\@tempb
3706     \let\ledllfill\relax
3707   \else
3708     \def\@tempb{r}%
3709     \ifx\@tempa\@tempb
      Right adjusted numbered lines
3710     \let\ledrlfill\relax
3711     \fi
3712   \fi

  Set up the footnoting.
3713   \def\@mpfn{mpfootnote}\def\thempfn{\thempfootnote}\c@mpfootnote\z@
3714   \let\@footnotetext\@mpfootnotetext
3715   \l@dfeetbeginmini%
3716 }{%
3717   \par
3718   \unskip
3719   \ifvoid\@mpfootins\else
3720     \l@dunboxmpfoot
3721   \fi
3722   \l@dfeetendmini%
3723 }
3724

```

`\ifledgroupnotesL@` These boolean tests check if we are in the notes of a ledgroup. If we are, we don't
`\ifledgroupnotesR@` number the lines.

```

3725 \newif\ifledgroupnotesL@
3726 \newif\ifledgroupnotesR@

```

36 Indexing

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

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

```

3727 \AtBeginDocument{%
3728   \unless\ifl@imakeidx%
3729     \ifpackageloaded{imakeidx}{\led@error@ImakeidxAfterEledmac}{}%
3730   \fi%
3731   \unless\ifl@indextools%
3732     \ifpackageloaded{indextools}{\led@error@indextoolsAfterEledmac}{}%
3733   \fi%
3734 }

```

`\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 3735 \newcommand{\pagelinesep}{-}
3736 \newcommand{\edindexlab}{$&}
3737 \newcounter{labidx}
3738 \setcounter{labidx}{0}

```

3739

`\doedindexlabel` This macro sets an `\edlabel`.

```
3740 \newcommand{\doedindexlabel}{\stepcounter{labidx}%
3741   \edlabel{\edindexlab\thelabidx}}
3742
```

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

```
3743 \newcommand{\thepageline}{%
3744   \thepage\pagelinesep\xlineref{\edindexlab\thelabidx}}
```

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

```
3745 \newcommand{\thestartpageline}{\l@dparsedstartpage\pagelinesep\l@dparsedstartline}
3746 \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.

```
3747 \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.

```
3748 \newcommand{\prepare@edindex@fornote}[1]{%
3749   \l@dp@rsefootspec#1}%
3750   \@edindex@fornote@true
3751 }
```

`\get@index@command` This macro is used to analyse if a text to be indexed has a command after a |.

```
3752 \def\get@index@command#1|#2+{%
3753   \gdef\@index@txt{#1}%
3754   \gdef\@index@command{#2}%
3755   \xdef\@index@parenthesis{}%
3756   \IfBeginWith{\@index@command}{(}{%
3757     \StrGobbleLeft{\@index@command}{1}[\@index@command@]%
3758     \global\let\@index@command\@index@command@%
3759     \xdef\@index@parenthesis{(%}%
3760   }{%
3761     \IfBeginWith{\@index@command}{)}{%
3762       \StrGobbleLeft{\@index@command}{1}[\@index@command@]%
3763       \global\let\@index@command\@index@command@%
3764       \xdef\@index@parenthesis{)%}%
3765     }{%
3766   }
```

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

```
\ledinnotehyperpage 3767 \newcommand{\ledinnote}[2]{\csuse{#1}{#2\emph{n}}}
3768 \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.

```
3769 \def\create@edindex@for@memoir{
3770   \g@addto@macro{\makememindexhook}{%
3771     \def\edindex{\@bsphack%
3772       \ifnextchar [{\l@d@index}{\l@d@index[\jobname]}}
3773   \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@d@wrindexm@m` instead of `\@wrindexm@m`.

```
3774 \def\l@d@index{##1}{%
3775   \@ifundefined{##1@idxfile}%
3776   {\ifreportnoidxfile
3777     \led@warn@NoIndexFile{##1}%
3778     \fi
3779   \begingroup
3780   \@sanitize
3781   \@nowrindex}%
3782   {\def\@idxfile{##1}%
3783     \doedindexlabel
3784     \begingroup
3785     \@sanitize
3786     \l@d@wrindexm@m}}
```

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

```
3787 \newcommand{\l@d@wrindexm@m}[1]{\l@d@wrindexhyp##1||\}
3788 \def\l@d@wrindexhyp##1|##2|##3\{\\%
3789   \ifshowindexmark\@showidx{##1}\fi
3790   \ifx\\##2\\%
```



```

3791 \if@edindex@fornote@%
3792 \protected@write\@auxout{%
3793 {\string\@wrindexm@\@idxfile}{##1|(\ledinnotehyperpage){\thestartpageline}}%
3794 \protected@write\@auxout{%
3795 {\string\@wrindexm@\@idxfile}{##1|)\ledinnotehyperpage}{\theendpageline}}%
3796 \else%
3797 \protected@write\@auxout{%
3798 {\string\@wrindexm@\@idxfile}{##1|hyperpage}{\thepageline}}%
3799 \fi%
3800 \else
3801 \def\Hy@temp@A{##2}%
3802 \ifx\Hy@temp@A\HyInd@ParenLeft
3803 \if@edindex@fornote@%
3804 \protected@write\@auxout{%
3805 {\string\@wrindexm@\@idxfile}{##1|(\ledinnotehyperpage{##2}){\thestartpageline}}%
3806 \protected@write\@auxout{%
3807 {\string\@wrindexm@\@idxfile}{##1|)\ledinnotehyperpage{##2}}{\theendpageline}}%
3808 \else%
3809 \protected@write\@auxout{%
3810 {\string\@wrindexm@\@idxfile}{##1|##2hyperpage}{\thepageline}}%
3811 \fi%
3812 \else
3813 \if@edindex@fornote@%
3814 \protected@write\@auxout{%
3815 {\string\@wrindexm@\@idxfile}{##1|(\ledinnote{##2}){\thestartpageline}}%
3816 \protected@write\@auxout{%
3817 {\string\@wrindexm@\@idxfile}{##1|)\ledinnote{##2}}{\theendpageline}}%
3818 \else%
3819 \protected@write\@auxout{%
3820 {\string\@wrindexm@\@idxfile}{##1|##2}{\thepageline}}%
3821 \fi%
3822 \fi
3823 \fi
3824 \endgroup
3825 \esphack}

```

This finishes the memoir-specific code.

```
3826 }
```

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.

```
3827 \def\create@edindex@notfor@memoir{
```

`\@wredindex` Write the index information to the `idx` file.

```

3828 \newcommand{\@wredindex}[2][1=\expandonce\jobname,usedefault]{%#1 = the index name, #2
3829 \global\let\old@Rlineflag\Rlineflag%
3830 \gdef\Rlineflag{}%
3831 \ifl@imakeidx%
3832 \if@edindex@fornote%
3833 \IfSubStr[1]{##2}{|}{\get@index@command##2+}{\get@index@command##2|+}%
3834 \expandafter\imki@wrindexentry{##1}{\@index@txt|(ledinnote{\@index@command}}{\the
3835 \expandafter\imki@wrindexentry{##1}{\@index@txt|)ledinnote{\@index@command}}{\the
3836 \else%
3837 \get@edindex@hyperref{##2}%
3838 \imki@wrindexentry{##1}{\@index@txt\@edindex@hyperref}{\thepageline}%
3839 \fi%
3840 \else%
3841 \if@edindex@fornote%
3842 \IfSubStr[1]{##2}{|}{\get@index@command##2+}{\get@index@command##2|+}%
3843 \expandafter\protected@write\@indexfile{%
3844 {\string\indexentry{\@index@txt|(ledinnote{\@index@command}}{\thestartpageline}
3845 }%
3846 \expandafter\protected@write\@indexfile{%
3847 {\string\indexentry{\@index@txt|)ledinnote{\@index@command}}{\theendpageline}
3848 }%
3849 \else%
3850 \protected@write\@indexfile{%
3851 {\string\indexentry{##2}{\thepageline}
3852 }%
3853 \fi%
3854 \fi%
3855 \endgroup
3856 \global\let\Rlineflag\old@Rlineflag%
3857 \@esphack}

```

Need to add the definition of `\edindex` to `\makeindex`, and initialise `\edindex` to do nothing.

```

3858 \pretocmd{\makeindex}{%
3859 \def\edindex{\@bsphack
3860 \doedindexlabel
3861 \begingroup
3862 \@sanitize
3863 \@wredindex}}{}{}
3864 \newcommand{\edindex}[1]{\@bsphack\@esphack}

3865 % That finishes the non-\Lpack{memoir} index code.
3866 }

```

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`

```

3867 \ifclassloaded{memoir}{%

```

```

3868 \@ifpackageloaded{imakeidx}%
3869   {\create@edindex@notfor@memoir}%
3870   {%
3871     \@ifpackageloaded{indextools}%
3872     {\create@edindex@notfor@memoir}%
3873     {\create@edindex@for@memoir}%
3874   }%
3875   }%
3876   {\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.

```

3877 \newcommand{\hyperlinkformat}[3]{%
3878   \ifstrempy{#1}%
3879   {\hyperlink{#2}{#3}}%
3880   {\csuse{#1}{\hyperlink{#2}{#3}}%
3881   }}

```

`\hyperlinkR` `\hyperlinkR` command is to be used to create a internal hyperlink and `\ledRflag`, when indexing.

```

3882 \newcommand{\hyperlinkR}[2]{%
3883   \hyperlink{#1}{#2\Rlineflag}%
3884 }%
3885

```

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

```

3886 \newcommand{\hyperlinkformatR}[3]{%
3887   \hyperlinkformat{#1}{#2}{#3\Rlineflag}%
3888 }%
3889

```

`\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`).

```

3890 \newcommand{\get@edindex@hyperref}[1]{%
3891   \ifdef{\hyperlink}%

```

We have to disable spaces to work with a xstring bug

```

3892   {%
3893     \edef\temp@{%
3894       \catcode'\ =9 %space need for catcode
3895       #1%
3896       \catcode'\ =10 % space need for catcode
3897     }%
3898     \IfSubStr{\temp@}{|}%
3899     {\get@index@command#1+%
3900     \ifledRcol%

```

```

3901      \gdef\@edindex@hyperref{|\@index@parenthesis %space kept
3902      hyperlinkformatR{\@index@command}%
3903      {\edindexlab\thelabidx}}%
3904      \else%
3905      \gdef\@edindex@hyperref{|\@index@parenthesis %space kept
3906      hyperlinkformat{\@index@command}%
3907      {\edindexlab\thelabidx}}%
3908      \fi%
3909      }%
3910      {\get@index@command#1|+%
3911      \ifledRcol%
3912      \gdef\@edindex@hyperref{|\hyperlinkR{\edindexlab\thelabidx}}%
3913      \else%
3914      \gdef\@edindex@hyperref{|\hyperlink{\edindexlab\thelabidx}}%
3915      \fi%
3916      }%
3917      }%
3918      {%
3919      \gdef\@index@txt{#1}%
3920      \gdef\@edindex@hyperref{}}%
3921      }

```

37 Macro as environment

The following is borrowed, and renamed, from the `amsmath` package. See also the CTT thread ‘`eeq` and `amstex`’, 1995/08/31, started by Keith Reckdahl and ended definitively by David M. Jones.

Several of the `[math]` macros scan their body twice. This means we must collect all text in the body of an environment form before calling the macro.

`\@emptytoks` This is actually defined in the `ams-gen` package.

```

3922 \newtoks\@emptytoks
3923

```

The rest is from `amsmath`.

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

```

3924 \newtoks\l@denvbody
3925

```

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

```

3926 \newcommand{\addtol@denvbody}[1]{%
3927   \global\l@denvbody\expandafter{\the\l@denvbody#1}}
3928

```

`\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`.

```

3929 \newcommand{\l@dcollect@body}[1]{%
3930   \l@denbody{\expandafter#1\expandafter{\the\l@denbody}}%
3931   \edef\processl@denbody{\the\l@denbody\noexpand\end{\@currenvir}}%
3932   \l@denbody\@emptytoks \def\l@dbegin@stack{b}%
3933   \begingroup
3934     \expandafter\let\csname\@currenvir\endcsname\l@dcollect@@body
3935     \edef\processl@denbody{\expandafter\noexpand\csname\@currenvir\endcsname}%
3936     \processl@denbody%
3937   }%
3938

```

`\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.

```

3939 \def\l@dpush@begins#1\begin#2{%
3940   \ifx\end#2\else b\expandafter\l@dpush@begins\fi
3941

```

`\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.

```

3942 \def\l@dcollect@@body#1\end#2{%
3943   \edef\l@dbegin@stack{\l@dpush@begins#1\begin\end
3944     \expandafter\@gobble\l@dbegin@stack}%
3945   \ifx\@empty\l@dbegin@stack
3946     \endgroup
3947     \@checkend{#2}%
3948     \addtol@denbody{#1}%
3949   \else
3950     \addtol@denbody{#1\end{#2}}%
3951   \fi
3952   \processl@denbody % A little tricky! Note the grouping
3953 }
3954

```

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#{%
  \@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 3955 \newcommand*{\hangingsymbol}{}
3956 \newif\ifinstanza
3957 \instanzafalse

\inserthangingsymbol The boolean \ifinserthangingsymbol 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 is made in \do@line before the printing of line but
after the line number calculation.

3958 \newif\ifinserthangingsymbol
3959 \newcommand{\inserthangingsymbol}{%
3960 \ifinserthangingsymbol%
3961   \ifinstanza%
3962     \hangingsymbol%
3963   \fi%
3964 \fi%
3965 }

& \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.

3966 \newcommand*{\ampersand}{\char'&}
3967

\stanza@count Before we can define the main macros we need to save and reset some category
\stanzaindentbase codes. To save the current values we use \next and \body from the \loop macro.

3968 \chardef\body=\catcode'\@
3969 \catcode'\@=11
3970 \chardef\next=\catcode'\&
3971 \catcode'\&=\active
3972

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 indenta-
tion; all stanza indentations are multiples of this value. The default value of
\stanzaindentbase is 20pt.

3973 \newcount\stanza@count
3974 \newlength{\stanzaindentbase}
3975 \setlength{\stanzaindentbase}{20pt}
3976

\strip@szacnt The indentations of stanza lines are non-negative integer multiples of the unit
\setstanzavalues called \stanzaindentbase. To make it easier for the user to specify these num-
bers, 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 appli-
cations, including penalties, which will be discussed below.

3977 \def\strip@szacnt#1,#2|{\def\@tempb{#1}\def\@tempa{#2|}}

```



```

3978 \newcommand*{\setstanzavalues}[2]{\def\@tempa{#2,,}}%
3979     \stanza@count\z@
3980     \def\next{\expandafter\strip@szacnt\@tempa
3981         \ifx\@tempb\empty\let\next\relax\else
3982         \expandafter\mathchardef\csname #1@\number\stanza@count
3983         \@endcsname\@tempb\relax
3984         \advance\stanza@count\@ne\fi\next}%
3985     \next}
3986

```

`\setstanzaindents` In the original `\setstanzavalues{sza}{...}` had to be called to set the indents, and similarly `\setstanzavalues{szp}{...}` to set the penalties. These
`\setstanzapenalties` two macros are a convenience to give the user one less thing to worry about (mis-
`\managestanza@modulo` spelling 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.

```

3987 \newcommand*{\setstanzaindents}[1]{\setstanzavalues{sza}{#1}}
3988 \newcommand*{\setstanzapenalties}[1]{\setstanzavalues{szp}{#1}}
3989
3990 \newcounter{stanzaindentsrepetition}
3991 \newcount\stanza@modulo
3992
3993 \newcommand*{\managestanza@modulo}[0]{
3994     \advance\stanza@modulo\@ne
3995     \ifnum\stanza@modulo>\value{stanzaindentsrepetition}
3996         \stanza@modulo\@ne
3997     \fi
3998 }

```

`\stanzaindent` The macro `\stanzaindent`, when called at the beginning of a verse, changes the
`\stanzaindent*` indentation normally defined for this verse by `\setstanzaindent`. The starred version skips the current verse for the repetition of stanza indent.

```

3999 \newcommand{\stanzaindent}[1]{%
4000     \hspace{\dimexpr#1\stanzaindentbase-\parindent\relax}%
4001     \ignorespaces%
4002 }%
4003 \WithSuffix\newcommand\stanzaindent*[1]{%
4004     \stanzaindent{#1}%
4005     \global\advance\stanza@modulo-\@ne%
4006     \ifnum\stanza@modulo=0%
4007         \global\stanza@modulo=\value{stanzaindentsrepetition}%
4008     \fi%
4009     \ignorespaces%
4010 }%

```

`\stanza@line` Now we arrive at the main works. `\stanza@line` sets the indentation for the
`\stanza@hang` line and starts a numbered paragraph—each line is treated as a paragraph.
`\sza@penalty`

`\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.

```

4011 \newcommand{\stanza@line}[1][1]{
4012   \ifnum\value{stanza@line}=0
4013     \parindent=\csname sza@\number\stanza@count
4014               @\endcsname\stanza@indentbase
4015   \else
4016     \parindent=\csname sza@\number\stanza@modulo
4017               @\endcsname\stanza@indentbase
4018     \managestanza@modulo
4019   \fi
4020   \pstart[#1]\stanza@hang\ignorespaces}
4021 \xdef\stanza@hang{\noexpand\leavevmode\noexpand\startlock
4022                 \hangindent\expandafter
4023                 \noexpand\csname sza@0@\endcsname\stanza@indentbase
4024                 \hangafter\@ne}
4025 \def\sza@penalty{\count@\csname szp@\number\stanza@count @\endcsname
4026                 \ifnum\count@>\@M\advance\count@-\@M\penalty-\else
4027                 \penalty\fi\count@}

```

`\startstanzahook` 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 `\endstanzaextra` 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.

```

4028 \let\startstanzahook\relax
4029 \let\endstanzaextra\relax
4030 \xdef\@startstanza[#1]{%
4031   \noexpand\instanzatrue\expandafter
4032   \begingroup\startstanzahook%
4033   \catcode'\noexpand\&\active%
4034   \global\stanza@count\@ne\stanza@modulo\@ne
4035   \noexpand\ifnum\expandafter\noexpand
4036   \csname sza@0@\endcsname=z@\let\noexpand\stanza@hang\relax
4037   \let\noexpand\endlock\relax\noexpand\else\interlinepenalty
4038   \@M\rightskip\z@ plus 1fil\relax\noexpand\fi\noexpand\ifnum
4039   \expandafter\noexpand\csname szp@0@\endcsname=z@

```

```

4040 \let\noexpand\sza@penalty\relax\noexpand\fi%
4041 \def\noexpand\falseverse{%
4042   \noexpand\led@war@FalseverseDeprecated%
4043   \global\advance\stanza@modulo-\@ne%
4044   \global\advance\stanza@count-\@ne%
4045   \relax\noexpand&\leavevmode\skipnumbering}
4046 \def\noexpand&{%
4047   \noexpand\newverse[] []}%
4048 \def\noexpand\&\noexpand\@stopstanza}%
4049 \noexpand\stanza@line[#1]}
4050
4051 \newcommandx{\stanza}[1][1,usedefault]{\@startstanza[#1]}
4052
4053 \newcommandx{\@stopstanza}[1][1,usedefault]{%
4054   \endlock%
4055   \pend[#1]%
4056   \endgroup%
4057   \instanzafalse%
4058   \endstanzaextra%
4059 }
4060
4061 \newcommandx*{\newverse}[2][1,2,usedefault]{%
4062   \endlock\pend[#1]\sza@penalty\global%
4063   \advance\stanza@count\@ne\stanza@line[#2]%
4064   }
4065

```

\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`.

```

4066 \newcommand*{\flagstanza}[2][\stanzaindentbase]{%
4067   \hskip -#1\llap{#2}\hskip #1\ignorespaces}
4068

```

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.

```

4069 \catcode'\&=\next
4070 \catcode'\@=\body
4071 %% \let\ampersand=\&
4072 \setstanzavalues{szp}{0}
4073

```

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, \footnote 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..

```

4074 \newcommand*{\l@dtabnoexpands}{%
4075   \let\rtab=0%
4076   \let\ctab=0%
4077   \let\ltab=0%
4078   \let\rtabtext=0%
4079   \let\ltabtext=0%
4080   \let\ctabtext=0%
4081   \let\edbeforetab=0%
4082   \let\edaftertab=0%
4083   \let\edatab=0%
4084   \let\edatabell=0%
4085   \let\edatleft=0%
4086   \let\edatright=0%
4087   \let\edvertline=0%
4088   \let\edvertdots=0%
4089   \let\edrowfill=0%
4090 }
4091

```

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

```

4092 \newcommand{\disable@familiarnotes}{%

```

```

4093 \def\do##1{%
4094     \csletcs{footnote@##1}{footnote##1}%
4095     \expandafter\renewcommand \csname footnote##1\endcsname[1]{%
4096         \protected@csxdef{@thefnmark##1}{\csuse{thefootnote##1}}%
4097         \csuse{@footnotemark##1}%
4098     }%
4099 }%
4100 \dolistloop{\@series}%
4101 }%
4102 \newcommand{\restore@familiarnotes}{%
4103     \def\do##1{%
4104         \csletcs{footnote##1}{footnote@##1}%
4105     }%
4106     \dolistloop{\@series}%
4107 }%
4108

```

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

```

\restore@sidenotes 4109 \newcommand{\disable@sidenotes}{%
4110     \let\@@ledrightnote\ledrightnote%
4111     \let\@@ledleftnote\ledleftnote%
4112     \let\@@ledsidenote\ledsidenote%
4113     \let\ledrightnote@gobble%
4114     \let\ledleftnote@gobble%
4115     \let\ledsidenote@gobble%
4116 }%
4117 \newcommand{\restore@sidenotes}{%
4118     \let\ledrightnote\@@ledrightnote%
4119     \let\ledleftnote\@@ledleftnote%
4120     \let\ledsidenote\@@ledsidenote%
4121 }%

```

`\disable@notes` Disable/restore side and familiar notes.

```

\restore@notes 4122 \newcommand{\disable@notes}{%
4123     \disable@sidenotes%
4124     \disable@familiarnotes%
4125 }%
4126 \newcommand{\restore@notes}{%
4127     \restore@sidenotes%
4128     \restore@familiarnotes%
4129 }%

```

`\l@dampcount` `\l@dampcount` is a counter for the & column dividers and `\l@dcolcount` is a counter for the columns. These were `\Undcount` and `\stellencount` respectively.

```

4130 \newcount\l@dampcount
4131 \l@dampcount=1\relax
4132 \newcount\l@dcolcount
4133 \l@dcolcount=0\relax
4134

```

```

\hilfsbox   Some (temporary) helper items.
\hilfsskip 4135 \newbox\hilfsbox
\Hilfsbox 4136 \newskip\hilfsskip
\hilfscount 4137 \newbox\Hilfsbox
           4138 \newcount\hilfscount
           4139

```

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).

```

4140 \newdimen\dcoli
4141 \newdimen\dcolii
4142 \newdimen\dcoliii
4143 \newdimen\dcoliv
4144 \newdimen\dcolv
4145 \newdimen\dcolvi
4146 \newdimen\dcolvii
4147 \newdimen\dcolviii
4148 \newdimen\dcolix
4149 \newdimen\dcolx
4150 \newdimen\dcolxi
4151 \newdimen\dcolxii
4152 \newdimen\dcolxiii
4153 \newdimen\dcolxiv
4154 \newdimen\dcolxv
4155 \newdimen\dcolxvi
4156 \newdimen\dcolxvii
4157 \newdimen\dcolxviii
4158 \newdimen\dcolxix
4159 \newdimen\dcolxx
4160 \newdimen\dcolxxi
4161 \newdimen\dcolxxii
4162 \newdimen\dcolxxiii
4163 \newdimen\dcolxxiv
4164 \newdimen\dcolxxv
4165 \newdimen\dcolxxvi
4166 \newdimen\dcolxxvii
4167 \newdimen\dcolxxviii
4168 \newdimen\dcolxxix
4169 \newdimen\dcolxxx
4170 \newdimen\dcolerr   % added for error handling
4171

```

`\l@dcolwidth` This is a cunning way of storing the columnwidths indexed by the column number `\l@dcolcount`, like an array. (was `\Dimenzuordnung`)

```

4172 \newcommand{\l@dcolwidth}{\ifcase \the\l@dcolcount \dcoli %???
4173   \or \dcoli \or \dcolii \or \dcoliii
4174   \or \dcoliv \or \dcolv \or \dcolvi
4175   \or \dcolvii \or \dcolviii \or \dcolix \or \dcolx

```

```

4176 \or \dcolxi \or \dcolxii \or \dcolxiii
4177 \or \dcolxiv \or \dcolxv \or \dcolxvi
4178 \or \dcolxvii \or \dcolxviii \or \dcolxix \or \dcolxx
4179 \or \dcolxxi \or \dcolxxii \or \dcolxxiii
4180 \or \dcolxxiv \or \dcolxxv \or \dcolxxvi
4181 \or \dcolxxvii \or \dcolxxviii \or \dcolxxix \or \dcolxxx
4182 \else \dcolerr \fi}
4183

