

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 LaTeX. The principal functions provided by the package are marginal line numbering and multiple series of footnotes and endnotes keyed to line numbers.

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

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

In October 2012, Maïeul Rouquette released the *eledform* package¹. Based on eledmac, this package provides tools for creating a formal description (formalism) of textual variants.

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 bug in English or in French.

You can subscribe to the eledmac mail list in:

<https://lists.berlios.de/pipermail/ledmac-users/>

*This file (eledmac.dtx) has version number v1.7.0, last revised 2013/11/08.

[†]herries dot press at earthlink dot net

[‡]maieul at maieul dot net

¹<http://www.ctan.org/eledform>.

Contents

1 Introduction	5
1.1 Overview	5
1.2 History	7
1.2.1 EDMAC	7
1.2.2 eledmac	8
2 The eledmac package	9
3 Numbering text lines and paragraphs	9
3.1 Lineation commands	12
3.2 Changing the line numbers	13
4 The apparatus	14
4.1 Commands	14
4.2 Alternate footnote formatting	17
4.3 Display options	17
4.3.1 Control line number printing	18
4.3.2 Separator between the lemma and the note content	19
4.3.3 Font style	19
4.3.4 Styles of notes content	20
4.3.5 Arbitrary code at the beginning of notes	20
4.3.6 Options for notes in columns	20
4.3.7 Options for paragraphed footnotes	20
4.3.8 Options for block of notes	21
4.4 Page layout	21
4.5 Fonts	21
4.6 Create a new series	23
5 Verse	23
5.1 Repeating stanza indents	23
5.2 Stanza breaking	24
5.3 False verse	25
5.4 Hanging symbol	25
5.5 Long verse and page break	25
5.6 Various tools	25
5.7 Hanging symbol	26
6 Grouping	26
7 Crop marks	27
8 Endnotes	27
9 Cross referencing	27

<i>Contents</i>	3
10 Side notes	29
11 Familiar footnotes	29
12 Indexing	30
13 Tabular material	31
14 Sectioning commands	34
15 Quotation environments	35
16 Page breaks	36
17 Miscellaneous	36
17.1 Known and suspected limitations	37
17.2 Use with other packages	38
17.3 Parallel typesetting	39
17.4 Notes for EDMAC users	39
18 Implementation overview	42
19 Preliminaries	42
19.1 Messages	44
20 Sectioning commands	46
21 Line counting	51
21.1 Choosing the system of lineation	51
21.2 List macros	56
21.3 Line-number counters and lists	57
21.4 Reading the line-list file	61
21.5 Commands within the line-list file	62
21.6 Writing to the line-list file	69
22 Marking text for notes	73
22.1 <code>\edtext</code> and <code>\critext</code> themselves	74
22.2 Substitute lemma	79
22.3 Substitute line numbers	79
23 Paragraph decomposition and reassembly	80
23.1 Boxes, counters, <code>\pstart</code> and <code>\pend</code>	80
23.2 Processing one line	84
23.3 Line and page number computation	85
23.4 Line number printing	88
23.5 Pstart number printing in side	91
23.6 Add insertions to the vertical list	93
23.7 Penalties	94

23.8 Printing leftover notes	94
24 Footnotes	95
24.1 Fonts	95
24.2 Outer-level footnote commands	96
24.3 Normal footnote formatting	97
24.4 Standard footnote definitions	103
24.5 Paragraphed footnotes	104
24.5.1 Insertion of the footnotes separator	109
24.6 Columnar footnotes	110
25 Familiar footnotes	114
25.1 Generality	114
25.2 Footnote formats	116
25.3 Two columns footnotes	119
25.4 Three columns footnotes	120
25.5 Paragraphed footnotes	121
25.6 Footnotes' output	124
26 Generate series	124
26.0.1 Test if series is still existing	125
26.0.2 Create all commands to memorize display options	125
26.0.3 Create inserts, needed to add notes in foot	126
26.0.4 Create command for critical apparatus, <code>\Xfootnote</code>	126
26.0.5 Create tools for familiar footnotes (<code>\footnoteX</code>)	127
26.0.6 The endnotes	127
26.0.7 Init standards series (A,B,C,D,E,Z)	128
26.0.8 Some tools	128
26.0.9 Old commands, kept for backward compatibility	131
26.0.10 Hooks for a particular footnote	131
26.0.11 Alias	132
26.0.12 Line number printing	132
27 Output routine	134
28 Cross referencing	140
29 Endnotes	144
30 Side notes	147
31 Minipages and such	150
32 Indexing	153
33 Macro as environment	157

34 Verse	160
35 Arrays and tables	164
36 Page breaking or no page breaking depending of specific lines	183
37 Long verse: prevents being separated by a page break	184
38 The End	185
Appendix A Some things to do when changing version	186
Appendix A.1 Migration from ledmac to eledmac	186
Appendix A.2 Migration to eledmac 1.5.1	186
References	188
Index	188
Change History	205

List of Figures

1 Introduction

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

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

Since 2011's Maïeul Rouquette begun to maintain and extend eledmac. As plain T_EX is used by little people, and L^AT_EX by more people eledmac and original EDMAC are more and more distant.

1.1 Overview

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

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

`eledmac` allows the scholar engaged in preparing a critical edition to focus attention wholly on the task of creating the critical text and evaluating the variant readings, text-critical notes and testimonia. \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 17.4); the complete source code for the package, with extensive documentation (in sections 18 and following) ; and an Index to the source code. We do not suggest that you need to read the source code for this package in order to use it; we provide this code primarily for reference, and many of our comments on it repeat material that is also found in the earlier sections. But no documentation, however thorough, can cover every question that comes up, and many can be answered quickly by consultation of the code. On a first reading, we suggest that you should read only the general documentation in sections 2, unless you are particularly interested in the innards of `eledmac`.

1.2 History

1.2.1 EDMAC

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

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

The present version was developed in the summer of 1990, with the intent of adding necessary features, streamlining and documenting the code, and further generalizing it to make it easily adaptable to the needs of editors in different disciplines. John did most of the general reworking and documentation, with the financial assistance of the Division of the Humanities and Social Sciences, California Institute of Technology. Dominik adapted the code to the conventions of Frank Mittelbach’s `doc` option, and added some documentation, multiple-column footnotes, cross-references, and crop marks.² A description by John and Dominik of this version of EDMAC was published as ‘An overview of EDMAC: a PLAIN T_EX format for critical editions’, *TUGboat* 11 (1990), pp. 623–643.

From 1991 through 1994, the macros continued to evolve, and were tested at a number of sites. We are very grateful to all the members of the (now defunct) `edmac@mailbase.ac.uk` discussion group who helped us with smoothing out bugs and infelicities in the macros. Ron Whitney and our anonymous reviewer at the TUG were both of great help in ironing out last-minute wrinkles, while Ron made some important suggestions which may help to make future versions of EDMAC even more efficient. Wayne Sullivan, in particular, provided several important fixes and contributions, including adapting the Mittelbach/Schöpf ‘New Font Selection

²This 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).

Scheme' for use with PLAIN T_EX and EDMAC. Another project Wayne has worked on is a DVI post-processor which works with an EDMAC that has been slightly modified to output \specials. This combination enables you to recover to some extent the text of each line, as ASCII code, facilitating the creation of concordances, an *index verborum*, etc.

At the time of writing (1994), we are pleased to be able to say that EDMAC is being used for real-life book production of several interesting editions, such as the Latin texts of Euclid's *Elements*,³ an edition of the letters of Nicolaus Copernicus,⁴ Simon Bredon's *Arithmetica*,⁵ a Latin translation by Plato of Tivoli of an Arabic astrolabe text,⁶ a Latin translation of part II of the Arabic *Algebra* by Abū Kāmil Shujā' b. Aslam,⁷ the Latin *Rithmachia* of Werinher von Tegernsee,⁸ a middle-Dutch romance epic on the Crusades,⁹ a seventeenth-century Hungarian politico-philosophical tract,¹⁰ an anonymous Latin compilation from Hungary entitled *Sermones Compilati in Studio Generali Quinqueecclesiensi in Regno Ungarie*,¹¹ the collected letters and papers of Leibniz,¹² Theodosius's *Spherics*, the German *Algorismus* of Sacrobosco, the Sanskrit text of the *Kāśikāvṛtti* of Vāmana and Jayāditya,¹³ 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.

³Gerhard 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).

⁴Being prepared at the German Copernicus Research Institute, Munich.

⁵Being prepared by Menso Folkerts *et al.*, at the Institut für Geschichte der Naturwissenschaften in Munich.

⁶Richard Lorch, Gerhard Brey *et al.*, at the same Institute.

⁷Richard Lorch, 'Abū Kāmil on the Pentagon and Decagon' in *Vestigia Mathematica*, ed. M. Folkerts and J. P. Hogendijk (Amsterdam, Atlanta: Rodopi, 1993).

⁸Menso Folkerts, 'Die *Rithmachia* des Werinher von Tegernsee', *ibid.*

⁹Geert H. M. Claassens, *De Middelnederlandse Kruisvaartromans*, (Amsterdam: Schipphower en Brinkman, 1993).

¹⁰Emil Hargittay, *Csáky István: Politica philosophiai Okoskodás-szerint való rendes életnek példája (1664–1674)* (Budapest: Argumentum Kiadó, 1992).

¹¹Being produced, as was the previous book, by Gyula Mayer in Budapest.

¹²Leibniz, *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>)

¹³Being prepared at Poona and Lausanne Universities.

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 4.3 p.17). They can make some little troubles with old customization. That is why a new name was selected: *eledmac*. To migrate from ledmac to eledmac, please read Appendix Appendix A.1 (p.186).

2 The eledmac package

eledmac is a three-pass package like LaTeX itself. Although your textual apparatus and line numbers will be printed even on the first run, it takes two more passes through LaTeX 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 LaTeX 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 Numbering text lines and paragraphs

<code>\beginnumbering</code> <code>\endnumbering</code>	<p>Each section of numbered text must be preceded by <code>\beginnumbering</code> and followed by <code>\endnumbering</code>, like:</p> <pre> \beginnumbering <text> \endnumbering </pre>
--	---

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).

`\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:

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

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

The lines of this paragraph are
not numbered.

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

1 This is a sample paragraph
2 with lines numbered
3 automatically.
4 This paragraph too
5 has its lines automatically
6 numbered.
7 And here the numbering
8 begins again.

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

```
\begingroup
\beginnumbering
\autopar

A paragraph of numbered text.

Another paragraph of numbered
text.

\endnumbering
\endgroup
```

1 A paragraph of numbered
2 text.
3 Another paragraph of
4 numbered text.

`\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.¹⁴

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

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

`\firstsublinenum` There are similar commands, `\firstsublinenum{<num>}` and `\sublinenumincrement{<num>}` for controlling sub-line numbering.

`\sublinenumincrement` `eledmac` stores a lot of information about line numbers and footnotes in memory as it goes through a numbered section. But at the end of such a section, it empties its memory out, so to speak. If your text has a very long numbered section it is possible that your LaTeX may reach its memory limit. There are two solutions to this. The first is to get a larger LaTeX with increased memory. The second solution is to split your long section into several smaller ones. The trouble with this is that your line numbering will start again at zero with each new section. To avoid this problem, we provide `\pausenumbering` and `\resumenumbering` which are just like `\endnumbering ... \beginnumbering`, except that they arrange for your line numbering to continue across the break. Use `\pausenumbering` only between numbered paragraphs:

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

\resumenumbering
\pstart
3 Another paragraph.
\pend
\endnumbering
```

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

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

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

`\numberpstarttrue` It's possible to insert a number at every `\pstart` command. You must use the `\numberpstarttrue` command to have it. You can stop the numbering with

¹⁴For a detailed study of the reasons for this restriction, see Barbara Beeton, 'Initiation rites', *TUGboat* **12** (1991), pp. 257–258.

`\numberpstartfalse` `\numberpstartfalse`. You can redefine the command `\thepstart` to change style. On each `\beginnumbering` the numbering restarts.

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

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

3.1 Lineation commands

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

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

```
\linenummargin{left}
```

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

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

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

`\linenumberlist` You can define `\linenumberlist` to specify a non-uniform distribution of printed line numbers. For example:

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

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

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

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

and `linenumincrement` counter values.

`\leftlinenum` When a marginal line number is to be printed, there are a lot of ways to
`\rightlinenum` display it. You can redefine `\leftlinenum` and `\rightlinenum` to change the
`\linenumsep` 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.

3.2 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-
`\endsub` 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
`\endlock` 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
`\advanceline` calculated: if you are printing only fragments of a work but want to print line num-
bers appropriate to a complete version, for example. The `\setline{<num>}` and
`\advanceline{<num>}` commands may be used to change the current line's num-
ber (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 out-
side such a group, for example between a `\pend` and a `\pstart`. It sets the line

number to $\langle num \rangle$. It has no effect if used within a `\pstart... \pend` group

`\linenumberstyle` Line numbers are normally printed as arabic numbers. You can use `\linenumberstyle{\langle style \rangle}`

`\sublinenumberstyle` to change the numbering style. $\langle style \rangle$ 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{\langle style \rangle}` 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.

If you want to use the feature in a stanza, you should look at the `\falseverse` macro (p. 25).

4 The apparatus

4.1 Commands

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

`\edtext{\langle lemma \rangle}{\langle commands \rangle}`

The $\langle lemma \rangle$ argument is the lemma in the main text: `\edtext` both prints this as part of the text, and makes it available to the $\langle commands \rangle$ 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 $\langle lemma \rangle$ 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> <code>\edtext{Smith}{\Afootnote{Jones</code> <code>C, D.}} on Tuesday.}{</code> <code>\Bfootnote{The date was</code> <code>July 16, 1954.}</code> <code>}</code>	1 I saw my friend 2 Smith on Tuesday. <hr/> 2 Smith] Jones C, D. <hr/> 1–2 I saw my friend 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, `\langle commands \rangle`, may contain a series of subsidiary commands that generate various kinds of notes.

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

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

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

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

<code>\Aendnote</code> <code>\Bendnote</code> <code>\Cendnote</code> <code>\Dendnote</code> <code>\Eendnote</code>	The package also maintains five separate series of endnotes. Like footnotes each macro takes a single argument like <code>\Aendnote{\langle text \rangle}</code> . Normally, none of them are printed: you must use the <code>\doendnotes</code> macro described below (p. 27) to call for their output at the appropriate point in your document.
--	--

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

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

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

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

`\linenum` You can use `\linenum{<arg>}` to change the line numbers passed to the notes. The notes are actually given seven parameters: the page, line, and sub-line 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. 27) 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:

```
\let\variant=\Afootnote
\let\explanatory=\Bfootnote
\let\trivial=\Aendnote
\let\testimonia=\Cfootnote
```


Formalism for textual criticism If your notes are for textual criticism, you should use the *eledform* package¹⁵.

This package provides tools to describes the textual variants in a formal way. It is based on *eledmac* for the typographical aspect.

4.2 Alternate footnote formatting

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

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

- `\footparagraph` formats all the footnotes of a series as a single paragraph;
- `\foottwocol` formats them as separate paragraphs, but in two columns;
- `\footthreecol`, in three columns.

Each of these macros takes one argument: a letter (between A and E) for the series of notes you want changed. So a text with three layers of notes might begin thus:

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

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

4.3 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 minus b minus c**. The final length *m* is calculated by L^AT_EX to have: $b - a \leq m \leq b + c$. If you use relative unity¹⁶, it will be relative to fontsize of the footnote.

4.3.1 Control line number printing

<code>\numberonlyfirstinline</code>	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 <code>\numberonlyfirstinline[⟨s⟩]</code> . Use <code>\numberonlyfirstinline[⟨s⟩][⟨false⟩]</code> to cancel it (⟨s⟩ can be empty if you want to disable it for every series).
<code>\numberonlyfirstintwolines</code>	Suppose you have a lemma on line 2 and a lemma between line 2 and line 3. With <code>\numberonlyfirstinline</code> , the second lemma is considered to be on the same line as the first lemma. But if you use both <code>\numberonlyfirstinline[⟨s⟩]</code> and <code>\numberonlyfirstintwolines[⟨s⟩]</code> , the distinction is made. Use <code>\numberonlyfirstintwolines[⟨s⟩]</code> to cancel it (⟨s⟩ can be empty if you want to disable it for every series).
<code>\symlinenum</code>	For setting a particular symbol in place of the line number, you can use <code>\symlinenum[⟨s⟩]{⟨symbol⟩}</code> in combination with <code>\numberonlyfirstinline[⟨s⟩]</code> . From the second lemma of the same line, the symbol will be used instead of line number.
<code>\nonumberinfootnote</code>	You can use <code>\nonumberinfootnote[⟨s⟩]</code> if you don't want to have the line number in a footnote. To cancel it, use <code>\nonumberinfootnote[⟨s⟩][⟨false⟩]</code> .
<code>\pstartinfootnote</code>	You can use <code>\pstartinfootnote[⟨s⟩]</code> if you want to print the pstart number in the footnote, before the line and subline number. Use <code>\pstartinfootnote[⟨s⟩][⟨false⟩]</code> 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 : <ul style="list-style-type: none"> • If you use lineation by pstart, the option is enabled. • If you use lineation by section or by page, the option is disabled.
<code>\onlypstartinfootnote</code>	In combination with <code>\pstartinfootnote</code> , you can use <code>\onlypstartinfootnote[⟨s⟩]</code> if you want to print only the pstart number in the footnote, and not the line and subline number. Use <code>\onlypstartinfootnote[⟨s⟩][⟨false⟩]</code> to cancel it (⟨s⟩ can be empty if you want to disable it for every series).
<code>\beforenumberinfootnote</code>	With <code>\beforenumberinfootnote[⟨s⟩]{⟨l⟩}</code> , 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.
<code>\afternumberinfootnote</code>	With <code>\afternumberinfootnote[⟨s⟩]{⟨l⟩}</code> 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.
<code>\nonbreakableafternumber</code>	By default, the space defined by <code>\afternumberinfootnote</code> is breakable. With <code>\nonbreakableafternumber[⟨s⟩]</code> it becomes nonbreakable. Use <code>\nonbreakableafternumber[⟨s⟩]</code> to cancel it (⟨s⟩ can be empty if you want to disable it for every series).
<code>\beforesymlinenum</code>	With <code>\beforesymlinenum[⟨s⟩]{⟨l⟩}</code> you can add some space before the line symbol in a footnote. The default value is value set by <code>\beforenumberinfootnote</code> .
<code>\aftersymlinenum</code>	With <code>\aftersymlinenum[⟨s⟩]{⟨l⟩}</code> you can add some space before the line symbol in a footnote. The default value is value set by <code>\afternumberinfootnote</code> .
<code>\inplaceofnumber</code>	If no number or symbolic line number is printed, you can add a space, with <code>\inplaceofnumber[⟨s⟩]{⟨l⟩}</code> . The default value is 1 em.
<code>\boxlinenum</code>	It could be useful to put the line number inside a fixed box: the content of

¹⁵<http://www.ctan.org/pkg/eledform>.

¹⁶Like `em` which is the width of a M.

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.

4.3.2 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⟩]{⟨l⟩}` to change it. The optional argument can be used to specify in which series it is applied. Note that there is a non-breakable space between lemma and separator, but **breakable** space between separator and lemma.

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

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

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

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

4.3.3 Font style

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

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

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

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

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

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

4.3.4 Styles of notes content

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

`\hangindentX` For familiar notes NOT paragraphed you can define an indent with `\Xhangindent[$\langle s \rangle$]{ $\langle l \rangle$ }`, 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.

4.3.5 Arbitrary code at the beginning of notes

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

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

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

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

`\bhookXendnote` `\bhookXendnote[$\langle series \rangle$]{ $\langle code \rangle$ }` is to be used at the beginning of the end-notes.

4.3.6 Options for notes in columns

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

`\hsizetwocol` `\hsizetwocol[$\langle s \rangle$]{ $\langle l \rangle$ }` is used to change width of a column when critical notes are displaying in two columns. Default value is `.45 \hspace`.

`\hsizethreecol` `\hsizethreecol[$\langle s \rangle$]{ $\langle l \rangle$ }` is used to change width of a column when critical notes are displaying in three columns. Default value is `.3 \hspace`.

`\hsizetwocolX` `\hsizetwocol[$\langle s \rangle$]{ $\langle l \rangle$ }` is used to change width of a column when familiar notes are displaying in two columns. Default value is `.45 \hspace`.

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

4.3.7 Options for paragraphed footnotes

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

`\parafootsep` For paragraphed footnotes (see below), you can choose the separator between

each note by `\parafootsep[⟨s⟩]{⟨l⟩}`. A common separator is double pipe (`$||$`), which you can set by `\parafootsep$||$`.

4.3.8 Options for block of notes

<code>\txtbeforeXnotes</code>	You can add some text before critical notes with <code>\textbeforeXnotes[⟨s⟩]{⟨text⟩}</code> .
<code>\beforeXnotes</code>	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.
<code>\beforenotesX</code>	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.
<code>\preXnotes</code>	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> . You can disable this feature by setting the length to 0 pt.
<code>\prenotesX</code>	You can want the space before the first printed (in a page) series of familiar notes not to be the same as before other series. You can do it with <code>\prenotesX{⟨l⟩}</code> . You can disable this feature by setting the length to 0 pt.
<code>\maxhXnotes</code>	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> .
<code>\maxhnotesX</code>	<code>\maxhnotesX[⟨s⟩]{⟨l⟩}</code> is the same as previous, but for familiar footnotes. Be careful with the two previous commands. Actually, for technical purposes, one paragraphed note is considered as one block. Consequently, it can't be broken between two pages, even if you used these commands. The debug is in the todolist.

4.4 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.¹⁷

4.5 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.

¹⁷There 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 107 explains why this restriction is necessary.

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

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

`eledmac` uses `\select@lemmafnt` 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.

4.6 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 Sullivan¹⁸ 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]. I have made a few changes to enable his macros to be used in the LaTeX `eledmac` package.

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

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

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

`\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. 25.

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:

¹⁸Department of Mathematics, University College, Dublin 4, Ireland

```
\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.2 p. 186.