```

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

```

4184 \newcommand*{\stepl@dcolcount}{\advance\l@dcolcount\@ne
4185 \ifnum\l@dcolcount>30\relax
4186 \led@err@TooManyColumns
4187 \fi}
4188

```

`\l@dsetmaxcolwidth` Sets the column width to the maximum value seen so far. (was `\dimenzuordnung`)

```

4189 \newcommand{\l@dsetmaxcolwidth}{%
4190 \ifdim\l@dcolwidth < \wd\hilfsbox
4191 \l@dcolwidth = \wd\hilfsbox
4192 \else \relax \fi}
4193

```

`\EDTEXT` We need to be able to modify the `\edtext` and `\critext` macros and also restore `\xedtext` their original definitions.

```

\CRITEXT 4194 \let\EDTEXT=\edtext
\xcritext 4195 \newcommand{\xedtext}[2]{\EDTEXT{#1}{#2}}
4196 \let\CRITEXT=\critext
4197 \long\def\xcritext #1#2/{\CRITEXT{#1}{#2}/}

```

`\EDLABEL` We need to be able to modify and restore the `\edlabel` macro.

```

\xedlabel 4198 \let\EDLABEL=\edlabel
4199 \newcommand*{\xedlabel}[1]{\EDLABEL{#1}}

```

`\EDINDEX` Macros supporting modification and restoration of `\edindex`.

```

\xedindex 4200 \let\EDINDEX=\edindex
\nulledindex 4201 \ifl@dmemoir
4202 \newcommand{\xedindex}{\@bsphack%
4203 \ifnextchar [{\l@d@index}{\l@d@index[\jobname]}}
4204 \newcommand{\nulledindex}[2][\jobname]{\@bsphack\@esphack}
4205 \else
4206 \newcommand{\xedindex}{\@bsphack%
4207 \doedindexlabel
4208 \begingroup
4209 \@sanitize
4210 \wredindex}
4211 \newcommand{\nulledindex}[1]{\@bsphack\@esphack}
4212 \fi
4213

```

```

\@line@num Macro supporting restoration of \linenum.
4214 \let\@line@num=\linenum

\l@dgobbledarg \l@dgobbledarg replaces its delineated argument by \relax (was \verschwinden).
\l@dgobblearg \l@dgobbleoptarg[\langle arg \rangle]{\langle arg \rangle} replaces these two arguments (first is optional)
by \relax.
4215 \def\l@dgobbledarg #1/{\relax}
4216 \newcommand*\l@dgobbleoptarg[2][]{\relax}%
4217

\Relax
\NEXT 4218 \let\Relax=\relax
\@hilfs@count 4219 \let\NEXT=\next
4220 \newcount\@hilfs@count
4221

\measuremcell Measure (recursively) the width required for a math cell. (was \messen)
4222 \def\measuremcell #1{%
4223   \ifx #1\ \ifnum\l@dc@count=0\let\NEXT\relax%
4224   \else\l@dcheckcols%
4225   \l@dc@count=0%
4226   \let\NEXT\measuremcell%
4227   \fi%
4228   \else\setbox\hilfsbox=\hbox{$\displaystyle{#1}$}%
4229   \step\l@dc@count%
4230   \l@dsetmaxcolwidth%
4231   \let\NEXT\measuremcell%
4232   \fi\NEXT}
4233

\measuretcell Measure (recursively) the width required for a text cell. (was \messentext)
4234 \def\measuretcell #1{%
4235   \ifx #1\ \ifnum\l@dc@count=0\let\NEXT\relax%
4236   \else\l@dcheckcols%
4237   \l@dc@count=0%
4238   \let\NEXT\measuretcell%
4239   \fi%
4240   \else\setbox\hilfsbox=\hbox{#1}%
4241   \step\l@dc@count%
4242   \l@dsetmaxcolwidth%
4243   \let\NEXT\measuretcell%
4244   \fi\NEXT}
4245

\measuremrow Measure (recursively) the width required for a math row. (was \Messen)
4246 \def\measuremrow #1\{%
4247   \ifx #1\let\NEXT\relax%
4248   \else\measuremcell #1\&\&\&%
4249   \let\NEXT\measuremrow%
4250   \fi\NEXT}

```


`\measuretrow` Measure (recursively) the width required for a text row. (was `\Messentext`)

```
4251 \def\measuretrow #1\{\%
4252   \ifx #1&\let\NEXT\relax%
4253   \else\measuretcell #1&\&\&\&%
4254     \let\NEXT\measuretrow%
4255   \fi\NEXT}
4256
```

`\edtabcolsep` The length `\edtabcolsep` controls the distance between columns. (was `\abstand`)

```
4257 \newskip\edtabcolsep
4258 \global\edtabcolsep=10pt
4259
```

`\NEXT`

```
\Next 4260 \let\NEXT\relax
4261 \let\Next=\next
```

`\variab`

```
4262 \newcommand{\variab}{\relax}
4263
```

`\l@dcheckcols` Check that the number of columns is consistent. (was `\tabfehlermeldung`)

```
4264 \newcommand*{\l@dcheckcols}{\%
4265   \ifnum\l@dcolcount=1\relax
4266   \else
4267     \ifnum\l@dampcount=1\relax
4268     \else
4269       \ifnum\l@dcolcount=\l@dampcount\relax
4270       \else
4271         \l@d@err@UnequalColumns
4272       \fi
4273     \fi
4274     \l@dampcount=\l@dcolcount
4275   \fi}
4276
```

`\l@dmodforcritext` Modify and restore various macros for when `\critext` is used.

```
\l@drestoreforcritext 4277 \newcommand{\l@dmodforcritext}{\%
4278   \let\critext\relax%
4279   \def\do##1{\global\csletcs{##1footnote}{\l@dgobbledarg}}
4280   \dolistloop{\@series}%
4281   \let\edindex\nulledindex%
4282   \let\linenum@gobble}
4283 \newcommand{\l@drestoreforcritext}{\%
4284   \def\do##1{\csdef{##1footnote}##1##2/{\csuse{##1@footnote}{##1}{##2}}}}
4285   \dolistloop{\@series}%
4286   \let\edindex\xedindex}
4287
```

```

\l@dmodforedtext  Modify and restore various macros for when \edtext is used.
\l@drestoreforedtext 4288 \newcommand{\l@dmodforedtext}{%
4289   \let\edtext\relax
4290   \def\do##1{\global\csletcs{##1footnote}{\l@gobbleoptarg}}%
4291   \dolistloop{\@series}%
4292   \let\edindex\nulledindex
4293   \let\linenum@gobble}
4294 \newcommand{\l@drestoreforedtext}{%
4295   \def\do##1{\global\csletcs{##1footnote}{##1@footnote}}
4296   \dolistloop{\@series}%
4297   \let\edindex\xedndex}

\l@dnullfills  Nullify and restore some column fillers, etc.
\l@drestorefills 4298 \newcommand{\l@dnullfills}{%
4299   \def\edlabel##1{%
4300     \def\edrowfill##1##2##3{%
4301     }
4302   \newcommand{\l@drestorefills}{%
4303     \def\edrowfill##1##2##3{\@EDROWFILL@{##1}{##2}{##3}}%
4304   }
4305
```

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.

```

4306 \newcommand{\letsforverteilen}{%
4307   \let\critext\xcritext
4308   \let\edtext\xedtext
4309   \let\edindex\xedindex
4310   \def\do##1{\global\csletcs{##1footnote}{##1@footnote}}
4311   \dolistloop{\@series}%
4312   \let\linenum\@line@num
4313   \hilfsskip=\l@dcwidth%
4314   \advance\hilfsskip by -\wd\hilfsbox
4315   \def\edlabel##1{\xílabel{##1}}
4316 }

```

\setmcellright Typeset (recursively) cells of display math right justified. (was *\rverteilen*)

```

4317 \def\setmcellright #1&{\def\edlabel##1{%
4318   \let\edindex\nulledindex
4319   \ifx #1\ \ifnum\l@dcwidth=0%\removelastskip
4320     \let\Next\relax%
4321   \else\l@dcwidth=0%
4322     \let\Next=\setmcellright%
4323   \fi%
4324   \else%
4325     \disablel@dtabfeet%
4326     \step1@dcwidth%
4327     \disable@notes%
4328     \setbox\hilfsbox=\hbox{$\displaystyle{#1}$}%
4329     \restore@notes%
4330     \letsforverteilen%
4331     \hskip\hilfsskip$\displaystyle{#1}$%
4332     \hskip\edtabcolsep%
4333     \let\Next=\setmcellright%
4334   \fi\Next}
4335 }

```

`\settcellright` Typeset (recursively) cells of text right justified. (was `\rverteilentext`)

```

4336 \def\settcellright #1&{\def\edlabel##1}%
4337     \let\edindex\nulledindex
4338     \ifx #1\\ \ifnum\l@dc@count=0%\removelastskip
4339         \let\Next\relax%
4340     \else\l@dc@count=0%
4341         \let\Next=\settcellright%
4342     \fi%
4343 \else%
4344     \disablel@dtabfeet%
4345     \step1@dc@count%
4346     \disable@notes%
4347     \setbox\hilfsbox=\hbox{#1}%
4348     \restore@notes%
4349     \letsforverteilen%
4350     \hskip\hilfsskip#1%
4351     \hskip\edtabcolsep%
4352     \let\Next=\settcellright%
4353 \fi\Next}

```

`\setmcellleft` Typeset (recursively) cells of display math left justified. (was `\lverteilen`)

```

4354 \def\setmcellleft #1&{\def\edlabel##1}%
4355     \let\edindex\nulledindex
4356     \ifx #1\\ \ifnum\l@dc@count=0 \let\Next\relax%
4357         \else\l@dc@count=0%
4358         \let\Next=\setmcellleft%
4359     \fi%
4360 \else \disablel@dtabfeet%
4361     \step1@dc@count%
4362     \disable@notes%
4363     \setbox\hilfsbox=\hbox{$\displaystyle{#1}$}%
4364     \restore@notes%
4365     \letsforverteilen%
4366     $\displaystyle{#1}$\hskip\hilfsskip\hskip\edtabcolsep%
4367     \let\Next=\setmcellleft%
4368 \fi\Next}
4369

```

`\settcellleft` Typeset (recursively) cells of text left justified. (was `\lverteilentext`)

```

4370 \def\settcellleft #1&{\def\edlabel##1}%
4371     \let\edindex\nulledindex
4372     \ifx #1\\ \ifnum\l@dc@count=0 \let\Next\relax%
4373         \else\l@dc@count=0%
4374         \let\Next=\settcellleft%
4375     \fi%
4376 \else \disablel@dtabfeet%
4377     \step1@dc@count%
4378     \disable@notes%
4379     \setbox\hilfsbox=\hbox{#1}%

```

```

4380         \restore@notes%
4381         \letsforverteilen%
4382         #1\hskip\hilfsskip\hskip\edtabcolsep%
4383         \let\Next=\settcclleft%
4384     \fi\Next}

```

\setmcellcenter Typeset (recursively) cells of display math centered. (was \zverteilen)

```

4385 \def\setmcellcenter #1&{\def\edlabel##1{}%
4386     \let\edindex\nulledindex
4387     \ifx #1\ \ifnum\l@dc@count=0\let\Next\relax%
4388         \else\l@dc@count=0%
4389             \let\Next=\setmcellcenter%
4390         \fi%
4391     \else \disabl@dtabfeet%
4392         \stepl@dc@count%
4393         \disabl@notes%
4394         \setbox\hilfsbox=\hbox{$\displaystyle{#1}$}%
4395         \restore@notes%
4396         \letsforverteilen%
4397         \hskip 0.5\hilfsskip$\displaystyle{#1}$\hskip0.5\hilfsskip%
4398         \hskip\edtabcolsep%
4399         \let\Next=\setmcellcenter%
4400     \fi\Next}
4401

```

\settcclcenter Typeset (recursively) cells of text centered. (new)

```

4402 \def\settcclcenter #1&{\def\edlabel##1{}%
4403     \let\edindex\nulledindex
4404     \ifx #1\ \ifnum\l@dc@count=0 \let\Next\relax%
4405         \else\l@dc@count=0%
4406             \let\Next=\settcclcenter%
4407         \fi%
4408     \else \disabl@dtabfeet%
4409         \stepl@dc@count%
4410         \disabl@notes%
4411         \setbox\hilfsbox=\hbox{#1}%
4412         \restore@notes%
4413         \letsforverteilen%
4414         \hskip 0.5\hilfsskip #1\hskip 0.5\hilfsskip%
4415         \hskip\edtabcolsep%
4416         \let\Next=\settcclcenter%
4417     \fi\Next}
4418

```

\NEXT

```

4419 \let\NEXT=\relax
4420

```

\setmrowright Typeset (recursively) rows of right justified math. (was \rsetzen)

```

4421 \def\setmrowright #1\{%
4422     \ifx #1& \let\NEXT\relax
4423     \else \centerline{\setmcellright #1&\&\&}
4424         \let\NEXT=\setmrowright
4425     \fi\NEXT}

```

\settroright Typeset (recursively) rows of right justified text. (was **\rsetzentext**)

```

4426 \def\settroright #1\{%
4427     \ifx #1& \let\NEXT\relax
4428     \else \centerline{\settcclright #1&\&\&}
4429         \let\NEXT=\settroright
4430     \fi\NEXT}
4431

```

\setmrowleft Typeset (recursively) rows of left justified math. (was **\lsetzen**)

```

4432 \def\setmrowleft #1\{%
4433     \ifx #1& \let\NEXT\relax
4434     \else \centerline{\setmcellleft #1&\&\&}
4435         \let\NEXT=\setmrowleft
4436     \fi\NEXT}

```

\settrorleft Typeset (recursively) rows of left justified text. (was **\lsetzentext**)

```

4437 \def\settrorleft #1\{%
4438     \ifx #1& \let\NEXT\relax
4439     \else \centerline{\settcclleft #1&\&\&}
4440         \let\NEXT=\settrorleft
4441     \fi\NEXT}
4442

```

\setmrowcenter Typeset (recursively) rows of centered math. (was **\zsetzen**)

```

4443 \def\setmrowcenter #1\{%
4444     \ifx #1& \let\NEXT\relax%
4445     \else \centerline{\setmcellcenter #1&\&\&}
4446         \let\NEXT=\setmrowcenter
4447     \fi\NEXT}

```

\settrorcenter Typeset (recursively) rows of centered text. (new)

```

4448 \def\settrorcenter #1\{%
4449     \ifx #1& \let\NEXT\relax
4450     \else \centerline{\settcclcenter #1&\&\&}
4451         \let\NEXT=\settrorcenter
4452     \fi\NEXT}
4453

```

\nullsetzen (was **\nullsetzen**)

```

4454 \newcommand{\nullsetzen}{%
4455     \stepl@dc@count%
4456     \l@dc@width=0pt%
4457     \ifnum\l@dc@count=30\let\NEXT\relax%

```

```

4458         \l@dcollcount=0\relax
4459     \else\let\NEXT\nullsetzen%
4460     \fi\NEXT}
4461
\edatleft  \edatleft[ $]{\langle symbol\rangle}{\langle len\rangle} (combination and generalisation of original
\seklam and \seklamgl). Left \langle symbol\rangle, 2\langle len\rangle high with prepended \langle math\rangle
vertically centered.
4462 \newcommand{\edatleft}[3][\@empty]{%
4463     \ifx#1\@empty
4464         \vbox to 10pt{\vss\hbox{\$ \left#2\vrule width0pt height #3
4465             depth 0pt \right. \$\hss}\vfil}
4466     \else
4467         \vbox to 4pt{\vss\hbox{\$#1\left#2\vrule width0pt height #3
4468             depth 0pt \right. \$}\vfil}
4469     \fi}

\edatright \edatright[ $]{\langle symbol\rangle}{\langle len\rangle} (combination and generalisation of origi-
nal \seklam and \seklamgl). Right \langle symbol\rangle, 2\langle len\rangle high with appended \langle math\rangle
vertically centered.
4470 \newcommand{\edatright}[3][\@empty]{%
4471     \ifx#1\@empty
4472         \vbox to 10pt{\vss\hbox{\$ \left.\vrule width0pt height #3
4473             depth 0pt \right#2 \$\hss}\vfil}
4474     \else
4475         \vbox to 4pt{\vss\hbox{\$ \left.\vrule width0pt height #3
4476             depth 0pt \right#2 #1 \$}\vfil}
4477     \fi}
4478

\edvertline \edvertline{\langle len\rangle} vertical line \langle len\rangle high. (was \sestrich)
4479 \newcommand{\edvertline}[1]{\vbox to 8pt{\vss\hbox{\vrule height #1}\vfil}}
4480

\edvertdots \edvertdots{\langle len\rangle} vertical dotted line \langle len\rangle high. (was \sepunkte)
4481 \newcommand{\edvertdots}[1]{\vbox to 1pt{\vss\vbox to #1%
4482     {\cleaders\hbox{\$ \m@th\hbox{.}\vbox to 0.5em{ }\$}\vfil}}}
4483$$ 
```

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`)

```
4484 \newdimen\edfilldimen
4485 \edfilldimen=0pt
4486
```

`\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...`

```
4487 \newcounter{addcolcount}
4488 \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 reimplementaion of the
 original `\@add@`.

```
4489 \newcommand{\l@dtabaddcols}[2]{%
4490   \l@dcheckstartend{#1}{#2}%
4491   \ifl@dstartendok
4492     \setcounter{addcolcount}{#1}%
4493     \@whilenum \value{addcolcount}<#2\relax \do
4494       {\advance\edfilldimen by \the\csname dcol\theaddcolcount\endcsname
4495        \advance\edfilldimen by \edtabcolsep
4496        \stepcounter{addcolcount}}%
4497     \advance\edfilldimen by \the\csname dcol\theaddcolcount\endcsname
4498     \fi
4499 }
4500
```

`\ifl@dstartendok` `\l@dcheckstartend{<startcol>}{<endcol>}` checks that the values of `<startcol>` and
`\l@dcheckstartend` `<endcol>` are sensible. If they are then `\ifl@dstartendok` is set TRUE, otherwise
 it is set FALSE.


```

4501 \newif\ifl@startendok
4502 \newcommand{\l@dccheckstartend}[2]{%
4503   \l@startendoktrue
4504   \ifnum #1<\@ne
4505     \l@startendokfalse
4506     \led@err@LowStartColumn
4507   \fi
4508   \ifnum #2>30\relax
4509     \l@startendokfalse
4510     \led@err@HighEndColumn
4511   \fi
4512   \ifnum #1>#2\relax
4513     \l@startendokfalse
4514     \led@err@ReverseColumns
4515   \fi
4516 }
4517

```

`\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.

```

4518 \newcommand*{\edrowfill}[3]{%
4519   \l@dtabaddcols{#1}{#2}%
4520   \hb@xt@ \the\l@dcwidth{\hb@xt@ \the\edfilldimen{#3}\hss}}
4521 \let\@edrowfill@=\edrowfill
4522 \def\@EDROWFILL@#1#2#3{\@edrowfill@{#1}{#2}{#3}}
4523

```

`\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`)

```

4524 \newcommand{\leftltab}[1]{%
4525   \hb@xt@ \z@{\vbox{\edtabindent%
4526     \moveleft\Hilfsskip\hbox{\ #1}\hss}}
4527

```

`\leftrtab` `\leftrtab{<text>}{<math>}` for `\edbeforetab` in `\rtab`. (was `\linksrtab`)

```

4528 \newcommand{\leftrtab}[2]{%
4529   #2\hb@xt@ \z@{\vbox{\edtabindent%
4530     \advance\Hilfsskip by\dcoli%
4531     \moveleft\Hilfsskip\hbox{\ #1}\hss}}
4532

```

`\leftctab` `\leftctab{<text>}{<math>}` for `\edbeforetab` in `\ctab`. (was `\linksztab`)

```

4533 \newcommand{\leftctab}[2]{%
4534     \hb@xt@\z@\vbox{\edtabindent\l@dcolcount=\l@dampcount%
4535     \advance\Hilfsskip by 0.5\dcoli%
4536     \setbox\hilfsbox=\hbox{\def\edlabel##1{%
4537     \disablel@dtabfeet$\displaystyle{#2}$}%
4538     \advance\Hilfsskip by -0.5\wd\hilfsbox%
4539     \moveleft\Hilfsskip\hbox{\ #1}}\hss}%
4540     #2}
4541

```

`\rightctab` `\rightctab{<math><math>}{<text>}` for `\edaftertab` in `\ctab`. (was `\rechtsztab`)

```

4542 \newcommand{\rightctab}[2]{%
4543     \setbox\hilfsbox=\hbox{\def\edlabel##1{%
4544     \disablel@dtabfeet#2}\l@dampcount=\l@dcolcount%
4545     #1\hb@xt@\z@\vbox{\edtabindent\l@dcolcount=\l@dampcount%
4546     \advance\Hilfsskip by 0.5\l@dcolwidth%
4547     \advance\Hilfsskip by -\wd\hilfsbox%
4548     \setbox\hilfsbox=\hbox{\def\edlabel##1{%
4549     \disablel@dtabfeet$\displaystyle{#1}$}%
4550     \advance\Hilfsskip by -0.5\wd\hilfsbox%
4551     \advance\Hilfsskip by \edtabcolsep%
4552     \moveright\Hilfsskip\hbox{ #2}}\hss}%
4553     }
4554

```

`\rightltab` `\rightltab{<math><math>}{<text>}` for `\edaftertab` in `\ltab`. (was `\rechtsltab`)

```

4555 \newcommand{\rightltab}[2]{%
4556     \setbox\hilfsbox=\hbox{\def\edlabel##1{%
4557     \disablel@dtabfeet#2}\l@dampcount=\l@dcolcount%
4558     #1\hb@xt@\z@\vbox{\edtabindent\l@dcolcount=\l@dampcount%
4559     \advance\Hilfsskip by\l@dcolwidth%
4560     \advance\Hilfsskip by-\wd\hilfsbox%
4561     \setbox\hilfsbox=\hbox{\def\edlabel##1{%
4562     \disablel@dtabfeet$\displaystyle{#1}$}%
4563     \advance\Hilfsskip by-\wd\hilfsbox%
4564     \advance\Hilfsskip by\edtabcolsep%
4565     \moveright\Hilfsskip\hbox{ #2}}\hss}%
4566     }
4567

```

`\rightrtab` `\rightrtab{<math><math>}{<text>}` for `\edaftertab` in `\rtab`. (was `\rechtsrtab`)

```

4568 \newcommand{\rightrtab}[2]{%
4569     \setbox\hilfsbox=\hbox{\def\edlabel##1{%
4570     \disablel@dtabfeet#2}%
4571     #1\hb@xt@\z@\vbox{\edtabindent%
4572     \advance\Hilfsskip by-\wd\hilfsbox%
4573     \advance\Hilfsskip by\edtabcolsep%
4574     \moveright\Hilfsskip\hbox{ #2}}\hss}%
4575     }
4576

```

`\rtab` `\rtab{⟨body⟩}` typesets $\langle body \rangle$ as an array with the entries right justified. (was `\edbeforetab` `\rtab`) (Here and elsewhere, `\edbeforetab` and `\edaftertab` were originally `\edavor` and `\edanach`) 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.

```

4577 \newcommand{\rtab}[1]{%
4578   \l@dnnullfills
4579   \def\edbeforetab##1##2{\lefttab{##1}{##2}}%
4580   \def\edaftertab##1##2{\righttab{##1}{##2}}%
4581   \measurebody{#1}%
4582   \l@drestorefills
4583   \variab
4584   \setmrowright #1\\&\\%
4585   \enablel@dtabfeet}
4586
```

`\measurebody` `\measurebody{⟨body⟩}` measures the array $\langle body \rangle$.

```

4587 \newcommand{\measurebody}[1]{%
4588   \disablel@dtabfeet%
4589   \l@dcolcount=0%
4590   \nullsetzen%
4591   \l@dcolcount=0
4592   \measuremrow #1\\&\\%
4593   \global\l@dampcount=1}
4594
```

`\rtabtext` `\rtabtext{⟨body⟩}` typesets $\langle body \rangle$ as a tabular with the entries right justified. (was `\rtabtext`)

```

4595 \newcommand{\rtabtext}[1]{%
4596   \l@dnnullfills
4597   \measuretbody{#1}%
4598   \l@drestorefills
4599   \variab
4600   \settroright #1\\&\\%
4601   \enablel@dtabfeet}
4602
```

`\measuretbody` `\measuretbody{⟨body⟩}` measures the tabular $\langle body \rangle$.