If you don't use the `stanzaindentsrepetition` counter, make sure you have at least one more numerical entry in `\setstanzavalues` than the number of lines in the stanza.

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

```
\setcounter{stanzaindentsrepetition}{0}
```

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

5.2 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.

5.3 False verse

In some special cases, you want to add false verse after true verse. This false verse:

1. Won't be numbered.
2. Won't affect the indent of the next verse.

It could be used, for example, to add some space between verses. To add this type of false verse, you have to finish the previous verse with `\falseverse` (and not with `&`). For example:

```
True verse&
True verse\falseverse
\vspace{3ex}&
True verse&
True verse
```

5.4 Hanging symbol

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

`\hangingsymbol`

```
\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. 36 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 else-

where) will typeset $\langle text \rangle$ at a distance $\langle len \rangle$ before the line. The default $\langle len \rangle$ is `\stanzaindentbase`.

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

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

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

5.7 Hanging symbol

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

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

6 Grouping

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

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

You can also put familiar footnotes (see section 11) in a `minipage` but unlike with `\footnote` the numbering scheme is unaltered.

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

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

```
\begin{ledgroupsize}[\langle pos \rangle]{\langle width \rangle}.
```

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

l (left) numbered text is flush left with respect to the normal `textwidth`. This is the default.

c (center) numbered text is in the center of the textwidth.

r (right) numbered text is flush right with respect to the normal textwidth.

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

`\begin{ledgroupsize}{\textwidth}` is effectively the same as `\begin{ledgroup}`

7 Crop marks

The `eledmac` package does not provide crop marks. These are available with either the memoir class [Wil02] or the `crop` package.

8 Endnotes

`\doendnotes` `\doendnotes{<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@lemmafонт` to select fonts, just as the footnote macros do (see p. 95 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.¹⁹

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

An `\edlabel` command may appear in the main text, or in the first argument of `\edtext`, but not in the apparatus itself. But `\edpageref`, `\lineref` and

¹⁹More precisely, you should stick to characters in the T_EX categories of 'letter' and 'other'.

`\sublineref` commands can also be used in the apparatus to refer to `\edlabel`'s in the text.

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

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

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

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

`\xpageref` However, there are situations in which you'll want `eledmac` to return a number
`\xlineref` without displaying any warning messages about undefined labels or the like: if
`\xsublineref` you want to use the reference in a context where L^AT_EX is looking for a number,
such a warning will lead to a complaint that the number is missing. This is the
case for references used within the argument to `\linenum`, for example. For this
situation, three variants of the reference commands, with the `x` prefix, are supplied:
`\xpageref`, `\xlineref`, and `\xsublineref`. They have these limitations: they
will not tell you if the label is undefined, and 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.

`\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.16 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 docu-
ment, so that you can see them at a glance.

`\label` The normal `\label`, `\ref` and `\pageref` macros may be used within num-
`\ref`
`\pageref`

bered text, and operate in the familiar fashion.

10 Side notes

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

`\ledleftnote` `\ledleftnote{<text>}` will put `<text>` into the left margin level with where the command was issued. Similarly, `\ledrightnote{<text>}` puts `<text>` in the right margin. `\ledsidenote{<text>}` will put `<text>` into the margin specified by the current setting of `\sidenotemargin{<location>}`. The permissible value for `<location>` is one out of the list `left`, `right`, `inner`, or `outer`, for example `\sidenotemargin{outer}`. The package's default setting is `\sidenotemargin{right}`

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

If two, say, `\ledleftnote`, commands are called in the same line the second `<text>` will obliterate the first. There is no problem though with having both a left and a right sidenote on the same line.

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

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

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

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

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

These can of course be changed to suit.

`\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}{\textsuperscript{\normalfont,}}
```

and can be changed if necessary.

`\footnoteA` As well as the standard LaTeX footnotes generated via `\footnote`, the package also provides three series of additional footnotes called `\footnoteA` through `\footnoteE`. These have the familiar marker in the text, and the marked text at the foot of the page can be formatted using any of the styles described for the critical footnotes. Note that the ‘regular’ footnotes have the series letter at the end of the macro name whereas the critical footnotes have the series letter at the start of the name.

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

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

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

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

```
\newcommand*{\bodyfootmarkA}{%
  \hbox{\textsuperscript{\normalfont\@nameuse{@thefnmarkA}}}}
```

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

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

There are similar command triples for the other series.

Additional footnote series can be easily defined: you just have to use `\newseries`, defined above (see 4.6 p.23).

12 Indexing

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

If the `memoir` class or the `imakeidx` 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`.
2. Load `eledmac`.

3. Declare the index with the macro `\makeindex` of `imakeidx`.

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

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

13 Tabular material

LaTeX'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 analagous command
for use in `edtabular` environments.

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

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

```


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

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
a	└───		d
A	B	C	D

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

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

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

`\edbeforetab` `\edbeforetab{ $\langle text \rangle$ }{ $\langle entry \rangle$ }`, where $\langle entry \rangle$ is an entry in the leftmost column, typesets $\langle text \rangle$ left justified before the $\langle entry \rangle$. Similarly `\edaftertab{ $\langle entry \rangle$ }{ $\langle text \rangle$ }`,

where $\langle entry \rangle$ is an entry in the rightmost column, typesets $\langle text \rangle$ right justified after the $\langle entry \rangle$.

For example:

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

	A 1 2 3 B 1 3 6 C 1 4 8 D 1 5 0	After
--	--	-------

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

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

However, `eledmac` provides the following commands:

- `\ledchapter[$\langle text \rangle$]{ $\langle critical text \rangle$ }`
- `\ledchapter*`
- `\ledsection[$\langle text \rangle$]{ $\langle critical text \rangle$ }`

- `\ledsection*`
- `\ledsubsection[<text>]{<critical text>}`
- `\ledsubsection*`
- `\ledsubsubsection[<text>]{<critical text>}`
- `\ledsubsubsection*`

Which are the equivalent of the standard LaTeX commands, but be careful. Note the following points:

1. All these commands close a `\pstart`, and open a new one. The content of the command itself is between `\pstart` and `\pend`.
2. Don't try to make `\let\chapter\ledchapter`, or other things like it: the `\ledsection` commands call the standard commands.
3. For the non-starred sections, use the optional argument *<text>* to provide the text to the table of contents.
4. The `\ledchapter` doesn't open a new page. You must use `\beforeledchapter` before. This also closes a `\pstart` and opens a new.

`\ledsectnotoc` You can create a table of contents that indexes only the titles that appear on the left side of the edition: for instance, titles from the original language, not the translation. You could use `\ledsectnotoc` at the beginning of the side environment :

```
\begin{Rightside}
\ledsectnotoc
...
\end{Rightside}
```

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 the quotation environments inside a `\start-\pend` block, not outside.

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

`\ledpb` and `\ledpar` 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 `\ledpbsetting{after}`, the page break will be placed at the end of the long verse, and the page containing the long verse will have one extra line.

17 Miscellaneous

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

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

`\showlemma` The lemma within the text is printed via `\showlemma{lemma}`. Normally, or with the ‘final’ option, the definition of `\showlemma` is:
`\newcommand*{\showlemma}[1]{#1}`
 so it just produces its argument. With the ‘draft’ option it is defined as
`\newcommand*{\showlemma}[1]{\textit{#1}}`

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

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

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

17.1 Known and suspected limitations

In general, `eledmac`'s system for adding marginal line numbers breaks anything that makes direct use of the 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 17, p. 21, and described in more detail on p. 106, 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 LaTeX 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
```

²⁰Reported by Dirk-Jan Dekker in the CTT thread 'Incompatibility of "color" package' on 2003/08/28.

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

```
... \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 have developed the `Ledpar` package as an adjunct to `eledmac` specifically for parallel typesetting of critical texts. This also cooperates with the `babel` package for typesetting in multiple languages. The package is called *eledpar* since september 2012.

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

17.4 Notes for EDMAC users

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

The package still provides the original `\text` command, but it is (a) deprecated, and (b) its name has been changed²¹ 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

²¹A name like `\text` is likely to be defined by other LaTeX packages (it certainly is by the AMS packages) and it seems sensible to try and avoid clashes with other definitions.

to generate notes. The / at the end terminates the command; it is part of the macro's definition so that spaces after the macro will be treated as significant.

For example:

I saw my friend \critext{Smith}	1 I saw my friend
\Afootnote{Jones C, D.}/	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 `\critext` commands. Nesting makes it possible to print an explanatory note on a long passage together with notes on variants for individual words within the passage. For example:

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

The second argument of the `\critext` macro, *commands*, is the same as the second argument to the `\edtext` macro.

It is possible to define aliases for `\critext`, which can be easier to type. You can make a single character substitute for `\critext` by saying this:

```
\catcode'\<=\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.

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 27) finish the main part of the processing; cross-referencing (Section 28) and endnotes (Section 29) complete the story.

In what follows, macros with an `@` in their name are more internal to the workings of `eledmac` than those made up just of ordinary letters, just as in PLAIN `TEX` (see *The TeXbook*, p.344). You are meant to be able to make free with ordinary macros, but the '`@`' ones should be treated with more respect, and changed only if you are pretty sure of what you are doing.

19 Preliminaries

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

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

```
1 (*code)
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{eledmac}[2013/11/08 v1.7.0 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.

`\ifledfinal` Use this to remember which option is used, set and execute the options with final
`\ifparapparatus@` as the default.

```

4 \newif\ifledfinal
5 \newif\ifparapparatus@
6 \newif\ifnoquotation@
7 \newif\iflednopbinverse
8 \parapparatus@false
9 \DeclareOption{noquotation}{\noquotation@true}
10 \DeclareOption{final}{\ledfinaltrue}
11 \DeclareOption{draft}{\ledfinalfalse}
12 \DeclareOption{parapparatus}{\parapparatus@true}
13 \DeclareOption{nopbinverse}{\lednopbinversetrue}
14 \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.

```

15 \ProcessOptions*\relax
16

```

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.

```

17 \RequirePackage{xargs}
18 \RequirePackage{etoolbox}
19 \reserveinserts{32}
20 \RequirePackage{suffix}
21 \RequirePackage{xstring}
22 \RequirePackage{iflutex}

```

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

```

23 \ifcsdef{if@RTL}{-}{\newif\if@RTL}

```

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

```

24 \ifledfinal
25   \newcommand*{\showlemma}[1]{#1}
26 \else
27   \newcommand*{\showlemma}[1]{\underline{#1}}
28 \fi
29

```

`\linenumberlist` The code for the `\linenumberlist` mechanism was given to Peter Wilson by Wayne Sullivan on 2004/02/11.
Initialize it as `\empty`
30 `\let\linenumberlist=\empty`
31

`\@l@tempcnta` In imitation of L^AT_EX, we create a couple of scratch counters.
`\@l@tempcntb` LaTeX 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).
32 `\newcount\@l@tempcnta \newcount\@l@tempcntb`

`\ifl@dmemoir` Define a flag for if the memoir class has been used.
33 `\newif\ifl@dmemoir`
34 `\@ifclassloaded{memoir}{\l@dmemoirtrue}{\l@dmemoirfalse}`
35

`\ifl@imakeidx` Define a flag for if the imakeidx package has been used.
36 `\newif\ifl@imakeidx`
37 `\@ifpackageloaded{imakeidx}{\l@imakeidxtrue}{\l@imakeidxfalse}`

19.1 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.
38 `\newcommand{\eledmac@warning}[1]{\PackageWarning{eledmac}{#1}}`

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

`\led@err@NumberingStarted`
`\led@err@NumberingNotStarted` 40 `\newcommand*{\led@err@NumberingStarted}{%`
`\led@err@NumberingShouldHaveStarted` 41 `\eledmac@error{Numbering has already been started}{\@ehc}}`
42 `\newcommand*{\led@err@NumberingNotStarted}{%`
43 `\eledmac@error{Numbering was not started}{\@ehc}}`
44 `\newcommand*{\led@err@NumberingShouldHaveStarted}{%`
45 `\eledmac@error{Numbering should already have been started}{\@ehc}}`

`\led@mess@NotesChanged`
46 `\newcommand*{\led@mess@NotesChanged}{%`
47 `\typeout{eledmac reminder: }%`
48 `\typeout{ The number of the footnotes in this section`
49 `has changed since the last run.}%`
50 `\typeout{ You will need to run LaTeX two more times`
51 `before the footnote placement}%`
52 `\typeout{ and line numbering in this section are`
53 `correct.}}`

```

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

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

\led@warn@BadLineation
\led@warn@BadLinenummargin 59 \newcommand*{\led@warn@BadLineation}{%
\led@warn@BadLockdisp      60   \eledmac@warning{Bad \string\lineation\space argument}}
\led@warn@BadSublockdisp 61 \newcommand*{\led@warn@BadLinenummargin}{%
62   \eledmac@warning{Bad \string\linenummargin\space argument}}
63 \newcommand*{\led@warn@BadLockdisp}{%
64   \eledmac@warning{Bad \string\lockdisp\space argument}}
65 \newcommand*{\led@warn@BadSublockdisp}{%
66   \eledmac@warning{Bad \string\sublockdisp\space argument}}

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

\led@warn@BadAdvancelineSubline
\led@warn@BadAdvancelineLine 69 \newcommand*{\led@warn@BadAdvancelineSubline}{%
70   \eledmac@warning{\string\advanceline\space produced a sub-line
71     number less than zero.}}
72 \newcommand*{\led@warn@BadAdvancelineLine}{%
73   \eledmac@warning{\string\advanceline\space produced a line
74     number less than zero.}}

\led@warn@BadSetline
\led@warn@BadSetlinenum 75 \newcommand*{\led@warn@BadSetline}{%
76   \eledmac@warning{Bad \string\setline\space argument}}
77 \newcommand*{\led@warn@BadSetlinenum}{%
78   \eledmac@warning{Bad \string\setlinenum\space argument}}

\led@err@PstartNotNumbered
\led@err@PstartInPstart 79 \newcommand*{\led@err@PstartNotNumbered}{%
\led@err@PendNotNumbered 80   \eledmac@error{\string\pstart\space must be used within a
\led@err@PendNoPstart    81     numbered section}{\@ehc}}
\led@err@AutoparNotNumbered 82 \newcommand*{\led@err@PstartInPstart}{%
83   \eledmac@error{\string\pstart\space encountered while another
84     \string\pstart\space was in effect}{\@ehc}}
85 \newcommand*{\led@err@PendNotNumbered}{%
86   \eledmac@error{\string\pend\space must be used within a
87     numbered section}{\@ehc}}
88 \newcommand*{\led@err@PendNoPstart}{%
89   \eledmac@error{\string\pend\space must follow a \string\pstart}{\@ehc}}

```

```

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

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

\led@warn@DuplicateLabel
\led@warn@RefUndefined 95 \newcommand*{\led@warn@DuplicateLabel}[1]{%
96   \eledmac@warning{Duplicate definition of label ‘#1’ on page \the\pageno.}}
97 \newcommand*{\led@warn@RefUndefined}[1]{%
98   \eledmac@warning{Reference ‘#1’ on page \the\pageno\space undefined.
99     Using ‘000’.}}

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

\led@warn@BadSidenotemargin
102 \newcommand*{\led@warn@BadSidenotemargin}{%
103   \eledmac@warning{Bad \string\sidenotemmargin\space argument}}

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

\led@err@TooManyColumns
\led@err@UnequalColumns 106 \newcommand*{\led@err@TooManyColumns}{%
\led@err@LowStartColumn 107   \eledmac@error{Too many columns}{\@ehc}}
\led@err@HighEndColumn 108 \newcommand*{\led@err@UnequalColumns}{%
\led@err@ReverseColumns 109   \eledmac@error{Number of columns is not equal to the number
110     in the previous row (or \protect\space forgotten?)}{\@ehc}}
111 \newcommand*{\led@err@LowStartColumn}{%
112   \eledmac@error{Start column is too low}{\@ehc}}
113 \newcommand*{\led@err@HighEndColumn}{%
114   \eledmac@error{End column is too high}{\@ehc}}
115 \newcommand*{\led@err@ReverseColumns}{%
116   \eledmac@error{Start column is greater than end column}{\@ehc}}

```

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. LaTeX 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.

```
117 \newcount\section@num
118 \section@num=0
119 \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.

```
120 \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.

```
\l@dpairingtrue
\l@dpairingfalse 121 \newif\ifl@dpairing
\ifp@rtedL 122 \l@dpairingfalse
\p@rtedLtrue 123 \newif\ifp@rtedL
\p@rtedLfalse 124 \p@rtedLfalse
\l@dnumpstartsL 125 \newcount\l@dnumpstartsL
\ifledRcol 126 \newif\ifledRcol
```

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

```
127 \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.

```
128 \newcommand*{\beginnumbering}{%
129   \ifnumbering
130   \led@err@NumberingStarted
131   \endnumbering
```

```

132 \fi
133 \global\numberingtrue
134 \global\advance\section@num \@ne
135 \initnumbering@reg
136 \message{Section \the\section@num }%
137 \line@list@stuff{\jobname.\extensionchars\the\section@num}%
138 \l@dend@stuff
139 \setcounter{pstart}{1}
140 \ifl@dpairing\else\begin@group\fi
141 \initnumbering@sectcmd
142 }
143 \newcommand*\initnumbering@reg{%
144 \global\pst@rtedLfalse
145 \global\l@dnum@startsL \z@
146 \global\abs@line@num \z@
147 \gdef\normal@page@break{}
148 \gdef\l@prev@pb{}
149 \gdef\l@prev@nopb{}
150 \global\line@num \z@
151 \global\sub@line@num \z@
152 \global\@lock \z@
153 \global\sub@lock \z@
154 \global\sub@lines@false
155 \global\let\next@page@num=\relax
156 \global\let\sub@change=\relax
157 \resetprevline@
158 }
159

```

`\initnumbering@sectcmd` `\initnumbering@sectcmd` defines sectioning commands inside numbered section.

`\ledsection` It also defines quotation environment. Note: this supposes that the user didn't

`\ledsection*` change `\chapter`. If he did, he should redefine `\initnumbering@sectcmd`.

`\ledsubsection` 160 `\newcommand{\initnumbering@sectcmd}{`

`\ledsubsection*` 161 `\newcommand{\ledsection}[2] []{%`

`\ledsubsubsection` 162 `\leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\skipnumbering%`

`\ledsubsubsection*` 163 `\pstart%`

`\ledchapter` 164 `\leavevmode\section{##1}{##2}\leavevmode\vspace{2.3ex \@plus .2ex}\skipnumbering%`

`\ledchapter*` 165 `\vspace{-2\parskip}\vspace{-2\baselineskip}%`

`\quotation` 166 `\ifautopar\else\pstart\fi`

`\endquotation` 167 `}`

`\quote` 168 `\WithSuffix\newcommand\ledsection*[1]{%`

`\endquote` 169 `\leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\skipnumbering%`

170 `\pstart%`

171 `\leavevmode\section*{##1}\leavevmode\vspace{2.3ex \@plus .2ex}\skipnumbering\pend%`

172 `\vspace{-2\parskip}\vspace{-2\baselineskip}%`

173 `\ifautopar\else\pstart\fi`

174 `}`

175 `\newcommand{\ledsubsection}[2] []{%`

176 `\leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\skipnumbering%`

177 `\pstart%`