```

4603 \newcommand{\measuretbody}[1]{%
4604   \disable@notes%
4605   \disablel@dtabfeet%
4606   \l@dcolcount=0%
4607   \nullsetzen%
4608   \l@dcolcount=0
4609   \measuretror #1\\&\\%
4610   \restore@notes%
4611   \global\l@dampcount=1}
4612
```

```

\ltab Array with entries left justified. (was \ltab)
\edbeforetab 4613 \newcommand{\ltab}[1]{%
\edaftertab 4614 \l@dnnullfills
4615 \def\edbeforetab##1##2{\leftltab{##1}{##2}}%
4616 \def\edaftertab##1##2{\rightltab{##1}{##2}}%
4617 \measuretbody{#1}%
4618 \l@drestorefills
4619 \variab
4620 \setmrowleft #1\\&\\%
4621 \enablel@dtabfeet}
4622

\ltabtext Tabular with entries left justified. (was \ltabtext)
4623 \newcommand{\ltabtext}[1]{%
4624 \l@dnnullfills
4625 \measuretbody{#1}%
4626 \l@drestorefills
4627 \variab
4628 \setthrowleft #1\\&\\%
4629 \enablel@dtabfeet}
4630

\ctab Array with centered entries. (was \ztab)
\edbeforetab 4631 \newcommand{\ctab}[1]{%
\edaftertab 4632 \l@dnnullfills
4633 \def\edbeforetab##1##2{\leftctab{##1}{##2}}%
4634 \def\edaftertab##1##2{\rightctab{##1}{##2}}%
4635 \measuretbody{#1}%
4636 \l@drestorefills
4637 \variab
4638 \setmrowcenter #1\\&\\%
4639 \enablel@dtabfeet}
4640

\ctabtext Tabular with entries centered. (new)
4641 \newcommand{\ctabtext}[1]{%
4642 \l@dnnullfills
4643 \measuretbody{#1}%
4644 \l@drestorefills
4645 \variab
4646 \setthrowcenter #1\\&\\%
4647 \enablel@dtabfeet}
4648

\spreadtext (was \breitertext)
4649 \newcommand{\spreadtext}[1]{%\l@dcolcount=\l@dampcount%
4650 \hb@xt@ \the\l@dcolwidth{\hbox{#1}\hss}}

\spreadmath (was \breiter, 'breiter' = 'broadly')

```

```

4651 \newcommand{\spreadmath}[1]{%
4652   \hb@xt@ \the\l@dcolwidth{\hbox{$\displaystyle{#1}$}\hss}}
4653

```

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`)

```

4654 \def\tabellzwischen #1{%
4655   \ifx #1\ \let\NEXT\relax \l@dcolcount=0
4656   \else \stepl@dcolcount%
4657         \l@dcolwidth = #1 mm
4658         \let\NEXT=\tabellzwischen
4659   \fi \NEXT }
4660

```

`\edatabell` For example `\edatabell 4 & 19 & 8 \` specifies 3 columns with widths of 4, 19, and 8mm. (was `\atabell`)

```

4661 \def\edatabell #1\\{%
4662   \tabellzwischen #1\\&}

```

`\Setzen` (was `\Setzen`, 'setzen' = 'set')

```

4663 \def\Setzen #1{%
4664   \ifx #1\relax \let\NEXT=\relax
4665   \else \stepl@dcolcount%
4666         \let\tabelskip=\l@dcolwidth
4667         \EDTAB #1|
4668         \let\NEXT=\Setzen
4669   \fi\NEXT}
4670

```

`\EDATAB` (was `\ATAB`)

```

4671 \def\EDATAB #1\\{%
4672   \ifx #1\Relax \centerline{\Setzen #1\relax&}
4673         \let\Next\relax
4674   \else \centerline{\Setzen #1&\relax&}
4675         \let\Next=\EDATAB
4676   \fi\Next}

```

`\edatab` (was `\atab`)

```

4677 \newcommand{\edatab}[1]{%
4678   \variab%
4679   \EDATAB #1\\Relax\\}
4680

```

`\HILFSskip` More helpers.

```

\Hilfsskip 4681 \newskip\HILFSskip
4682 \newskip\Hilfsskip
4683

```

```

4684 \newcommand{\EDTABINDENT}{-%
4685     \ifnum\l@dcolcount=30\let\NEXT\relax\l@dcolcount=0%
4686     \else\step1@dcolcount%
4687         \advance\Hilfsskip by\l@dcolwidth%
4688         \ifdim\l@dcolwidth=0pt\advance\hilfscount\@ne
4689         \else\advance\Hilfsskip by \the\hilfscount\edtabcolsep%
4690         \hilfscount=1\fi%
4691         \let\NEXT=\EDTABINDENT%
4692     \fi\NEXT}%

```

```

4693 \newcommand{\edtabindent}{%
4694   \l@dcolcount=0\relax
4695   \Hilfsskip=0pt%
4696   \hilfscount=1\relax
4697   \EDTABINDENT%
4698   \hilfsskip=\hsize%
4699   \advance\hilfsskip -\Hilfsskip%
4700   \Hilfsskip=0.5\hilfsskip%
4701 }%
4702

```

```

4703 \def\EDTAB #1|#2|{%
4704     \setbox\tabhilfbox=\hbox{$\displaystyle\{#1\}$}%
4705     \setbox\tabHilfbox=\hbox{$\displaystyle\{#2\}$}%
4706     \advance\tabelskip -\wd\tabhilfbox%
4707     \advance\tabelskip -\wd\tabHilfbox%
4708     \unhbox\tabhilfbox\hskip\tabelskip%
4709     \unhbox\tabHilfbox}%
4710

```

```

4711 \def\EDTABtext #1\#2{%
4712   \setbox\tabhilfbox=\hbox{#1}%
4713   \setbox\tabHilfbox=\hbox{#2}%
4714   \advance\tabelskip -\wd\tabhilfbox%
4715   \advance\tabelskip -\wd\tabHilfbox%
4716   \unhbox\tabhilfbox\hskip\tabelskip%
4717   \unhbox\tabHilfbox}%

```

```
\tabHilfbox 4718 \newbox\tabhilfbox
4719 \newbox\tabHilfbox
4720
```

```
% That finishes tabmac
```

%%%

edarrayl The ‘environment’ forms for \ltab, \ctab and \rtab.

```
edarrayc 4721 \newenvironment{edarrayl}{\l@collect@body\ltab}{}
edarrayr 4722 \newenvironment{edarrayc}{\l@collect@body\ctab}{}
          4723 \newenvironment{edarrayr}{\l@collect@body\rtab}{}
          4724
```

edtabularl The ‘environment’ forms for \ltabtext, \ctabtext and \rtabtext.

```
edtabularc 4725 \newenvironment{edtabularl}{\l@collect@body\ltabtext}{}
edtabularr 4726 \newenvironment{edtabularc}{\l@collect@body\ctabtext}{}
          4727 \newenvironment{edtabularr}{\l@collect@body\rtabtext}{}
          4728
```

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 4729 \newcommand{\usingcritext}{%
\usingedtext 4730 \def\disablel@dtabfeet{\l@modforcritext}%
          4731 \def\enablel@dtabfeet{\l@drestoreforcritext}}
          4732 \newcommand{\usingedtext}{%
          4733 \def\disablel@dtabfeet{\l@modforedtext}%
          4734 \def\enablel@dtabfeet{\l@drestoreforedtext}}
          4735
          4736 \usingedtext
          4737
```

40 Section’s title commands

40.1 Deprecated commands

\initnumbering@sectcmd \initnumbering@sectcmd defines \ledxxx commands. These commands are deprecated.

\ledsection It also defines quotation environment. Note: this assumes that the user

\ledsection* didn’t change \chapter. If he did, he should redefine \initnumbering@sectcmd.

```
\ledsubsection 4738 \newcommand{\initnumbering@sectcmd}{
\ledsubsection* 4739 \newcommand{\ledsection}[2][{}]{%
\ledsubsubsection 4740 \led@war@ledxxxDeprecated{section}%
\ledsubsubsection* 4741 \leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\ifl@dpairing\else\skipnumbering\fi%
\ledchapter 4742 \pstart%
\ledchapter* 4743 \leavevmode\ifledseccolnolinenumber\skipnumbering\fi\section{##1}{##2}\leavevmode\vspace{2.3ex}
\@patchforledchapter 4744 \vspace{-2\parskip}\vspace{-2\baselineskip}%
\quotation 4745 \ifautopar\else\pstart\fi
\endquotation 4746 }
\quote 4747 \WithSuffix\newcommand\ledsection*[1]{%
\endquote 4748 \led@war@ledxxxDeprecated{section*}%
          4749 \leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\ifl@dpairing\else\skipnumbering\fi%
```

```

4750     \pstart%
4751     \leavevmode\ifledsecnolinenumber\skipnumbering\fi\section*{##1}\leavevmode\vspace{
4752     \vspace{-2\parskip}\vspace{-2\baselineskip}}%
4753     \ifautopar\else\pstart\fi
4754 }
4755 \newcommand{\ledsubsection}[2][]{%
4756     \led@war@ledxxxDeprecated{subsection}%
4757     \leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\ifl@dpairing\else\skipnumbe
4758     \pstart%
4759     \leavevmode\ifledsecnolinenumber\skipnumbering\fi\subsection[##1]{##2}\leavevmode\
4760     \vspace{-2\parskip}\vspace{-2\baselineskip}}%
4761     \ifautopar\else\pstart\fi
4762 }
4763 \WithSuffix\newcommand\ledsubsection*[1]{%
4764     \led@war@ledxxxDeprecated{subsection}%
4765     \leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\ifl@dpairing\else\skipnumbe
4766     \pstart%
4767     \leavevmode\ifledsecnolinenumber\skipnumbering\fi\subsection*{##1}\leavevmode\vspace{
4768     \vspace{-2\parskip}\vspace{-2\baselineskip}}%
4769     \ifautopar\else\pstart\fi
4770 }
4771 \newcommand{\ledsubsubsection}[2][]{%
4772     \led@war@ledxxxDeprecated{subsubsection}%
4773     \leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\ifl@dpairing\else\skipnumbe
4774     \pstart%
4775     \leavevmode\ifledsecnolinenumber\skipnumbering\fi\subsubsection[##1]{##2}\leavevmode\
4776     \vspace{-2\parskip}\vspace{-2\baselineskip}}%
4777     \ifautopar\else\pstart\fi
4778 }
4779 \WithSuffix\newcommand\ledsubsubsection*[1]{%
4780     \led@war@ledxxxDeprecated{subsubsection}%
4781     \leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\ifl@dpairing\else\skipnumbe
4782     \pstart%
4783     \leavevmode\ifledsecnolinenumber\skipnumbering\fi\subsubsection*{##1}\leavevmode\
4784     \vspace{-2\parskip}\vspace{-2\baselineskip}}%
4785     \ifautopar\else\pstart\fi
4786 }
4787 \newcommand\ledchapter[2][]{%
4788     \led@war@ledxxxDeprecated{chapter}%
4789     \ifl@dmemoir%
4790     \gdef\ch@pt@c{##1}%
4791     \fi%
4792     ~\pend\skipnumbering%
4793     \pstart%
4794     \@patchforledchapter\chapter[##1]{##2}%
4795     \pend\pstart}
4796 \WithSuffix\newcommand\ledchapter*[1]{%
4797     \led@war@ledxxxDeprecated{chapter}%
4798     ~\pend\skipnumbering%
4799     \pstart%

```



```

4800     \@patchforledchapter\chapter*{##1}\pend%
4801     \pstart}
4802 \def\@patchforledchapter{
4803     \patchcmd{\@makeschapterhead}{1\par}{1}{-}{-}
4804     \pretocmd{\@makeschapterhead}{\par}{-}{-}
4805     \apptocmd{\@makeschapterhead}{\par}{-}{-}
4806     \patchcmd{\@makeschapterhead}{\vskip 40\p@}{-}{-}{-}
4807     \patchcmd{\@makechapterhead}{1\par}{1}{-}{-}
4808     \pretocmd{\@makechapterhead}{\par}{-}{-}
4809     \apptocmd{\@makechapterhead}{\par}{-}{-}
4810     \patchcmd{\@makechapterhead}{\vskip 40\p@}{-}{-}{-}
4811     \apptocmd{\@chapter}{\par\leavevmode\vspace{40 \p@}\skipnumbering}{-}{-}
4812     \apptocmd{\@schapter}{\par\leavevmode\vspace{40 \p@}\skipnumbering}{-}{-}
4813     \newcommand\beforeledchapter{\pend\cleardoublepage\pstart}
4814     \patchcmd{\chapter}{\cleardoublepage}{\relax}{-}{-}
4815     \patchcmd{\chapter}{\clearpage}{\relax}{-}{-}
4816 }
4817 \ifnoquotation@ \else
4818 \renewcommand{\quotation}{\par\leavevmode%
4819     \parindent=1.5em%
4820     \skipnumbering%
4821     \ifautopar%
4822         \vskip-\parskip%
4823     \else%
4824         \vskip\topsep%
4825     \fi%
4826     \global\leftskip=\leftmargin%
4827     \global\rightskip=\leftmargin%
4828 }
4829 \renewcommand{\endquotation}{\par%
4830     \global\leftskip=0pt%
4831     \global\rightskip=0pt%
4832     \leavevmode%
4833     \skipnumbering%
4834     \ifautopar%
4835         \vskip-\parskip%
4836     \else%
4837         \vskip\topsep%
4838     \fi%
4839 }
4840 \renewcommand{\quote}{\par\leavevmode%
4841     \parindent=0pt%
4842     \skipnumbering%
4843     \ifautopar%
4844         \vskip-\parskip%
4845     \else%
4846         \vskip\topsep%
4847     \fi%
4848     \global\leftskip=\leftmargin%
4849     \global\rightskip=\leftmargin%

```

```

4850 }
4851 \renewcommand{\endquote}{\par%
4852     \global\leftskip=0pt%
4853     \global\rightskip=0pt%
4854     \leavevmode%
4855     \skipnumbering%
4856     \ifautopar%
4857         \vskip-\parskip%
4858     \else%
4859         \vskip\topsep%
4860     \fi%
4861 }
4862 \fi
4863 }

```

`\ledsectnotoc` The `\ledsectnotoc` only disables the `\addcontentsline` macro.

```

4864 \newcommand{\ledsectnotoc}{\let\addcontentsline\@gobblethree}

```

`\ledsectnomark` The `\ledsectnomark` only disables the `\chaptermark`, `\sectionmark` and `\subsectionmark` macros.

```

4865 \newcommand{\ledsectnomark}{%
4866     \let\chaptermark\@gobble%
4867     \let\sectionmark\@gobble%
4868     \let\subsectionmark\@gobble%
4869 }

```

40.2 New commands : `\eledxxx`

The new system of `\eledxxxx` commands to section text work like this:

1. When one of these commands is called, `eledmac` writes to an auxiliary files:
 - The section level.
 - The section title.
 - The side (when `eledpar` is used).
 - The `pstart` where the command is called.
 - If we have starred version or not.
2. `eledmac` adds the title of the section to `pstart`, as normal content. This is to enable critical notes.
3. When \LaTeX is run a other time, this file is read. That:
 - Adds the `pstart` number to a list of `pstarts` where a sectioning command is used.
 - Defines a command, the name of which contains the `pstart` number, and which calls the normal \LaTeX sectioning command.

4. This last command is called when the pstart is effectively printed.

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

```
4870 \ifl@dmemoir
4871   \newcommand\beforeeledchapter{\clearforchapter}
4872 \else
4873   \newcommand\beforeeledchapter{\if@openright\cleardoublepage\else\clearpage\fi}
4874 \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.

```
4875 \newif\if@eled@sectioning
```

`\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.

```
4876 \def\print@rightmargin@eledsection{%
4877   \if@eled@sectioning%
4878     \begingroup%
4879     \if@RTL%
4880       \let\llap\rlap%
4881       \let\leftlinenum\rightlinenum%
4882       \let\leftlinenumR\rightlinenumR%
4883       \let\l@drd@ta\l@dld@ta%
4884       \let\l@drsn@te\l@dlsn@te%
4885     \fi%
4886     \hfill\l@drd@ta \csuse{LR}{\l@drsn@te}%
4887   \endgroup%
4888 \fi%
4889 }%
4890
4891 \def\print@leftmargin@eledsection{%
4892   \if@eled@sectioning%
4893     \leavevmode%
4894     \begingroup%
4895     \if@RTL%
4896       \let\rlap\llap%
4897       \let\rightlinenum\leftlinenum%
4898       \let\rightlinenumR\leftlinenumR%
4899       \let\l@dld@ta\l@drd@ta%
4900       \let\l@dlsn@te\l@drsn@te%
4901     \fi%
4902     \l@dld@ta\csuse{LR}{\l@dlsn@te}%
4903   \endgroup%
4904 \fi%
4905 }%
4906
```

<code>\chapter</code> <code>\M@sect</code> <code>\@mem@old@ssect</code> <code>\@makechapterhead</code> <code>\@makechapterhead</code> <code>\@makeschapterhead</code> <code>\@sect</code> <code>\@ssect</code>	<p>We have to patch L^AT_EX, book and memoir sectioning commands in order to:</p> <ul style="list-style-type: none"> • Disable <code>\edtext</code> inside. • Disable page breaking (for <code>\chapter</code>). • Add line numbers and sidenotes. <p>Unfortunately, Maïeul Rouquette was not able to try if memoir is loaded. That is why <code>eledmac</code> tries to define for both standard class and memoir class.</p>
---	--

```

4907 \catcode'\#=12 % Space NEEDS by \catcode
4908 \AtBeginDocument{%
4909 \patchcmd{\chapter}{\clearforchapter}{%
4910   \if@eled@sectioning\else%
4911     \clearforchapter
4912   \fi%
4913 }
4914 {}
4915 {}
4916
4917 \pretocmd{\M@sect}
4918   {\let\old@edtext=\edtext%
4919    \let\edtext=\dummy@edtext@showlemma%
4920   }
4921 {}
4922 {}
4923
4924 \apptocmd{\M@sect}
4925   {\let\edtext=\old@edtext}
4926 {}
4927 {}
4928
4929 \patchcmd{\M@sect}
4930   { #9}
4931   { #9%
4932     \print@rightmargin@eledsection%
4933   }
4934 {}
4935 {}
4936
4937 \patchcmd{\M@sect}
4938   {\hskip #3\relax}
4939   {\hskip #3\relax%
4940     \print@leftmargin@eledsection%
4941   }
4942 {}
4943 {}
4944
4945
4946

```

```

4947 \patchcmd{\@mem@old@ssect}
4948   {#5}
4949   {#5%
4950   \print@leftmargin@eledsection%
4951   }
4952   {}
4953   {}
4954
4955 \patchcmd{\@mem@old@ssect}
4956   {\hskip #1}
4957   {\hskip #1%
4958   \print@rightmargin@eledsection%
4959   }
4960   {}
4961   {}
4962
4963 \patchcmd{\chapter}{\if@openright\cleardoublepage\else\clearpage\fi}{%
4964   \if@eled@sectioning\else%
4965     \if@openright\cleardoublepage\else\clearpage\fi%No clearpage inside a \eledsection : will keep cr
4966   \fi%
4967   }%
4968   {}%
4969   {}%
4970
4971 \patchcmd{\@makechapterhead}
4972   {#1}
4973   {\print@leftmargin@eledsection%
4974     #1%
4975   \print@rightmargin@eledsection%
4976   }
4977   {}
4978   {}
4979
4980 \patchcmd{\@makechapterhead}% For BIDI
4981   {\if@RTL\raggedleft\else\raggedright\fi}%
4982   {\if@eled@sectioning\else%
4983     \if@RTL\raggedleft\else\raggedright\fi%
4984   \fi%
4985   }%
4986   {}%
4987   {}%
4988
4989 \patchcmd{\@makeschapterhead}
4990   {#1}
4991   {\print@leftmargin@eledsection%
4992     #1%
4993   \print@rightmargin@eledsection%
4994   }
4995   {}
4996   {}

```

```

4997
4998 \pretocmd{\@sect}
4999   {\let\old@edtext=\edtext
5000    \let\edtext=\dummy@edtext@showlemma%
5001   }
5002   {}
5003   {}
5004
5005 \apptocmd{\@sect}
5006   {\let\edtext=\old@edtext}
5007   {}
5008   {}
5009
5010 \pretocmd{\@ssect}
5011   {\let\old@edtext=\edtext%
5012    \let\edtext=\dummy@edtext@showlemma%
5013   }
5014   {}
5015   {}
5016
5017 \apptocmd{\@ssect}
5018   {\let\edtext=\old@edtext}
5019   {}
5020   {}
5021

```

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.

```

5022 \@ifpackageloaded{nameref}{
5023
5024   \patchcmd{\NR@sect}
5025     {#8}
5026     {#8%
5027      \print@rightmargin@eledsection%
5028     }
5029     {}
5030     {}
5031
5032   \patchcmd{\NR@sect}
5033     {\hskip #3\relax}
5034     {\hskip #3\relax%
5035      \print@leftmargin@eledsection%
5036     }
5037     {}
5038     {}
5039
5040   \patchcmd{\NR@ssect}
5041     {#5}
5042     {#5%

```

```

5043 \print@rightmargin@eledsection%
5044 }
5045 {}
5046 {}
5047
5048 \patchcmd{\NR@ssect}
5049 {\hskip #1}
5050 {\hskip #1%
5051 \print@leftmargin@eledsection%
5052 }
5053 {}
5054 {}
5055 }%
5056 {
5057 \patchcmd{\@sect}
5058 {#8}
5059 {#8%
5060 \print@rightmargin@eledsection%
5061 }
5062 {}
5063 {}
5064
5065 \patchcmd{\@sect}
5066 {\hskip #3\relax}
5067 {\hskip #3\relax%
5068 \print@leftmargin@eledsection%
5069 }
5070 {}
5071 {}
5072
5073 \patchcmd{\@ssect}
5074 {#5}
5075 {#5%
5076 \print@rightmargin@eledsection%
5077 }
5078 {}
5079 {}
5080
5081 \patchcmd{\@ssect}
5082 {\hskip #1}
5083 {\hskip #1%
5084 \print@leftmargin@eledsection%
5085 }
5086 {}
5087 {}
5088 }%
5089 }
5090 \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.

```
5091 \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.