```

178     \leavevmode\subsection[##1]{##2}\leavevmode\vspace{1.5ex \@plus .2ex}\skipnumbering\pend%
179     \vspace{-2\parskip}\vspace{-2\baselineskip}%
180     \ifautopar\else\pstart\fi
181 }
182 \WithSuffix\newcommand\ledsubsection*[1]{%
183     \leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\skipnumbering%
184     \pstart%
185     \leavevmode\subsection*{##1}\leavevmode\vspace{1.5ex \@plus .2ex}\skipnumbering\pend%
186     \vspace{-2\parskip}\vspace{-2\baselineskip}%
187     \ifautopar\else\pstart\fi
188 }
189 \newcommand{\ledsubsubsection}[2][]{%
190     \leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\skipnumbering%
191     \pstart%
192     \leavevmode\subsubsection[##1]{##2}\leavevmode\vspace{1.5ex \@plus .2ex}\skipnumbering\pend%
193     \vspace{-2\parskip}\vspace{-2\baselineskip}%
194     \ifautopar\else\pstart\fi
195 }
196 \WithSuffix\newcommand\ledsubsubsection*[1]{%
197     \leavevmode\pend\vspace{3.5ex \@plus 1ex \@minus .2ex}\skipnumbering%
198     \pstart%
199     \leavevmode\subsubsection*{##1}\leavevmode\vspace{1.5ex \@plus .2ex}\skipnumbering\pend%
200     \vspace{-2\parskip}\vspace{-2\baselineskip}%
201     \ifautopar\else\pstart\fi
202 }
203 \newcommand\ledchapter[2][]{\ifl@dmemoir\gdef\ch@pt@c{##1}\fi~\pend\skipnumbering\pstart\chapter{
204 \WithSuffix\newcommand\ledchapter*[1]{~\pend\skipnumbering\pstart\chapter*{##1}\pend\pstart}
205 \patchcmd{\@makeschapterhead}{1\par}{1}{-}{-}
206 \pretocmd{\@makeschapterhead}{\par}{-}{-}
207 \apptocmd{\@makeschapterhead}{\par}{-}{-}
208 \patchcmd{\@makeschapterhead}{\vskip 40\p@}{-}{-}{-}
209 \patchcmd{\@makechapterhead}{1\par}{1}{-}{-}
210 \pretocmd{\@makechapterhead}{\par}{-}{-}
211 \apptocmd{\@makechapterhead}{\par}{-}{-}
212 \patchcmd{\@makechapterhead}{\vskip 40\p@}{-}{-}{-}
213 \apptocmd{\@chapter}{\par\leavevmode\vspace{40 \p@}\skipnumbering}{-}{-}
214 \apptocmd{\@schapter}{\par\leavevmode\vspace{40 \p@}\skipnumbering}{-}{-}
215 \newcommand\beforeledchapter{\pend\cleardoublepage\pstart}
216 \patchcmd{\chapter}{\cleardoublepage}{\relax}{-}{-}
217 \patchcmd{\chapter}{\clearpage}{\relax}{-}{-}
218 \ifnoquotation@else
219 \renewcommand{\quotation}{\par\leavevmode%
220     \parindent=1.5em%
221     \skipnumbering%
222     \ifautopar%
223         \vskip-\parskip%
224     \else%
225         \vskip\topsep%
226     \fi%
227     \global\leftskip=\leftmargin%

```

```

228             \global\rightskip=\leftmargin%
229         }
230     \renewcommand{\endquotation}{\par%
231         \global\leftskip=0pt%
232         \global\rightskip=0pt%
233         \leavevmode%
234         \skipnumbering%
235         \ifautopar%
236             \vskip-\parskip%
237         \else%
238             \vskip\topsep%
239         \fi%
240     }
241     \renewcommand{\quote}{\par\leavevmode%
242         \parindent=0pt%
243         \skipnumbering%
244         \ifautopar%
245             \vskip-\parskip%
246         \else%
247             \vskip\topsep%
248         \fi%
249         \global\leftskip=\leftmargin%
250         \global\rightskip=\leftmargin%
251     }
252     \renewcommand{\endquote}{\par%
253         \global\leftskip=0pt%
254         \global\rightskip=0pt%
255         \leavevmode%
256         \skipnumbering%
257         \ifautopar%
258             \vskip-\parskip%
259         \else%
260             \vskip\topsep%
261         \fi%
262     }
263     \fi
264 }

```

`\ledsectnotoc` The `\ledsectnotoc` only disables the `\addcontentsline` macro.

```

265 \newcommand{\ledsectnotoc}{\let\addcontentsline\@gobblethree}

```

`\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.

```

266 \def\endnumbering{%
267     \ifnumbering
268         \global\numberingfalse
269         \normal@pars
270         \ifl@dpairing

```

```

271     \global\pst@rtedLfalse
272   \else
273     \ifx\insertlines@list\empty\else
274       \global\noteschanged@true
275     \fi
276     \ifx\line@list\empty\else
277       \global\noteschanged@true
278     \fi
279   \fi
280   \ifnoteschanged@
281     \led@mess@NotesChanged
282   \fi
283 \else
284   \led@err@NumberingNotStarted
285 \fi
286 \autoparfalse\ifl@dpairing\else\endgroup\fi}

```

`\pausenumbering` The `\pausenumbering` macro is just the same as `\endnumbering`, but with the `\resumenumbering` `\ifnumbering` flag set to `true`, to show that numbering continues across the gap.²²

```

287 \newcommand{\pausenumbering}{%
288   \endnumbering\global\numberingtrue}

```

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

```

289 \newcommand*{\resumenumbering}{%
290   \ifnumbering
291     \global\pst@rtedLtrue
292     \global\advance\section@num \@ne
293     \led@mess@SectionContinued{\the\section@num}%
294     \line@list@stuff{\jobname.\extensionchars\the\section@num}%
295     \l@dend@stuff
296   \else
297     \led@err@NumberingShouldHaveStarted
298     \endnumbering
299     \beginnumbering
300   \fi}
301

```

21 Line counting

21.1 Choosing the system of lineation

Sometimes you want line numbers that start at 1 at the top of each page; sometimes you want line numbers that start at 1 at each `\pstart`; other times you want line numbers that start at 1 at the start of each section and increase regardless of page

²²Our thanks to Wayne Sullivan, who suggested the idea behind these macros.

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

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

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

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

```

302 \newif\ifbypage@
303 \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.
304 \newcommand*{\lineation}[1]{%
305   \ifnumbering
306     \led@err@LineationInNumbered
307   \else
308     \def\@tempa{#1}\def\@tempb{page}%
309     \ifx\@tempa\@tempb
310       \global\bypage@true
311       \global\bypstart@false
312       \pstartinfootnote[] [false]
313     \else
314       \def\@tempb{pstart}%
315       \ifx\@tempa\@tempb
316         \global\bypage@false
317         \global\bypstart@true
318         \pstartinfootnote
319       \else
320         \def\@tempb{section}
321         \ifx\@tempa\@tempb
322           \global\bypage@false
323           \global\bypstart@false
324           \pstartinfootnote[] [false]
325         \else
326           \led@warn@BadLineation
327         \fi
328       \fi
329     \fi
330   \fi}}

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

book.) You can change this within a numbered section, but the change may not take effect just when you'd like; if it's done between paragraphs nothing surprising should happen.

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

```

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

`\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.

```

361 \newcounter{firstlinenum}
362 \setcounter{firstlinenum}{5}
363 \newcounter{linenumincrement}
364 \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.

```

365 \newcounter{firstsublinenum}
366 \setcounter{firstsublinenum}{5}
367 \newcounter{sublinenumincrement}
368 \setcounter{sublinenumincrement}{5}
369

```

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

```

\linenumincrement 370 \newcommand*{\firstlinenum}[1]{\setcounter{firstlinenum}{#1}}
\firstsublinenum 371 \newcommand*{\linenumincrement}[1]{\setcounter{linenumincrement}{#1}}
\sublinenumincrement 372 \newcommand*{\firstsublinenum}[1]{\setcounter{firstsublinenum}{#1}}
373 \newcommand*{\sublinenumincrement}[1]{\setcounter{sublinenumincrement}{#1}}
374

```

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

`\lock@disp` encodes the selection: 0 for first, 1 for last, 2 for all.

```

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

```

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

```

401 \newcount\sublock@disp
402 \newcommand{\sublockdisp}[1]{%

```

```

403 \l@edgetlock@disp{#1}%
404 \ifnum\@l@dttempcntb>\m@ne
405   \global\sublock@disp=\@l@dttempcntb
406 \else
407   \led@warn@BadSublockdisp
408 \fi}}
409

```

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

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

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

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

`\sublinenumr@p`

```

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

```

Initialise the number styles to arabic.

```

414 \linenumberstyle{arabic}
415 \let\linenumr@p\linenumrep
416 \sublinenumberstyle{arabic}
417 \let\sublinenumr@p\sublinenumrep
418

```

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

`\rightlinenum`

`\linenumsep` They're made easy to access and change, since you may often want to change the styling in some way. These standard versions illustrate the general sort of thing that will be needed; they're based on the `\leftheadline` macro in *The TeXbook*, p. 416.

`\numlabfont`

`\ledlinenum`

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

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

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

```

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

```

```

424 \ifsublines@
425   \ifnum\subline@num>0\relax
426     \unskip\fullstop\sublinenumrep{\subline@num}%
427   \fi
428 \fi}
429 \newcommand*{\leftlinenum}{%
430   \ledlinenum
431   \kern\linenumsep}
432 \newcommand*{\rightlinenum}{%
433   \kern\linenumsep
434   \ledlinenum}
435

```

21.2 List macros

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

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

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

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

```

436 \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`.

```

437 \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. It creates global control sequences, like `\xdef`, and uses two temporary token-list registers, `\@toksa` and `\@toksb`.

```

438 \newtoks\@toksa \newtoks\@toksb
439 \global\@toksa={\}
440 \long\def\xright@appenditem#1\to#2{%
441   \global\@toksb=\expandafter{#2}%
442   \xdef#2{\the\@toksb\the\@toksa\expandafter{#1}}%
443   \global\@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`.

```

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

```



```

445 \global\@toksb=\expandafter{#2}%
446 \xdef#2{\the\@toksa\expandafter{#1}\the\@toksb}%
447 \global\@toksb={}}

```

\glp The **\glp** macro removes the leftmost item from a list and places it in a control sequence. You say **\glp\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 **\glp** are all global.

```

448 \def\glp#1\to#2{\expandafter\glpoff#1\glpoff#1#2}
449 \long\def\glpoff\#1#2\glpoff#3#4{\gdef#4{#1}\gdef#3{#2}}
450

```

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**.

```

451 \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.

```

452 \newcount\subline@num

```

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

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

of English Renaissance verse drama, in which stage directions are given sub-line numbers: a single line of verse may be interrupted by several stage directions.

453 `\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.

454 `\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.

455 `\newcount\@lock`

456 `\newcount\sub@lock`

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Here are the commands to create these lists:

```

457     \list@create{\line@list}
458     \list@create{\insertlines@list}
459     \list@create{\actionlines@list}
460     \list@create{\actions@list}
461
    \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
462 \newcount\page@num
463 \newcount\endpage@num
464 \newcount\endline@num
465 \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.
466 \newif\ifnoteschanged@

```

`\resetprevline@` Inside the apparatus, at each note, the line number is memorized 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@
467 \newcommand*{\resetprevline@}{%
468   \def\do##1{\global\csundef{prevline##1}}%
469   \dolistloop{\@series}%
470 }
```

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.

```
471 \newread\@inputcheck
472 \newcommand*{\read@linelist}[1]{%
473   \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 LaTeX context. We ignore carriage returns, since if we're in horizontal mode they can get interpreted as spaces to be printed.

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

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

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

```
474   \get@linelistfile{#1}%
475   \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.

```
476   \global\page@num=\m@ne
477   \ifx\actionlines@list\empty
478     \gdef\next@actionline{1000000}%
479   \else
480     \gl@p\actionlines@list\to\next@actionline
481     \gl@p\actions@list\to\next@action
```

```

482 \fi}
483
\list@clearing@reg  Clears the lists for \read@linelist
484 \newcommand*{\list@clearing@reg}{%
485   \list@clear{\line@list}%
486   \list@clear{\insertlines@list}%
487   \list@clear{\actionlines@list}%
488   \list@clear{\actions@list}}

\get@linelistfile  eledmac can take advantage of the LaTeX ‘safe file input’ macros to get the line-list
file.
489 \newcommand*{\get@linelistfile}[1]{%
490   \InputIfFileExists{#1}{%
491     \global\noteschanged@false
492     \begingroup
493       \catcode'\[=1 \catcode'\]=2
494       \makeatletter \catcode'\^M=9}{%
495     \led@warn@NoLineFile{#1}%
496     \global\noteschanged@true
497     \begingroup}%
498 }
499

```

This version of `\read@linelist` creates list macros containing data for the entire section, so they could get rather large. It would be no more difficult to read the line-list file incrementally rather than all at once: we could read, at the start of each paragraph, only the commands relating to that paragraph. But this would require that we have two line-lists open at once, one for reading, one for writing, and on systems without version numbers we’d have to do some file renaming outside of LaTeX for that to work. We’ve retained this slower approach to avoid that sort of hacking about, but have provided the `\pausenumbering` and `\resumenumbering` macros to help you if you run into macro memory limitations (see p. 11 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, `\@l`, 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`.

`\@l` `\@l` does everything related to the start of a new line of numbered text.
`\@l@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:

`\@l{\<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.

```
500 \newcommand{\@l}[2]{%
501   \fix@page{#1}%
502   \@l@reg}
503 \newcommand*{\@l@reg}{%
504   \ifx\l@dchset@num\relax \else
505     \advance\absline@num \@ne
506     \set@line@action
507     \let\l@dchset@num=\relax
508     \advance\absline@num \m@ne
509     \advance\line@num \m@ne
510   \fi
```

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

```
511   \advance\absline@num \@ne
512     \ifx\next@page@num\relax \else
513       \page@action
514       \let\next@page@num=\relax
515     \fi
516     \ifx\sub@change\relax \else
517       \ifnum\sub@change>\z@
518         \sublines@true
519       \else
520         \sublines@false
521       \fi
522       \sub@action
523       \let\sub@change=\relax
524     \fi
```

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

```
525     \ifcase\@lock
526       \or
```

```

527         \@lock \tw@
528     \or \or
529         \@lock \z@
530 \fi
531 \ifcase\sub@lock
532     \or
533         \sub@lock \tw@
534     \or \or
535         \sub@lock \z@
536 \fi

```

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

```

537 \ifsublines@
538     \ifnum\sub@lock<\tw@
539         \advance\subline@num \@ne
540     \fi
541 \else
542     \ifnum\@lock<\tw@
543         \advance\line@num \@ne \subline@num \z@
544     \fi
545 \fi}
546

```

`\@page` `\@page{<num>}` marks the start of a new output page; its argument is the number of that page.

First we reset the visible line numbers, if we're numbering by page, and store the page number itself in a count.

```

547 \newcommand*{\@page}[1]{%
548     \ifbypage@
549         \line@num \z@ \subline@num \z@
550     \fi
551     \page@num=#1\relax

```

And we set a flag that tells `\@1` that a new page number is to be set, because other associated actions shouldn't occur until the next line-start occurs.

```

552 \def\next@page@num{#1}}
553

```

`\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 `\@1`.

```

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

```



```

564 \def\next@page@num{#1}%
565 \listcsxadd{normal@page@break}{\the\absline@num}
566 \fi}
567

```

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

\@lopL

\@lopR

```

568 \newcommand*{\@pend}[1]{ }
569 \newcommand*{\@pendR}[1]{ }
570 \newcommand*{\@lopL}[1]{ }
571 \newcommand*{\@lopR}[1]{ }
572

```

\sub@on The `\sub@on` and `\sub@off` macros turn sub-lineation on and off: but not directly, since such changes don't really take effect until the next line of text. Instead they set a flag that notifies `\@l` of the necessary action.

\sub@off

```

573 \newcommand*{\sub@on}{\ifsublines@
574   \let\sub@change=\relax
575   \else
576     \def\sub@change{1}%
577   \fi}
578 \newcommand*{\sub@off}{\ifsublines@
579   \def\sub@change{-1}%
580   \else
581     \let\sub@change=\relax
582   \fi}
583

```

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

```

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

```

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

```

599 \newcommand*{\@set}[1]{\ifsublines@
600     \subline@num=#1\relax
601     \else
602     \line@num=#1\relax
603     \fi
604     \set@line@action}
605

```

`\l@d@set` The `\l@d@set{<num>}` macro sets the line number for the next `\pstart...` to the value specified as its argument. This is used to implement `\setlinenum`.
`\l@dchset@num` is a flag to the `\@l` macro. If it is not `\relax` then a linenumber change is to be done.

```

606 \newcommand*{\l@d@set}[1]{%
607     \line@num=#1\relax
608     \advance\line@num \@ne
609     \def\l@dchset@num{#1}}
610 \let\l@dchset@num\relax
611

```

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

```

612 \newcommand*{\page@action}{%
613     \xright@appenditem{\the\absline@num}\to\actionlines@list
614     \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.

```

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

```

624 \newcommand*{\sub@action}{%
625     \xright@appenditem{\the\absline@num}\to\actionlines@list
626     \ifsublines@
627         \xright@appenditem{-1001}\to\actions@list
628     \else
629         \xright@appenditem{-1002}\to\actions@list
630     \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.

```

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

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

```

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

```

```

674      \@lock \z@
675      \fi
676    \fi}
677 \newcommand*{\do@lockoff}{\do@lockoffL}
678 \newcommand*{\skip@lockoff}{\global\let\lock@off=\do@lockoff}
679 \global\let\lock@off=\do@lockoff
680

```

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

```

681 \newcommand*{\n@num}{\n@num@reg}
682 \newcommand*{\n@num@reg}{%
683   \xright@appenditem{\the\absline@num}\to\actionlines@list
684   \xright@appenditem{-1007}\to\actions@list}
685

```

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

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

```

686   \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.

```

687 \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.

```

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

```

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

```

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

```

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

```

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

```

712 #2}
713

```

21.6 Writing to the line-list file

We've now defined all the counters, lists, and commands involved in reading the line-list file at the start of a section. Now we'll cover the commands that `eledmac` uses within the text of a section to write commands out to the line-list.

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

```

714 \newwrite\linenum@out

```

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

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

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

```

715 \newif\iffirst@linenum@out@
716 \first@linenum@out@true

```

`\line@list@stuff` The `\line@list@stuff{<file>}` macro, which is called by `\beginnumbering`, performs all the line-list operations needed at the start of a section. Its argument is the name of the line-list file.

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

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

```
718 \read@linelist{#1}%
```

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

```
719 \iffirst@linenum@out@
720 \immediate\closeout\linenum@out%
721 \global\first@linenum@out@false%
722 \immediate\openout\linenum@out=#1\relax%
723 \else
```

If we get here, then this is not the first line-list we've seen, so we don't open or close the files immediately, except if we are in a minipage and this minipage is not a ledgroup.

```
724 \if@minipage%
725 \if@ledgroup%
726 \closeout\linenum@out%
727 \openout\linenum@out=#1\relax%
728 \else%
729 \immediate\closeout\linenum@out%
730 \immediate\openout\linenum@out=#1\relax%
731 \fi
732 \else%
733 \closeout\linenum@out%
734 \openout\linenum@out=#1\relax%
735 \fi%
736 \fi}
737
```

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

```
738 \newcommand*{\new@line}{%
739 \IfStrEq{\led@pb@setting}{after}%
740 {\xifinlistcs{\the\absline@num}{l@prev@nopb}%
741 {\xifinlistcs{\the\absline@num}{normal@page@break}%
742 {\numgdef{\@next@page}{\thepage+1}%
743 \write\linenum@out{\string\@l[\@next@page][\@next@page]}}%
744 }%
745 {\write\linenum@out{\string\@l[\the\c@page][\thepage]}}}%
746 }%
747 {\write\linenum@out{\string\@l[\the\c@page][\thepage]}}}%
748 {}%
749 \IfStrEq{\led@pb@setting}{before}%
```

```

750   {%
751   \numdef{\next@absline}{\the\absline@num+1}
752   \xifinlistcs{\next@absline}{l@prev@nopb}%
753   {\xifinlistcs{\the\absline@num}{normal@page@break}%
754   {%
755     \numgdef{\nc@page}{\c@page+1}%
756     \write\linenum@out{\string\@l[\nc@page][\nc@page]}%
757   }%
758   {\write\linenum@out{\string\@l[\the\c@page][\thepage]}}%
759   }%
760   {\write\linenum@out{\string\@l[\the\c@page][\thepage]}}%
761 }%
762 {}%
763 \IfStrEqCase{\led@pb@setting}{\{before\}{\relax}{\after\}{\relax}}{\write\linenum@out{\string\@l[\the\c@page][\thepage]}}%
764 }
765

```

\flag@start We enclose a lemma marked by **\edtext** in **\flag@start** and **\flag@end**: these
\flag@end 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.

```

766 \newcommand*{\flag@start}{%
767   \edef\next{\write\linenum@out{%
768     \string\@ref[\the\insert@count] []}%
769   \next}
770 \newcommand*{\flag@end}{\write\linenum@out{[]}}

```

\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.

```

771 \newcommand*{\page@start}{%
772

```

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

We tinker with **\lastskip** because a command of either sort really needs to be attached to the last word preceding the change, not the first word that follows the change. This is because sub-lineation will often turn on and off in mid-line—stage

directions, for example, often are mixed with dialogue in that way—and when a line is mixed we want to label it using the system that was in effect at its start. But when sub-lineation begins at the very start of a line we have a problem, if we don't put in this code.

```

773 \newcommand*{\startsub}{\dimen0\lastskip
774 \ifdim\dimen0>0pt \unskip \fi
775 \write\linenum@out{\string\sub@on}%
776 \ifdim\dimen0>0pt \hskip\dimen0 \fi}
777 \def\endsub{\dimen0\lastskip
778 \ifdim\dimen0>0pt \unskip \fi
779 \write\linenum@out{\string\sub@off}%
780 \ifdim\dimen0>0pt \hskip\dimen0 \fi}
781

```

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

```

782 \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.