```
5092 \newif\if@noeled@sec%
```

```
5093 \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 5094 \newcommand{\eledchapter}[2] [] {%
\eledsubsubsection 5095   #2%
  \eledchapter* 5096   \ifledRcol%
  \eledsection* 5097     \immediate\write\eled@sectioningR@out{%
\eledsubsection* 5098       \string\eled@chapter{#1}{\unexpanded{#2}}{\the\l@dumpstartsR}{R}
\eledsubsubsection* 5099       }%
5100     \else%
5101       \immediate\write\eled@sectioning@out{%
5102         \string\eled@chapter{#1}{\unexpanded{#2}}{\the\l@dumpstartsL}{L}
5103       }%
5104     \fi%
5105 }
5106
5107 \newcommand{\eledsection}[2] [] {%
5108   #2%
5109   \ifledRcol%
5110     \immediate\write\eled@sectioningR@out{%
5111       \string\eled@section{#1}{\unexpanded{#2}}{\the\l@dumpstartsR}{R}
5112     }%
5113   \else%
5114     \immediate\write\eled@sectioning@out{%
5115       \string\eled@section{#1}{\unexpanded{#2}}{\the\l@dumpstartsL}{L}
5116     }%
5117   \fi%
5118 }
5119
5120 \newcommand{\eledsubsection}[2] [] {%
5121   #2%
5122   \ifledRcol%
5123     \immediate\write\eled@sectioningR@out{%
5124       \string\eled@subsection{#1}{\unexpanded{#2}}{\the\l@dumpstartsR}{R}
5125     }%
5126   \else%
5127     \immediate\write\eled@sectioning@out{%
5128       \string\eled@subsection{#1}{\unexpanded{#2}}{\the\l@dumpstartsL}{L}
5129     }%
5130   \fi%
5131 }
5132 \newcommand{\eledsubsubsection}[2] [] {%
```



```

5133 #2%
5134 \ifledRcol%
5135   \immediate\write\eled@sectioningR@out{%
5136     \string\eled@subsubsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR}{*}{R}
5137   }%
5138 \else%
5139   \immediate\write\eled@sectioning@out{%
5140     \string\eled@subsubsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL}{*}{L}
5141   }%
5142 \fi%
5143 }
5144
5145
5146 \WithSuffix\newcommand\eledchapter*[2] [] {%
5147   #2%
5148   \ifledRcol%
5149     \immediate\write\eled@sectioningR@out{%
5150       \string\eled@chapter{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR}{*}{R}
5151     }%
5152   \else%
5153     \immediate\write\eled@sectioning@out{%
5154       \string\eled@chapter{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL}{*}{L}
5155     }%
5156   \fi%
5157 }
5158
5159 \WithSuffix\newcommand\eledsection*[2] [] {%
5160   #2%
5161   \ifledRcol%
5162     \immediate\write\eled@sectioningR@out{%
5163       \string\eled@section{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR}{*}{R}
5164     }%
5165   \else%
5166     \immediate\write\eled@sectioning@out{%
5167       \string\eled@section{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL}{*}{L}
5168     }%
5169   \fi%
5170 }
5171
5172 \WithSuffix\newcommand\eledsubsection*[2] [] {%
5173   #2%
5174   \ifledRcol%
5175     \immediate\write\eled@sectioningR@out{%
5176       \string\eled@subsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR}{*}{R}
5177     }%
5178   \else%
5179     \immediate\write\eled@sectioning@out{%
5180       \string\eled@subsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL}{*}{L}
5181     }%
5182   \fi%

```

```

5183 }
5184
5185 \WithSuffix\newcommand\eledsubsubsection*[2][]{%
5186   #2%
5187   \ifledRcol%
5188     \immediate\write\eled@sectioningRout{%
5189       \string\eled@subsubsection{#1}{\unexpanded{#2}}{\the\l@dumpstartsR}{*}{R}
5190     }%
5191   \else%
5192     \immediate\write\eled@sectioningLout{%
5193       \string\eled@subsubsection{#1}{\unexpanded{#2}}{\the\l@dumpstartsL}{*}{L}
5194     }%
5195   \fi%
5196 }

```

<code>\eled@chapter</code> <code>\eled@section</code> <code>\eled@subsection</code> <code>\eled@subsubsection</code>	<p>The sectioning macros, called in the auxiliary file. They have five arguments:</p> <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.
---	---

```

5197 \def\eled@chapter#1#2#3#4#5{%
5198   \ifstrempy{#4}%
5199   {%
5200     \ifstrempy{#1}%
5201     {%
5202       \global\csdef{eled@sectioning@#3#5}{\let\edtext=\dummy@edtext@showlemma\chapter{#1}}
5203       \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext{}\chaptermark{#2}}%
5204       }%Need for \pairs, because of using parbox.
5205     {%
5206       \global\csdef{eled@sectioning@#3#5}{\let\edtext=\dummy@edtext@showlemma\chapter{#1}}
5207       \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext{}\chaptermark{#2}}%Need
5208       }%
5209     }%
5210   }%
5211   \ifstrempy{#1}%
5212   {\global\csdef{eled@sectioning@#3#5}{\let\edtext=\dummy@edtext@showlemma\chapter*{#1}}
5213   {\global\csdef{eled@sectioning@#3#5}{\let\edtext=\dummy@edtext@showlemma\chapter*{#1}}
5214   }%
5215   \listcsadd{eled@sections#5@@}{#3}%
5216   }
5217 \def\eled@section#1#2#3#4#5{%
5218   \ifstrempy{#4}%
5219   {\ifstrempy{#1}%
5220     {%

```

```

5221     \global\csdef{eled@sectioning@#3#5}{\section{#2}}%
5222     \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext{}\sectionmark{#2}}%Need for \pair
5223     }%
5224     {%
5225     \global\csdef{eled@sectioning@#3#5}{\section{#1}{#2}}%
5226     \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext{}\sectionmark{#1}}%Need for \pair
5227     }%
5228     }%
5229     {\ifstrempy{#1}%
5230     {\global\csdef{eled@sectioning@#3#5}{\section*{#2}}}%
5231     {\global\csdef{eled@sectioning@#3#5}{\section*{#1}{#2}}}%Bug in LaTeX!
5232     }
5233     \listcsadd{eled@sections#5@@}{#3}%
5234     }
5235 \def\eled@subsection#1#2#3#4#5{%
5236     \ifstrempy{#4}%
5237     {\ifstrempy{#1}%
5238     {%
5239     \global\csdef{eled@sectioning@#3#5}{\subsection{#2}}%
5240     \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext{}\csuse{subsectionmark}{#2}}%Need
5241     }%
5242     {%
5243     \global\csdef{eled@sectioning@#3#5}{\subsection{#1}{#2}}%
5244     \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext{}\csuse{subsectionmark}{#1}}%Need
5245     }%
5246     }%
5247     {\ifstrempy{#1}%
5248     {\global\csdef{eled@sectioning@#3#5}{\subsection*{#2}}}%
5249     {\global\csdef{eled@sectioning@#3#5}{\subsection*{#1}{#2}}}%Bug in LaTeX!
5250     }
5251     \listcsadd{eled@sections#5@@}{#3}%
5252     }
5253 \def\eled@subsubsection#1#2#3#4#5{%
5254     \ifstrempy{#4}%
5255     {\ifstrempy{#1}%
5256     {\global\csdef{eled@sectioning@#3#5}{\subsubsection{#2}}}%
5257     {\global\csdef{eled@sectioning@#3#5}{\subsubsection{#1}{#2}}}%
5258     }%
5259     {\ifstrempy{#1}%
5260     {\global\csdef{eled@sectioning@#3#5}{\subsubsection*{#2}}}%
5261     {\global\csdef{eled@sectioning@#3#5}{\subsubsection*{#1}{#2}}}%Bug in LaTeX!
5262     }
5263     \listcsadd{eled@sections#5@@}{#3}%
5264     }
5265

```

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.

```
5266 \def\normal@page@break{}
```

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

```
5267 \def\l@prev@pb{}
```

```
5268 \def\l@prev@nopb{}
```

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

```
5269 \newcommand{\ledpb}{\write\linenum@out{\string\led@pb}}
```

```
5270 \newcommand{\ledpbnum}[1]{\write\linenum@out{\string\led@pbnum{#1}}}
```

```
5271 \newcommand{\lednopb}{\write\linenum@out{\string\led@nopb}}
```

```
5272 \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` adds the argument in the `\prev@pb` list. The `\led@nopb` adds the absolute line number in the `\prev@nopb` list. The `\led@nopbnum` adds the argument in the `\prev@nopb` list.

```
5273 \newcommand{\led@pb}{\listxadd{\l@prev@pb}{\the\absline@num}}
```

```
5274 \newcommand{\led@pbnum}[1]{\listxadd{\l@prev@pb}{#1}}
```

```
5275 \newcommand{\led@nopb}{\listxadd{\l@prev@nopb}{\the\absline@num}}
```

```
5276 \newcommand{\led@nopbnum}[1]{\listxadd{\l@prev@nopb}{#1}}
```

`\ledpbsetting` The `\ledpbsetting` macro only changes the value of `\led@pb@macro`, for which the default value is `before`.

```
5277 \def\led@pb@setting{before}
```

```
5278 \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`.

```
5279 \newcommand{\led@check@pb}{\xifinlist{\the\absline@num}{\l@prev@pb}{\pagebreak[4]}{}}
```

```

5280 \newcommand{\led@check@nopb}{%
5281   \IfStrEq{\led@pb@setting}{before}{%
5282     \xifinlist{\the\absline@num}{\l@prev@nopb}%
5283       {\numdef{\abs@prevline}{\the\absline@num-1}%
5284       \xifinlist{\abs@prevline}{\normal@page@break}%
5285       {\nopagebreak[4]\enlargethispage{\baselineskip}}%
5286       {}}%
5287     {}}%
5288   {}%
5289 {}%
5290 \IfStrEq{\led@pb@setting}{after}{%
5291   \xifinlist{\the\absline@num}{\l@prev@nopb}{%
5292     \xifinlist{\the\absline@num}{\normal@page@break}%
5293     {\nopagebreak[4]\enlargethispage{\baselineskip}}%
5294     {}}%
5295 }%
5296   {}}%
5297   {}%
5298   {}%
5299 }

```

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.

```

5300 \newcommand{\check@pb@in@verse}{%
5301   \ifinstanza\iflednopbinverse\ifinserthangingsymbol% Using stanzas and enabling page breaks in ver
5302     \ifnum\page@num=\last@page@num\else%If we have change page
5303       \IfStrEq{\led@pb@setting}{before}{%
5304         \numgdef{\abs@line@verse}{\the\absline@num-1}%
5305         \ledpbnum{\abs@line@verse}%
5306       }{}%
5307       \IfStrEq{\led@pb@setting}{after}{%
5308         \numgdef{\abs@line@verse}{\the\absline@num-1}%
5309         \lednopbnum{\abs@line@verse}%
5310       }{}%
5311     \fi%
5312   \fi\fi\fi%
5313 }

```

43 The End

i/code_i

Appendix A Some things to do when changing version

Appendix A.1 Migrating from edmac

If you have never used 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 ededmac.

The package still provides the original `\text` command, but it is (a) deprecated, and (b) its name has been changed³⁰ 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:

<code>I saw my friend \critext{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 `\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</u> Smith] 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 `⟨lemma⟩` argument of another `\critext` must end there, too. (The `\lemma` and `\linenum` commands may be used to generate overlapping notes if necessary.)

³⁰A 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 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\}\variant{Jones}/`. This of course destroys the ability to use `<` in any new macro definitions, so long as it remains in effect; hence it should be used with care.

Changing the character at the end of the command requires more work:

```
\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.

Appendix A.2 Migration from ledmac to eledmac

In eledmac, some changes were made in the code to allow for easy customization. This can cause problems for people who have made their own customizations. The next sections explain how to correct this.

If you created your own series using `\addfootins` and `\addfootinsX`, you should instead use the `\newseries` command (see 4.7 p.26). You must delete your `\Xfootnote` command.

If you customized the `\XXXXXXfmt` command, you should see if commands for display options (4.4 p.20) and options in `\Xfootnote` (4.1 p.17) can't do the same things. If not, you can add a new ticket in Github to request a new function it³¹.

If for some reason you don't want to make the modifications to use eledmac new functions, you can continue to use your own `\XXXXXXfmt` command, but you must replace:

```
\renewcommand*{XXXXXfmt}[3]
```

³¹<https://github.com/maieu/ledmac/issues>

with

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

If you don't do that, you will see a spurious [X], where X is series letter.

If you used a `\protect` command inside a `\footnote` command inside a numbered section, you must change the `\protect` to `\noexpand`. If you don't, the command after the `\protect` won't be displayed.

Appendix A.3 Migration to eledmac 1.5.1

The version 1.5.1 corrects a bug with `stanzaindentsrepetition` (cf. p. 27). This bug had two consequences:

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

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

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

This code, in a version older than 1.5.1, made that the first verse had an indent of 0, the second verse of 1, the third verse of 0, the fourth verse of 1 etc.

But instead the code should have assigned the reverse: the first verse had an indent of 1, the second verse of 0, the third verse of 1, the fourth verse of 0 etc.

So version 1.5.1 corrected this bug. If you want to keep the older presentation, you must change:

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

by:

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

Appendix A.4 Migration to eledmac 1.12.0

The migration to eledmac 1.12.0 is easy:

- You must delete all the auxiliary files, and so one, make the normal three runs.
- If you have modified `\l@reg`, which is not advisable, you must rename it to `\@nl@reg`.

Anyway, there is another problem. If you have text in brackets just after `\pstart` or `\pend`, the text will be considered an optional argument of `\pstart` or `\pend` (see 3.2.2, p. 11). In this case, just add a `\relax` between `\pstart/\pend` and the brackets.

The version 1.12.0 adds new best way to manage section title inside numbered text. Please read § 14 (p. 39).

Appendix A.5 Migration to eledmac 17.1

The version change the default behavior of `\pstartinfofootnote`. Henceforth, the `pstart` will be printed if footnote only for the section of text where you have called `\numberpstarttrue`.

We don't see any reason to print it in other section. However, if you want to print the `pstart` number in all footnote, with or without `\numberpstarttrue`, you can use `\pstartinfofootnoteeverytime`.

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	
<code>\#</code>	4907, 5090
<code>\&</code>	22, 3966, 3970, 3971, 4033, 4048, 4069, 4071
<code>\@ledleftnote</code>	4111, 4119
<code>\@ledrightnote</code>	4110, 4118
<code>\@ledsidenote</code>	4112, 4120
<code>\@line</code>	1928
<code>\@wrindexm@m</code>	3793, 3795, 3798, 3805, 3807, 3810, 3815, 3817, 3820
<code>\@EDROWFILL@</code>	4303, <u>4518</u>
<code>\@M</code>	1928, 4026, 4038

- \@MM 1579
- \@adv 588, 797
- \@arabic 1027
- \@auxout 3265, 3278, 3792, 3794, 3797,
3804, 3806, 3809, 3814, 3816, 3819
- \@botlist 3213, 3215
- \@cclv 3118, 3122, 3123, 3211, 3212, 3240
- \@chapter 4811
- \@checkend 3947
- \@colht 3099, 3216, 3228
- \@colroom 3216
- \@combinefloats 3094
- \@currentlabel
..... 1066, 2163, 2287, 2354, 2471
- \@currenvir 3931, 3934, 3935
- \@currlist 3217, 3220
- \@dbldeferlist 3226, 3231, 3233
- \@dblfloatplacement 3230
- \@dbltoplist 3226, 3227
- \@deferlist 3213, 3222, 3223
- \@docclearpage 3200
- \@edindex@fornote@true 3750
- \@edindex@hyperref 3838, 3890
- \@edrowfill@ 4518
- \@edtext@false 874
- \@edtext@true 851
- \@ehb 3219
- \@ehd 173, 176, 179
- \@eled@sectioningfalse 1177
- \@eled@sectioningtrue 1175
- \@emptytoks 3922, 3932
- \@fnpos 2560, 3136, 3139
- \@footnotemark 2092
- \@footnotetext
2091, 2105, 2120, 3651, 3687, 3714
- \@freelist 3092
- \@gobble . 702, 830, 831, 2619, 3944,
4113–4115, 4282, 4293, 4866–4868
- \@gobblefour 181, 2793
- \@gobblethree 181, 4864
- \@gobbletwo 707
- \@h 1926
- \@hilfs@count 4218
- \@idxfile . 3782, 3793, 3795, 3798,
3805, 3807, 3810, 3815, 3817, 3820
- \@ifclassloaded
..... 44, 2090, 3167, 3192, 3867
- \@ifnextchar 3772, 4203
- \@ifpackageloaded 47, 49,
3114, 3729, 3732, 3868, 3871, 5022
- \@iiiminipage 3640
- \@iiiparbox 3667
- \@index@command
. 3754, 3756–3758, 3761–3763,
3834, 3835, 3844, 3847, 3902, 3906
- \@index@command@ 3757, 3758, 3762, 3763
- \@index@parenthesis
..... 3755, 3759, 3764, 3901, 3905
- \@index@txt 3753,
3834, 3835, 3838, 3844, 3847, 3919
- \@indexfile 3843, 3846, 3850
- \@inputcheck 479
- \@insert 1465–1467, 1501–1503
- \@k 1926
- \@kludgeins 3096, 3164
- \@l@dttempcnta . . 185, 622, 624, 626,
627, 1248, 1249, 1251, 1253,
1256, 1257, 1272, 1308–1312,
1314, 1321–1325, 1327, 1330,
1333, 1335, 1339, 1370, 1374,
1378, 1385, 1389, 1393, 1474,
1478, 1482, 1485, 1488, 1491, 1492
- \@l@dttempcntb
. 185, 338, 339, 344, 348, 352,
356, 359, 382, 383, 390, 394,
398, 400, 408, 409, 1306, 1318,
1339, 1347–1349, 1351, 1370,
1374, 1378, 1385, 1389, 1393,
1423–1425, 1427, 1480, 1481,
3386, 3388, 3390, 3396, 3400,
3404, 3408, 3411, 3498–3500,
3503–3505, 3508, 3516–3518,
3521–3523, 3526, 3576–3578, 3580
- \@lab 706, 3256, 3269, 3311
- \@latexerr 3219
- \@led@extranofeet . . 3189, 3198, 3206
- \@led@nofootfalse 3202–3204
- \@led@nofoottrue 3201
- \@led@testifnofoot 3200
- \@lemmacommand@false 875
- \@lemmacommand@true 903
- \@line@num 4214, 4312
- \@listdepth 3653
- \@lock . 242, 459, 536, 538, 540, 553,
655, 656, 658, 659, 675, 676,
678, 1141, 1218, 1278, 1280,
1281, 1283, 1382, 1397, 1399, 1401
- \@lopL 572, 702
- \@lopR 572
- \@makechapterhead . . 4807–4810, 4907

- \@makecol 3171
 - \@makefcolumn .. 3222, 3223, 3231, 3233
 - \@makeschapterhead .. 4803–4806, 4907
 - \@makespecialcolbox 3097
 - \@maxdepth 3112, 3121
 - \@mem@extranofeet 3193
 - \@mem@nofootfalse 3194, 3195
 - \@mem@old@ssect 4907
 - \@midlist 3092, 3093
 - \@minipagefalse 3664
 - \@minipagerestore 3654
 - \@minus 2713, 2714,
4741, 4749, 4757, 4765, 4773, 4781
 - \@mpargs 3644, 3667
 - \@mpfn 3650, 3686, 3713
 - \@mpfnpos 2560, 3600, 3603
 - \@mpfootins 3660,
3670, 3676, 3678, 3682, 3692, 3719
 - \@mpfootnotetext ... 3651, 3687, 3714
 - \@mplistdepth 3653
 - \@nameuse 415, 417, 1584,
1585, 1722, 1829, 1830, 1875,
1983, 2051, 2135, 2139, 2141,
2150, 2153, 2154, 2160, 2164,
2168, 2172, 2195, 2200, 2220,
2232, 2238, 2284, 2288, 2298,
2306, 2351, 2355, 2365, 2373,
2435, 2449, 2457, 2458, 2463,
2472, 2476, 2489, 2594, 2595,
2597, 2598, 2603, 3179, 3180,
3182, 3183, 3185, 3187, 3624, 3626
 - \@next@page 741, 742
 - \@nl .. 511, 742, 744, 746, 753, 757, 760
 - \@nl@reg 511
 - \@nobreakfalse 1040, 1174
 - \@nobreaktrue 1038, 1042, 1174
 - \@noeled@ssecttrue 5093
 - \@noneed@Footnotefalse 873
 - \@noneed@Footnotetrue 2786
 - \@nowrindex 3781
 - \@oldnobreak 1038, 1040, 1094
 - \@opcol 3223, 3241
 - \@opxtrafeetii 3148, 3149, 3178
 - \@outputbox
. 2577, 2578, 2592, 2593, 3099,
3101, 3102, 3118, 3120, 3146, 3147
 - \@outputpage 3232
 - \@parboxrestore 1589, 2158, 3649
 - \@patchforledchapter 4738
 - \@pboxswfalse 3642
 - \@pend 572
 - \@pendR 572
 - \@plus 1600,
2294, 2361, 2713, 2714, 4741,
4743, 4749, 4751, 4757, 4759,
4765, 4767, 4773, 4775, 4781, 4783
 - \@ref 690, 768, 777
 - \@ref@reg 692
 - \@reinserts 3172
 - \@schapter 4812
 - \@ssect 4907
 - \@series 473, 477,
2580, 2585, 2669, 2671, 2801,
2810, 2811, 2817, 2819, 2843,
2857, 3151, 3158, 3197, 3205,
3610, 3631, 3636, 3639, 4100,
4106, 4280, 4285, 4291, 4296, 4311
 - \@set 603, 802
 - \@setminipage 3655
 - \@showidx 3789
 - \@ssect 4907
 - \@startstanza 4028
 - \@stopstanza 4028
 - \@sw 707, 928, 945, 963
 - \@tag 839, 852,
904, 2113, 2114, 2742, 2755, 2790
 - \@tempboxa 3211, 3212, 3645, 3667
 - \@tempdima 3122, 3643, 3647
 - \@templ@d 3566, 3568
 - \@templ@n 3567, 3568
 - \@textbottom 3104
 - \@texttop 3100
 - \@toplist 3213, 3214
 - \@whilenum 4493
 - \@whilesw 3223, 3232
 - \@wredindex 3828, 4210
 - \@x@sf 2084, 2087, 2095, 2101, 2125, 2131
 - \@xloop 1499, 1506
 - \@xympar 3374
 - \^ 505
 - _ 3894, 3896, 4526, 4531, 4539
- A**
- \abs@line@verse 5304, 5305, 5308, 5309
 - \abs@prevline 5283, 5284
 - \absline@num
. 236, 458, 516, 519, 522, 569,
617, 620, 629, 643, 665, 687,

697, 739, 740, 749, 751, 965,
966, 984–986, 1209, 1230, 1231,
1239, 1464, 5273, 5275, 5279,
5282, 5283, 5291, 5292, 5304, 5308
`\absline@numR` 973, 974, 998–1000
 Abu Kamil Shuja' b. Aslam 4
`\actionlines@list`
 461, 485, 488, 495, 617,
 620, 629, 643, 665, 687, 1261, 1264
`\actions@list`
 . 461, 489, 496, 618, 627, 631,
 633, 645, 654, 667, 674, 688, 1265
`\add@inserts` 1158, 1172, 1453
`\add@inserts@next` 1453
`\add@penalties` 1169, 1474
`\addcontentsline` 4864
`\addfootins` 3175
`\addfootinsX` 2588
`\addtocounter` 1095
`\addtol@denvbody` . . . 3926, 3948, 3950
 Adelard II 4
`\advancelabel@refs` . . 3263, 3276, 3284
`\advanceline` . . . 11, 89, 92, 797, 820, 831
`\advancepageno` 3087
`\Aendnote` 13
`\affixline@num` 1147, 1300
`\affixpstart@num` 1154, 1412
`\affixside@note` 1158, 1172, 3553
`\Afootnote` 13
`\afterlemmaseparator` 17
`\afternote` 19
`\afternumberinfootnote` 17
`\afterruleX` 20
`\aftersymlinenum` 17
`\afterXrule` 20
`\allowbreak` 1976, 2040, 2299, 2366
`\ampersand` 24, 3966, 4071
`\appto` 3559, 3560
`\apptocmd` 2091, 2110, 4805,
 4809, 4811, 4812, 4924, 5005, 5017
`\at@every@pend` 1096, 1099
`\at@every@pstart` . . . 1023, 1024, 1033
`\AtBeginDocument`
 3114, 3307, 3324, 3727, 4908
`\AtEveryPend` 7, 1099
`\AtEveryPstart` 7, 1022
`\autopar` 7, 110, 285, 1108
`\autopar@pausetrue` 281
`\autopar@false` 271, 1109
`\autopar@true` 1122

B

`\ballast` 37
`\ballast@count` . . 1225, 1228, 1233, 1474
 Beeton, Barbara Ann Neuhaus Friend 8
`\beforeeledchapter` 4870
`\beforeeledchapter` 4813
`\beforelemmaseparator` 17
`\beforenotesX` 19
`\beforenumberinfootnote` 16
`\beforesymlinenum` 17
`\beforeXnotes` 19
`\beginnumbering` 6, 204, 302, 1045, 1119
`\beginnumberingR` 1114
`\Bendnote` 13
`\Bfootnote` 13
`\bfseries` 1027
`\bhooknoteX` 19
`\bhookXendnote` 19
`\bhookXnote` 19
`\body` 1507, 1508, 3968, 4070
`\bodyfootmarkA` 30
`\box` . . . 1201, 1203, 1824, 1839, 1908,
 1927, 2453, 2467, 3118, 3212, 3240
`\boxfootnotennumbers` . . 3046, 3051, 3056
`\boxlinenum` 17
`\boxmaxdepth` 3121
`\boxsymlinenum` 17
 Bredon, Simon 4
 Breger, Herbert 2, 5, 182
 Brey, Gerhard 4
`\brokenpenalty` 1104
 Burt, John 3
 Busard, Hubert L. L. 4
`\bypage@false` 306, 320, 326
`\bypage@true` 306, 314
`\bypstart@false` 306, 315, 327
`\bypstart@true` 306, 321

C

`\c@addcolcount` 4487
`\c@ballast` 1225, 1233
`\c@firstlinenum`
 365, 1320, 1322, 1325, 1327
`\c@firstsublinenum`
 369, 1307, 1309, 1312, 1314
`\c@labidx` 3735
`\c@linenumincrement` . . 365, 1323, 1324
`\c@mpfootnote` 3650, 3686, 3713
`\c@page` 744,
 746, 752, 755, 757, 760, 3432, 3440