```

783 \newcommand*{\setline}[1]{%
784 \ifnum#1<\z@
785 \led@warn@BadSetline
786 \else
787 \write\linenum@out{\string\@set[#1]}%
788 \fi}
789

```

`\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.

```

790 \newcommand*{\setlinenum}[1]{%
791 \ifnum#1<\z@
792 \led@warn@BadSetlinenum
793 \else
794 \write\linenum@out{\string\l@d@set[#1]}%
795 \fi}
796

```

`\startlock` You can use `\startlock` or `\endlock` in running text to start or end line number locking at the current line. They decide whether line numbers or sub-line numbers are affected, depending on the current state of the sub-lineation flags.

```

797 \newcommand*{\startlock}{\write\linenum@out{\string\lock@on}}
798 \def\endlock{\write\linenum@out{\string\lock@off}}
799

```

`\ifl@dskipnumber` In numbered text `\skipnumbering` will suspend the numbering for that particular line.
`\l@dskipnumbertrue`
`\l@dskipnumberfalse` 800 `\newif\ifl@dskipnumber`
`\skipnumbering`
`\skipnumbering@reg`


```

801 \l@dskipnumberfalse
802 \newcommand*{\skipnumbering}{\skipnumbering@reg}
803 \newcommand*{\skipnumbering@reg}{%
804   \write\linenum@out{\string\n@num}%
805   \advanceline{-1}}
806

```

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.84). 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.

An example where some 'memory' of line numbers might be required is where there are several variant readings per line of text, and you do not wish the line number to be repeated for each lemma in the notes. After the first occurrence of the line number, you might want the symbol '||' instead of further occurrences, for instance. This can easily be done by a macro like `\printlines`, if it saves the last value of `\l@d@nums` that it saw, and then performs a simple conditional test to see whether to print a number or a '||'.

22.1 `\edtext` and `\critext` themselves

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

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

`\end@lemmas` is intended for the few things that need to be associated with the end of the lemma, like footnote numbers. Such numbers are not implemented

in the current version, and indeed no use is currently made of `\end@lemmas` or of the `\aftergroup` trick. The general approach would be to define a macro to be used within the second argument of `\critext` that would add the appropriate command to `\end@lemmas`.

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

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

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

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

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

`\dummy@edtext` LaTeX users are not used to delimited arguments, so I provide a `\edtext` macro as well.

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

We’re going to need another macro that takes one argument and ignores it entirely. This is supplied by the LaTeX `\@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.²³ This is done because we want to pass into our notes the generic commands to change to roman or whatever, and not their expansions that will ask for a particular style at a specified size. The notes may well be in a smaller font, so the command should be expanded later, when the note’s environment is in effect.

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

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

²³Since ‘control sequences equivalent to characters are not expandable’—*The TeXbook*, answer to Exercise 20.14.

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

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

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

```

810 \newcommand*{\no@expands}{%
811   \let\select@lemmafnt=0%
812   \let\startsub=\relax \let\endsub=\relax
813   \let\startlock=\relax \let\endlock=\relax
814   \let\edlabel=\@gobble
815   \let\setline=\@gobble \let\advanceline=\@gobble
816   \let\critext=\dummy@text
817   \let\edtext=\dummy@edtext
818   \l@dtabnoexpands
819   \morenoexpands}
820 \let\morenoexpands=\relax
821
```

`\@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.

```

822 \newcommand{\@tag}{}
823 % \end{macrocode}
824 % \end{macro}
825 % \begin{macro}{\critext}
826 % Now we begin \cs{critext} itself. The definition requires a \verb"/" after
827 % the arguments: this eliminates the possibility of problems about
828 % knowing where \verb"#2" ends. This also changes the handling of spaces
829 % following an invocation of the macro: normally such spaces are
830 % skipped, but in this case they're significant because \verb"#2" is
831 % a ‘delimited parameter’. Since \cs{critext} is always used in running
832 % text, it seems more appropriate to pay attention to spaces than to
833 % skip them.
834 %
```

```

835 % When executed, \cs{critext} first ensures that we're in
836 % horizontal mode.
837 %   \begin{macrocode}
838 \long\def\critext#1#2/{\leavevmode

```

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

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

```

839 \begingroup
840   \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`.

```

841   \set@line

```

`\insert@count` will be altered by the note-generating macros: it counts the number of deferred footnotes or other insertions generated by this instance of `\critext`.

```

842   \global\insert@count=0

```

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.

```

843   \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.

```

844   \@ifundefined{xpg@main@language}{%if not polyglossia
845     \flag@start}%
846     {\if@RTL\flag@end\else\flag@start\fi% With polyglossia, you must track whether the language re
847   }
848   \endgroup
849   \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.

```

850   \ifx\end@lemmas\empty \else
851     \glp\end@lemmas\to\x@lemma

```

```

852 \x@lemma
853 \global\let\x@lemma=\relax
854 \fi
855 \@ifundefined{xpg@main@language}{%if not polyglossia
856     \flag@end}%
857     {\ifRTL\flag@start\else\flag@end\fi% With polyglossia, you must track whether the
858     }
859 }

```

`\edtext`

```

860 \newcommand{\edtext}[2]{\leavevmode
861 \begingroup
862 \global\renewcommand{\@tag}{\no@expands #1}%
863 \set@line
864 \global\insert@count=0
865 \ignorespaces #2\relax
866 \@ifundefined{xpg@main@language}{%if not polyglossia
867     \flag@start}%
868     {\ifRTL\flag@end\else\flag@start\fi% With polyglossia, you must track whether the
869     }%
870 \endgroup
871 \showlemma{#1}%
872 \ifx\end@lemmas\empty \else
873 \glp\end@lemmas\to\x@lemma
874 \x@lemma
875 \global\let\x@lemma=\relax
876 \fi
877 \@ifundefined{xpg@main@language}{%if not polyglossia
878     \flag@end}%
879     {\ifRTL\flag@start\else\flag@end\fi% With polyglossia, you must track whether the
880     }%
881 }
882

```

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

```

883 \newif\ifnumberline
884 \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.

```

885 \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.

```

886 \ifx\line@list\empty
887   \global\noteschanged>true
888   \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
889 \else
890   \gl@p\line@list\to\@tempb
891   \xdef\l@d@nums{\@tempb|\edfont@info}%
892   \global\let\@tempb=\undefined
893 \fi}
894

```

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

```

895 \newcommand*{\edfont@info}{\f@encoding/\f@family/\f@series/\f@shape}
896

```

22.2 Substitute lemma

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

```

897 \newcommand*{\lemma}[1]{\global\renewcommand{\@tag}{\no@expands #1}}

```

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.58): 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.

```

898 \newcommand*{\linenum}[1]{%
899   \xdef\@tempa{#1|||||\\noexpand\\l@d@nums}%
900   \global\let\l@d@nums=\empty
901   \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.

```

902 \def\line@set#1|#2|#3|#4|{%
903   \gdef\@tempb{#1}%
904   \ifx\@tempb\empty
905     \l@d@add{#3}%
906   \else

```

```

907      \l@d@add{#1}%
908    \fi
909    \gdef\@tempb{#4}%
910    \ifx\@tempb\empty\else
911      \l@d@add{|\line@set#2\|#4\}%
912    \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`.

```

913 \newcommand{\l@d@add}[1]{\xdef\l@d@nums{\l@d@nums#1}}
914

```

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`

<pre> \raw@text \ifnumberedpar@ \numberedpar@true \numberedpar@false \num@lines \one@line \par@line </pre>	<p>Here are numbers and flags that are used internally in the course of the paragraph decomposition.</p> <p>When we first form the paragraph, it goes into a box register, <code>\raw@text</code>, instead of onto the current vertical list. The <code>\ifnumberedpar@</code> flag will be <code>true</code> while a paragraph is being processed in that way. <code>\num@lines</code> will store the number of lines in the paragraph when it's complete. When we chop it up into lines, each line in turn goes into the <code>\one@line</code> register, and <code>\par@line</code> will be the number of that line within the paragraph.</p> <pre> 915 \newbox\raw@text 916 \newif\ifnumberedpar@ 917 \newcount\num@lines 918 \newbox\one@line 919 \newcount\par@line </pre>
<pre> \pstart \numberpstarttrue \numberpstartfalse \labelpstarttrue \labelpstartfalse thepstart </pre>	<p><code>\pstart</code> starts the paragraph by clearing the <code>\inserts@list</code> list and other relevant variables, and then arranges for the subsequent text to go into the <code>\raw@text</code> box. <code>\pstart</code> needs to appear at the start of every paragraph that's to be numbered; the <code>\autopar</code> command below may be used to insert these commands automatically.</p> <p>Beware: everything that occurs between <code>\pstart</code> and <code>\pend</code> is happening within a group; definitions must be global if you want them to survive past the end of the paragraph.</p>

You can use the command `\numberpstarttrue` to insert a number on every `\pstart`. To stop the numbering, you must use `\numberpstartfalse`. To reset the numebering of `\pstarts`, insert

```
\setcounter{pstart}{0}
```

```

920
921 \newcounter{pstart}
922 \renewcommand{\thepstart}{\bfseries\@arabic\c@pstart}. }
923 \newif\ifnumberpstart
924 \numberpstartfalse
925 \newif\iflabelpstart
926 \labelpstartfalse
927 \newcommand*{\pstart}{%
928 \if@nobreak%
929   \let\@oldnobreak\@nobreaktrue%
930 \else%
931   \let\@oldnobreak\@nobreakfalse%
932 \fi%
933 \@nobreaktrue%
934 \ifnumbering \else%
935   \led@err@PstartNotNumbered%
936   \beginnumbering%
937   \fi%
938   \ifnumberedpar@%
939     \led@err@PstartInPstart%
940     \pend%
941   \fi%
942   \list@clear{\inserts@list}%
943   \global\let\next@insert=\empty%
944   \begingroup\normal@pars%
945   \global\setbox\raw@text=\vbox\bgroup%
946     \ifautopar\else%
947       \ifnumberpstart%
948         \ifinstanza\else%
949           \ifsidepstartnum\else%
950             \thepstart%
951             \fi%
952           \fi%
953         \fi%
954       \fi%
955     \numberedpar@true%
956     \iflabelpstart\protected@edef\@currentlabel%
957       {\p@pstart\thepstart}\fi%
958   }
```

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

```

959 \newcommand*{\pend}{\ifnumbering \else%
960   \led@err@PendNotNumbered%
```

```

961 \fi%
962 \ifnumberedpar@ \else%
963   \led@err@PendNoPstart%
964 \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.

```

965 \l@dzeropenalties%
966 \endgraf\global\num@lines=\prevgraf\egroup%
967 \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`.

```

968 \csnumdef{pstartline}{0}%
969 \loop\ifvbox\raw@text%
970   \csnumdef{pstartline}{\pstartline+1}%
971   \do@line%
972   \ifbypstart%
973     \ifnumequal{pstartline}{1}{\setline{1}\resetprevline@}{}%
974   \fi%
975 \repeat%

```

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

```

976 \flush@notes%
977 \endgroup%
978 \ignorespaces%
979 \ifnumberpstart%
980 \pstartnumtrue%
981 \fi%
982 \@oldnobreak%
983 \addtocounter{pstart}{1}}
984

```

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

```

985 \newcommand*{\l@dzeropenalties}{%
986   \brokenpenalty \z@ \clubpenalty \z@
987   \displaywidowpenalty \z@ \interlinepenalty \z@ \predisplaypenalty \z@
988   \postdisplaypenalty \z@ \widowpenalty \z@}
989

```

`\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.

```

990 \newif\ifautopar
991 \autoparfalse
992 \newcommand*{\autopar}{
993   \ifledRcol
994     \ifnumberingR \else
995       \led@err@AutoparNotNumbered
996       \beginnumberingR
997       \fi
998     \else
999       \ifnumbering \else
1000        \led@err@AutoparNotNumbered
1001        \beginnumbering
1002        \fi
1003      \fi
1004      \autopartrue
1005      \everypar={\setbox0=\lastbox
1006        \endgraf \vskip-\parskip
1007        \pstart \noindent \kern\wd0 \ifnumberpstart\ifinstanza\else\thepstart\fi\fi
1008        \let\par=\pend}%
1009      \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.

```

1010 \newcommand*{\normal@pars}{\everypar={}\let\par\endgraf}
1011

```

23.2 Processing one line

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

```

1012 \newcommand*{\l@dunhbox@line}[1]{\unhbox #1}
1013 \newcommand*{\do@line}{%
1014   {\vbadness=10000
1015    \splittopskip=\z@
1016    \do@linehook
1017 \l@emptyd@ta
1018   \global\setbox\one@line=\vsplit\raw@text to\baselineskip}%
1019   \unvbox\one@line \global\setbox\one@line=\lastbox
1020   \getline@num
1021   \IfStrEq{\led@pb@setting}{before}{\led@check@pb\led@check@nopb}{%
1022     \ifnum\@lock>\@ne
1023       \inserthangingsymboltrue
1024     \else
1025       \inserthangingsymbolfalse
1026     \fi
1027     \check@pb@in@verse
1028     \affixline@num
1029     \affixpstart@num
1030     \hb@xt@ \linewidth{\do@insidelinehook\l@dld@ta\add@inserts\affixside@note
1031       \l@dlsn@te
1032       {\ledllfill\hb@xt@ \wd\one@line{\new@line\inserthangingsymbol\l@dunhbox@line{\one@line
1033         \l@drsn@te
1034       }}}
1035     \IfStrEq{\led@pb@setting}{after}{\led@check@pb\led@check@nopb}{%
1036     }%

```

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

```

1037 \newcommand*{\do@linehook}{%
1038 \newcommand*{\do@insidelinehook}{%

```

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

```

\l@dld@ta \l@dld@ta 1039 \newcommand*{\l@emptyd@ta}{%
\l@dcsnotetext 1040 \gdef\l@dld@ta{}%
1041 \gdef\l@dld@ta{}%
1042 \gdef\l@dcsnotetext{}%
1043

```

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

```

\l@drsn@te 1044 \newcommand*{\l@dlsn@te}{%
1045   \hb@xt@ \z@{\hss\box\l@dldp@rbox\kern\ledlsnotesep}}
1046 \newcommand*{\l@drsn@te}{%
1047   \hb@xt@ \z@{\kern\ledrsnotesep\box\l@drp@rbox\hss}}
1048

```

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

```
1049 \newcommand*{\ledllfill}{\hfil}
1050 \newcommand*{\ledrlfill}{\hfil}
1051
```

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.

```
1052 \newcommand*{\getline@num}{%
1053   \global\advance\absline@num \@ne
1054   \do@actions
1055   \do@ballast
1056   \ifnumberline
1057   \ifsublines@
1058     \ifnum\sub@lock<\tw@
1059       \global\advance\subline@num \@ne
1060     \fi
1061   \else
1062     \ifnum\@lock<\tw@
1063       \global\advance\line@num \@ne
1064       \global\subline@num \z@
1065     \fi
1066   \fi
1067   \fi
1068 }
```

`\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.94).

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

```
1069 \newcount\ballast@count
1070 \newcounter{ballast}
1071 \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.

```
1072 \newcommand*{\do@ballast}{\global\ballast@count \z@
1073   \begingroup
1074     \advance\absline@num \@ne
```

```

1075     \ifnum\next@actionline=\absline@num
1076         \ifnum\next@action>-1001\relax
1077             \global\advance\ballast@count by -\c@ballast
1078         \fi
1079     \fi
1080 \endgroup}

```

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

`\do@actions@next` It may call itself recursively, and to do this efficiently (using T_EX'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`.

```

1081 \newcommand*{\do@actions}{%
1082     \global\let\do@actions@next=\relax
1083     \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.

```

1084     \ifnum\next@action>-1001
1085         \global\page@num=\next@action
1086         \ifbypage@
1087             \global\line@num=\z@ \global\subline@num=\z@
1088             \resetprevline@
1089         \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`.)

```

1090     \else
1091         \ifnum\next@action<-4999
1092             \@l@dttempcnta=-\next@action
1093             \advance\@l@dttempcnta by -5001
1094             \ifsublines@
1095                 \global\subline@num=\@l@dttempcnta
1096             \else
1097                 \global\line@num=\@l@dttempcnta
1098             \fi

```

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

```

1099     \else
1100         \@l@dttempcnta=-\next@action
1101         \advance\@l@dttempcnta by -1000
1102         \do@actions@fixedcode
1103     \fi
1104 \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.

```

1105   \ifx\actionlines@list\empty
1106       \gdef\next@actionline{1000000}%
1107   \else
1108       \gl@p\actionlines@list\to\next@actionline
1109       \gl@p\actions@list\to\next@action
1110       \global\let\do@actions@next=\do@actions
1111   \fi
1112 \fi

```

Make the recursive call, if necessary.

```

1113 \do@actions@next}
1114

```

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

```

1115 \newcommand*{\do@actions@fixedcode}{%
1116   \ifcase\@l@dttempcnta
1117   \or% % 1001
1118       \global\sublines@true
1119   \or% % 1002
1120       \global\sublines@false
1121   \or% % 1003
1122       \global\@lock=\@ne
1123   \or% % 1004
1124       \ifnum\@lock=\tw@
1125           \global\@lock=\thr@@
1126       \else
1127           \global\@lock=\z@
1128       \fi
1129   \or% % 1005
1130       \global\sub@lock=\@ne
1131   \or% % 1006
1132       \ifnum\sub@lock=\tw@
1133           \global\sub@lock=\thr@@
1134       \else
1135           \global\sub@lock=\z@
1136       \fi
1137   \or% % 1007
1138       \l@dskipnumbertrue
1139   \else
1140       \led@warn@BadAction
1141   \fi}
1142
1143

```

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 `\line@num ≤ \firstlinenum`, we compare the two directly instead of making these calculations.

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

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

```
1144 \newcommand*{\affixline@num}{%
```

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

```
1145 \ifnumberline
1146 \ifl@dskipnumber
1147   \global\l@dskipnumberfalse
1148 \else
1149   \ifsublines@
1150     \@l@tempcntb=\subline@num
1151     \ifnum\subline@num>\c@firstsublinenum
1152       \@l@tempcnta=\subline@num
1153       \advance\@l@tempcnta by-\c@firstsublinenum
1154       \divide\@l@tempcnta by\c@sublinenumincrement
1155       \multiply\@l@tempcnta by\c@sublinenumincrement
1156       \advance\@l@tempcnta by\c@firstsublinenum
1157     \else
1158       \@l@tempcnta=\c@firstsublinenum
1159     \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.

```
1160   \ch@cksubl@ck
```

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

```
1161 \else
```



```
1162 \l@ldtempcntb=\line@num
```

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

```
1163 \ifx\linenumberlist\empty
1164 \ifnum\line@num>\c@firstlinenum
1165 \l@ldtempcnta=\line@num
1166 \advance\l@ldtempcnta by-\c@firstlinenum
1167 \divide\l@ldtempcnta by\c@linenumincrement
1168 \multiply\l@ldtempcnta by\c@linenumincrement
1169 \advance\l@ldtempcnta by\c@firstlinenum
1170 \else
1171 \l@ldtempcnta=\c@firstlinenum
1172 \fi
1173 \else
```

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

```
1174 \l@ldtempcnta=\line@num
1175 \edef\rem@inder{\,\linenumberlist,\number\line@num,}%
1176 \edef\sc@n@list{\def\noexpand\sc@n@list
1177   ###1,\number\l@ldtempcnta,###2|{\def\noexpand\rem@inder{###2}}}%
1178 \sc@n@list\expandafter\sc@n@list\rem@inder|}%
1179 \ifx\rem@inder\empty\advance\l@ldtempcnta\@ne\fi
1180 \fi
```

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

```
1181 \ch@ck@l@ck
1182 \fi
```

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

```
1183 \ifnum\l@ldtempcnta=\l@ldtempcntb
```

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

For LaTeX we have to consider two column documents as well. In this case I think we need to put the numbers at the outside of the column — the left of the first column and the right of the second. Do the 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 1184 \if@twocolumn
1185 \if@firstcolumn
1186 \gdef\l@dld@ta{\llap{\leftlinenum}}}%
1187 \else
1188 \gdef\l@drd@ta{\rlap{\rightlinenum}}}%
1189 \fi
```

1190 `\else`

Continuing the original code ...

```
1191 \l@dttempcntb=\line@margin
1192 \ifnum\l@dttempcntb>\@ne
1193 \advance\l@dttempcntb \page@num
1194 \fi
```

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

```
1195 \ifodd\l@dttempcntb
1196 \gdef\l@drd@ta{\rlap{{\rightlinenum}}}%
1197 \else
1198 \gdef\l@dld@ta{\llap{{\leftlinenum}}}%
1199 \fi
1200 \fi
1201 \else
```

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

```
1202 %% #1%
1203 \fi
```

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

```
1204 \f@x@l@cks
1205 \fi
1206 \fi
1207 }
1208
```

`\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.

```
1209 \newcommand*{\ch@cksub@l@ck}{%
1210 \ifcase\sub@lock
1211 \or
1212 \ifnum\sublock@disp=\@ne
1213 \l@dttempcntb=\z@ \l@dttempcnta=\@ne
1214 \fi
1215 \or
1216 \ifnum\sublock@disp=\tw@ \else
1217 \l@dttempcntb=\z@ \l@dttempcnta=\@ne
1218 \fi
1219 \or
1220 \ifnum\sublock@disp=\z@
1221 \l@dttempcntb=\z@ \l@dttempcnta=\@ne
1222 \fi
1223 \fi}
```

Similarly for line numbers.

```
1224 \newcommand*{\ch@ck@l@ck}{%
1225 \ifcase\@lock
```

```

1226     \or
1227     \ifnum\lock@disp=\@ne
1228         \@l@dttempcntb=\z@ \@l@dttempcnta=\@ne
1229     \fi
1230     \or
1231     \ifnum\lock@disp=\tw@ \else
1232         \@l@dttempcntb=\z@ \@l@dttempcnta=\@ne
1233     \fi
1234     \or
1235     \ifnum\lock@disp=\z@
1236         \@l@dttempcntb=\z@ \@l@dttempcnta=\@ne
1237     \fi
1238 \fi}

```

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

```

1239 \newcommand*{\fx@locks}{%
1240   \ifcase\@lock
1241   \or
1242     \global\@lock=\tw@
1243   \or \or
1244     \global\@lock=\z@
1245   \fi
1246   \ifcase\sub@lock
1247   \or
1248     \global\sub@lock=\tw@
1249   \or \or
1250     \global\sub@lock=\z@
1251   \fi}
1252

```

`\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.

```

1253 \newcommand{\pageparbreak}{\pend\newpage\pstart\noindent}
1254

```

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 1255
\ifsidepstartnum 1256 \newif\ifsidepstartnum
1257 \newcommand*{\affixpstart@num}{%
1258   \ifsidepstartnum
1259     \if@twocolumn
1260       \if@firstcolumn
1261         \gdef\l@dld@ta{\llap{\leftpstartnum}}}%
1262       \else
1263         \gdef\l@drd@ta{\rlap{\rightpstartnum}}}%
1264       \fi
1265     \else
1266       \l@dttempcntb=\line@margin
1267       \ifnum\l@dttempcntb>\@one
1268         \advance\l@dttempcntb \page@num
1269       \fi
1270       \ifodd\l@dttempcntb
1271         \gdef\l@drd@ta{\rlap{\rightpstartnum}}}%
1272       \else
1273         \gdef\l@dld@ta{\llap{\leftpstartnum}}}%
1274       \fi
1275     \fi
1276   \fi
1277 }
1278 }
1279 %
1280
1281 \newif\ifpstartnum
1282 \pstartnumtrue
1283 \newcommand*{\leftpstartnum}{
1284   \ifpstartnum\thepstart
1285   \kern\linenumsep\fi
1286   \global\pstartnumfalse
1287 }
1288 \newcommand*{\rightpstartnum}{
1289   \ifpstartnum
1290   \kern\linenumsep
1291   \thepstart
1292   \fi
1293   \global\pstartnumfalse
1294 }

```

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.

```
1295 \list@create{\inserts@list}
```

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

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

```
1296 \newcommand*{\add@inserts}{%
```

```
1297   \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.

```
1298   \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.

```
1299   \ifx\next@insert\empty
```

```
1300     \ifx\insertlines@list\empty
```

```
1301       \global\noteschanged@true
```

```
1302       \gdef\next@insert{100000}%
```

```
1303     \else
```

```
1304       \glp\insertlines@list\to\next@insert
```

```
1305     \fi
```

```
1306   \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.

```
1307   \ifnum\next@insert=\absline@num
```

```
1308     \glp\inserts@list\to\@insert
```

```
1309     \@insert
```

```
1310     \global\let\@insert=\undefined
```

```
1311     \global\let\next@insert=\empty
```

```
1312     \global\let\add@inserts@next=\add@inserts
```

```
1313   \fi
```

```
1314 \fi
```

Make the recursive call, if necessary.

```
1315 \add@inserts@next}
```

```
1316
```

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@tempcnta` 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.85). Finally, the penalty is checked to see that it doesn't go below -10000 .

```

1317 \newcommand*{\add@penalties}{\@l@tempcnta=\ballast@count
1318   \ifnum\num@lines>\@ne
1319     \global\advance\par@line \@ne
1320     \ifnum\par@line=\@ne
1321       \advance\@l@tempcnta \clubpenalty
1322     \fi
1323     \@l@tempcntb=\par@line \advance\@l@tempcntb \@ne
1324     \ifnum\@l@tempcntb=\num@lines
1325       \advance\@l@tempcnta \widowpenalty
1326     \fi
1327     \ifnum\par@line<\num@lines
1328       \advance\@l@tempcnta \interlinepenalty
1329     \fi
1330   \fi
1331   \ifnum\@l@tempcnta=\z@
1332     \relax
1333   \else
1334     \ifnum\@l@tempcnta>-10000
1335       \penalty\@l@tempcnta
1336     \else
1337       \penalty -10000
1338     \fi
1339   \fi}
1340
```

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.

```

1341 \newcommand*{\flush@notes}{%

```

```

1342 \@xloop
1343     \ifx\inserts@list\empty \else
1344         \gl@p\inserts@list\to\@insert
1345         \@insert
1346         \global\let\@insert=\undefined
1347     \repeat}
1348

```

\@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.

```

1349 \def\@xloop#1\repeat{%
1350     \def\body{#1\expandafter\body\fi}%
1351     \body}
1352

```

24 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.

```

1353 \def\select@lemmafont#1|#2|#3|#4|#5|#6|#7|{\select@@lemmafont#7|}
1354 \def\select@@lemmafont#1/#2/#3/#4|%
1355     {\fontencoding{#1}\fontfamily{#2}\fontseries{#3}\fontshape{#4}%
1356     \selectfont}
1357

```

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.

```

1358 \newcommandx*{\footnoteoptions@}[3][1=L,usedefault]{%
1359   \def\do##1{%
1360     \ifstrequal{##1}{L}{% In Leftside
1361       \xright@appenditem{\global\noexpand\settoggle{##1@}{#3}}\to\inserts@list% Switch
1362       \global\advance\insert@count \@ne% Increment the left insert counter.
1363     }%
1364     {%
1365       \xright@appenditem{\global\noexpand\settoggle{##1@}{#3}}\to\inserts@listR% Swit
1366       \global\advance\insert@countR \@ne% Increment the right insert counter insert.
1367     }%
1368   }%
1369   \notblank{#2}{\docsvlist{#2}}}% Parsing all options
1370 }
```

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

```

1371 \newcommandx*{\footnotelang@lua}[1][1=L,usedefault]{%
1372   \ifstrequal{#1}{L}{%
1373     \xright@appenditem{{\csxdef{footnote@luatextextdir}{\the\luatextextdir}}}\to\inserts@l
1374     \global\advance\insert@count \@ne%
1375     \xright@appenditem{{\csxdef{footnote@luatexpardir}{\the\luatexpardir}}}\to\inserts@l
1376     \global\advance\insert@count \@ne%
1377   }%
1378   {%
1379     \xright@appenditem{{\csxdef{footnote@luatextextdir}{\the\luatextextdir}}}\to\inserts@l
1380     \global\advance\insert@countR \@ne%
1381     \xright@appenditem{{\csxdef{footnote@luatexpardir}{\the\luatexpardir}}}\to\inserts@l
1382     \global\advance\insert@countR \@ne%
1383   }%
1384 }
```

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

```

1385 \newcommandx*{\footnotelang@poly}[1][1=L,usedefault]{%
1386   \ifstrequal{#1}{L}{%
1387     \if@RTL%
1388       \xright@appenditem{{\csxdef{footnote@dir}{@RTLtrue}}}\to\inserts@list%Know the lan
1389       \global\advance\insert@count \@ne%
1390     \else
1391       \xright@appenditem{{\csxdef{footnote@dir}{@RTLfalse}}}\to\inserts@list%Know the
1392       \global\advance\insert@count \@ne%
1393     \fi%
1394     \xright@appenditem{{\csxdef{footnote@lang}{\csexpandonce{language}}}\to\inserts@
1395     \global\advance\insert@count \@ne%
1396   }%

```



```

1397 {%
1398   \if@RTL
1399     \xright@appenditem{{\csxdef{footnote@dir}{@RTLtrue}}}\to\inserts@listR%Know the language of 1
1400     \global\advance\insert@countR \@ne%
1401   \else
1402     \xright@appenditem{{\csxdef{footnote@dir}{@RTLfalse}}}\to\inserts@listR%Know the language of
1403     \global\advance\insert@countR \@ne%
1404   \fi
1405   \xright@appenditem{{\csxdef{footnote@lang}{\csexpandonce{language name}}}\to\inserts@listR%Know t
1406   \global\advance\insert@countR \@ne%
1407 }%
1408 }

```

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.

```

1409 \notbool{parapparatus@}{\newcommand*}{\newcommand}{\normalvfootnote}[2]{%
1410   \insert\csname #1footins\endcsname\bgroup
1411   \csuse{bhookXnote@#1}
1412   \csuse{Xnotefontsize@#1}
1413   \footplitskips
1414   \spaceskip=\z@skip \xspaceskip=\z@skip
1415   \csname #1footfmt\endcsname #2[#1]\egroup}

```

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

```

1416 \newcommand*{\footplitskips}{%
1417   \interlinepenalty=\interfootnotelinepenalty
1418   \floatingpenalty=\@MM
1419   \splittopskip=\ht\strutbox \splitmaxdepth=\dp\strutbox

```

```

1420 \leftskip=\z@skip \rightskip=\z@skip}
1421

```

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

```

1422 \notbool{parapparatus@}{\newcommand*}{\newcommand}{\mpnormalvfootnote}[2]{%
1423   \global\setbox\@nameuse{mp#1footins}\vbox{%
1424     \unvbox\@nameuse{mp#1footins}
1425     \csuse{bhookXnote@#1}
1426     \csuse{Xnotefontsize@#1}
1427     \hsize\columnwidth
1428     \@parboxrestore
1429     \color@begingroup
1430     \csname #1footfmt\endcsname #2[#1]\color@endgroup}}
1431

```

`\ledsetnormalparstuff` `\normalfootfmt` is a ‘normal’ macro to take the footnote line and page number information (see p.58), 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).

```

1432 \newcommand*{\ledsetnormalparstuff}{%
1433   \ifluatex%
1434     \luatextextdir\footnote@luatextextdir%
1435     \luatexpardir\footnote@luatexpardir%
1436   \fi%
1437   \csuse{\csuse{footnote@dir}}}%
1438   \normal@pars%
1439   \noindent \parfillskip \z@ \@plus 1fil}
1440
1441 \notbool{parapparatus@}{\newcommandx*}{\newcommandx}{\normalfootfmt}[4][4=Z]{% 4th arg is
1442   \ledsetnormalparstuff%
1443   \hangindent=\csuse{Xhangindent@#4}
1444   \strut{\printlinefootnote{#1}{#4}}%
1445   {\select@lemmafont#1|#2}%
1446   \iftoggle{nosep@}{\hskip\csuse{inplaceoflemmaseparator@#4}}{\ifcsemtyp{lemmaseparator@#4}
1447     {\hskip\csuse{inplaceoflemmaseparator@#4}}%
1448     {\nobreak\hskip\csuse{beforelemmaseparator@#4}\csuse{lemmaseparator@#4}\hskip\csuse{a
1449   }}%
1450   #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
`\fullstop` does not contain an en-dash or square brackets, and its period and comma are in
`\rbracket`

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.

```

1451 \def\endashchar{\textnormal{--}}
1452 \newcommand*{\fullstop}{\textnormal{.}}
1453 \newcommand*{\rbracket}{\textnormal{}}
1454 \csuse{text}\csuse{footnote@lang}}{%
1455     \ifluatex%
1456     \ifdefstring{\footnote@luatextextdir}{TRT}{\thinspace[]{\thinspace}}%
1457     \else%
1458     \thinspace}%
1459     \fi}%
1460 }%
1461 }
1462
```

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

```

1463 \newcommand{\printpstart}[0]{%
1464     \ifl@dpairing%
1465     \ifledRcol%
1466     \thepstartR%
1467     \else%
1468     \thepstartL%
1469     \fi%
1470 \else%
1471     \thepstart%
1472     \fi%
1473 }
```

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 58: the starting page, line, and sub-line numbers, followed by the ending page, line, and sub-line numbers, and then the font specifier for the lemma.

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

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

- `\@pnum` for page numbers;
- `\@ssub` for starting sub-line;

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

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

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

```

\ifl@d@pnum
\ifl@d@ssub 1474 \newif\ifl@d@pnum
\ifl@d@elin 1475 \l@d@pnumfalse
\ifl@d@esl 1476 \newif\ifl@d@ssub
\ifl@d@dash 1477 \l@d@ssubfalse
1478 \newif\ifl@d@elin
1479 \l@d@elinfalse
1480 \newif\ifl@d@esl
1481 \l@d@eslfalse
1482 \newif\ifl@d@dash
1483 \l@d@dashfalse

\l@dp@rsefootsspec \l@dp@rsefootsspec{<spec>}{<lemma>}{<text>} parses a footnote specification.
\l@dp@rsefootsspec <lemma> and <text> are the lemma and text respectively. <spec> is the line and
\l@dp@rsefootsspec page number and lemma font specifier in \l@d@nums style format. The real work
\l@dp@rsefootsspec is done by \l@dp@rsefootsspec which defines macros holding the numeric values.
\l@dp@rsefootsspec 1484 \newcommand*{\l@dp@rsefootsspec}[3]{\l@dp@rsefootsspec#1|}
\l@dp@rsefootsspec 1485 \def\l@dp@rsefootsspec#1|#2|#3|#4|#5|#6|#7|{%
\l@dp@rsefootsspec 1486 \gdef\l@dp@rsefootsspec#1|#2|#3|#4|#5|#6|#7|{%
\l@dp@rsefootsspec 1487 \gdef\l@dp@rsefootsspec#1|#2|#3|#4|#5|#6|#7|{%
1488 \gdef\l@dp@rsefootsspec#1|#2|#3|#4|#5|#6|#7|{%
1489 \gdef\l@dp@rsefootsspec#1|#2|#3|#4|#5|#6|#7|{%
1490 \gdef\l@dp@rsefootsspec#1|#2|#3|#4|#5|#6|#7|{%
1491 \gdef\l@dp@rsefootsspec#1|#2|#3|#4|#5|#6|#7|{%
1492 }

Initialise the several number value macros.
1493 \def\l@dp@rsefootsspec#1|#2|#3|#4|#5|#6|#7|{%
1494 \def\l@dp@rsefootsspec#1|#2|#3|#4|#5|#6|#7|{%
1495 \def\l@dp@rsefootsspec#1|#2|#3|#4|#5|#6|#7|{%
1496 \def\l@dp@rsefootsspec#1|#2|#3|#4|#5|#6|#7|{%
1497 \def\l@dp@rsefootsspec#1|#2|#3|#4|#5|#6|#7|{%
1498 \def\l@dp@rsefootsspec#1|#2|#3|#4|#5|#6|#7|{%
1499

\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 | subtitle | end-page | line | subtitle | 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`.

```
1500 \newcommand*\setprintlines}[6]{%
1501   \l@dpnumfalse \l@ddashfalse
1502   \ifbypage@
1503     \ifnum#4=#1 \else
1504       \l@dpnumtrue
1505       \l@ddashtrue
1506     \fi
1507   \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.

```
1508   \ifl@dpnum \l@d@elintrue \else \l@d@elinfalse \fi
1509   \ifnum#2=#5 \else
1510     \l@d@elintrue
1511     \l@ddashtrue
1512   \fi
```

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

```
1513   \l@d@ssubfalse
1514   \ifnum#3=0 \else
1515     \l@d@ssubtrue
1516   \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.

```
1517   \l@d@eslfalse
1518   \ifnum#6=0 \else
1519     \ifnum#6=#3
1520       \ifl@d@elin \l@d@esltrue \else \l@d@eslfalse \fi
1521     \else
1522       \l@d@esltrue
1523       \l@ddashtrue
1524     \fi
1525   \fi}
```

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

```
1526 \def\printlines#1|#2|#3|#4|#5|#6|#7|{\begingroup
1527   \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 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).

```
1528   \ifl@dpnum #1\fullstop\fi
1529   \linenumrep{#2}

1530   \ifl@d@ssub \fullstop \sublinenumrep{#3}\fi
1531   \ifl@ddash \endashchar\fi
```

```

1532 \ifl@d@pnum #4\fullstop\fi
1533 \ifl@d@elin \linenumrep{#5}\fi
1534 \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumrep{#6}\fi
1535 \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. T_EX 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`.

```

1536 \newcommand*{\normalfootstart}[1]{%
1537   \ifdimequal{0pt}{\preXnotes@}{}%
1538   {%
1539     \iftoggle{preXnotes@}{%
1540       \togglefalse{preXnotes@}\skip\csname #1footins\endcsname=\csuse{preXnotes@}}%
1541     }%
1542   }%
1543   \vskip\skip\csname #1footins\endcsname%
1544   \leftskip0pt \rightskip0pt
1545   \csname #1footnoterule\endcsname\noindent\leavevmode}

```

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

```

1546 \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.

```

1547 \newcommand*{\normalfootgroup}[1]{\csuse{Xnotefontsize@#1}\noindent\csuse{txtbeforeXnotes@#1}}
1548

```

`\mpnormalfootgroup` A somewhat different version for minipages.

```

1549 \newcommand*{\mpnormalfootgroup}[1]{%
1550   \vskip\skip\@nameuse{mp#1footins}
1551   \normalcolor
1552   \@nameuse{#1footnoterule}
1553   {\csuse{Xnotefontsize@#1}\noindent\csuse{txtbeforeXnotes@#1}}
1554   \unvbox\csname mp#1footins\endcsname}
1555

```

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\vsiz
\let\vAfootnote=\normalvfootnote \let\Afootfmt=\normalfootfmt
\let\Afootstart=\normalfootstart \let\Afootgroup=\normalfootgroup
\let\Afootnoterule=\normalfootnoterule
```

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

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

```
1556 \newcommand*{\ledfootinsdim}{0.8\vsiz} % kept for backward compatibility, should'nt be used anymore.
```

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

```
1557 \newtoggle{preXnotes@}
1558 \toggletrue{preXnotes@}
1559 \newcommand{\preXnotes@}{0pt}
1560 \newcommand*{\preXnotes}[1]{\renewcommand{\preXnotes@}{#1}}
```

The same, but for familiar footnotes.