- \c@pstart 1027, 3279
- \c@pstartR 3266
- \c@sublinenumincrement 369, 1310, 1311
- \Cendnote 13
- \centerline 4423, 4428,
4434, 4439, 4445, 4450, 4672, 4674
- \Cfootnote 13
- \ch@ck@l@ck 1337, 1366
- \ch@cksub@l@ck 1316, 1366
- \ch@pt@c 4790
- \changes 843, 2824, 2827, 2828
- \chapter 4794, 4800, 4814,
4815, 4907, 5202, 5206, 5212, 5213
- \chaptermark 4866, 5203, 5207
- \char 3966
- \chardef 2664, 3968, 3970
- \check@pb@in@verse 1146, 5300
- Chester, Robert of 4
- Claassens, Geert H. M. 4
- class 1 feet 134, 151
- class 2 feet 151, 152
- \cleaders 4482
- \cleardoublepage
.... 4813, 4814, 4873, 4963, 4965
- \clearforchapter ... 4871, 4909, 4911
- \closeout 273, 726, 733, 2608
- \clubpenalty 1104, 1478
- \color@begingroup
1590, 1835, 2159, 2463, 3125, 3646
- \color@endgroup
1591, 1835, 2160, 2463, 3129, 3665
- \columns@position 223, 224,
295, 296, 2527, 2540, 2550, 2556
- \columnwidth 1588, 1795,
2157, 2419, 2532, 2545, 3648, 3701
- \content 2730, 2742, 2755, 2770, 2774,
2787, 2790, 3453, 3456, 3460,
3468, 3471, 3475, 3483, 3486, 3490
- Copernicus, Nicolaus 4
- \count 1763, 1768, 1780, 1786, 1951,
1954, 2020, 2048, 2252, 2257,
2273, 2276, 2341, 2344, 2405, 2410
- \countdef 3087
- \cr 1929, 1932
- \create@edindex@for@memoir 3769, 3873
- \create@edindex@notfor@memoir ..
..... 3827, 3869, 3872, 3876
- \CRITEXT 4194
- \critext 218, 832, 848, 4196, 4278, 4307
- \cs 843, 844,
2798, 2827–2829, 2951, 3054, 3071
- \csdef 4284, 5202,
5203, 5206, 5207, 5212, 5213,
5221, 5222, 5225, 5226, 5230,
5231, 5239, 5240, 5243, 5244,
5248, 5249, 5256, 5257, 5260, 5261
- \csgdef ... 1756, 1774, 1940, 2009,
2242, 2263, 2331, 2398, 2675–
2693, 2699, 2701, 2702, 2704–
2706, 2708–2719, 2765, 2782, 2851
- \cslet 2707, 2793
- \csletcs 2797, 3609, 3635,
4094, 4104, 4279, 4290, 4295, 4310
- \csnumdef 1080, 1082
- \csnumgdef 927, 944, 968, 976
- \csundef 472, 1178
- \csuse 223, 224,
295, 296, 927, 928, 944, 945,
970, 978, 986, 1000, 1176, 1568,
1569, 1586, 1587, 1598, 1604,
1607–1609, 1615, 1701, 1712,
1716, 1734, 1740, 1743, 1764,
1765, 1769, 1770, 1781, 1782,
1787, 1788, 1791, 1804, 1811,
1816, 1817, 1831, 1832, 1850,
1857–1859, 1867, 1868, 1887,
1893, 1899, 1900, 1915, 1917–
1919, 1921, 1959, 1964, 1968,
1972–1974, 1978, 1995, 2001,
2003, 2023, 2028, 2032, 2036–
2038, 2041, 2042, 2063, 2069,
2071, 2142, 2143, 2150, 2155,
2156, 2167, 2168, 2178, 2190,
2212, 2218, 2224, 2253, 2254,
2258, 2259, 2281, 2291, 2292,
2298, 2301, 2318, 2324, 2349,
2357, 2359, 2365, 2368, 2385,
2391, 2406, 2407, 2411, 2412,
2415, 2428, 2439, 2445, 2446,
2459, 2460, 2476, 2485, 2502,
2508, 2515, 2527, 2540, 2550,
2556, 2566, 2572, 2579, 2584,
2616, 2704, 2705, 2741, 2754,
2760, 2772–2774, 2837, 2838,
2853, 2854, 3001, 3007, 3018,
3020–3024, 3045, 3057, 3061,
3081, 3085, 3150, 3157, 3194,
3195, 3203, 3204, 3621, 3629,
3638, 3767, 3768, 3880, 4096,

- 4097, 4284, 4886, 4902, 5240, 5244
`\csxdef` 965, 973, 1530,
 1532, 1536, 1538, 1545, 1548,
 1551, 1556, 1559, 1562, 1796, 3037
`\ctab` 4076, 4631, 4722
`\ctabtext` 4080, 4641, 4726
- D**
- `\dcolerr` 4170, 4182
`\dcoli` ... 4140, 4172, 4173, 4530, 4535
`\dcolii` 4141, 4173
`\dcoliii` 4142, 4173
`\dcoliv` 4143, 4174
`\dcolix` 4148, 4175
`\dcolv` 4144, 4174
`\dcolvi` 4145, 4174
`\dcolvii` 4146, 4175
`\dcolviii` 4147, 4175
`\dcolx` 4149, 4175
`\dcolxi` 4150, 4176
`\dcolxii` 4151, 4176
`\dcolxiii` 4152, 4176
`\dcolxiv` 4153, 4177
`\dcolxix` 4158, 4178
`\dcolxv` 4154, 4177
`\dcolxvi` 4155, 4177
`\dcolxvii` 4156, 4178
`\dcolxviii` 4157, 4178
`\dcolxx` 4159, 4178
`\dcolxxi` 4160, 4179
`\dcolxxii` 4161, 4179
`\dcolxxiii` 4162, 4179
`\dcolxxiv` 4163, 4180
`\dcolxxix` 4168, 4181
`\dcolxxv` 4164, 4180
`\dcolxxvi` 4165, 4180
`\dcolxxvii` 4166, 4181
`\dcolxxviii` 4167, 4181
`\dcolxxx` 4169, 4181
`\DeclareOption` 12–18
Dekker, Dirk-Jan 3, 39
`\Dendnote` *13*
`\Dfootnote` *13*
`\dimen` 788, 789,
 791–793, 795, 1764, 1769, 1781,
 1787, 1793–1795, 1797, 1934–
 1936, 1952, 1955, 2021, 2049,
 2253, 2258, 2274, 2277, 2342,
 2345, 2406, 2411, 2417–2419, 2422
`\dimen@` 3101, 3103
`\dimexpr` 4000
`\dingdef` 3624, 3626, 3676, 3678
`\disable@familiarnotes` .. 4092, 4124
`\disable@notes` 4122, 4327,
 4346, 4362, 4378, 4393, 4410, 4604
`\disable@sidenotes` 4109, 4123
`\disablel@dtabfeet`
 4325, 4344, 4360, 4376,
 4391, 4408, 4537, 4544, 4549,
 4557, 4562, 4570, 4588, 4605, 4729
`\displaystyle` 4228, 4328,
 4331, 4363, 4366, 4394, 4397,
 4537, 4549, 4562, 4652, 4704, 4705
`\displaywidowpenalty` 1105
`\divide` .. 1310, 1323, 1795, 1935, 2419
`\do@actions` 1210, 1237
`\do@actions@fixedcode` ... 1258, 1271
`\do@actions@next` 1237
`\do@ballast` 1211, 1225
`\do@insidelinehook` .. 1156, 1189, 1191
`\do@line` 1083, 1131
`\do@linehook` 1135, 1188, 1191
`\do@lockoff` 664
`\do@lockoffL` 664
`\do@lockon` 635
`\do@lockonL` 635
`\docsvlist` 1526, 2667, 2846, 2860
`\doedindexlabel` 3740, 3783, 3860, 4207
`\doendnotes` *26*, 2657
`\doinsidelinehook` 1188
`\dolinehook` 1188
`\dolistloop`
 .. 473, 477, 2580, 2585, 2810,
 2817, 2843, 2857, 3151, 3158,
 3197, 3205, 3564, 3584, 3591,
 3610, 3631, 3636, 3639, 4100,
 4106, 4280, 4285, 4291, 4296, 4311
`\doreinextrafeeti` ... 2576, 2596, 3153
`\doreinextrafeetii` .. 3154, 3156, 3181
`\dosplits` 1926
Downes, Michael 38, 111, 113
`\doxtrafeet` 3135
`\doxtrafeeti`
 2576, 2591, 3137, 3140, 3141
`\doxtrafeetii` .. 3137, 3140, 3141, 3145
`\dp` 1580,
 1822, 1837, 2451, 2465, 3101, 3122
`\dummy@edtext` 824, 834,
 5203, 5207, 5222, 5226, 5240, 5244

- \dummy@edtext@showlemma 825, 4919,
5000, 5012, 5202, 5206, 5212, 5213
- \dummy@ref 691, 701
- \dummy@text 823, 832
- E**
- \edaftertab
.. 33, 196, 4082, 4577, 4613, 4631
- \edarrayc (environment) 31, 4721
- \edarrayl (environment) 31, 4721
- \edarrayr (environment) 31, 4721
- \EDATAB 4671, 4679
- \edatab 4083, 4677
- \edatabell 4084, 4661
- \edatleft 33, 4085, 4462
- \edatright 33, 4086, 4470
- \edbeforetab
.. 33, 196, 4081, 4577, 4613, 4631
- \edfilldimen
.... 4484, 4494, 4495, 4497, 4520
- \edfont@info 893, 896, 900
- \EDINDEX 4200
- \edindex 30, 3769, 4200,
4281, 4286, 4292, 4297, 4309,
4318, 4337, 4355, 4371, 4386, 4403
- \edindexlab 31, 3735,
3741, 3744, 3903, 3907, 3912, 3914
- \EDLABEL 4198
- \edlabel 27, 830, 3250,
3741, 4198, 4299, 4315, 4317,
4336, 4354, 4370, 4385, 4402,
4536, 4543, 4548, 4556, 4561, 4569
- \edlineref 27, 3323
- \edmakelabel 28, 3372
- \edpageref 27, 3320
- \edrowfill . 32, 4089, 4300, 4303, 4518
- \EDTAB 4667, 4703
- \edtabcolsep 32, 4257,
4332, 4351, 4366, 4382, 4398,
4415, 4495, 4551, 4564, 4573, 4689
- \EDTABINDENT 4684, 4697
- \edtabindent 4525,
4529, 4534, 4545, 4558, 4571, 4693
- \EDTABtext 4711
- \edtabularc (environment) 31, 4725
- \edtabularl (environment) 31, 4725
- \edtabularr (environment) 31, 4725
- \EDTEXT 4194
- \edtext . 12, 64, 834, 848, 849, 2107,
2236, 3428, 3429, 3447, 4194,
4289, 4308, 4918, 4919, 4925,
4999, 5000, 5006, 5011, 5012,
5018, 5202, 5203, 5206, 5207,
5212, 5213, 5222, 5226, 5240, 5244
- \edvertdots 34, 4088, 4481
- \edvertline 34, 4087, 4479
- \Eendnote 13
- \Efootnote 13
- \eled@chapter
.... 5098, 5102, 5150, 5154, 5197
- \eled@section
.... 5111, 5115, 5163, 5167, 5197
- \eled@sectioning@out
.. 230, 273, 5091, 5101, 5114,
5127, 5139, 5153, 5166, 5179, 5192
- \eled@sectioningR@out 5097, 5110,
5123, 5135, 5149, 5162, 5175, 5188
- \eled@sections@@ 227, 1148, 1174
- \eled@subsection
.... 5124, 5128, 5176, 5180, 5197
- \eled@subsubsection
.... 5136, 5140, 5189, 5193, 5197
- \eledchapter 5094
- \eledchapter* 5094
- \Eledmac 2795
- \eledmac@error . 56, 58, 60, 62, 64,
76, 99, 102, 105, 108, 110, 162,
164, 167, 169, 171, 173, 176, 179
- \eledmac@warning
.... 55, 79, 81, 83, 85, 87, 89,
92, 95, 97, 113, 115, 117, 120,
122, 124, 126, 129, 132, 136,
138, 143, 145, 150, 152, 156, 159
- \eledsection 4965, 5094
- \eledsection* 5094
- \eledsubsection 5094
- \eledsubsection* 5094
- \eledsubsubsection 5094
- \eledsubsubsection* 5094
- \emph 3767, 3768
- \empty 183, 188, 258, 261,
440, 441, 485, 864, 891, 908,
912, 918, 929, 946, 989, 1003,
1052, 1261, 1319, 1335, 1455–
1457, 1468, 1500, 3257, 3270, 3981
- \enablel@dtabfeet 4585,
4601, 4621, 4629, 4639, 4647, 4729
- \end@lemmas 822, 864, 865
- \endashchar 21, 1612, 1692, 2649
- \endgraf 1078, 1124, 1128

- \endline@num 466, 710, 716
- \endlock 11, 812, 829, 4037, 4054, 4062
- \endminipage 3657
- \endnumbering ... 6, 207, 251, 282, 301
- \endpage@num 466, 709, 716
- \endprint 26, 2616, 2660
- \endquotation 4738
- \endquote 4738
- \endstanzaextra 25, 4028
- \endsub 11, 788, 828
- \endsubline@num 466, 711, 717
- \enlargethispage 5285, 5293
- \enskip 2617
- \enspace . 2168, 2298, 2365, 2476, 2617
- environments:
 - edarrayc 31, 4721
 - edarrayl 31, 4721
 - edarrayr 31, 4721
 - edtabularc 31, 4725
 - edtabularl 31, 4725
 - edtabularr 31, 4725
 - ledgroup 26, 3684
 - ledgroupsize 26, 3698
 - minipage 26
- Euclid 4
- \ExecuteOptions 19
- \expandonce 1551, 1562, 2114, 2225, 2742, 2755, 2774, 2790, 3456, 3460, 3471, 3475, 3486, 3490, 3828
- \extensionchars 37, 186, 213, 289
- \extractendline@ 2992, 2995
- \extractendsubline@ 2993, 2995
- \extractline@ 2990, 2995, 2998
- \extractsubline@ ... 2991, 2995, 2998
- F**
- \f@encoding 900
- \f@family 900
- \f@series 900
- \f@shape 900
- \f@x@l@cks 1360, 1366
- Fairbairns, Robin 29
- \falseverse 156, 4028
- \first@linenum@out@false ... 721, 727
- \first@linenum@out@true 721
- \firstlinenum 9, 374
- \firstseries 2805
- \firstsublinenum 9, 374
- \fix@page 512, 558
- \flag@end 763, 860, 870, 871, 881
- \flag@start 763, 859, 860, 871
- \flagstanza 25, 4066
- \floatingpenalty 1579
- \flush@notes 1088, 1498, 3617
- \flush@notesR 3615
- \fnpos 30, 2560
- Folkerts, Menso 4
- \fontencoding 1512
- \fontfamily 1512
- \fontseries 1512
- \fontshape 1512
- \footfootmarkA 30
- \footfudgefiddle 38, 1790, 1795, 2419
- \footins . 3117, 3124, 3128, 3162, 3202
- \footnormal 1746, 2766, 3177
- \footnormalX 29, 2241, 2590, 2783
- \footnote@luatexpardir 1596
- \footnote@luatextextdir . 1595, 1617
- \footnoteA 29
- \footnoteB 29
- \footnoteC 29
- \footnoteD 29
- \footnoteE 29
- \footnotelang@lua .. 1528, 2734, 2747
- \footnotelang@poly .. 1542, 2738, 2751
- \footnoteoptions@ 1515, 2739, 2743, 2752, 2757
- \footnoterule .. 1714, 2192, 3127, 3672
- \footnotesize 2977, 3425, 3426
- \footparagraph 15, 1773
- \footparagraphX 29, 2397
- \footssplitskips 1570, 1577, 1818, 1833, 1960, 2024, 2144, 2282, 2350, 2447, 2461
- \footthreecol 15, 1939
- \footthreecolX 29, 2330
- \foottwocol 15, 2008
- \foottwocolX 29, 2262
- \foottwocolX 2262
- \fullstop 21, 430, 1612, 1689, 1691, 1693, 1695, 2648, 2652
- G**
- \g@addto@macro 2591, 2596, 2599, 2602, 3168, 3169, 3178, 3181, 3184, 3186, 3193, 3770
- Gädeke, Nora 5
- \get@edindex@hyperref ... 3837, 3890
- \get@index@command 3752, 3833, 3842, 3899, 3910

- `\get@linelistfile` 482, 500
`\getline@num` 1139, 1208
`\gl@p` 452, 488, 489, 865,
 895, 930, 947, 992, 1006, 1264,
 1265, 1461, 1465, 1501, 3260, 3273
`\gl@poff` 452, 453
- H**
- `\hangafter` 4024
`\hangindentX` 18
`\hangingsymbol` 24, 25, 3955, 3962
`\hb@xt@` 1155, 1160,
 1201, 1203, 4520, 4525, 4529,
 4534, 4545, 4558, 4571, 4650, 4652
`\hfilneg` 1928
`\Hilfsbox` 4135
`\hilfsbox` 4135, 4190, 4191,
 4228, 4240, 4314, 4328, 4347,
 4363, 4379, 4394, 4411, 4536,
 4538, 4543, 4547, 4548, 4550,
 4556, 4560, 4561, 4563, 4569, 4572
`\hilfscount` 4135, 4688–4690, 4696
`\HILFSskip` 4681
`\Hilfsskip` 4526,
 4530, 4531, 4535, 4538, 4539,
 4546, 4547, 4550–4552, 4559,
 4560, 4563–4565, 4572–4574,
 4681, 4687, 4689, 4695, 4699, 4700
`\hilfsskip`
 . 4135, 4313, 4314, 4331, 4350,
 4366, 4382, 4397, 4414, 4698–4700
`\hsize@fornote` 2523, 2528,
 2531, 2532, 2536, 2541, 2544, 2545
`\hsizethreecol` 19
`\hsizethreecolX` 19
`\hsizetwocol` 19
`\hsizetwocolX` 19
`\Hy@temp@A` 3801, 3802
`\HyInd@ParenLeft` 3802
`\hyperlink`
 3316, 3317, 3879, 3880, 3883, 3891
`\hyperlinkformat` 3877, 3887
`\hyperlinkformatR` 3886
`\hyperlinkR` 3882
`\hyperpage` 3768
`\hypertarget` 3267, 3280
- I**
- `\if@edindex@fornote@`
 3747, 3791, 3803, 3813, 3832, 3841
`\if@edindex@fornote@true` 3747
`\if@edtext@` 842, 847, 931, 948
`\if@eled@sectioning`
 4875, 4877, 4892, 4910, 4964, 4982
`\if@fcolmade` 3223, 3232
`\if@firstcolumn` 1341, 1417, 3225, 3570
`\if@led@nofoot` 3189, 3210
`\if@lemmacommand@` 905, 932, 949
`\if@nobreak` 1037
`\if@noeled@sec` 228, 272, 5092
`\if@noneed@Footnote` 763
`\if@openright` 4873, 4963, 4965
`\if@RTL` 29,
 54, 860, 871, 1179, 1544, 1555,
 1824, 2150, 4879, 4895, 4981, 4983
`\ifautopar` 281, 1056, 1108,
 3251, 4745, 4753, 4761, 4769,
 4777, 4785, 4821, 4834, 4843, 4856
`\ifautopar@pause` 285, 1130
`\IfBeginWith` 3756, 3761
`\ifboolexpr` 1625, 3074
`\ifbypage@` 306, 563, 1242, 1663
`\ifbypstart@` 306, 1084
`\ifcsdef` 476, 984, 998, 1916, 3012
`\ifcseempty`
 1607, 1857, 1918, 1972, 2036, 3016
`\ifcsequal` 3014
`\ifcsstring` 1864, 1865, 1896, 1897,
 2481, 2482, 2511, 2512, 2809, 2816
`\ifcsundef` 967, 975
`\ifdef` 29, 54, 3267, 3280, 3316, 3325, 3891
`\ifdefstring` 1617
`\ifdim` 789, 791, 793, 795, 2083, 4190, 4688
`\ifdimequal`
 1698, 1801, 2175, 2425, 3021, 3057
`\iffirst@linenum@out@` 721, 725
`\ifFN@bottom` 3114, 3124
`\ifhbox` 1907, 1912
`\ifhmode` 2094, 2101, 2124, 2131
`\ifinserthangingsymbol` .. 3958, 5301
`\ifinstanza` 1058, 1125, 3955, 3961, 5301
`\ifl@d@dash` 1635, 1692, 2649
`\ifl@d@elin` 1635,
 1681, 1694, 1695, 2639, 2651, 2652
`\ifl@d@esl` 1635, 1695, 2652
`\ifl@d@pnum`
 1635, 1669, 1689, 1693, 2627, 2650
`\ifl@d@ssub` 1635, 1691, 2648
`\ifl@dend@` 2605, 2611
`\ifl@dmemoir` 43, 4201, 4789, 4870

- \ifl@dpaging 190, 1571, 2145
- \ifl@dpairing . . 190, 216, 255, 275,
291, 1571, 1706, 1723, 1729,
1876, 1882, 1984, 1990, 2052,
2058, 2145, 2184, 2201, 2207,
2307, 2313, 2374, 2380, 2490,
2497, 3251, 3613, 3622, 3674,
4741, 4749, 4757, 4765, 4773, 4781
- \ifl@dprintingcolumns 190
- \ifl@dprintingpages 190
- \ifl@dskipnumber 815, 1302
- \ifl@dstartendok 4491, 4501
- \ifl@imakeidx 46, 3728, 3831
- \ifl@indextools 48, 3731
- \iflabelpstart 1030, 1066
- \ifledfinal 4, 30, 37
- \ifledgroupnotesL@ 1301, 3725
- \ifledgroupnotesR@ 3725
- \iflednopbinverse 4, 5300, 5301
- \ifledplinenumber 2976
- \ifledRcol
. 190, 766, 855, 882, 926, 964,
1111, 1626, 2732, 3255, 3387,
3455, 3470, 3485, 3614, 3623,
3675, 3900, 3911, 5096, 5109,
5122, 5134, 5148, 5161, 5174, 5187
- \ifledRcol@
135, 142, 149, 190, 983, 3497, 3515
- \ifledsecnolinenumber 4,
4743, 4751, 4759, 4767, 4775, 4783
- \ifleftnoteup 3537, 3550
- \ifluatex
1034, 1161, 1594, 1616, 2733, 2746
- \ifnoquotation@ 4, 4817
- \ifnoteschanged@ 265, 470
- \ifnumberedpar@
. 850, 1017, 1047, 1074,
2106, 2112, 2223, 2235, 2731,
2789, 2790, 3376, 3454, 3469, 3484
- \ifnumbering 189,
205, 252, 284, 309, 1043, 1071, 1117
- \ifnumberingR 190, 1112
- \ifnumberline 888, 1212, 1301
- \ifnumberpstart 1028, 1057, 1091, 1125
- \ifnumequal 1085, 1915, 1917, 3557
- \ifnumgreater
. 994, 1008, 3565, 3585, 3592
- \ifodd 1351,
1427, 3432, 3440, 3508, 3526, 3580
- \ifparapparatus@ 4
- \ifparledgroup 4, 1723, 1728,
1876, 1881, 1984, 1989, 2052,
2057, 2201, 2206, 2307, 2312,
2374, 2379, 2490, 2496, 3622, 3673
- \ifpst@rtedL 190
- \ifpstartnum 1438, 1441, 1446
- \ifreportnoidxfile 3776
- \ifrighnoteup 3448, 3545
- \ifshowindexmark 3789
- \ifsidepstartnum 1059, 1412
- \ifstrempty 1033,
1096, 2842, 2856, 3878, 5198,
5200, 5211, 5218, 5219, 5229,
5236, 5237, 5247, 5254, 5255, 5259
- \IfStrEq 738,
748, 1140, 1151, 3136, 3139,
3600, 3603, 5281, 5290, 5303, 5307
- \IfStrEqCase 760
- \ifstrequal 935,
952, 1517, 1529, 1543, 2835, 2852
- \ifsublines@
. 428, 457, 548, 577, 582, 588,
603, 621, 630, 644, 666, 715,
717, 1213, 1250, 1305, 3289, 3313
- \IfSubStr 3833, 3842, 3898
- \iftoggle 1607, 1700,
1803, 1857, 1972, 2036, 2177,
2427, 2525, 2538, 2549, 2555,
2994, 3000, 3005, 3010, 3078, 3084
- \ifvbox 1081, 3096, 3164
- \ifvmode 3262, 3275
- \ifvoid 2579, 2584, 2594, 2597, 2603,
3117, 3150, 3157, 3162, 3179,
3182, 3187, 3194, 3195, 3202–
3204, 3621, 3638, 3660, 3692, 3719
- \ifwidthliketwocolumns . . 4, 222, 294
- \imki@wrindexentry
. 52, 53, 3834, 3835, 3838
- \indexentry 3844, 3847, 3851
- \indtl@wrindexentry 52, 53
- \initnumbering@reg 204
- \initnumbering@sectcmd 221, 293, 4738
- \inplaceoflemmaseparator 17
- \inplaceofnumber 17
- \InputIfFileExists 229, 501
- \insert 1567, 1814, 1958,
2022, 2141, 2280, 2348, 2443,
2584, 2598, 3157, 3162, 3164, 3183
- \insert@count
. 689, 690, 777, 779, 856, 1519,

- 1531, 1533, 1546, 1549, 1552,
2116, 2227, 2756, 3462, 3477, 3492
`\insert@countR` 768, 770,
855, 1523, 1537, 1539, 1557,
1560, 1563, 2744, 3458, 3473, 3488
`\inserthangingsymbol` 1164, 3959
`\inserthangingsymbolfalse` 1144
`\inserthangingsymboltrue` 1142
`\inserthangingymbl` 3958
`\insertlines@list`
... 258, 461, 494, 697, 1457, 1461
`\insertparafootsep` .. 1853, 1914, 2474
`\inserts@list` 1051, 1452,
1455, 1465, 1500, 1501, 1518,
1530, 1532, 1545, 1548, 1551,
2115, 2226, 2755, 3461, 3476, 3491
`\inserts@listR`
..... 1522, 1536, 1538, 1556,
1559, 1562, 2742, 3457, 3472, 3487
`\instanzafalse` 3957, 4057
`\instanzatru` 4031
`\interfootnotelinepenalty` 1578
`\interlinepenalty`
..... 1105, 1485, 1578, 4037
`\interparanoteglue` 2984
`\ipn@skip` 2984
`\itemcount@` 136, 138,
143, 145, 150, 152, 3555, 3557,
3562, 3565, 3583, 3585, 3590, 3592
- J**
- Jayaditya 5
- K**
- Kabelschacht, Alois 100
Krukov, Alexej 70
- L**
- `\l@d@wrindexhyp` 3787
`\l@d@add` 913, 915, 919, 921
`\l@d@dashfalse` 1644, 1662, 2622
`\l@d@dashtrue`
1666, 1672, 1684, 2625, 2630, 2642
`\l@d@elinfalse` 1640, 1669, 2627
`\l@d@elintrue` .. 1669, 1671, 2627, 2629
`\l@d@end`
2605, 2607, 2608, 2614, 2664, 2788
`\l@d@err@UnequalColumns` 4271
`\l@d@eslfalse`
.... 1642, 1678, 1681, 2636, 2639
`\l@d@esltrue` ... 1681, 1683, 2639, 2641
`\l@d@index` 3772, 3774, 4203
`\l@d@makecol` 3108, 3171, 3241
`\l@d@nums` . 854, 893, 896, 907, 908,
921, 2740, 2742, 2753, 2755, 2789
`\l@d@pnumfalse` 1636, 1662, 2622
`\l@d@pnumtrue` 1665, 2624
`\l@d@reinserts` 3161, 3172
`\l@d@section` 2614, 2616
`\l@d@set` 610, 809
`\l@d@ssubfalse` 1638, 1674, 2632
`\l@d@ssubtrue` 1676, 2634
`\l@d@wrindexm@` 3786, 3787
`\l@dampcount` 4130,
4267, 4269, 4274, 4534, 4544,
4545, 4557, 4558, 4593, 4611, 4649
`\l@d@begin@stack` 3932, 3943–3945
`\l@d@bfnote` 2107, 2111
`\l@d@checkcols` 4224, 4236, 4264
`\l@d@checkstartend` 4490, 4501
`\l@d@chset@num` 515, 518, 610
`\l@d@colcount` 4130, 4172,
4184, 4185, 4223, 4225, 4235,
4237, 4265, 4269, 4274, 4319,
4321, 4338, 4340, 4356, 4357,
4372, 4373, 4387, 4388, 4404,
4405, 4457, 4458, 4534, 4544,
4545, 4557, 4558, 4589, 4591,
4606, 4608, 4649, 4655, 4685, 4694
`\l@d@collect@@body` 3934, 3942
`\l@d@collect@body`
.... 3929, 4721–4723, 4725–4727
`\l@d@colwidth` 4172, 4190, 4191,
4313, 4456, 4520, 4546, 4559,
4650, 4652, 4657, 4666, 4687, 4688
`\l@d@csnote` 3447, 3448
`\l@d@csnotetext`
1193, 3511, 3527, 3532, 3564, 3567
`\l@d@csnotetext@l` 1193, 3509, 3567, 3584
`\l@d@csnotetext@r` 1193, 3529, 3567, 3591
`\l@d@ddoreinxtrafeet` 3152, 3163, 3169
`\l@d@ddofootinsert` 3109, 3115
`\l@d@ddoxtrafeet` 3132, 3135, 3168
`\l@d@dedbeginmini` 3184, 3598, 3608
`\l@d@dedendmini`
.... 3186, 3601, 3604, 3605, 3608
`\l@d@emptyd@ta` 1136, 1193
`\l@d@end@close` 2607, 2657
`\l@d@end@false` 2605, 2608
`\l@d@end@open` 2607, 2612

- \l@dend@stuff ... 214, 290, 2610, 2663
- \l@dend@true 2605, 2607
- \l@denbody 3924, 3927, 3930–3932
- \l@dfambeginmini ... 2599, 3598, 3634
- \l@dfamendmini
.... 2602, 3601, 3604, 3605, 3634
- \l@dfetbeginmini
..... 3598, 3652, 3688, 3715
- \l@dfetendmini 3598, 3663, 3695, 3722
- \l@dgetline@margin 335
- \l@dgetlock@disp 379, 407
- \l@dgetref@num
.. 3320, 3321, 3323, 3327, 3329,
3330, 3332, 3333, 3340, 3362, 3367
- \l@dgetsidenote@margin 3383
- \l@dobblearg 4215
- \l@dobblearg 4215
- \l@dobbleoptarg 4216
- \l@dlabel@parse 3346, 3349
- \l@dld@ta 1157, 1193,
1340, 1418, 1430, 4883, 4899, 4902
- \l@dlp@rbox 1201, 3418, 3536
- \l@dlsn@te 1159, 1200, 4884, 4900, 4902
- \l@dlsnote 3428, 3448
- \l@dmake@labels 3279, 3299, 3308
- \l@dmake@labelsR 3266
- \l@dmemoirfalse 44
- \l@dmemoirtrue 44
- \l@dmodforcritext 4277, 4730
- \l@dmodforedtext 4288, 4733
- \l@dnullfills 4298,
4578, 4596, 4614, 4624, 4632, 4642
- \l@dnumpstartsl
190, 217, 235, 276, 1054, 1148,
1173, 1176, 1178, 5102, 5115,
5128, 5140, 5154, 5167, 5180, 5193
- \l@dnumpstartsr 5098, 5111,
5124, 5136, 5150, 5163, 5176, 5189
- \l@dold@xympar 3374
- \l@doldold@footnotetext 2105
- \l@dp@rsefootspec 1645, 3749
- \l@dpagingfalse 190
- \l@dpagingtrue 190
- \l@dpairingfalse 190
- \l@dpairingtrue 190
- \l@dparsedendline 1645, 3746
- \l@dparsedendpage 1645, 3746
- \l@dparsedendsub 1645
- \l@dparsedstartline 1645, 3745
- \l@dparsedstartpage 1645, 3745
- \l@dparsedstartsub 1645
- \l@dparsefootspec 1645
- \l@dpprintingcolumnfalse 190
- \l@dpprintingcolumnstrue 190
- \l@dpprintingpagesfalse 190
- \l@dpprintingpagetrue 190
- \l@dpush@begins 3939, 3943
- \l@drd@ta 1166, 1193,
1340, 1420, 1428, 4883, 4886, 4899
- \l@dref@undefined
.... 3320, 3323, 3329, 3332, 3335
- \l@drestorefills 4298,
4582, 4598, 4618, 4626, 4636, 4644
- \l@drestoreforcritext ... 4277, 4731
- \l@drestoreforedtext ... 4288, 4734
- \l@drp@rbox 1203, 3418, 3544
- \l@drsn@te 1167, 1200, 4884, 4886, 4900
- \l@drsnote 3429, 3448
- \l@dsetmaxcolwidth .. 4189, 4230, 4242
- \l@dskipnumberfalse 815, 1303
- \l@dskipnumbertrue 815, 1294
- \l@dstartendokfalse . 4505, 4509, 4513
- \l@dstartendoktrue 4503
- \l@dtabaddcols 4489, 4519
- \l@dtabnoexpands 835, 4074
- \l@dunboxmpfoot 3661, 3669, 3693, 3720
- \l@dunhbox@line 1131, 1165, 1181, 1184
- \l@dzeropenalties .. 1069, 1077, 1103
- \l@imakeidxtrue 47, 51
- \l@indextoolstrue 50
- \l@luatextextdir@L 1035, 1162
- \l@prev@nopb 239, 739,
750, 5268, 5275, 5276, 5282, 5291
- \l@prev@pb 238, 5267, 5273, 5274, 5279
- Lück, Uwe 3
- \label 28
- \label@refs 3258, 3260, 3266, 3271,
3273, 3279, 3287, 3290, 3292, 3294
- \labelpstartfalse 1022
- \labelpstarttrue 1022
- \labelref@list . 3246, 3270, 3273, 3313
- \labelref@listR 3257, 3260
- \labelrefsparseline 3284
- \labelrefsparsesubline 3284
- \languagenam 1551, 1562
- \last@page@num 558, 5302
- \lastbox . 1123, 1138, 1845, 1906, 1911
- \lastkern 2083
- \lastskip 788, 792
- Lavagnino, John 2, 3

- Leal, Jeronimo@Leal, Jerónimo 3
- \led 159
- \led@check@nopb 1140, 1151, 5279
- \led@check@pb 1140, 1151, 5279
- \led@err@AutoparNotNumbered
 98, 1113, 1118
- \led@err@edtextoutsidepstart 63, 877
- \led@err@EdtextWithoutFootnote
 172, 772, 781
- \led@err@HighEndColumn . . . 161, 4510
- \led@err@LineationInNumbered 75, 310
- \led@err@LowStartColumn . . 161, 4506
- \led@err@ManyLeftnotes . . . 134, 3585
- \led@err@ManyRightnotes . . 134, 3592
- \led@err@ManySidenotes . . . 134, 3565
- \led@err@NumberingNotStarted 57, 269
- \led@err@NumberingShouldHaveStarted
 57, 300
- \led@err@NumberingStarted . . . 57, 206
- \led@err@PendNoPstart 98, 1075
- \led@err@PendNotNumbered . . . 98, 1072
- \led@err@PstartInPstart . . . 98, 1048
- \led@err@PstartNotNumbered . 98, 1044
- \led@err@ReverseColumns . . 161, 4514
- \led@err@TooManyColumns . . 161, 4186
- \led@err@UnequalColumns 161
- \led@error@ImakeidxAfterEledmac
 175, 3729
- \led@error@IndextoolsAfterEledmac
 178
- \led@error@indextoolsAfterEledmac
 3732
- \led@mess@NotesChanged 65, 266
- \led@mess@SectionContinued . . 73, 288
- \led@nopb 5271, 5273
- \led@nopbnum 5272, 5273
- \led@pb 5269, 5273
- \led@pb@setting 738, 748, 760, 1140,
 1151, 5277, 5281, 5290, 5303, 5307
- \led@pbnum 5270, 5273
- \led@toksa 442, 450
- \led@toksb 442, 449–451
- \led@war@FalseverseDeprecated
 155, 4042
- \led@war@ledxxxDeprecated
 155, 4740, 4748,
 4756, 4764, 4772, 4780, 4788, 4797
- \led@warn@AddfootinsObsolete
 128, 3176
- \led@warn@Addfootinsobsolete . . . 125
- \led@warn@AddfootinsXObsolete
 125, 2589
- \led@warn@AddfootinsXobsolete . . 125
- \led@warn@BadAction 112, 1296
- \led@warn@BadAdvancelineLine 88, 597
- \led@warn@BadAdvancelineSubline
 88, 591
- \led@warn@BadLineation 78, 330
- \led@warn@BadLinenummargin . . 78, 358
- \led@warn@BadLockdisp 78, 385
- \led@warn@BadSetline 94, 800
- \led@warn@BadSetlinenum 94, 807
- \led@warn@BadSidenotemargin 121, 3410
- \led@warn@BadSubblockdisp . . . 78, 411
- \led@warn@DuplicateLabel . . . 114, 3302
- \led@warn@NoIndexFile 123, 3777
- \led@warn@NoLineFile 86, 506
- \led@warn@NoMarginpars 119, 3377
- \led@warn@RefUndefined 114, 3337
- \led@warn@SeriesStillExist 131, 2671
- \ledchapter 4738
- \ledchapter* 4738
- \ledfinalfalse 14
- \ledfinaltrue 13
- \ledfootinsdim 1746, 2718, 2719
- ledgroup (environment) 26, 3684
- ledgroupsized (environment) . . 26, 3698
- \ledinnernote 28, 3428
- \ledinnote 3767
- \ledinnotehyperpage 3767
- \ledleftnote 28, 3428, 4111, 4114, 4119
- \ledlinenum 423
- \ledllfill 1160, 1205, 3702, 3706
- \ledlsnotefontsetup 29, 3421, 3535
- \ledlsnotesep 29, 1201, 3421
- \ledlsnotewidth 29, 3421, 3535
- \lednopb 36, 5269
- \lednopbinversetrue 16, 36
- \lednopbnum 5269, 5309
- \ledouternote 28, 3439
- \ledouterote 3428
- \ledpb 36, 5269
- \ledpbnum 5269, 5305
- \ledpbsetting 36, 5277
- \ledplinumtrue 2979
- \ledrightnote 28, 3428, 4110, 4113, 4118
- \ledrlfill 1166, 1205, 3703, 3710
- \ledrsnotefontsetup 29, 3421, 3543
- \ledrsnotesep 29, 1203, 3421
- \ledrsnotewidth 29, 3421, 3543

- \ledsecnolinenumbertrue 17
 - \ledsection [4738](#)
 - \ledsection* [4738](#)
 - \ledsectnomark [4865](#)
 - \ledsectnotoc [4864](#)
 - \ledsetnormalparstuff
 - [1593](#), 1854, 2166, 2475
 - \ledsidenote 28, [3428](#), 4112, 4115, 4120
 - \ledsubsection [4738](#)
 - \ledsubsection* [4738](#)
 - \ledsubsubsection [4738](#)
 - \ledsubsubsection* [4738](#)
 - \left 4464, 4467, 4472, 4475
 - \leftctab [4533](#), 4633
 - \leftlinenum
 - .. 10, [423](#), 1342, 1354, 4881, 4897
 - \leftlinenumR 4882, 4898
 - \leftltab [4524](#), 4615
 - \leftmargin 4826, 4827, 4848, 4849
 - \leftnoteupfalse 29
 - \leftnoteuptrue 3551
 - \leftpstartnum [1412](#)
 - \leftrtab [4528](#), 4579
 - Leibniz 5
 - \lemma 13, [902](#)
 - \lemmaseparator 17, 2983
 - \letsforverteilen [4306](#),
 - 4330, 4349, 4365, 4381, 4396, 4413
 - \line@list 261, [461](#), 493, 717, 891, 895
 - \line@list@stuff 213, 289, [723](#)
 - \line@margin [335](#), 1347, 1423
 - \line@num . 138, 145, 152, 240, 427,
 - [455](#), 520, 554, 564, 595, 596,
 - 598, 606, 611, 612, 624, 710,
 - 714, 1219, 1243, 1253, 1318,
 - 1320, 1321, 1330, 1331, 1917, 3312
 - \line@numR 136, 143, 150
 - \line@set 909, [910](#)
 - \lineation 10, 76, 79, [308](#)
 - \lineinfo@ 2995, 2998, 3037
 - \linenum 13,
 - 906, 3369, 4214, 4282, 4293, 4312
 - \linenum@out [720](#),
 - 726, 728, 733, 734, 742, 744,
 - 746, 753, 755, 757, 760, 776,
 - 790, 794, 797, 802, 809, 812,
 - 813, 819, 885, 928, 3269, 5269–5272
 - \linenum@outR 767, 883, 945, 3256
 - \linenumberlist ... 9, [183](#), 1319, 1331
 - \linenumberstyle 11, [414](#)
 - \linenumincrement 9, [374](#)
 - \linenummargin 10, 81, [335](#)
 - \linenumr@p [414](#)
 - \linenumrep [414](#),
 - 427, 1690, 1694, 2647, 2651, 3312
 - \linenumsep
 - .. 10, [423](#), 1442, 1447, 3423, 3424
 - \lineref [3323](#)
 - \linewidth 1155, 2520, 2523, 2524, 2537
 - \list@clear [441](#), 493–498, 1051
 - \list@clearing@reg 481, [492](#)
 - \list@create [440](#),
 - 461–464, 822, 923, 924, 1452, 3246
 - \listadd 2809, 2816
 - \listcsgadd 5215, 5233, 5251, 5263
 - \listeadd 2808, 2818
 - \listgadd 3509, 3511, 3527, 3529, 3532
 - \listxadd 569, 2801, 5273–5276
 - \lock@disp [379](#), 1384, 1388, 1392
 - \lock@off 637, 638, [664](#), 813
 - \lock@on [635](#), 812
 - \lockdisp 11, 83, [379](#)
 - Lorch, Richard 4
 - \Lpack 3865
 - \ltab 4077, [4613](#), 4721
 - \ltabtext 4079, [4623](#), 4725
 - \luatexpardir 1532, 1538, 1596
 - \luatextextdir
 - 1035, 1162, 1530, 1536, 1595
 - Luecking, Dan 41
- ## M
- \m@m@makecolfloats [3091](#), 3110
 - \m@m@makecolintro [3091](#)
 - \m@m@makecoltext [3091](#), 3111
 - \m@mdodoreinextrafeet 3169
 - \m@mdoextrafeet 3168
 - \m@mmf@check [2082](#), 2096, 2126
 - \m@mmf@prepare
 - [2079](#), 2091, 2100, 2130, 2774
 - \M@sect [4907](#)
 - \m@th 4482
 - \makehboxoffhboxes
 - 1866, 1898, [1903](#), 2483, 2513
 - \makememindexhook 3770
 - \managestanza@modulo [3987](#), 4018
 - \marg 845, 2829
 - \marginparwidth 3421, 3422

- \marks 1724–1726, 1877–1879, 1985–1987, 2053–2055, 2202–2204, 2308–2310, 2375–2377, 2492–2494
- \mathchardef 3982
- \maxdepth 3112
- \maxdimen 1819, 1834, 2448, 2462
- \maxhnotesX 20
- \maxhXnotes 20
- Mayer, Gyula 5
- \measurebody .. 4581, 4587, 4617, 4635
- \measurecell 4222, 4248
- \measuremrow 4246, 4592
- \measuretbody .. 4597, 4603, 4625, 4643
- \measuretbody .. 4597, 4603, 4625, 4643
- \measuretbody .. 4597, 4603, 4625, 4643
- \measuretbody .. 4597, 4603, 4625, 4643
- \measuretbody .. 4597, 4603, 4625, 4643
- \measuretbody .. 4597, 4603, 4625, 4643
- \message 74, 212
- Middleton, Thomas 5, 57
- minipage (environment) 26
- Mittelbach, Frank 4
- \morenoexpands 38, 826
- \moveleft 2566, 2572, 4526, 4531, 4539
- \moveright 4552, 4565, 4574
- \mpfnpos 30, 2560
- \mpnrmalfootgroup 1721, 1767
- \mpnrmalfootgroupX 2199, 2256
- \mpnrmalvfootnote 1583, 1766, 1945, 2014
- \mpnrmalvfootnoteX 2152, 2255, 2268, 2336
- \mppara@footgroup 1785, 1873
- \mppara@footgroupX 2409, 2479
- \mppara@vfootnote 1784, 1828
- \mppara@vfootnoteX 2408, 2442
- \mpthreecolfootgroup 1946, 1982
- \mpthreecolfootgroupX ... 2337, 2368
- \mpthreecolfootsetup 1947, 1953
- \mpthreecolfootsetupX ... 2338, 2340
- \mptwocolfootgroup 2015, 2047
- \mptwocolfootgroupX 2269, 2301
- \mptwocolfootsetup 2016, 2047
- \mptwocolfootsetupX 2270, 2272
- \multfootsep 29, 2076, 2086
- \multiplefootnotemarker 2076, 2080, 2081, 2083
- N**
- \n@l 755
- \n@num 685, 819
- \n@num@reg 685
- \nc@page 752, 753
- \NeedsTeXFormat 2
- \new 2807–2809, 2811, 2815, 2816, 2818, 2819
- \new@line 737, 1160, 1181, 1184
- \newbox 1017, 1020, 3418, 3419, 4135, 4137, 4718, 4719
- \newcommandx 925, 1032, 1071, 1515, 1528, 1542, 1602, 1843, 1852, 1962, 2026, 2832, 2849, 2954, 2968, 2983, 3828, 4011, 4051, 4053, 4061
- \newcounter 365, 367, 369, 371, 1026, 1226, 2777, 3284, 3285, 3737, 3990, 4487
- \newhookcommand@series 2863
- \newhookcommand@series@reload . 2936
- \newhooktoggle@series 2950, 2953, 2958–2966, 2967
- \newhooktoggle@series@reload ... 2967, 2973, 2974
- \newif 4–10, 29, 43, 46, 48, 54, 189, 190, 192, 194, 196, 198, 201–203, 306, 307, 457, 470, 721, 763, 815, 847, 888, 905, 1018, 1028, 1030, 1108, 1130, 1413, 1438, 1635, 1637, 1639, 1641, 1643, 2606, 2978, 3114, 3189, 3448, 3550, 3725, 3726, 3747, 3956, 3958, 4501, 4875, 5092
- \newinsert 2723–2726
- \newlength 423, 3974
- \newlinechar 2785
- \newread 479
- \newrobustcmd 3315
- \newseries 2665, 2803, 2804
- \newseries@ 2666, 2670
- \newtoggle 1747, 1751, 2673, 2674, 2694–2698, 2700, 2703, 2720, 2721, 2981, 2982
- \newverse 156, 4028
- \newwrite 720, 2605, 5091
- \NEXT 4218, 4223, 4226, 4231, 4232, 4235, 4238, 4243, 4244, 4247, 4249, 4250, 4252, 4254, 4255, 4260, 4419, 4422, 4424, 4425, 4427, 4429, 4430, 4433, 4435, 4436, 4438, 4440, 4441, 4444, 4446, 4447, 4449, 4451, 4452, 4457,

- 4459, 4460, 4655, 4658, 4659,
4664, 4668, 4669, 4685, 4691, 4692
\Next 4260, 4320,
4322, 4333, 4334, 4339, 4341,
4352, 4353, 4356, 4358, 4367,
4368, 4372, 4374, 4383, 4384,
4387, 4389, 4399, 4400, 4404,
4406, 4416, 4417, 4673, 4675, 4676
\next@absline 749, 750
\next@action 113, 489, 1232,
1240, 1241, 1247, 1248, 1256, 1265
\next@actionline
486, 488, 1231, 1239, 1262, 1264
\next@insert
1052, 1456, 1459, 1461, 1464, 1468
\next@page@num 245, 523, 525, 568, 618
\no@expands 826, 853, 904
\noalign 1931
\noeledsec 35, 5092
\noendnotes 27, 2663
\noindent 1023, 1033, 1096,
1100, 1125, 1410, 1600, 1713,
1716, 1812, 1820, 1824, 1835,
1869, 1901, 1978, 2003, 2042,
2071, 2150, 2449, 2463, 2486, 2516
\nointerlineskip 2565, 2567, 2571, 2573
\nolemmaseparator 17, 2983
\nonbreakableafternumber 17
\nonum@ 2981
\nonumberinfootnote 16
\noquotation@true 12
\normal@footnotemarkX ... 2133, 2244
\normal@page@break
237, 569, 740, 751, 5266, 5284, 5292
\normal@pars 254, 1053,
1128, 1599, 1963, 2027, 2290, 2358
\normalbfnoteX 2222, 2236
\normalbodyfootmarkX 2138, 2245
\normalcolor 1727, 1880, 1988, 2056,
2205, 2311, 2378, 2495, 3126, 3671
\normalfont 425, 2077, 2139, 2976
\normalfootfmt 1593, 1759
\normalfootfmtX 2162, 2248
\normalfootfootmarkX 2171, 2249
\normalfootgroup 1715, 1760
\normalfootgroupX 2194, 2250
\normalfootnoterule 1714, 1762
\normalfootnoteruleX 2192, 2251, 2404
\normalfootstart 1697, 1757
\normalfootstartX 2174, 2243
\normalvfootnote 1566, 1758
\normalvfootnoteX 2140, 2246
\nosep@ 2982
\notblank 1526
\notbool 1566, 1583, 1602,
1957, 1962, 2022, 2026, 2616, 2728
\notefontsetup
.... 1977, 2686–2688, 2976, 2986
\notefontsizeX 18
\notenumfont 2683–2685, 2976
\notenumfontX 18
\noteschanged@false 470, 502
\noteschanged@true
.... 259, 262, 470, 507, 892, 1458
\notesXwidthliketwocolumns 21
\nottoggle 1606, 1856, 1971, 2035, 2617
\NR@ssect 5024, 5032
\NR@sssect 5040, 5048
\nulledindex 4200, 4281, 4292,
4318, 4337, 4355, 4371, 4386, 4403
\nullsetzen 4454, 4590, 4607
\num@lines 1017, 1078, 1475, 1481, 1484
\numberedpar@false 1017
\numberedpar@true 1017, 1065
\numberingfalse 189, 253
\numberingtrue 189, 209, 282
\numberlinefalse 9
\numberlinetrue 9, 889
\numberonlyfirstinline 16
\numberonlyfirstintwolines 16
\numberpstartfalse 8, 1022
\numberpstarttrue 8, 1022
\numdef 749,
966, 970, 974, 978, 985, 986,
988, 999, 1000, 1002, 1173, 5283
\numgdef 741, 752,
3555, 3562, 3583, 3590, 5304, 5308
\numlabfont 21, 423
- O**
- \old@edtext
4918, 4925, 4999, 5006, 5011, 5018
\old@hsize
1707, 1718, 1870, 2185, 2196, 2518
\old@Rlineflag 3829, 3856
\one@line 1017,
1137, 1138, 1160, 1165, 1181, 1184
\onlypstartinfootnote 16
\openout 230, 728, 734, 2607