```
\preXnotes
\preXnotes@ 1561 \newtoggle{prenotesX@}
1562 \toggletrue{prenotesX@}
1563 \newcommand{\prenotesX@}{0pt}
1564 \newcommand*{\prenotesX}[1]{\renewcommand{\prenotesX@}{#1}}
```

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

```
1565 \newcommand*{\footnormal}[1]{%
1566 \csgdef{series@display#1}{normal}}
```

```

1567 \expandafter\let\csname #1footstart\endcsname=\normalfootstart
1568 \expandafter\let\csname v#1footnote\endcsname=\normalvfootnote
1569 \expandafter\let\csname #1footfmt\endcsname=\normalfootfmt
1570 \expandafter\let\csname #1footgroup\endcsname=\normalfootgroup
1571 \expandafter\let\csname #1footnoterule\endcsname=%
1572                                     \normalfootnoterule
1573 \count\csname #1footins\endcsname=1000
1574 \dimen\csname #1footins\endcsname=\csuse{maxhXnotes@#1}
1575 \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).

```

1576 \expandafter\let\csname mpv#1footnote\endcsname=\mpnormalvfootnote
1577 \expandafter\let\csname mp#1footgroup\endcsname=\mpnormalfootgroup
1578 \count\csname mp#1footins\endcsname=1000
1579 \dimen\csname mp#1footins\endcsname=\csuse{maxhXnotes@#1}
1580 \skip\csname mp#1footins\endcsname=\csuse{beforeXnotes@#1}
1581 }
1582

```

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

24.5 Paragraphed footnotes

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

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

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

```

1583 \newcommand*{\footparagraph}[1]{%
1584   \csgdef{series@display#1}{paragraph}
1585   \expandafter\newcount\csname prevpage#1@num\endcsname
1586   \expandafter\let\csname #1footstart\endcsname=\parafootstart

```



```

1587 \expandafter\let\csname v#1footnote\endcsname=\para@vfootnote
1588 \expandafter\let\csname #1footfmt\endcsname=\parafootfmt
1589 \expandafter\let\csname #1footgroup\endcsname=\para@footgroup
1590 \count\csname #1footins\endcsname=1000
1591 \para@footsetup{#1}

```

And the extra setup for minipages.

```

1592 \expandafter\let\csname mpv#1footnote\endcsname=\mppara@vfootnote
1593 \expandafter\let\csname mp#1footgroup\endcsname=\mppara@footgroup
1594 \count\csname mp#1footins\endcsname=1000
1595 }

```

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

```
1596 \providecommand{\footfudgefiddle}{64}
```

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

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

```

1597 \newcommand*{\para@footsetup}[1]{\csuse{Xnotefontsize@#1}
1598   \dimen0=\baselineskip
1599   \multiply\dimen0 by 1024
1600   \divide \dimen0 by \columnwidth \multiply\dimen0 by \footfudgefiddle\relax
1601   \expandafter
1602   \xdef\csname #1footfudgefactor\endcsname{%
1603     \expandafter\strip@pt\dimen0 }}
1604

```

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

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

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

```

1605 \newcommand*{\parafootstart}[1]{%
1606   \rightskip=0pt \leftskip=0pt \parindent=0pt
1607   \ifdimequal{0pt}{\preXnotes@}{}%
1608   {%

```

```

1609      \iftoggle{preXnotes@}{%
1610          \togglefalse{preXnotes@}\skip\csname #1footins\endcsname=\csuse{preXnotes@}}%
1611          {}%
1612      }%
1613      \vskip\skip\csname #1footins\endcsname%
1614      \csname #1footnoterule\endcsname\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 T_EX 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.²⁴

Wayne Sullivan pointed out to us another subtle problem that arises from the same cause: T_EX also leaves the `\language` whatsit nodes out of the horizontal list.²⁵ 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.²⁶ Michael's unboxing macro is called `\unvvh`: `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.²⁷ In other words, be very careful not to say `\break`, or `\penalty-10000`,

²⁴Michael Downes, 'Line Breaking in `\unhboxed` Text', *TUGboat* 11 (1990), pp. 605–612.

²⁵See *The TeXbook*, p. 455 (editions after January 1990).

²⁶Wayne supplied his own macros to do this, but since they were almost identical to Michael's, we have used the latter's `\unvvh` macro since it is publicly documented.

²⁷'Line Breaking', p. 610.

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. 102 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.

```

1615 \newcommand*{\para@vfootnote}[2]{%
1616   \insert\csname #1footins\endcsname
1617   \bgroup
1618     \csuse{bhookXnote@#1}
1619     \csuse{Xnotefontsize@#1}
1620     \footplitskips
1621     \setbox0=\vbox{\hsize=\maxdimen
1622       \noindent\csname #1footfmt\endcsname#2[#1]}%
1623     \setbox0=\hbox{\unvvh0[#1]}%
1624     \dp0=0pt
1625     \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.

```

1626   \if@RTL\noindent \leavevmode\fi\box0%
1627   \penalty0
1628   \egroup}
1629

```

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 T_EXbook*, 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 `\makehboxoffhboxes`. 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.

```

1630 \newcommand*{\mppara@vfootnote}[2]{%
1631   \global\setbox\@nameuse{mp#1footins}\vbox{%
1632     \unvvhbox\@nameuse{mp#1footins}%
1633     \csuse{bhookXnote@#1}

```

```

1634 \csuse{Xnotefontsize@#1}
1635 \footsplitskips
1636 \setbox0=\vbox{\hsize=\maxdimen
1637   \noindent\color@begingroup\csname #1footfmt\endcsname #2[#1]\color@endgroup}%
1638 \setbox0=\hbox{\unvxh0[#1]}%
1639 \dp0=\z@
1640 \ht0=\csname #1footfudgefactor\endcsname\wd0
1641 \box0
1642 \penalty0
1643 }}
1644

```

`\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). `\unvxh` cancels these unwanted paragraph-final items using `\unskip` and `\unpenalty`.

```

1645 \newcommand*{\unvxh}[2][2=Z]{% 2th is optional for retro-compatibility
1646   \setbox0=\vbox{\unvbox#1%
1647     \global\setbox1=\lastbox}%
1648   \unhbox1
1649   \unskip           % remove \rightskip,
1650   \unskip           % remove \parfillskip,
1651   \unpenalty        % remove \penalty of 10000,
1652   \hskip\csuse{afternote@#2}} % but add the glue to go between the notes
1653

```

`\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).

```

1654 \newcommand*{\parafootfmt}[4][4=Z]{%
1655   \insertparafootsep{#4}%
1656   \ledsetnormalparstuff%
1657   \printlinefootnote{#1}{#4}%
1658   {\select@lemmafont#1|#2}%
1659   \iftoggle{nosep@}{\hskip\csuse{inplaceoflemmaseparator@#4}}{\ifcsempy{lemmaseparator@#4}
1660     {\hskip\csuse{inplaceoflemmaseparator@#4}}%
1661     {\nobreak\hskip\csuse{beforelemmaseparator@#4}\csuse{lemmaseparator@#4}\hskip\csuse{af
1662   }}%
1663   #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.

```

1664 \newcommand*{\para@footgroup}[1]{%
1665   \unvbox\csname #1footins\endcsname
1666   \makehboxofhboxes
1667   \setbox0=\hbox{\csuse{Xnotefontsize@#1}\csuse{txtbeforeXnotes@#1}}\unhbox0 \removehboxes}%
1668   \csuse{Xnotefontsize@#1}
1669   \noindent\unhbox0\par}
1670
```

`\mppara@footgroup` The minipage version.

```

1671 \newcommand*{\mppara@footgroup}[1]{%
1672   \vskip\skip\@nameuse{mp#1footins}
1673   \normalcolor
1674   \@nameuse{#1footnoterule}%
1675   \unvbox\csname mp#1footins\endcsname
1676   \makehboxofhboxes
1677   \setbox0=\hbox{\csuse{Xnotefontsize@#1}\csuse{txtbeforeXnotes@#1}}\unhbox0 \removehboxes}%
1678   \csuse{Xnotefontsize@#1}
1679   \noindent\unhbox0\par}}
1680
```

`\makehboxofhboxes`

```

\removehboxes 1681 \newcommand*{\makehboxofhboxes}{\setbox0=\hbox{}%
1682   \loop
1683     \unpenalty
1684     \setbox2=\lastbox
1685     \ifhbox2
1686       \setbox0=\hbox{\box2\unhbox0}%
1687     \repeat}
1688
1689 \newcommand*{\removehboxes}{\setbox0=\lastbox
1690   \ifhbox0{\removehboxes}\unhbox0 \fi}
1691
```

24.5.1 Insertion of the footnotes separator

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

`\prevpage@num`

```

\insertparafootsep 1692 \newcommand{\insertparafootsep}[1]{%
1693   \ifnumequal{\csuse{prevpage#1@num}}{\page@num}%
1694     {\ifcsdef{prevline#1}% Be sur \prevline#1 exists.
1695       {\ifnumequal{\csuse{prevline#1}}{\line@num}%
1696         {\ifcsempy{symplinenumber}{\csuse{parafootsep@#1}}{}}}%

```

```

1697         {\csuse{parafootsep@#1}}%
1698     }%
1699     {\csuse{parafootsep@#1}}%
1700 }%
1701 {}%
1702 \global\csname prevpage#1@num\endcsname=\page@num%
1703 }

```

24.6 Columnar footnotes

\rigidbalance We will now define macros for three-column notes and two-column notes. Both sets of macros will use **\rigidbalance**, which splits a box (#1) into a number (#2) of columns, each with a space (#3) between the top baseline and the top of the **\vbox**. The **\rigidbalance** macro is taken from *The TeXbook*, p.397, with a slight change to the syntax of the arguments so that they don't depend on white space. Note also the extra unboxing in **\splitoff**, which allows the new **\vbox** to have its natural height as it goes into the alignment.

The LaTeX **\line** macro has no relationship to the TeX **\line**. The LaTeX equivalent is **\@@line**.

```

1704 \newcount\@k \newdimen\@h
1705 \newcommand*\rigidbalance}[3]{\setbox0=\box#1 \@k=#2 \@h=#3
1706 \@@line{\splittopskip=\@h \vbadness=\@M \hfilneg
1707 \valign{##\vfil\cr\dosplits}}}%
1708
1709 \newcommand*\dosplits{\ifnum\@k>0 \noalign{\hfil}\splitoff
1710 \global\advance\@k-1\cr\dosplits\fi}
1711
1712 \newcommand*\splitoff{\dimen0=\ht0
1713 \divide\dimen0 by\@k \advance\dimen0 by\@h
1714 \setbox2 \vsplit0 to \dimen0
1715 \unvbox2 }
1716

```

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.

```

1717 \newcommand*\footthreecol}[1]{%
1718 \csgdef{series@display#1}{threecol}
1719 \expandafter\let\csname v#1footnote\endcsname=\threecolvfootnote
1720 \expandafter\let\csname #1footfmt\endcsname=\threecolfootfmt
1721 \expandafter\let\csname #1footgroup\endcsname=\threecolfootgroup
1722 \threecolfootsetup{#1}

```

The additional setup for minipages.

```

1723 \expandafter\let\csname mpv#1footnote\endcsname=\mpnormalvfootnote
1724 \expandafter\let\csname mp#1footgroup\endcsname=\mpthreecolfootgroup

```

```

1725 \mpthreecolfootsetup{#1}
1726 }
1727

```

The `\footstart` and `\footnoterule` macros for these notes assume the normal values (p.102 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 T_EX is accumulating material for the page and checking that limit, it doesn't apply the `\count` scaling.

```

1728 \newcommand*{\threecolfootsetup}[1]{%
1729   \count\csname #1footins\endcsname 333
1730   \multiply\dimen\csname #1footins\endcsname \thr@@}

```

`\mpthreecolfootsetup` The setup for minipages.

```

1731 \newcommand*{\mpthreecolfootsetup}[1]{%
1732   \count\csname mp#1footins\endcsname 333
1733   \multiply\dimen\csname mp#1footins\endcsname \thr@@}
1734

```

`\threecolvf footnote` `\threecolvf footnote` 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).

```

1735 \notbool{parapparatus@}{\newcommand*{\newcommand}{\threecolvf footnote}[2]{%
1736   \insert\csname #1footins\endcsname\bgroup
1737   \csuse{Xnotefontsize@#1}
1738   \footplitskips
1739   \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.

```

1740 \notbool{parapparatus@}{\newcommandx*}{\newcommandx}{\threecolfootfmt}[4][4=Z]{%
1741   \normal@pars
1742   \hsize \csuse{hsizethreecol@#4}
1743   \parindent=0pt
1744   \tolerance=5000
1745   \raggedright
1746   \hangindent=\csuse{Xhangindent@#4}
1747   \leavevmode
1748   \strut{\printlinefootnote{#1}{#4}}%
1749   {\select@lemmafnt#1|#2}%
1750   \iftoggle{nosep@}{\hskip\csuse{inplaceoflemmaseparator@#4}}{\ifcsemtylemmaseparator@#4}
1751     {\hskip\csuse{inplaceoflemmaseparator@#4}}%
1752     {\nobreak\hskip\csuse{beforelemmaseparator@#4}\csuse{lemmaseparator@#4}\hskip\csuse{a
1753   }}%
1754   #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.

```

1755 \newcommand*{\threecolfootgroup}[1]{\notefontsetup
1756   {\csuse{Xnotefontsize@#1}\noindent\csuse{txtbeforeXnotes@#1}}\par
1757   \splittopskip=\ht\strutbox
1758   \expandafter
1759   \rigidbalance\csname #1footins\endcsname \thr@@ \splittopskip}}

```

`\mpthreecolfootgroup` The setup for minipages.

```

1760 \newcommand*{\mpthreecolfootgroup}[1]{\%
1761   \vskip\skip\@nameuse{mp#1footins}
1762   \normalcolor
1763   \@nameuse{#1footnoterule}
1764   {\csuse{Xnotefontsize@#1}\noindent\csuse{txtbeforeXnotes@#1}}\par
1765   \splittopskip=\ht\strutbox
1766   \expandafter
1767   \rigidbalance\csname mp#1footins\endcsname \thr@@ \splittopskip}}
1768

```

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.

```
1769 \newcommand*{\foottwocol}[1]{%
1770   \csgdef{series@display#1}{twocol}
1771   \expandafter\let\csname v#1footnote\endcsname=\twocolvfootnote
1772   \expandafter\let\csname #1footfmt\endcsname=\twocolfootfmt
1773   \expandafter\let\csname #1footgroup\endcsname=\twocolfootgroup
1774   \twocolfootsetup{#1}
```

The additional setup for minipages.

```
1775   \expandafter\let\csname mpv#1footnote\endcsname=\mpnormalvfootnote
1776   \expandafter\let\csname mp#1footgroup\endcsname=\mptwocolfootgroup
1777   \mptwocolfootsetup{#1}
1778 }
1779
```

`\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 1780 \newcommand*{\twocolfootsetup}[1]{%
1781   \count\csname #1footins\endcsname 500
1782   \multiply\dimen\csname #1footins\endcsname \tw@}

1783 \notbool{parapparatus@}{\newcommand*{\newcommand}{\twocolvfootnote}[2]{\insert\csname #1footins\endcsname
1784   \csuse{Xnotefontsize@#1}
1785   \footssplitskips
1786   \csname #1footfmt\endcsname #2[#1]\egroup}

1787 \notbool{parapparatus@}{\newcommand*{\newcommandx}{\twocolfootfmt}[4][4=Z]{% 4th arg is optional, f
1788   \normal@pars
1789   \hsize \csuse{hsizetwocol@#4}
1790   \parindent=0pt
1791   \tolerance=5000
1792   \raggedright
1793   \hangindent=\csuse{Xhangindent@#4}
1794   \leavevmode
1795   \strut{\printlinefootnote{#1}{#4}}%
1796   {\select@lemmafnt#1|#2}%
1797   \iftoggle{nosep@}{\hskip\csuse{inplaceoflemmaseparator@#4}}{\ifcsemtyp{lemmaseparator@#4}%
1798     {\hskip\csuse{inplaceoflemmaseparator@#4}}%
1799     {\nobreak\hskip\csuse{beforelemmaseparator@#4}\csuse{lemmaseparator@#4}\hskip\csuse{afterlemmasep
1800   }}%
1801   #3\strut\par\allowbreak}

1802 \newcommand*{\twocolfootgroup}[1]{\csuse{Xnotefontsize@#1}
1803   {\csuse{Xnotefontsize@#1}\noindent\csuse{txtbeforeXnotes@#1}}\par
1804   \splittopskip=\ht\strutbox
1805   \expandafter
1806   \rigidbalance\csname #1footins\endcsname \tw@ \splittopskip}}
1807
```

```

\mptwocolfootsetup The versions for minipages.
\mptwocolfootgroup 1808 \newcommand*{\mptwocolfootsetup}[1]{%
1809   \count\csname mp#1footins\endcsname 500
1810   \multiply\dimen\csname mp#1footins\endcsname \tw@}

1811 \newcommand*{\mptwocolfootgroup}[1]{%
1812   \vskip\skip\@nameuse{mp#1footins}
1813   \normalcolor
1814   \@nameuse{#1footnoterule}
1815   {\csuse{Xnotefontsize@#1}\noindent\csuse{txtbeforeXnotes@#1}}\par
1816   \splittopskip=\ht\strutbox
1817   \expandafter
1818   \rigidbalance\csname mp#1footins\endcsname \tw@ \splittopskip}}
1819

```

25 Familiar footnotes

25.1 Generality

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

First, though, the `footmisc` package has an option whereby two or more consecutive `\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
\multfootsep footmisc package and perhaps by other footnote packages.

1820 \providecommand*{\multiplefootnotemarker}{3sp}
1821 \providecommand*{\multfootsep}{\textsuperscript{\normalfont,}}
1822

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

1823 \providecommand*{\m@mmf@prepare}{%
1824   \kern-\multiplefootnotemarker
1825   \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.

1826 \providecommand*{\m@mmf@check}{%
1827   \ifdim\lastkern=\multiplefootnotemarker\relax
1828     \edef\@x@sf{\the\spacefactor}%
1829     \unkern
1830     \multfootsep
1831     \spacefactor\@x@sf\relax
1832   \fi}
1833

```

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

```
1834 \ifclassloaded{memoir}{}{%
```

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

```
1835 \let\l@dold@footnotetext\@footnotetext
1836 \renewcommand{\@footnotetext}[1]{%
1837   \l@dold@footnotetext{#1}%
1838   \m@mmf@prepare}
```

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

```
1839 \renewcommand*{\@footnotemark}{%
1840   \leavevmode
1841   \ifhmode
1842     \edef\x@sf{\the\spacefactor}%
1843     \m@mmf@check
1844     \nobreak
1845     \fi
1846     \@makefnmark
1847     \m@mmf@prepare
1848     \ifhmode\spacefactor\x@sf\fi
1849     \relax}
```

Finished the modifications for the non-memoir case.

```
1850 }
1851
```

`\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.

```
1852 \let\l@doldold@footnotetext\@footnotetext
1853 \renewcommand{\@footnotetext}[1]{%
1854   \ifnumberedpar@
1855     \edtext{}{\l@dbfnote{#1}}%
1856   \else
1857     \l@doldold@footnotetext{#1}%
1858   \fi}
```

`\l@dbfnote` `\l@dbfnote` adds the footnote to the insert list, and `\v1@dbfnote` calls the original

`\v1@dbfnote` `\@footnotetext`.

```
1859 \newcommand{\l@dbfnote}[1]{%
1860   \ifnumberedpar@
1861   \gdef\@tag{#1}%
1862   \xright@appenditem{\noexpand\v1@dbfnote{\csexpandonce{\@tag}}}{\@thefnmark}}%
1863   \to\inserts@list
1864   \global\advance\insert@count \@ne
1865   \fi\ignorespaces}
1866 \newcommand{\v1@dbfnote}[2]{%
1867   \def\@thefnmark{#2}%
1868   \l@doldold@footnotetext{#1}}
```

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 1869 \newcommand*{\prebodyfootmark}{%
1870   \leavevmode
1871   \ifhmode
1872     \edef\x@sf{\the\spacefactor}%
1873     \m@mmf@check
1874     \nobreak
1875   \fi}
1876 \newcommand{\postbodyfootmark}{%
1877   \m@mmf@prepare
1878   \ifhmode\spacefactor\x@sf\fi\relax}
1879

\normal@footnotemarkX  \normal@footnotemarkX{<series>} sets up the typesetting of the marker at the
                        point where the footnote is called for.
1880 \newcommand*{\normal@footnotemarkX}[1]{%
1881   \prebodyfootmark
1882   \@nameuse{bodyfootmark#1}%
1883   \postbodyfootmark}
1884

\normalbodyfootmarkX  The \normalbodyfootmarkX{<series>} really typesets the in-text marker. The
                        style is the normal superscript.
1885 \newcommand*{\normalbodyfootmarkX}[1]{%
1886   \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>.
1887 \newcommand*{\normalvfootnoteX}[2]{%
1888   \insert\@nameuse{footins#1}\bgroup
1889     \csuse{bhooknoteX@#1}
1890     \csuse{notefontsizeX@#1}
1891     \footsplitskips
1892     \spaceskip=\z@skip \xspaceskip=\z@skip
1893     \csuse{\csuse{footnote@dir}}\ifRTL\else\noindent\leavevmode\fi\@nameuse{footfmt#1}{#1}
1894

\mpnormalvfootnoteX  The minipage version.
1895 \newcommand*{\mpnormalvfootnoteX}[2]{%
1896   \global\setbox\@nameuse{mpfootins#1}\vbox{%
1897     \unvbox\@nameuse{mpfootins#1}
1898     \csuse{bhooknoteX@#1}
1899     \csuse{notefontsizeX@#1}

```

```

1900 \hsize\columnwidth
1901 \@parboxrestore
1902 \color@begingroup
1903 \@nameuse{footfmt#1}{#1}{#2}\color@endgroup}}
1904

```

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

```

1905 \newcommand*\normalfootfmtX[2]{%
1906   \protected@edef\@currentlabel{%
1907     \@nameuse{@thefnmark#1}%
1908   }%
1909   \ledsetnormalparstuff
1910   \hangindent=\csuse{hangindentX@#1}%
1911   {{\csuse{notenumfontX@#1}\@nameuse{footfootmark#1}}\strut%\enspace
1912     #2\strut\par}}
1913

```

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

```

1914 \newcommand*\normalfootfootmarkX[1]{%
1915   \textsuperscript{\@nameuse{@thefnmark#1}}}
1916