P

- \p@pstart 1067
- \PackageError 56
- \PackageWarning 55
- \page@action 524, 616, 703
- \page@num 138, 145,
152, 466, 484, 566, 709, 714,
1241, 1349, 1425, 1915, 1924,
3432, 3440, 3505, 3523, 3578, 5302
- \page@numR .. 136, 143, 150, 3500, 3518
- \page@start 786, 3116
- \pagebreak 5279
- \pagelinesep 31, 3735, 3744–3746
- \pageno 115, 117, 3087
- \pageparbreak 38, 1410
- \pageref 28
- \pairs 5204, 5207, 5222, 5226, 5240, 5244
- \paperwidth 1180, 1183
- \par@line
1017, 1079, 1476, 1477, 1480, 1484
- \para@footgroup 1779, 1862
- \para@footgroupX 2403, 2479
- \para@footsetup 1783, 1791
- \para@footsetupX 2413, 2415
- \para@vfootnote 1777, 1813
- \para@vfootnoteX 2401, 2442
- \parafootfmt 1778, 1852
- \parafootfmtX 2402, 2470
- \parafootftmsep 2712, 2988
- \parafootsep 19
- \parafootstart 1776, 1799
- \parafootstartX 2400, 2424
- \parapparatus@false 11
- \parapparatus@true 15
- \parledgroup@ 1724, 1877,
1985, 2053, 2202, 2308, 2375, 2492
- \parledgroup@beforenotesL 3626, 3678
- \parledgroup@beforenotesR 3624, 3676
- \parledgroup@series .. 1725, 1878,
1986, 2054, 2203, 2309, 2376, 2493
- \parledgroup@type ... 1726, 1879,
1987, 2055, 2204, 2310, 2377, 2494
- \patchcmd 4803, 4806,
4807, 4810, 4814, 4815, 4909,
4929, 4937, 4947, 4955, 4963,
4971, 4980, 4989, 5024, 5032,
5040, 5048, 5057, 5065, 5073, 5081
- \pausenumbering 8, 280
- \pend ... 6, 64, 105, 108, 1049, 1071,
1126, 1410, 4055, 4062, 4741,
4743, 4749, 4751, 4757, 4759,
4765, 4767, 4773, 4775, 4781,
4783, 4792, 4795, 4798, 4800, 4813
- Plato of Tivoli 4
- \postbodyfootmark 2122, 2136
- \postdisplaypenalty 1106
- \prebodyfootmark 2122, 2134
- \predisdisplaypenalty 1105
- \prenotesX 20, 1754
- \prenotesX@ 1753, 1754, 2175, 2425
- \prepare@edindex@fornote
..... 2740, 2753, 3748
- \pretocmd 2105,
3858, 4804, 4808, 4917, 4998, 5010
- \prev@line 966–968, 970,
974–976, 978, 985, 986, 999, 1000
- \prev@nopb 5267
- \prev@pb 5267
- \prevgraf 1078
- \prevline 1916, 3013
- \prevpage@num 1914
- \preXnotes 20, 1747, 1751
- \preXnotes@ 1698, 1747, 1751, 1801
- \print@eledsection 1149, 1171
- \print@footnoterule
. 2189, 2211, 2217, 2317, 2323,
2384, 2390, 2438, 2501, 2507, 2564
- \print@leftmargin@eledsection ..
..... 4876, 4940, 4950,
4973, 4991, 5035, 5051, 5068, 5084
- \print@line 1150, 1153
- \print@rightmargin@eledsection ..
..... 4876, 4932, 4958,
4975, 4993, 5027, 5043, 5060, 5076
- \print@Xfootnoterule
. 1711, 1733, 1739, 1810, 1886,
1892, 1994, 2000, 2062, 2068, 2564
- \printafternumberinfootnote
..... 3047, 3083
- \printbeforenumberinfootnote ...
..... 3044, 3080
- \printendlines 2616, 2645
- \printlinefootnote
.... 1605, 1855, 1970, 2034, 2989
- \printlinefootnotearea
..... 3027, 3031, 3035, 3043
- \printlinefootnotenumbers
..... 3058, 3062, 3068, 3073
- \printlines 1687, 3078
- \printnpnum 26, 2647, 2650, 2655

`\printpstart` 1624, 3077
`\processl@denbody`
 3931, 3935, 3936, 3952
`\ProcessOptions` 20
`\protected@csxdef` 2772, 4096
`\protected@edef`
 1066, 2163, 2287, 2354, 2471
`\protected@write`
 .. 928, 945, 3265, 3278, 3792,
 3794, 3797, 3804, 3806, 3809,
 3814, 3816, 3819, 3843, 3846, 3850
`\protected@xdef` 2224
`\ProvidesPackage` 3
`\pst@rtedLfalse` ... 190, 218, 234, 256
`\pst@rtedLtrue` 190, 286
`\pstart` ... 6, 64, 99, 102, 103, 108,
 1022, 1125, 1410, 4020, 4742,
 4745, 4750, 4753, 4758, 4761,
 4766, 4769, 4774, 4777, 4782,
 4785, 4793, 4795, 4799, 4801, 4813
`\pstarteref` 3332
`\pstartinfootnote` .. 16, 316, 322, 328
`\pstartinfootnoteeverytime` 16
`\pstartline` 1082, 1085
`\pstartnum` 1412
`\pstartnumfalse` 1443, 1450
`\pstartnumtrue` 1092, 1439
`\pstartref` 27, 3332
`\reserveinserts` 24, 38
`\resetprevline@` 60, 247, 471, 1085, 1244
`\resetprevpage@` 475
`\resetprevpage@num` . 60, 248, 475, 3685
`\restore@familiarnotes` .. 4092, 4128
`\restore@notes` 4122, 4329,
 4348, 4364, 4380, 4395, 4412, 4610
`\restore@sidenotes` 4109, 4127
`\resumenumbering` 8, 280
`\right` 4465, 4468, 4473, 4476
`\rightctab` 4542, 4634
`\rightlinenum`
 .. 10, 423, 1344, 1352, 4881, 4897
`\rightlinenumR` 4882, 4898
`\rightltab` 4555, 4616
`\rightnoteupfalse` 29
`\rightnoteuptrue` 3449
`\rightpstartnum` 1420, 1428, 1445
`\rightrtab` 4568, 4580
`\rightstartnum` 1412
`\rigidbalance` ... 1926, 1981, 2006,
 2045, 2074, 2304, 2328, 2371, 2395
`\rlap` 1344, 1352, 1420, 1428, 4880, 4896
`\Rlineflag` 3829, 3830, 3856, 3883, 3887
Robinson, Peter 2
`\roman` 4488
`\rtab` 4075, 4577, 4723
`\rtabtext` 4078, 4595, 4727

Q

`\quotation` 4738
`\quote` 4738

R

`\RaggedLeft` 1864, 1896, 2481, 2511
`\RaggedRight` ... 1865, 1897, 2482, 2512
`\raggedright` 1967,
 2031, 2296, 2363, 3426, 4981, 4983
`\raggedX` 19
`\raw@text` 1017, 1055, 1081, 1137
`\rbracket` 21, 1612, 2707
`\read@linelist` 479, 724
`\ref` 28
`\Relax` 4218, 4672, 4679
`\rem@inder` 1331, 1333–1335
`\removehboxes`
 1867, 1899, 1903, 2484, 2514
`\removelastskip` 4319, 4338
`\RequirePackage`
 22, 23, 25–28, 36, 37, 39–42

S

Sacrobosco 5
`\sameword` 14, 833, 925
`\sameword@inedtext` 833, 982
`\sc@n@list` 1332, 1334
Schöpf, Rainer 4
`\section@num` 186, 210,
 212, 213, 229, 230, 287–289,
 965, 967, 968, 970, 984, 986, 2614
`\section@numR`
 973, 975, 976, 978, 998, 1000
`\sectionmark` 4867, 5222, 5226
`\select@lemmafонт` 827, 1510
`\select@lemmafонт` 21,
 1510, 1606, 1856, 1971, 2035, 2617
`\series` .. 2665, 2806, 2808, 2814, 2818
`\seriesatbegin` 2805
`\seriesatend` 2813
`\set@line` 854, 890
`\set@line@action` 517, 601, 608, 619, 705
`\setcommand@series` .. 2849, 2865, 2938

- \setl@dlp@rbox . 3534, 3571, 3586, 3588
- \setl@drp@rbox . 3542, 3573, 3581, 3593
- \setl@drpr@box 3534
- \setline 11, 95, 798, 831, 1085
- \setlinenum 11, 97, 805
- \setmcellcenter 4385, 4445
- \setmcellleft 4354, 4434
- \setmcellright 4317, 4423
- \setmrowcenter 4443, 4638
- \setmrowleft 4432, 4620
- \setmrowright 4421, 4584
- \setnotesXpositionliketwocolumns@
 - 2148,
 - 2188, 2210, 2216, 2316, 2322,
 - 2383, 2389, 2437, 2500, 2506, 2548
- \setnotesXwidthliketwocolumns@ .
 - 2146, 2187, 2209,
 - 2215, 2315, 2321, 2382, 2388,
 - 2416, 2436, 2488, 2499, 2505, 2518
- \setprintendlines 2621, 2646
- \setprintlines 1661, 1688
- \setstanzaindents 22, 3987
- \setstanzapenalties 24, 3987
- \setstanzavalues 3977, 3987, 3988, 4072
- \settcclcenter 4402, 4450
- \settcclleft 4370, 4439
- \settcclright 4336, 4428
- \settoggle 1518, 1522, 2834
- \settoggle@series
 - 2826, 2832, 2955, 2969
- \setthrowcenter 4448, 4646
- \setthrowleft 4437, 4628
- \setthrowright 4426, 4600
- \setXnotespositionliketwocolumns@
 - 1574,
 - 1710, 1732, 1738, 1809, 1885,
 - 1891, 1993, 1999, 2061, 2067, 2548
- \setXnoteswidthliketwocolumns@ .
 - 1572, 1709, 1731,
 - 1737, 1792, 1808, 1874, 1884,
 - 1890, 1992, 1998, 2060, 2066, 2518
- \Setzen 4663, 4672, 4674
- \showlemma 30, 37, 825, 863, 877
- \showwordrank 15, 995, 1009, 1013
- \sidenote@margin 3383, 3503, 3521, 3576
- \sidenote@marginR . . 3388, 3498, 3516
- \sidenotecontent@
 - . 3554, 3559, 3560, 3571, 3573,
 - 3581, 3582, 3586, 3588, 3589, 3593
- \sidenotemargin 28, 3383
- \sidenotemmargin 122
- \sidenotesep 29, 3552, 3560
- \skip . 1701, 1704, 1722, 1765, 1770,
 - 1782, 1788, 1804, 1807, 1875,
 - 1983, 2051, 2178, 2181, 2200,
 - 2254, 2259, 2306, 2373, 2407,
 - 2412, 2428, 2431, 2435, 2489,
 - 3124, 3624, 3626, 3670, 3676, 3678
- \skip@lockoff 638, 664
- \skipnumbering 12, 815,
 - 4045, 4741, 4743, 4749, 4751,
 - 4757, 4759, 4765, 4767, 4773,
 - 4775, 4781, 4783, 4792, 4798,
 - 4811, 4812, 4820, 4833, 4842, 4855
- \skipnumbering@reg 815
- \spacefactor
 - 2084, 2087, 2095, 2101, 2125, 2131
- \spaceskip 1575, 2149, 2283
- \splitmaxdepth 1580
- \splitoff 1926
- \splittopskip . . . 1134, 1580, 1928,
 - 1979, 1981, 2004, 2006, 2043,
 - 2045, 2072, 2074, 2302, 2304,
 - 2326, 2328, 2369, 2371, 2393, 2395
- \spreadmath 32, 4651
- \spreadtext 32, 4649
- \stanza 22, 4028
- \stanza@count . . . 3968, 3979, 3982,
 - 3984, 4013, 4025, 4034, 4044, 4063
- \stanza@chang 4011, 4036
- \stanza@line 4011, 4049, 4063
- \stanza@modulo 3991, 3994–
 - 3996, 4005–4007, 4016, 4034, 4043
- \stanzaindent 23, 3999
- \stanzaindent* 23, 3999
- \stanzaindentbase 22,
 - 3968, 4000, 4014, 4017, 4023, 4066
- \startlock 11, 812, 829, 4021
- \startstanzahook 25, 4028
- \startsub 11, 788, 828
- \stepcounter
 - 2771, 3288, 3291, 3740, 4496
- \stepl@dcclcount 4184, 4229,
 - 4241, 4326, 4345, 4361, 4377,
 - 4392, 4409, 4455, 4656, 4665, 4686
- \StrGobbleLeft 3757, 3762
- \strip@pt 1797, 2422
- \strip@szacnt 3977
- \sub@action 533, 628, 704

- \sub@change 246,
527, 528, 534, 578, 580, 583, 585
 - \sub@clock 243, 459, 542, 544,
546, 549, 646, 647, 649, 650,
668, 669, 671, 1214, 1286, 1288,
1289, 1291, 1367, 1403, 1405, 1407
 - \sub@off 577, 794
 - \sub@on 577, 790
 - \subline@num 241, 429, 430,
456, 550, 554, 564, 589, 590,
592, 604, 622, 711, 715, 1215,
1220, 1243, 1251, 1306–1308, 3313
 - \sublinenumberstyle 11, 414
 - \sublinenumincrement 9, 374
 - \sublinenumr@p 414
 - \sublinenumrep 414,
430, 1691, 1695, 2648, 2652, 3313
 - \sublineref 27, 3329
 - \sublines@false .. 244, 457, 531, 1276
 - \sublines@true 457, 529, 1274
 - \sublock@disp .. 405, 1369, 1373, 1377
 - \sublockdisp 85, 405
 - \subsection 2823,
4759, 4767, 5239, 5243, 5248, 5249
 - \subsectionmark 4868, 5240, 5244
 - \subsubsection 2825,
4775, 4783, 5256, 5257, 5260, 5261
 - Sullivan, Wayne 4,
5, 22, 37, 46, 50, 112, 113, 155, 178
 - \sw@atthisline
.. 986, 988, 994, 1000, 1002, 1008
 - \sw@list 497, 923, 929, 930, 971
 - \sw@list@inedtext
..... 498, 923, 933, 935, 989, 992
 - \sw@list@inedtextR 950, 952, 1003, 1006
 - \sw@listR 946, 947, 979
 - \symlinenum 16
 - \symplinenum 2699, 2976
 - \sza@penalty 4011, 4040, 4062
- T**
- \tabellzwischen 4654, 4662
 - \tabelskip 4666, 4706–4708, 4714–4716
 - \tabHilfbox 4705,
4707, 4709, 4713, 4715, 4717, 4718
 - \tabhilfbox 4704,
4706, 4708, 4712, 4714, 4716, 4718
 - Tapp, Christian 3
 - \temp@ 1173, 1174, 3893, 3898
 - \textbf 877
 - \textheight 3228
 - \textnormal 1612–1614
 - \textsc 877
 - \textsuperscript 1015, 2077, 2139, 2172
 - \textwidth 3648, 3700
 - \the@sw 970, 971, 978,
979, 990, 992, 995, 1004, 1006, 1009
 - \theaddcolcount 4487, 4494, 4497
 - \theendpageline
3745, 3795, 3807, 3817, 3835, 3847
 - \thefootnoteA 30
 - \thelabidx
3741, 3744, 3903, 3907, 3912, 3914
 - \theline 3292, 3294
 - \thempfn 3650, 3686, 3713
 - \thempfootnote 3650, 3686, 3713
 - Theodosius 5
 - \thepage 741, 744,
746, 755, 757, 760, 3266, 3279, 3744
 - \thepageline
3743, 3798, 3810, 3820, 3838, 3851
 - \thepstart
.. 8, 1022, 1125, 1441, 1448, 1632
 - \thepstartL 1629
 - \thepstartR 1627
 - \thestartpageline
3745, 3793, 3805, 3815, 3834, 3844
 - \thesubline 3292
 - \thinspace 1617, 1619
 - \thisfootnote 2224, 2225
 - \thr@@ 356, 649, 658, 669, 676,
1281, 1289, 1952, 1955, 1981,
2006, 2342, 2345, 2371, 2395, 3408
 - \threecolfootfmt 1942, 1962
 - \threecolfootfmtX 2333, 2353
 - \threecolfootgroup 1943, 1977
 - \threecolfootgroupX 2334, 2368
 - \threecolfootsetup 1944, 1950
 - \threecolfootsetupX 2335, 2340
 - \threecolvfootnote 1941, 1957
 - \threecolvfootnoteX 2332, 2347
 - \togglefalse ... 1701, 1804, 2178, 2428
 - \toggletrue 1748, 1752
 - \tolerance 1966, 2030, 2295, 2362
 - \twocolfootfmt 2011, 2019
 - \twocolfootfmtX 2265, 2286
 - \twocolfootgroup 2012, 2019
 - \twocolfootgroupX 2266, 2301
 - \twocolfootsetup 2013, 2019
 - \twocolfootsetupX 2267, 2272

- `\twocolvfootnote` 2010, [2019](#)
`\twocolvfootnoteX` 2264, [2279](#)
`\txtbeforeXnotes` 19
- U**
- `\unexpanded` 1023, 1100,
 5098, 5102, 5111, 5115, 5124,
 5128, 5136, 5140, 5150, 5154,
 5163, 5167, 5176, 5180, 5189, 5193
`\unhbox` 1131, 1846, 1867, 1869, 1899,
 1901, 1908, 1912, 2484, 2486,
 2514, 2516, 4708, 4709, 4716, 4717
`\unkern` 2085
`\unless` 926, 929,
 932, 946, 949, 964, 983, 3728, 3731
`\unpenalty` 1849, 1905
`\unvbox` ... 1138, 1585, 1717, 1744,
 1830, 1844, 1863, 1895, 1937,
 2154, 2195, 2220, 2458, 2480,
 2510, 2578, 2584, 2593, 2598,
 3102, 3123, 3128, 3147, 3157,
 3162, 3164, 3183, 3211, 3667, 3682
`\unvxh` ... 1821, 1836, [1843](#), 2450, 2464
`\usingcritext` [4729](#)
`\usingedtext` [4729](#)
- V**
- `\valign` 1929
`\value` 3995, 4007, 4012, 4493
`Vamana` 5
`\variab` [4262](#), 4583,
 4599, 4619, 4627, 4637, 4645, 4678
`\vbadness` 1133, 1928
`\vbfnoteX` 2225, [2230](#)
`\vbox` 1055,
 1584, 1819, 1829, 1834, 1844,
 2153, 2448, 2457, 2462, 2566,
 2572, 2577, 2592, 3099, 3120,
 3146, 3236, 3240, 3538, 3540,
 3546, 3548, 3645, 4464, 4467,
 4472, 4475, 4479, 4481, 4482,
 4525, 4529, 4534, 4545, 4558, 4571
`\vfil` 1929, 3240,
 4465, 4468, 4473, 4476, 4479, 4482
`\vfill` 3124
`\vl@dbfnote` [2111](#)
`\vl@dcsnote` 3486, 3490, [3496](#)
`\vl@dlsnote` 3456, 3460, [3496](#)
`\vl@drsnote` 3471, 3475, [3496](#)
`\vnumfootnoteX` [2234](#), 2247
- W**
- `\wd` 1125, 1160, 1823,
 1838, 2452, 2466, 4190, 4191,
 4314, 4538, 4547, 4550, 4560,
 4563, 4572, 4706, 4707, 4714, 4715
`Whitney, Ron` 4
`\widowpenalty` 1106, 1482
`\widthliketwocolumnstrue` 18
`\WithSuffix` 4003, 4747, 4763,
 4779, 4796, 5146, 5159, 5172, 5185
`\wrap@edcrossref`
 [3315](#), 3320, 3323, 3329, 3332
`Wujastyk, Dominik` 2, 3
- X**
- `\x@lemma` 865–867
`\xcritext` 4194, 4307
`\xedindex` [4200](#), 4286, 4297, 4309
`\xedlabel` [4198](#), 4315
`\xedtext` [4194](#), 4308
`\Xendlemmadisablefontselection` . 18
`\Xendnotefontsize` 18
`\Xendnotenumfont` 18
`\Xhangindent` 18
`\xifinlist`
 739, 740, 750, 751, 1148, 1174,
 2671, 5279, 5282, 5284, 5291, 5292
`\xleft@appenditem` [448](#)
`\Xlemmadisablefontselection` 18
`\xlineref` 27, [3323](#), 3744
`\Xnotefontsize` 18
`\Xnotenumfont` 18
`\Xnoteswidthliketwocolumns` 21
`\xpageref` 27, [3320](#)
`\xpstartref` 27, [3332](#)
`\Xragged` 19
`\xright@appenditem` [442](#), 617,
 618, 620, 627, 629, 631, 633,
 643, 645, 654, 665, 667, 674,
 687, 688, 697, 713, 933, 935,
 950, 952, 971, 979, 1518, 1522,
 1530, 1532, 1536, 1538, 1545,
 1548, 1551, 1556, 1559, 1562,
 2114, 2225, 2740, 2753, 3311,
 3456, 3460, 3471, 3475, 3486, 3490
`\xspaceskip` 1575, 2149, 2283

<code>\xsublineref</code>	27, 3329	Z
<code>\xxref</code>	28, 3357	<code>\z@skip</code> 1575, 1581, 2149, 2283
		<code>\zz@@@</code> ... 3247 , 3258, 3271, 3360, 3365

Change History

v0.1.	they disabled page/line refs in a footnotes 75
General: First public release	1
v0.2.	<code>\zz@@@</code> : Minor change to <code>\zz@@@</code> . 155
General: Added tabmac code, and extended indexing	1
<code>\eledmac@error</code> : Added <code>\eledmac@error</code> and replaced error messages ..	43
<code>\ifl@dmemoir</code> : Added <code>\ifl@dmemoir</code> for memoir class having been used	42
<code>\morenoexpands</code> : Added <code>\l@dtabnoexpands</code> to <code>\no@expands</code>	75
v0.2.1.	
<code>\@lab</code> : Removed page setting from <code>\@lab</code>	157
General: Added text about normal labeling	28
Bug fixes and match with mempatch v1.8	1
Major changes to insert code when memoir is loaded	152
<code>\doextrafeet</code> : Renamed <code>\doextrafeet</code> to <code>\l@ddoextrafeet</code>	151
<code>\edlabel</code> : Tweaked <code>\edlabel</code> to get correct page numbers ...	155
<code>\l@d@makecol</code> : Rewrote <code>\@makecol</code> , calling it <code>\l@d@makecol</code>	150
<code>\l@ddodoreinextrafeet</code> : Renamed <code>\dodoreinextrafeet</code> to <code>\l@ddodoreinextrafeet</code>	152
<code>\l@ddofootinsert</code> : Renamed <code>\dofootinsert</code> as <code>\l@ddofootinsert</code>	150
<code>\m@m@makecolintro</code> : Added <code>\m@m@makecolfloats</code> , <code>\m@m@makecoltext</code> and <code>\m@m@makecolintro</code> ...	150
<code>\morenoexpands</code> : Removed some <code>\lets</code> from <code>\no@expands</code> . These were in EDMAC but I feel that they should not have been as	
v0.2.2.	
General: Improved paragraph footnotes	1
New Dekker example	1
Used <code>\providecommand</code> for <code>\@gobblethree</code> to avoid clash with the amsfonts package ...	46
<code>\footfudgefiddle</code> : Added <code>\footfudgefiddle</code>	110
<code>\line@list@stuff</code> : Added initial write of page number in <code>\line@list@stuff</code>	69
<code>\para@footsetup</code> : Added <code>\footfudgefiddle</code> to <code>\para@footsetup</code>	111
<code>\para@footsetupX</code> : Added <code>\footfudgefiddle</code> to <code>\para@footsetupX</code>	130
v0.3.	
<code>\@lab</code> : Replaced <code>\the\line@num</code> by <code>\linenumr@p\line@num</code> in <code>\@lab</code> , and similar for sub-lines	157
<code>\@nl@reg</code> : Added a bunch of code to <code>\@nl</code> for handling <code>\setlinenum</code>	62
General: Includes edstanza and more	1
<code>\ledlinenum</code> : Added <code>\linenumr@p</code> and <code>\sublinenum@rep</code> to <code>\leftlinenum</code> and <code>\rightlinenum</code>	54
<code>\linenumberlist</code> : Added <code>\linenumberlist</code> mechanism .	46
<code>\printendlines</code> : Added <code>\linenumr@p</code> and <code>\sublinenumr@p</code> to <code>\printendlines</code>	136
<code>\printlines</code> : Added <code>\linenumr@p</code> and <code>\sublinenumr@p</code> to	