```

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

```

1917 \newcommand*\normalfootstartX[1]{%
1918   \ifdimequal{0pt}{\prenotesX@}{}%
1919   {%
1920     \iftoggle{prenotesX@}{%
1921       \togglefalse{prenotesX@} \skip\csname footins#1\endcsname=\csuse{prenotesX@}}%
1922     }%
1923   }%
1924   \vskip\skip\csname footins#1\endcsname%
1925   \leftskip=\z@
1926   \rightskip=\z@
1927   \@nameuse{footnoterule#1}}
1928

```

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

```

1929 \let\normalfootnoteruleX=\footnoterule
1930

```

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

```

1931 \newcommand*\normalfootgroupX[1]{%
1932   \unvbox\@nameuse{footins#1}}
1933

```

`\mpnormalfootgroupX` The minipage version.

```
1934 \newcommand*{\mpnormalfootgroupX}[1]{%
1935   \vskip\skip\@nameuse{mpfootins#1}
1936   \normalcolor
1937   \@nameuse{footnoterule#1}
1938   \unvbox\@nameuse{mpfootins#1}}
1939
```

`\normalbfnoteX`

```
1940 \newcommand{\normalbfnoteX}[2]{%
1941   \ifnumberedpar@
1942     \protected@csxdef{thisfootnote}{\csuse{thefootnote#1}}%
1943     \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\csexpandonce{thisfootnote}}}%
1944                       \to\inserts@list
1945     \global\advance\insert@count \@ne
1946   \fi\ignorespaces}
1947
```

`\vbfnoteX`

```
1948 \newcommand{\vbfnoteX}[3]{%
1949   \@namedef{@thefnmark#1}{#3}%
1950   \@nameuse{regvfootnote#1}{#1}{#2}}
1951
```

`\vnumfootnoteX`

```
1952 \newcommand{\vnumfootnoteX}[2]{%
1953   \ifnumberedpar@
1954     \edtext{}{\normalbfnoteX{#1}{#2}}%
1955   \else
1956     \@nameuse{regvfootnote#1}{#1}{#2}%
1957   \fi}
1958
```

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

```
1959 \newcommand*{\footnormalX}[1]{%
1960   \csgdef{series@displayX#1}{normalX}
1961   \expandafter\let\csname footstart#1\endcsname=\normalfootstartX
1962   \@namedef{@footnotemark#1}{\normal@footnotemarkX{#1}}
1963   \@namedef{bodyfootmark#1}{\normalbodyfootmarkX{#1}}
1964   \expandafter\let\csname regvfootnote#1\endcsname=\normalvfootnoteX
1965   \expandafter\let\csname vfootnote#1\endcsname=\vnumfootnoteX
1966   \expandafter\let\csname footfmt#1\endcsname=\normalfootfmtX
1967   \@namedef{footfootmark#1}{\normalfootfootmarkX{#1}}
1968   \expandafter\let\csname footgroup#1\endcsname=\normalfootgroupX
1969   \expandafter\let\csname footnoterule#1\endcsname=\normalfootnoteruleX
1970   \count\csname footins#1\endcsname=1000
1971   \dimen\csname footins#1\endcsname=\csuse{maxhnotesX@#1}
1972   \skip\csname footins#1\endcsname=\csuse{beforenotesX@#1}
```

Additions for minipages.

```

1973 \expandafter\let\csname mpvfootnote#1\endcsname=\mpnormalvfootnoteX
1974 \expandafter\let\csname mpfootgroup#1\endcsname=\mpnormalfootgroupX
1975 \count\csname mpfootins#1\endcsname=1000
1976 \dimen\csname mpfootins#1\endcsname=\csuse{maxhnotesX@#1}
1977 \skip\csname mpfootins#1\endcsname=\csuse{beforenotesX@#1}
1978 }
1979

```

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>}

1980 \newcommand*{\foottwocolX}[1]{%
1981 \csgdef{series@displayX#1}{twocol}
1982 \expandafter\let\csname regvfootnote#1\endcsname=\twocolvfootnoteX
1983 \expandafter\let\csname footfmt#1\endcsname=\twocolfootfmtX
1984 \expandafter\let\csname footgroup#1\endcsname=\twocolfootgroupX
1985 \twocolfootsetupX{#1}
1986 \expandafter\let\csname mpvfootnote#1\endcsname=\mpnormalvfootnoteX
1987 \expandafter\let\csname mpfootgroup#1\endcsname=\mptwocolfootgroupX
1988 \mptwocolfootsetupX{#1}}
1989

\twocolfootsetupX \twocolfootsetupX{<series>}
\mptwocolfootsetupX 1990 \newcommand*{\twocolfootsetupX}[1]{%
1991 \count\csname footins#1\endcsname 500
1992 \multiply\dimen\csname footins#1\endcsname by \tw@}
1993 \newcommand*{\mptwocolfootsetupX}[1]{%
1994 \count\csname mpfootins#1\endcsname 500
1995 \multiply\dimen\csname mpfootins#1\endcsname by \tw@}
1996

\twocolvfootnoteX \twocolvfootnoteX{<series>}

1997 \newcommand*{\twocolvfootnoteX}[2]{%
1998 \insert\csname footins#1\endcsname\bgroup
1999 \csuse{notefontsizeX@#1}
2000 \footsplitskips
2001 \spaceskip=\z@skip \xspaceskip=\z@skip
2002 \@nameuse{footfmt#1}{#1}{#2}\egroup}
2003

\twocolfootfmtX \twocolfootfmtX{<series>}

2004 \newcommand*{\twocolfootfmtX}[2]{%
2005 \protected@edef\@currentlabel{%
2006 \@nameuse{@thefnmark#1}}%
2007 }%

```

```

2008 \normal@pars
2009 \hangindent=\csuse{hangindentX@#1}%
2010 \hsize \csuse{hsizetwocolX@#1}
2011 \parindent=\z@
2012 %% \parfillskip=0pt \@plus 1fil
2013 \tolerance=5000\relax
2014 \raggedright
2015 \leavevmode
2016 {\csuse{notenumfontX@#1}\@nameuse{footfootmark#1}\strut%\enspace
2017 #2\strut\par}\allowbreak}
2018

\twocolfootgroupX \twocolfootgroupX{<series>}
\mptwocolfootgroupX 2019 \newcommand*{\twocolfootgroupX}[1]{\csuse{notefontsizeX@#1}
2020 \splittopskip=\ht\strutbox
2021 \expandafter
2022 \rigidbalance\csname footins#1\endcsname \tw@ \splittopskip}}
2023 \newcommand*{\mptwocolfootgroupX}[1]{\%
2024 \vskip\skip\@nameuse{mpfootins#1}
2025 \normalcolor
2026 \@nameuse{footnoterule#1}
2027 \splittopskip=\ht\strutbox
2028 \expandafter
2029 \rigidbalance\csname mpfootins#1\endcsname \tw@ \splittopskip}}
2030

```

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>}
2031 \newcommand*{\footthreecolX}[1]{\%
2032 \csgdef{series@displayX#1}{threecol}
2033 \expandafter\let\csname regvfootnote#1\endcsname=\threecolvfootnoteX
2034 \expandafter\let\csname footfmt#1\endcsname=\threecolfootfmtX
2035 \expandafter\let\csname footgroup#1\endcsname=\threecolfootgroupX
2036 \threecolfootsetupX{#1}
2037 \expandafter\let\csname mpvfootnote#1\endcsname=\mpnormalvfootnoteX
2038 \expandafter\let\csname mpfootgroup#1\endcsname=\mpthreecolfootgroupX
2039 \mpthreecolfootsetupX{#1}}
2040

\threecolfootsetupX \threecolfootsetupX{<series>}
\mpthreecolfootsetupX 2041 \newcommand*{\threecolfootsetupX}[1]{\%
2042 \count\csname footins#1\endcsname 333
2043 \multiply\dimen\csname footins#1\endcsname by \thr@@
2044 \newcommand*{\mpthreecolfootsetupX}[1]{\%
2045 \count\csname mpfootins#1\endcsname 333

```



```

2046 \multiply\dimen\csname mpfootins#1\endcsname by \thr@@}
2047
\threecolvffootnoteX \threecolvffootnoteX{\series}\{<text>\}
2048 \newcommand*\threecolvffootnoteX}[2]{%
2049 \insert\csname footins#1\endcsname\bgroup
2050 \csuse{notefontsizeX@#1}
2051 \footplitskips
2052 \@nameuse{footfmt#1}{#1}{#2}\egroup}
2053
\threecolfootfmtX \threecolfootfmtX{\series}
2054 \newcommand*\threecolfootfmtX}[2]{%
2055 \protected@edef\@currentlabel{%
2056 \@nameuse{@thefnmark#1}%
2057 }%
2058 \hangindent=\csuse{hangindentX@#1}%
2059 \normal@pars
2060 \hsize \csuse{hsizethreecolX@#1}
2061 \parindent=\z@
2062 %%% \parfillskip=0pt \@plus 1fil
2063 \tolerance=5000\relax
2064 \raggedright
2065 \leavevmode
2066 {\csuse{notennumfontX@#1}\@nameuse{footfootmark#1}\strut%\enspace
2067 #2\strut\par}\allowbreak}
2068
\threecolfootgroupX \threecolfootgroupX{\series}
\mpthreecolfootgroupX 2069 \newcommand*\threecolfootgroupX}[1]{\csuse{notefontsizeX@#1}
2070 \splittopskip=\ht\strutbox
2071 \expandafter
2072 \rigidbalance\csname footins#1\endcsname \thr@@ \splittopskip}}
2073 \newcommand*\mpthreecolfootgroupX}[1]{%
2074 \vskip\skip\@nameuse{mpfootins#1}
2075 \normalcolor
2076 \@nameuse{footnoterule#1}
2077 \splittopskip=\ht\strutbox
2078 \expandafter
2079 \rigidbalance\csname mpfootins#1\endcsname \thr@@ \splittopskip}}
2080

```

25.5 Paragraphed footnotes

The following macros set footnotes as one paragraph.

```

\footparagraphX \footparagraphX{\series}
2081 \newcommand*\footparagraphX}[1]{%
2082 \csgdef{series@displayX#1}{paragraph}

```

```

2083 \expandafter\newcount\csname prevpage#1@num\endcsname
2084 \expandafter\let\csname footstart#1\endcsname=\parafootstartX
2085 \expandafter\let\csname regvfootnote#1\endcsname=\para@vfootnoteX
2086 \expandafter\let\csname footfmt#1\endcsname=\parafootfmtX
2087 \expandafter\let\csname footgroup#1\endcsname=\para@footgroupX
2088 \expandafter\let\csname footnoterule#1\endcsname=\normalfootnoteruleX
2089 \count\csname footins#1\endcsname=1000
2090 \expandafter\let\csname mpvfootnote#1\endcsname=\mppara@vfootnoteX
2091 \expandafter\let\csname mpfootgroup#1\endcsname=\mppara@footgroupX
2092 \count\csname mpfootins#1\endcsname=1000
2093 \para@footsetupX{#1}}
2094

```

\para@footsetupX \para@footsetupX{<series>}

```

2095 \newcommand*{\para@footsetupX}[1]{\csuse{notefontsizeX@#1}
2096 \dimen0=\baselineskip
2097 \multiply\dimen0 by 1024
2098 \divide\dimen0 by \hsize \multiply\dimen0 by \footfudgefiddle\relax
2099 \expandafter
2100 \xdef\csname footfudgefactor#1\endcsname{%
2101 \expandafter\strip@pt\dimen0 }}
2102

```

\parafootstartX \parafootstartX{<series>}

```

2103 \newcommand*{\parafootstartX}[1]{%
2104 \ifdimequal{0pt}{\prenotesX@}{}%
2105 {%
2106 \iftoggle{prenotesX@}{%
2107 \togglefalse{prenotesX@}\skip\csname footins#1\endcsname=\csuse{prenotesX@}}%
2108 }%
2109 }%
2110 \vskip\skip\csname footins#1\endcsname%
2111 \leftskip=\z@
2112 \rightskip=\z@
2113 \parindent=\z@
2114 \vskip\skip\@nameuse{footins#1}%
2115 \@nameuse{footnoterule#1}}
2116

```

\para@vfootnoteX \para@vfootnoteX{<series>}{<text>}

```

\mppara@vfootnoteX 2117 \newcommand*{\para@vfootnoteX}[2]{%
2118 \insert\csname footins#1\endcsname
2119 \bgroup
2120 \csuse{bhooknoteX@#1}
2121 \csuse{notefontsizeX@#1}
2122 \footsplitskips
2123 \setbox0=\vbox{\hsize=\maxdimen
2124 \noindent\@nameuse{footfmt#1}{#1}{#2}}%
2125 \setbox0=\hbox{\unvxh0[#1]}%

```

```

2126 \dp0=\z@
2127 \ht0=\csname footfudgefactor#1\endcsname\wd0
2128 \box0
2129 \penalty0
2130 \egroup}
2131 \newcommand*\mppara@vfootnoteX}[2]{%
2132 \global\setbox\@nameuse{mpfootins#1}\vbox{%
2133 \unvbox\@nameuse{mpfootins#1}
2134 \csuse{bhooknoteX@#1}
2135 \csuse{notefontsizeX@#1}
2136 \footsplitskips
2137 \setbox0=\vbox{\hsize=\maxdimen
2138 \noindent\color@begingroup\@nameuse{footfmt#1}-{#1}-{#2}\color@endgroup}%
2139 \setbox0=\hbox{\unvxh0[#1]}%
2140 \dp0=\z@
2141 \ht0=\csname footfudgefactor#1\endcsname\wd0
2142 \box0
2143 \penalty0}}
2144

```

`\parafootfmtX` `\parafootfmtX{<series>}`

```

2145 \newcommand*\parafootfmtX}[2]{%
2146 \protected@edef\@currentlabel{%
2147 \@nameuse{@thefnmark#1}%
2148 }%
2149 \insertparafootsep{#1}%
2150 \ledsetnormalparstuff
2151 {\csuse{notenumfontX@#1}\csuse{notenumfontX@#1}\@nameuse{footfootmark#1}\strut%\enspace
2152 #2\penalty-10}}
2153

```

`\para@footgroupX` `\para@footgroupX{<series>}`

```

\mppara@footgroupX 2154 \newcommand*\para@footgroupX}[1]{%
2155 \unvbox\csname footins#1\endcsname
2156 \makehboxofhboxes
2157 \setbox0=\hbox{\unhbox0 \removehboxes}%
2158 \csuse{notefontsizeX@#1}
2159 \noindent\unhbox0\par}
2160 \newcommand*\mppara@footgroupX}[1]{%
2161 \vskip\skip\@nameuse{mpfootins#1}
2162 \normalcolor
2163 \@nameuse{footnoterule#1}
2164 \unvbox\csname mpfootins#1\endcsname
2165 \makehboxofhboxes
2166 \setbox0=\hbox{\unhbox0 \removehboxes}%
2167 \csuse{notefontsizeX@#1}
2168 \noindent\unhbox0\par}}
2169

```

25.6 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.

```

2170 \newcommand*{\doxtrafeeti}{%
2171   \setbox\@outputbox \vbox{%
2172     \unvbox\@outputbox
2173     \def\do##1{\ifvoid\csuse{footins##1}\else\csuse{footstart##1}{##1}\csuse{footgroup##1}%
2174     \dolistloop{\@series}%
2175   }}
2176
2177 \newcommand{\doreinextrafeeti}{%
2178   \def\do##1{\ifvoid\csuse{footins##1}\else\insert\csuse{footins##1}{\unvbox\csuse{footins##1}%
2179   \dolistloop{\@series}%
2180   }
2181
```

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

```

2182 \newcommand*{\addfootinsX}[1]{%
2183   \eledmac@warning{AddfootinsX is obsolete in eledmac 1.0. Use newseries instead.}%
2184   \footnormalX{#1}%
2185   \g@addto@macro{\doxtrafeeti}{%
2186     \setbox\@outputbox \vbox{%
2187       \unvbox\@outputbox
2188       \ifvoid\@nameuse{footins#1}\else
2189         \@nameuse{footstart#1}{#1}\@nameuse{footgroup#1}{#1}\fi}}%as
2190   \g@addto@macro{\doreinextrafeeti}{%
2191     \ifvoid\@nameuse{footins#1}\else
2192       \insert\@nameuse{footins#1}{\unvbox\@nameuse{footins#1}}\fi}%
2193   \g@addto@macro{\l@dfambeginmini}{%
2194     \expandafter\expandafter\expandafter\let\expandafter\expandafter
2195       \csname footnote#1\endcsname \csname mpfootnote#1\endcsname}%
2196   \g@addto@macro{\l@dfamendmini}{%
2197     \ifvoid\@nameuse{mpfootins#1}\else\@nameuse{mpfootgroup#1}{#1}\fi}%
2198 }
```

26 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@`.

```

2199 \newcommand{\newseries}[1]{%
2200   \def\do##1{\newseries@{##1}}%
2201   \docsvlist{#1}
2202 }
```

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

```

2203 \newcommand{\@series}{}
```

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

`\newseries@`

```
2204 \newcommand{\newseries@}[1]{
```

26.0.1 Test if series is still existing

```
2205 \xifinlist{#1}{\@series}{\eledmac@warning{Series #1 is still existing !}}
2206 {%
```

26.0.2 Create all commands to memorize display options

```
2207 \csgdef{Xhangindent@#1}{0pt}%
2208 \csgdef{hangindentX@#1}{0pt}
2209 \csgdef{hsizetwocol@#1}{0.45 \hsize}%
2210 \csgdef{hsizetwocolX@#1}{0.45 \hsize}%
2211 \csgdef{hsizethreecol@#1}{.3 \hsize}%
2212 \csgdef{hsizethreecolX@#1}{.3 \hsize}%
2213 \csgdef{Xnotenumfont@#1}{\notenumfont}%
2214 \csgdef{Xendnotenumfont@#1}{\notenumfont}%
2215 \csgdef{notenumfontX@#1}{\notenumfont}%
2216 \csgdef{Xnotefontsize@#1}{\notefontsetup}%
2217 \csgdef{notefontsizeX@#1}{\notefontsetup}%
2218 \csgdef{Xendnotefontsize@#1}{\notefontsetup}%
2219 \csgdef{bhooknoteX@#1}{}%
2220 \csgdef{bhookXnote@#1}{}%
2221 \csgdef{bhookXendnote@#1}{}%
2222 \csgdef{boxlinenum@#1}{0pt}%
2223 \csgdef{boxsymlinenum@#1}{0pt}%
2224 \newtoggle{numberonlyfirstinline@#1}%
2225 \newtoggle{numberonlyfirstintwolines@#1}%
2226 \newtoggle{onlypstartinfootnote@#1}%
2227 \newtoggle{pstartinfootnote@#1}%
2228 \csgdef{symlinenum@#1}{\symlinenum}%
2229 \newtoggle{nonnumberinfootnote@#1}%
2230 \csgdef{beforenumberinfootnote@#1}{0pt}%
2231 \csgdef{afternumberinfootnote@#1}{0.5em}%
2232 \newtoggle{nonbreakableafternumber@#1}%
2233 \csgdef{beforesymlinenum@#1}{\csuse{beforenumberinfootnote@#1}}%
2234 \csgdef{aftersymlinenum@#1}{\csuse{afternumberinfootnote@#1}}%
2235 \csgdef{inplaceofnumber@#1}{1em}%
2236 \global\cslet{lemmaseparator@#1}{\rbracket}%
2237 \csgdef{beforelemmaseparator@#1}{0em}%
2238 \csgdef{afterlemmaseparator@#1}{0.5em}%
2239 \csgdef{inplaceoflemmaseparator@#1}{1em}%
2240 \csgdef{afternote@#1}{1em plus.4em minus.4em}%
2241 \csgdef{parafootsep@#1}{\parafootftmsep}%
2242 \csgdef{beforeXnotes@#1}{1.2em \@plus .6em \@minus .6em}
2243 \csgdef{beforenotesX@#1}{1.2em \@plus .6em \@minus .6em}
2244 \csgdef{txtbeforeXnotes@#1}{%
2245 \csgdef{maxhnotesX@#1}{\ledfootinsdim}%
```

```
2246 \csgdef{maxhXnotes@#1}{\ledfootinsdim}
```

26.0.3 Create inserts, needed to add notes in foot

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

```
2247
2248 \expandafter\newinsert\csname mpfootins#1\endcsname
2249 \expandafter\newinsert\csname footins#1\endcsname
2250 \expandafter\newinsert\csname #1footins\endcsname
2251 \expandafter\newinsert\csname mp#1footins\endcsname
```