<code>\printlines</code>	106	<code>\xympar</code> : Eliminated <code>\marginpar</code>	
<code>\sublinenumr@p</code> : Added <code>\linenumberstyle</code>		disturbance	160
and <code>\sublinenumberstyle</code> . . .	54	General: Added left and right side	
v0.3.1.		notes	160
General: Not released. Added re-		Added sidenotes, familiar foot-	
marks about the parallel pack-		notes in numbered text	1
age	1	v0.5.1.	
v0.4.		General: Added moveable side note	160
<code>\@iiiminipage</code> : Modified ker-		Fixed right line numbers killed in	
nel <code>\@iiiminipage</code> and		v0.5	1
<code>\endminipage</code> to cater for criti-		<code>\affixline@num</code> : Changed	
cal footnotes	167	<code>\affixline@num</code> to cater for	
General: Added minipage, etc., sup-		sidenotes	93
port	1	<code>ledgroupsize</code> : Only change	
<code>ledgroup</code> : Added <code>ledgroup</code> environ-		<code>\hsize</code> in <code>ledgroupsize</code> envi-	
ment	168	ronment otherwise page number	
<code>ledgroupsize</code> : Added <code>ledgroup</code> -		can be in wrong place	168
sized environment	168	<code>\l@dgetsidenote@margin</code> : Added	
<code>\edtext</code> : Added <code>\showlemma</code> to		<code>\sidenotemargin</code> and	
<code>\edtext</code> (and <code>\critext</code>) . . .	76	<code>\sidenote@margin</code>	160
<code>\footnormal</code> : Added minpage foot-		v0.6.	
note setup to <code>\footnormal</code> .	109	<code>\@lopR</code> : Added <code>\@pend</code> , <code>\@pendR</code> ,	
<code>\ifledsecnolinenum</code> : Added fi-		<code>\@lopL</code> and <code>\@lopR</code> in anticipa-	
nal/draft options	41	tion of parallel processing . . .	64
<code>\l@dfeetendmini</code> : Added		<code>\@nl@reg</code> : Added <code>\fix@page</code> to	
<code>\l@dfeetbeginmini</code> , <code>\l@dfeetendmini</code>		<code>\@nl</code>	62
and all their supporting code	166	Extended <code>\@nl</code> to include the	
<code>\mpnormalfootgroup</code> : Added		page number	62
<code>\mpnormalfootgroup</code>	107	General: Fixed long paragraphs	
<code>\mpnormalvfootnote</code> : Added		looping	1
<code>\mpnormalvfootnote</code>	102	Fixed minor typos	1
<code>\showlemma</code> : Added <code>\showlemma</code> .	41	Prepared for <code>eledpar</code> package . .	1
v0.4.1.		<code>\fix@page</code> : Added <code>\last@page@num</code>	
<code>\@opxtrafeetii</code> : Added <code>\@opxtrafeetii</code>		and <code>\fix@page</code>	63
.	151	<code>\new@line</code> : Extended <code>\new@line</code> to	
General: Added code for changing		output page numbers	69
<code>\@doclearpage</code>	153	<code>\page@start</code> : Made <code>\page@start</code> a	
Not released. Minor editorial im-		no-op	70
provements and code tweaks . .	1	<code>\vl@dbfnote</code> : Changed <code>\l@dbfnote</code>	
Only change <code>\@footnotetext</code>		and <code>\vl@dbfnote</code> as originals	
and <code>\@footnotemark</code> if memoir		could give incorrect markers in	
not used	121	the footnotes	122
<code>\doxtrafeetii</code> : Changed		v0.7.	
<code>\doxtrafeetii</code> code for easier		<code>\@nl@reg</code> : Added <code>\@nl@reg</code>	62
extensions	151	<code>\@ref@reg</code> : Added <code>\@ref@reg</code> . . .	67
<code>\edindex</code> : Let <code>eledmac</code> take advan-		General: <code>eledmac</code> having been avail-	
tage of memoir's indexing . .	171	able for 2 years, deleted the	
v0.5.		commented out original <code>edmac</code>	
<code>\@footnotetext</code> : Enabled regular		texts	1
<code>\footnote</code> in numbered text	122		

Maïeul Rouquette new main-tainer	1	\ledlinenum: Added \ledlinenum for use by \leftlinenum and \rightlinenum	54
Made macros of all messages ..	42		
Replaced all \interAfootnotelinepenalty\line@list@stuff: Deleted etc., by just \interfootnotelinepenalty \page@start from \line@list@stuff	1		69
Tidying up for eledpar and ledarab packages	1	\list@clearing@reg: Added	
\affixline@num: Added skipnum-mering to \affixline@num ..	93	\list@clearing@reg	61
\do@actions@fixedcode: Added		\n@num@reg: Added \n@num	67
\do@actions@fixedcode	92	\normalbfnoteX: Removed extraneous space from \normalbfnoteX	125
\do@actions@next: Added number skipping to \do@actions	91	\resumenummering: Changed	
\do@insidelinehook: Added		\resumenummering for eledpar	50
\do@linehook for use in \do@line	89	\setprintendlines: Added	
\endnumbering: Changed		\setprintendlines for use by \printendlines	136
\endnumbering for eledpar ..	49	\setprintlines: Added \setprintlines for use by \printlines	105
\f@x@l@cks: Added \ch@cksub@l@ck, \ch@ck@l@ck and \f@x@l@cks	95	\skipnumbering@reg: Added	
\footplitskips: Added		\skipnumbering and supports	72
\footplitskips for use in many footnote styles	102	\sublinenumincrement: Added	
\get@linelistfile: Added		\firstlinenum, \linenumincrement, \firstsublinenum and \linenumincrement	53
\get@linelistfile	61	\sublinenumr@p: Using \linenumrep instead of \linenumr@p	54
\ifledRcol@: Added \l@dnumpstartsL, \ifl@dpairing and \ifpst@rted for/from eledpar	47	Using \sublinenumrep instead of \sublinenumr@p	54
\initnumbering@reg: Added		\vnumfootnoteX: Removed extraneous space from \vnumfootnoteX	125
\initnumbering@reg	48		
\l@dcstotext@r: Added			
\l@demptyd@ta	89		
\l@ddofootinsert: Deleted			
\page@start from \l@ddofootinsert	150		
\l@dgetline@margin: Added			
\l@dgetline@margin	52		
\l@dgetlock@disp: Added			
\l@dgetlock@disp	53		
\l@dgetsidenote@margin: Added			
\l@dgetsidenote@margin ..	160		
\l@drsn@te: Added \l@dlsn@te and \l@drsn@te for use in \do@line	89		
\l@dunboxmpfoot: Added			
\l@dunboxmpfoot containing some common code	167		
\l@dzeropenalties: Added			
\l@dzeropenalties	85		

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 use non-switching commands 1

v0.9.
General: No more ledpatch. All patches are now in the main file. 1

v0.9.1.
General: Fix some bugs linked to integrating ledpatch on the main file. 1

v0.10.	General: Corrections to <code>\section</code> and other titles in numbered sections	1	<code>\edlabel</code> : Tweaked <code>\edlabel</code> to get correct line number if the command is first element of a paragraph.	155
v0.11.	General: Makes it possible to add a symbol on each verse's hanging, as in French typography. Redefines the command <code>\hangingsymbol</code> to define the character.	1	v0.15.	General: Line numbering can be reset at each <code>pstart</code>
				Possibility to print <code>\pstart</code> number inside.
				<code>\affixline@num</code> : Line numbering can be disabled.
				<code>\ifinserthangingsymbol</code> : New management of <code>hangingsymbol</code> insertion, preventing undesirable insertions.
				<code>\printlines</code> : Line numbering can be reset at each <code>pstart</code>
v0.12.	General: For compatibility with <code>eledpar</code> , possibility to use <code>\autopar</code> on the right side. . . .	1	v0.17.	<code>\ifinserthangingsymbol</code> : New new management of hanging-symbol insertion, preventing undesirable insertions.
	Possibility to number <code>\pstart</code> . . .	8		
	Possibility to number the <code>pstart</code> with the commands <code>\numberpstarttrue</code>	1	v1.0.	General: <code>\lemma</code> can contain commands.
	<code>\ifledRcol@</code> : Added <code>\ifledRcol</code> and <code>\ifnumberingR</code> for/from <code>eledpar</code>	47		Debug in lineation command . . .
v0.12.1.	General: Don't number <code>\pstarts</code> of stanza.	1		New generic commands to customize footnote display.
	The numbering of <code>\pstarts</code> restarts on each <code>\beginnumbering</code>	1		Options <code>nonum</code> and <code>nosep</code> in <code>\Xfootnote</code>
v0.13.	General: New <code>stanzaindentsrepetition</code> counter to repeat stanza indents every n verses.	23		Options of <code>\Xfootnotes</code>
	New <code>stanzaindentsrepetition</code> counter: to repeat stanza indents every n verses.	1		Possibility to have commands in sidenotes.
	<code>\managestanza@modulo</code> : New <code>stanzaindentsrepetition</code> counter to repeat stanza indents every n verses.	180		Some compatibility break with <code>eledmac</code> . Change of name: <code>eledmac</code>
v0.13.1.	General: <code>\thepstartL</code> and <code>\thepstartR</code> use now <code>\bfseries</code> and not <code>\bf</code> , which is deprecated and makes conflicts with memoir class.	1		<code>\morenoexpands</code> : Change to be compatible with new features . . .
v0.14.	General: Tweaked <code>\edlabel</code> to get correct line number if the command is first element of a paragraph.	1	v1.0.1.	General: Correction on <code>\numberonlyfirstinline</code> with lineation by <code>pstart</code> or by page.
			v1.1.	General: Add <code>\labelpstarttrue</code> . . .
				Add <code>\numberonlyfirstintwolines</code>
				Add <code>\pstartinfootnote</code> and <code>\onlypstartinfootnote</code>
				New hook to add arbitrary code at the beginning of the notes . .

New options for block of notes.	19	v1.4.3.	General: Add <code>\nonbreakableafternumber</code>	17
New package option: <code>parapparat</code>	1		Spurious space after familiar footnotes.	1
New tools to change order of series	141	v1.4.4.	General: Label inside familiar footnotes.	1
<code>\ledfootinsdim</code> : Deprecated	109	v1.4.5.	General: Bug with <code>komasscript</code> + <code>eledpar</code> + chapter.	1
<code>\preXnotes</code> : New skip <code>\preXnotes@</code>	109	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>	202
General: Sectioning commands.	34	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. . .	125
<code>\endquote</code> : Compatibility of <code>\ledchapter</code> with the <i>memoir</i> class.	202	v1.5.	General: Correct indexing when the call is made in critical notes. .	169
<code>\preXnotes</code> : Debug in familiar footnotes (but introduced by v1.1). .	109		<code>\do@insidelinehook</code> : Added <code>\do@insidelinehook</code> for use in <code>\do@line</code>	89
v1.3.			<code>\edindex</code> : Compatibility with <code>imakeidx</code> package, and possibility to use multiple index with <code>\edindex</code>	171
<code>\endquote</code> : <i>Quotation</i> and quote environment inside the numbering sections.	202		<code>\ifFN@bottom</code> : Use the bottom option of <code>footmisc</code> package. . .	150
v1.4.		v1.5.1.	<code>\managestanza@modulo</code> : Correct <code>stanzaindentsrepetition</code> counter	180
General: Compatibility with LuaTeX of RTL notes.	41, 42		<code>\normalvfootnoteX</code> : Fix bug with normal familiar footnotes when mixing RTL and LTR text. .	123
<code>\edtext</code> : Compatibility of <code>\edtext</code> (and <code>\critext</code>) with the right-to-left direction (with <i>Polyglossia</i>).	76	v1.6.0.	<code>\falseverse</code> : Add <code>\falseverse</code> macro.	181
<code>\newseries@</code> : Remembers the language of the lemma, in order to create a correct direction for the footnote separator.	139	v1.6.1.	General: Corrects a false hanging verse when a verse is exactly the	
<code>\normalfootfmt</code> : Direction of footnotes with <i>polyglossia</i>	103			
<code>\rbracket</code> : Switch the right bracket to a left bracket when the lemma is RTL (needs <i>polyglossia</i> or <i>LuaTeX</i>).	104			
v1.4.1.				
<code>\affixside@note</code> : Remove spurious spaces.	164			
<code>\endquote</code> : New option <i>noquotation</i>	202			
<code>\labelrefsparsesubline</code> : Fix bug with <code>\edlabel</code>	156			
v1.4.2.				
General: Debug with some special classes.	1			

length of a line.	1	<code>\l@dfheetendmini</code> : Add <code>\mpfnpos</code> to choice the order of footnotes in minipage / ledgroup. . . .	166
<code>\AtEveryPstart</code> : Spurious space in <code>\pstart</code>	83	v1.10.0. General: Add <code>\pstartref</code> and <code>\xpstartref</code> to refer to a pstart number (extension of <code>\edlabel</code>).	1
<code>\ifinserthangingsymbol</code> : Hang verse is now not automatically flush right.	179	<code>\endquote</code> : Correction of sectioning commands in parallel texts. . .	202
<code>\l@dunhbox@line</code> : Move the call to <code>\inserthangingsymbol</code> to al- low use <code>\hfill</code> inside.	87	v1.10.1. General: Compatibility with <code>clev- eref</code>	1
<code>\pend</code> : Spurious space in <code>\pend</code> . .	84	v1.10.2. General: Compatibility of stanza with v1.8a of <code>babel-greek</code>	1
v1.7.0. General: New features for managing page breaks.	36	v1.10.3. General: Debug of cross- referencing.	1
v1.8.0. General: Compatibility with <code>parled-</code> <code>group</code> option of <code>eledpar</code> pack- age.	1	v1.10.4. General: Debug of critical notes in <code>edtabular</code> environment.	1
If <code>imakeidx</code> and <code>hyperref</code> are loaded, adds <code>hyperref</code> in the in- dex.	169	v1.10.5. General: Debug of <code>\pausenumbering</code>	1
<code>\endquote</code> : Correction of sectioning commands in parallel texts. . .	202	Debug of <code>\xxref</code>	1
<code>\get@index@command</code> : Debug <code>\get@index@command</code> and com- patibility with <code>hyperref</code> pack- age.	170	v1.10.6. General: Debug of interac- tion between <code>\autopar</code> and <code>\pausenumbering</code>	1
<code>\newhookcommand@series@reload</code> : Debug <code>\beforenotesX</code> and <code>\maxhnotesX</code> which didn't work.	144	v1.11.0. General: Add hooks to disable the font selection for lemma in foot- note.	18
<code>\prevpage@num</code> : Correct <code>\parafootsep</code> when using with <code>ledgroup</code> . . .	116	v1.11.1. General: Correct a bug when a crit- ical note starts with plus or mi- nus.	1
v1.8.1. General: Debug endnotes when more than one series is used (change the position where tools for endnotes are defined). . .	135	v1.12.0. <code>\@nl@reg</code> : To ensure compatibility with <code>\musixtex</code> , <code>\@l</code> becomes <code>\@1</code> . Consequently, <code>\@l@reg</code> be- comes <code>\@nl@reg</code>	62
v1.8.2. General: Debug compatibility prob- lem with <code>hebrew</code> option of <code>babel</code> package.	1	General: Add <code>\ledinnernote</code> and <code>\ledouternote</code> commands. . .	28
v1.8.3. General: Fixes spurious spaces added by v1.7.0.	1	Add hyperlink to crossref (needs <code>hyperref</code> package).	27
v1.8.5. General: Debug indexing in right column, with <code>eledpar</code>	169	Compatibility with <code>musixtex</code> . . .	1
v1.9.0. <code>\doextrafeet</code> : Add <code>\fnpos</code> to choice the order of footnotes.	151		

Debug <code>\eledmac</code> sectioning command after using <code>\resumenumbering</code>	1	<code>\Stanza</code> can have an optional argument: content to be printed before.	181
Ensure that <code>imakeidx</code> is loaded before <code>\eledmac</code>	169	Add <code>\newverse</code> macro, <code>\falseverse</code> deprecated. . . .	181
New hooks: <code>\afterXrule</code> and <code>\afterruleX</code>	20	v1.12.1.	
New options for ragged-paragraph notes.	19	<code>\wrap@edcrossref</code> : Fix spurious spaces.	157
New sectioning commands. . . .	34	v1.12.2.	
Optional arguments for <code>\pstart</code> and <code>\pend</code>	7	<code>\l@dunhbox@line</code> : Fix a bug with critical notes at the tops of pages (added by v12.0.0)	87
<code>\AtEveryPstart</code> : New optional argument for <code>\pstart</code> , to execute code before it.	83	v1.12.3.	
<code>\edindex</code> : Use correctly default index when <code>imakeidx</code> is loaded. . . .	171	General: Add macros for new messages since v0.7	42
<code>\endquote</code> : <code>\ledxxx</code> sectioning commands are deprecated and replaced by <code>\eledxxx</code> commands.	202	Correct bug with side and familiar notes in tabular environments.	1
<code>\ifledRcol@</code> : Add <code>\ifledRcol@</code> for <code>\eledpar</code>	47	Debug <code>\eledxxx</code> with some paper size.	1
<code>\initnumbering@reg</code> : <code>\beginnumbering</code> is defined only on <code>\eledmac</code> , not on <code>\eledpar</code>	48	Debug <code>\ledinnernote</code> and <code>\ledouternote</code> commands in the top of pages.	28
<code>\l@dcsnote</code> : <code>\l@dlsnote</code> , <code>\l@drsnote</code> and <code>\l@dcsnote</code> defined only one time, in <code>\eledmac</code> , including needs for <code>\eledpar</code> case.	162	Debug left and right notes (bugs added by 1.12.0)	1
<code>\l@dgetsidenote@margin</code> : <code>\sidenotemargin</code> is now directly defined in <code>\eledmac</code> to be able to manage <code>\eledpar</code>	160	Underline lemma in <code>\eledxxx</code> when using draft mode.	1
<code>\l@dunhbox@line</code> : <code>\do@line</code> is split in more little commands.	87	<code>\eledmac@error</code> : Replaced error messages	43
<code>\newhookcommand@series@reload</code> : Debug <code>\beforenotesX</code> and <code>\maxhnotesX</code> which didn't work when called after <code>\footparagraphX</code>	144	<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>	70
Debug <code>\beforeXnotes</code> and <code>\maxhXnotes</code> which didn't work when called after <code>\footparagraph</code>	144	<code>\flag@start</code> send a error message when a <code>\edtext</code> is done without insert (note)	70
<code>\pend</code> : New optional argument for <code>\pend</code> , to execute code after it. . . .	84	v1.12.4.	
<code>\stanza</code> : & can have an optional argument: content to be printed after.	181	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	174
		<code>\@ssect</code> : Debug <code>\eledchapter</code> in parallel with <code>memoir</code>	207
		<code>\doinsidelinehook</code> : Added <code>\dolinehook</code> and <code>\doinsidelinehook</code>	89

<code>\endnumbering</code> : Allow to mix parallel columns and normal text when using <code>\pausenumbering</code> 49	
<code>\l@dgobblearg</code> : <code>\l@dgobblearg</code> becomes <code>\l@dgobbleoptarg</code> .. 187	
<code>\l@drestoreforedtext</code> : Debug optional arguments of <code>\Xfootnote</code> in tabular context 189	
<code>\resumenumbering</code> : Debug <code>\resumenumbering</code> 50	
v1.12.6.	
<code>\noeledsec</code> : Add <code>\noeledsec</code> macro. 211	
v1.12.7.	
<code>\wrap@edcrossref</code> : <code>\wrap@edcrossref</code> is now robust 157	
v1.12.8.	
<code>\flag@end</code> : <code>\flag@start</code> don't send a error message when a <code>\edtext</code> is done without insert (note) but have a endnote 70	
v1.13.0.	
General: Add <code>\Xnoteswidthliketwocolumns</code> and <code>\notesXwidthliketwocolumns</code> 20	
<code>\ifledsecnolinenummer</code> : Added <code>widthliketwocolumns</code> option .. 41	
<code>\newhooktoggle@series</code> : Add <code>\newhookcommand@toggle@reload</code> 145	
<code>\para@footsetupX</code> : In <code>\para@footsetupX</code> , use <code>\columnwidth</code> instead of <code>\hsize</code> 130	
v1.13.1.	
General: Coming back of page and line breaking penalties's management, deleted by error in v0.17. 1	
Debug quotation environment inside of a <code>\pstart</code> preceded by a sectioning command. 1	
<code>\thepstart</code> : Add <code>\l@dzeropenalties</code> in <code>\pstart</code> 83	
v1.13.2.	
General: Fix bug with normal footnotes, added by v1.13.0. 1	
<code>\ifledRcol@</code> : Add <code>\ifl@dpaging</code> for <code>eledpar</code> 47	
v1.13.3.	
General: Fix extra spaces with paragraphed footnotes, added by v1.13.0. 1	
v1.13.4.	
General: Fix bug with index when memoir class is used without hyperref 1	
v1.14.0.	
General: Debug spurious characters before endnotes. 135	
Delete previous override of <code>\l@d@wrindexhyp</code> at the beginning of a document when hyperref is not loaded. 175	
Moves gobbling command 46	
Provide <code>\@gobblefour</code> 46	
<code>\edindex</code> : Let <code>eledmac</code> take advantage of <code>imakeidx</code> even when memoir class is used 171	
v1.14.1.	
<code>\@ssect</code> : Debug sectioning commands when using both <code>handout</code> and <code>hyperref</code> package. 209	
v1.14.2.	
<code>\@ssect</code> : Debug <code>\edtext</code> after starred sectioning commands when using <code>memoir</code> class. ... 207	
v1.15.0.	
General: Fix bug with footnotes layout when using some options of the geometry package (bug add by v1.13.0). 1	
New commands <code>\AtEveryPstart</code> and <code>\AtEveryPend</code> 7	
New tools to prevent ambiguous references in lemma 14	
<code>\endsub</code> : Restore subline feature (disabled by mistake in v1.8.0). 71	
<code>\footparagraphX</code> : Correct bug with paragraphed familiar footnotes setting. 129	
<code>\if@lemmacommand@</code> : New boolean <code>\iflemmacommand@</code> 78	
v1.15.1.	
<code>\line@list@stuff</code> : Revert modification of 1.5.2 which makes bug with numbering. Leave vertical mode to solve spurious space before minipage. 69	

v1.16.0.	
General: Compatibility of standard footnotes with some biblatex styles.	1
New <code>\stanzaindent</code> command. .	1
<code>\critext</code> : <code>\critext</code> and <code>\edtext</code> are now defined only in <code>eledmac</code> , not in <code>eledpar</code> . Debug wrong numbering when using <code>\sameword + eledpar + \tag</code> command.	76
v1.16.1.	
<code>\lineref</code> : <code>\lineref</code> is not defined if defined by some other package, like <code>lineno</code> . <code>Eledmac</code> provides <code>\edlineref</code> instead. . .	157
v1.17.0.	
<code>\critext</code> : The historical <code>\critext</code> now just refers to <code>\edtext</code> (code refactoring).	76
<code>\edtext</code> : Error message when calling <code>\edtext</code> outside of a numbered paragraph.	76
v1.18.0.	
<code>\@edindex@hyperref</code> : Fix spurious space with <code>\edindex</code> when using <code>imakeidx/indextools + hy-</code>	
<code>perref</code>	174
General: Add <code>\pstartinfootnoteeverytime</code>	16
Compatibility with Lua ^A T _E X RTL languages.	1
Debug <code>\onlypstartinfootnote</code> when using <code>\numberonlyfirstinline</code> and the current line number differs from the previous.	16
<code>\edlabel</code> : <code>\edlabel</code> is now defined only one time for both <code>eledmac</code> and <code>eledpar</code>	155
<code>\ifledRcol@</code> : Add <code>\ifl@dprintingpages</code> and <code>\@dprintingcolumns</code> for <code>eledpar</code>	47
<code>\l@d@section</code> : Option <code>parapparat</code> works for endnotes.	135
<code>\print@line</code> : Compatibility with Lua ^A T _E X RTL languages. . . .	87
<code>\printlinefootnote</code> : Code refactoring in <code>\printlinefootnote</code> : the printing of the numbers are factorized in <code>\printlinefootnotearea</code> . .	146
<code>\printpstart</code> : Debug <code>\pstartinfootnote</code> with parallel pages and columns (<code>eledpar</code>)	104