26.0.4 Create command for critical apparatus, \Xfootnote

Note the double # in command: it's because command is made inside another command.

```
2252
2253 \global\newcommand{\parapparatus@}{\expandafter\newcommand\expandafter *}{\expandafter\new
2254 \begingroup%
2255 \newcommand{\content}{##2}%
2256 \ifnumberedpar@
2257 \ifledRcol%
2258 \ifluatex%
2259 \footnotelang@lua[R]%
2260 \fi%
2261 \@ifundefined{xpg@main@language}%if polyglossia
2262 {}%
2263 {\footnotelang@poly[R]}%
2264 \footnoteoptions@[R]{##1}{true}%
2265 \xright@appenditem{\noexpand\prepare@edindex@fornote{\l@d@nums}%
2266 \noexpand\csuse{v#1footnote}{#1}%
2267 {\l@d@nums}{\csexpandonce{@tag}}{\csexpandonce{content}}}}\to\inserts@li
2268 \footnoteoptions@[R]{##1}{false}%
2269 \global\advance\insert@countR \@ne%
2270 \else%
2271 \ifluatex%
2272 \footnotelang@lua%
2273 \fi%
2274 \@ifundefined{xpg@main@language}%if polyglossia
2275 {}%
2276 {\footnotelang@poly}%
2277 \footnoteoptions@{##1}{true}%
2278 \xright@appenditem{\noexpand\prepare@edindex@fornote{\l@d@nums}%
2279 \noexpand\csuse{v#1footnote}{#1}%
2280 {\l@d@nums}{\csexpandonce{@tag}}{\csexpandonce{content}}}}\to\inserts@li
2281 \global\advance\insert@count \@ne%
2282 \footnoteoptions@{##1}{false}%
2283 \fi
2284 \else
2285 \csuse{v#1footnote}{#1}{\{0|0|0|0|0|0|0\}}{##1}%
2286 \fi%
```

```

2287     \ignorespaces%
2288     \endgroup
2289     }

```

Set standard display and remember the display.

```

2290     \csgdef{series@display#1}{-}
2291     \footnormal{#1}

```

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

First, create the `\footnoteX` command.

```

2292
2293     \global\expandafter\newcommand\csname footnote#1\endcsname[1]{%
2294         \begingroup%
2295             \newcommand{\content}{##1}%
2296             \stepcounter{footnote#1}%
2297             \protected@csxdef{@thefnmark#1}{\csuse{thefootnote#1}}%
2298             \csuse{@footnotemark#1}%
2299             \csuse{vfootnote#1}{#1}{\csexpandonce{content}}\m@mmf@prepare%
2300         \endgroup%
2301     }

```

The counters.

```

2302     \newcounter{footnote#1}
2303     \global\expandafter\renewcommand\csname thefootnote#1\endcsname{\arabic{footnote#1}}
2304 % \end{macrocode}
2305 % Don't forget to initialize series
2306 %     \begin{macrocode}
2307     \csgdef{series@displayX#1}{-}
2308     \footnormalX{#1}

```

26.0.6 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.

```

2309
2310     \global\expandafter\newcommand\csname #1endnote\endcsname[2]{\newlinechar='40
2311         \newcommand{\content}{##1}%
2312         \immediate\write\l@d@end{\expandafter\string\csname #1end\endcsname%
2313             {\ifnumberedpar@\l@d@nums\fi}%
2314             {\ifnumberedpar@\csexpandonce{@tag}\fi}{\csexpandonce{content}}{#1}}\ignorespaces%
2315     }

```

`\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.²⁸

```

2316
2317   \global\csletcs{#1end}{\@gobblethree}
2318 %\end{macrocode}
2319 % We need to be able to modify \Eledmac's footnote macros and restore their
2320
2321   \global\csletcs{#1@@footnote}{#1footnote}
2322 % \cs{Stock series in \cs{@series}
2323 %   \begin{macrocode}
2324
2325   \listxadd{\@series}{#1}
2326 }
2327 }% End of \newseries

```

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

```

2328 \newseries{A,B,C,D,E,Z}

```

26.0.8 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.

```

2329 \newcommand{\seriesatbegin}[1]{
2330   \edef\series{#1}
2331   \def\new{}
2332   \listead{\new}{\series}
2333   \def\do##1{\ifcsstring{series}{##1}{\listadd{\new}{##1}}}
2334   \dolistloop{\@series}
2335   \xdef\@series{\new}
2336 }
2337 % \end{macrocode}
2338 % \end{macro}
2339 % \begin{macro}{\seriesatend}
2340 % And \cs{seriesatend} moves the series to the end of the list.
2341 % \begin{macrocode}
2342 \newcommand{\seriesatend}[1]{
2343   \edef\series{#1}
2344   \def\new{}
2345   \def\do##1{\ifcsstring{series}{##1}{\listadd{\new}{##1}}}
2346   \dolistloop{\@series}
2347   \listead{\new}{\series}
2348   \xdef\@series{\new}
2349 }
2350 % \end{macrocode}
2351 % \end{macro}
2352 % \subsection{Display}

```

²⁸Christophe 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.


```

2353 % \changes{v1.0}{2012/09/15}{New generic commands to customize footnote display.}
2354 % \subsubsection{Options}
2355 % \begin{macro}{\settoggle@series}
2356 % \changes{v1.1}{2012/09/25}{\cs{settoggle@series} switch the global value of the toggle, not only th
2357 % \cs{settoggle@series}\cs{series}{toggle}{value} is a generic command to switch one toggle for one s
2358 % \begin{macrocode}
2359 \newcommand{\settoggle@series}[3]{%
2360   \def\do##1{\global\settoggle{#2@##1}{#3}}
2361   \ifstrempy{#1}{%
2362     \dolistloop{\@series}%
2363   }%
2364   {%
2365     \docsvlist{#1}%
2366   }%
2367 }

```

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

```

2368 \newcommandx{\setcommand@series}[4][4]{%
2369   \def\do##1{
2370     \csgdef{#2@##1}{#3}
2371     \ifstrequal{#4}{reload}{\csuse{foot\csuse{series@display##1}}{##1}}{}}
2372   \ifstrempy{#1}{%
2373     \dolistloop{\@series}%
2374   }%
2375   {%
2376     \docsvlist{#1}%
2377   }%
2378 }%

```

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

```

2379 \newcommand{\newhookcommand@series}[1]{%
2380   \global\expandafter\newcommand\expandafter*\csname #1\endcsname[2][\csuse{setcommand@series}{##1}]
2381 }
2382 \newhookcommand@series{Xhangindent}
2383
2384 \newhookcommand@series{hangindentX}
2385
2386 \newhookcommand@series{hsizetwocol}
2387
2388 \newhookcommand@series{hsizethreecol}
2389
2390 \newhookcommand@series{hsizetwocolX}
2391
2392 \newhookcommand@series{hsizethreecolX}
2393
2394 \newhookcommand@series{Xnotenumfont}
2395

```

```

2396 \newhookcommand@series{notenumfontX}
2397
2398 \newhookcommand@series{Xendnotenumfont}
2399
2400 \newhookcommand@series{bhooknoteX}
2401
2402 \newhookcommand@series{bhookXnote}
2403
2404 \newhookcommand@series{bhookXendnote}
2405
2406 \newhookcommand@series{Xnotefontsize}
2407
2408 \newhookcommand@series{notefontsizeX}
2409
2410 \newhookcommand@series{Xendnotefontsize}
2411
2412 \newhookcommand@series{boxlinenum}
2413
2414 \newhookcommand@series{boxsymlinenum}
2415
2416 \newhookcommand@series{parafootsep}
2417
2418 \newhookcommand@series{symlinenum}
2419
2420 \newhookcommand@series{beforenumberinfootnote}
2421
2422 \newhookcommand@series{afternumberinfootnote}
2423
2424 \newhookcommand@series{beforesymlinenum}
2425
2426 \newhookcommand@series{aftersymlinenum}
2427
2428 \newhookcommand@series{inplaceofnumber}
2429
2430 \newhookcommand@series{lemmaseparator}
2431
2432 \newhookcommand@series{beforelemmaseparator}
2433
2434 \newhookcommand@series{afterlemmaseparator}
2435
2436 \newhookcommand@series{inplaceoflemmaseparator}
2437
2438 \newhookcommand@series{afternote}
2439
2440 \newhookcommand@series{txtbeforeXnotes}
2441

```

`\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)
2442 \newcommand{\newhookcommand@series@reload}[1]{%
2443   \global\expandafter\newcommand\expandafter*\csname #1\endcsname[2][1]{%
2444     \csuse{setcommand@series}{##1}{#1}{##2}[reload]
2445   }%
2446 }
2447 \newhookcommand@series@reload{beforeXnotes}
2448
2449 \newhookcommand@series@reload{beforenotesX}
2450
2451 \newhookcommand@series@reload{maxhnotesX}
2452
2453 \newhookcommand@series@reload{maxhXnotes}
2454 % \end{macrocode}
2455 % \end{macro}
2456 % \begin{macro}{\newhooktoggle@series}
2457 % \cs{newhooktoggle@series}\cs{command names} is a generic command to add new commands for new toggle
2458 %   \begin{macrocode}
2459 \newcommand{\newhooktoggle@series}[1]{%
2460   \global\expandafter\newcommandx\expandafter*\csname #1\endcsname[2][1,2={true},usedefault]{\settoggle
2461 }
2462 \newhooktoggle@series{numberonlyfirstinline}
2463 \newhooktoggle@series{numberonlyfirstintwolines}
2464 \newhooktoggle@series{nonumberinfootnote}
2465 \newhooktoggle@series{pstartinfootnote}
2466 \newhooktoggle@series{onlypstartinfootnote}
2467 \newhooktoggle@series{nonbreakableafternumber}

```

26.0.9 Old commands, kept for backward compatibility

The next commands are kept for ascendant compatibily, but should'nt be used anymore.

```

\notenumfont
\notefontsetup 2468 \newcommand*\notenumfont{\normalfont}
\ifledplinenum 2469 \newcommand*\notefontsetup{\footnotesize}
\symplinenum 2470 \newif\ifledplinenum
2471   \ledplinenumtrue
2472 \newcommand*\symplinenum{}

```

26.0.10 Hooks for a particular footnote

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

```
2473 \newtoggle{\nonum@}
```

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

```
2474 \newtoggle{\nosep@}
```

26.0.11 Alias

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

```
2475 \newcommand*{\nolemmaseparator}[1][1]{\lemmaseparator[#1]{}}
```

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

```
2476 \newskip\ipn@skip
2477 \newcommand*{\interparanoteglue}[1]{%
2478     {\notefontsetup\global\ipn@skip=#1 \relax}}
2479 \interparanoteglue{1em plus.4em minus.4em}
```

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

```
2480 \newcommand{\parafootftmsep}{}%
```

26.0.12 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.

```
2481 \newcommand{\printlinefootnote}[2]{%
2482     \def\extractline@##1|##2|##3|##4|##5|##6|##7|{##2}%
2483     \def\extractsubline@##1|##2|##3|##4|##5|##6|##7|{##3}%
2484     \def\extractendline@##1|##2|##3|##4|##5|##6|##7|{##5}%
2485     \def\extractendsubline@##1|##2|##3|##4|##5|##6|##7|{##6}%
2486     \iftoggle{numberonlyfirstintwolines@#2}{%
2487         \edef\lineinfo@{\extractline@ #1| - \extractsubline@ #1| - \extractendline@ #1| - \extractendsubline@ #1|}%
2488         }%
2489         {%
2490         \edef\lineinfo@{\extractline@ #1| - \extractsubline@ #1|}%
2491         }%
2492     \iftoggle{nonum@}{%Try if the line number must printed for this specific not (by default)}%
2493     \hspace{\csuse{inplaceofnumber@#2}}%
2494     }%
2495     {%
2496     }%
2497     \iftoggle{nonumberinfofootnote@#2}{%Try if the line number must printed (by default)}%
2498     {%
2499     \hspace{\csuse{inplaceofnumber@#2}}%
2500     }%
2501     {%
2502         {\iftoggle{numberonlyfirstinline@#2}% If for this series the line number must printed}
2503         {%
2504             \ifcsdef{prevline#2}%
2505             {%Be sure the \prevline exists.
2506             \ifcsequal{prevline#2}{\lineinfo@}%Try it
2507             }%
```

```

2508 \ifcempty{symlinenumber@#2}% Try if a symbol is define
2509 {%
2510 \hspace{\csuse{inplaceofnumber@#2}}%
2511 }%
2512 {\hspace{\csuse{beforenumberinfootnote@#2}}\csuse{Xnotenumfont@#2}}%
2513 \ifdimequal{\csuse{boxsymlinenumber@#2}}{0pt}%
2514 {\csuse{symlinenumber@#2}}%
2515 {\hbox to \csuse{boxsymlinenumber@#2}{\csuse{symlinenumber@#2}\hfill}}%
2516 \hspace{\csuse{aftersymlinenumber@#2}}}%
2517 }%
2518 {%
2519 \hspace{\csuse{beforenumberinfootnote@#2}}\csuse{Xnotenumfont@#2}}%
2520 \ifdimequal{\csuse{boxsymlinenumber@#2}}{0pt}%
2521 \iftoggle{pstartinfootnote@#2}{\printpstart}{}%
2522 \printlines#1}%
2523 {%
2524 \hbox to \csuse{boxsymlinenumber@#2}{%
2525 \iftoggle{pstartinfootnote@#2}{\printpstart}{}%
2526 \iftoggle{onlypstartinfootnote@#2}{\printlines#1}}%
2527 \hfill}%
2528 }%
2529 \iftoggle{nonbreakableafternumber@#2}{\nobreak}{\hspace{\csuse{afternumberinfootnote@#2}}}%
2530 }%
2531 }%
2532 {%
2533 \hspace{\csuse{beforenumberinfootnote@#2}}\csuse{Xnotenumfont@#2}}%
2534 \ifdimequal{\csuse{boxsymlinenumber@#2}}{0pt}%
2535 \iftoggle{pstartinfootnote@#2}{\printpstart}{}%
2536 \iftoggle{onlypstartinfootnote@#2}{\printlines#1}}%
2537 {%
2538 \hbox to \csuse{boxsymlinenumber@#2}{%
2539 \iftoggle{pstartinfootnote@#2}{\printpstart}{}%
2540 \iftoggle{onlypstartinfootnote@#2}{\printlines#1}}%
2541 \hfill}%
2542 }%
2543 \iftoggle{nonbreakableafternumber@#2}{\nobreak}{\hspace{\csuse{afternumberinfootnote@#2}}}%
2544 }%
2545 }%
2546 {%
2547 \hspace{\csuse{beforenumberinfootnote@#2}}\csuse{Xnotenumfont@#2}}%
2548 \ifdimequal{\csuse{boxsymlinenumber@#2}}{0pt}%
2549 \iftoggle{pstartinfootnote@#2}{\printpstart}{}%
2550 \iftoggle{onlypstartinfootnote@#2}{\printlines#1}}%
2551 }%
2552 {%
2553 \hbox to \csuse{boxsymlinenumber@#2}{%
2554 \iftoggle{pstartinfootnote@#2}{\printpstart}{}%
2555 \iftoggle{onlypstartinfootnote@#2}{\printlines#1}}%
2556 \hfill}%
2557 }%

```

```

2558             \iftoggle{nonbreakableafternumber@#2}{\nobreak}{\hspace{\csuse{afternumber:
2559             }%
2560             \csxdef{prevline#2}{\lineinfo@}%
2561             }%
2562     }%
2563 }%
2564 }%
2565 }

```

27 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.

```

2566 \countdef\pageno=0 \pageno=1
2567 \newcommand*{\advancepageno}{\ifnum\pageno<\z@ \global\advance\pageno\m@ne
2568   \else\global\advance\pageno\@ne\fi}
2569

```

The next portion is probably the trickiest part of moving from TeX to LaTeX. The original code is below, but we need something very different.

This is a new output routine, with changes to handle printing all our footnotes. Those changes have not been added directly, but are in macros that get called here: that should make it easier to see what would need to be taken over to a different output routine. We continue to use the `\pagebody`, `\makeheadline`, `\makefootline`, and `\dosupereject` macros of PLAIN TeX; for those macros, and the original version of `\output`, see *The TeXbook*, p. 364.

```

\output{\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 2570 \providecommand{\m@m@makecolfloats}{%
2571   \xdef\@freelist{\@freelist\@midlist}%
2572   \global \let \@midlist \@empty
2573   \@combinefloats}
```

```

2574 \providecommand{\m@m@makecoltext}{%
2575   \ifvbox\@kludgeins
2576     \@makespecialcolbox
2577   \else
2578     \setbox\@outputbox \vbox to\@colht {%
2579       \@texttop
2580       \dimen@ \dp\@outputbox
2581       \unvbox\@outputbox
2582       \vskip -\dimen@
2583       \@textbottom}%
2584   \fi}
2585 \providecommand{\m@m@makecolintro}{%
2586

```

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

```

2587 \gdef\l@d@makecol{%
2588   \l@ddofootinsert
2589   \m@m@makecolfloats
2590   \m@m@makecoltext
2591   \global \maxdepth \@maxdepth}
2592

```

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

```

2593 \AtBeginDocument{\@ifpackageloaded{footmisc}{\newif\ifFN@bottom}}

```

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

```

2594 \newcommand*{\l@ddofootinsert}{%
2595   %% \page@start
2596   \ifvoid\footins
2597     \setbox\@outputbox \box\@cclv
2598   \else
2599     \setbox\@outputbox \vbox {%
2600       \boxmaxdepth \@maxdepth
2601       \@tempdima\dp\@cclv
2602       \unvbox \@cclv
2603       \ifFN@bottom\vfill\fi\vskip \skip\footins%% If the option bottom of loadmisc pack
2604       \color@begingroup
2605         \normalcolor
2606         \footnoterule
2607         \unvbox \footins
2608       \color@endgroup
2609     }%
2610   \fi

```

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


```

2611 \l@ddoxtrafeet
2612 }
2613

```

`\doxtrafeet` `\doxtrafeet` is the code extending `\@makecol` to cater for the extra `eledmac` feet. We have two classes of extra footnotes. We order the footnote inserts so that the regular footnotes are first, then class 1 (familiar footnotes) and finally class 2 (critical footnotes).

```

2614 \newcommand*{\l@ddoxtrafeet}{%
2615 \doxtrafeeti
2616 \doxtrafeetii}
2617

```

`\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’.

```

2618 \newcommand*{\doxtrafeetii}{%
2619 \setbox\@outputbox \vbox{%
2620 \unvbox\@outputbox
2621 \@opxtrafeetii}}

```

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

```

2622 \newcommand*{\@opxtrafeetii}{%
2623 \def\do##1{\ifvoid\csuse{##1footins}\else\csuse{##1footstart}{##1}\csuse{##1footgroup}{##1}\fi}
2624 \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`.

```

2625 \newcommand*{\l@ddodoreinextrafeet}{%
2626 \doreinextrafeeti
2627 \doreinextrafeetii}
2628

```

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

```

2629 \newcommand*{\doreinextrafeetii}{%
2630 \def\do##1{\ifvoid\csuse{##1footins}\else\insert\csuse{##1footins}{\unvbox\csuse{##1footins}}\fi}
2631 \dolistloop{\@series}
2632 }
2633

```

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

```

2634 \gdef \l@d@reinserts{%
2635 \ifvoid\footins\else\insert\footins{\unvbox\footins}\fi
2636 \l@ddodoreinextrafeet
2637 \ifvbox\@kludgeins\insert\@kludgeins{\unvbox\@kludgeins}\fi
2638 }
2639

```

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.)

```
2640 \@ifclassloaded{memoir}{%
    memoir is loaded so we use memoir’s built in hooks.
2641   \g@addto@macro{\m@mdoextrafeet}{\l@ddextrafeet}%
2642   \g@addto@macro{\m@mdodoreinextrafeet}{\l@ddodoreinextrafeet}%
2643 }{%
    memoir has not been loaded, so redefine \@makecol and \@reinserts.
2644   \gdef\@makecol{\l@d@makecol}%
2645   \gdef\@reinserts{\l@d@reinserts}%
2646 }
2647

\addfootins   \addfootins is for backward compatibility, but should’nt be used anymore.
2648 \newcommand*{\addfootins}[1]{%
2649   \eledmac@warning{addfootins is deprecated, use newseries instead}
2650   \footnnormal{#1}
2651   \g@addto@macro{\@opxtrafeetii}{%
2652     \ifvoid\@nameuse{#1footins}\else
2653       \@nameuse{#1footstart{#1}}\@nameuse{#1footgroup}{#1}\fi}
2654   \g@addto@macro{\doreinextrafeetii}{%
2655     \ifvoid\@nameuse{#1footins}\else
2656       \insert\@nameuse{#1footins}{\unvbox\@nameuse{#1footins}}\fi}
2657   \g@addto@macro{\l@dedbeginmini}{%
2658     \expandafter\let\csname #1footnote\endcsname = \@nameuse{mp#1footnote}}
2659   \g@addto@macro{\l@dedendmini}{%
2660     \ifvoid\@nameuse{mp#1footins}\else\@nameuse{mpfootgroup#1{#1}}\fi}
2661 }
```

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.

```
2662 \newif\if@led@nofoot
2663 \newcommand*{\@led@extranofeet}{}
2664
2665 \@ifclassloaded{memoir}{%
    If the memoir class is loaded we hook into its modified \@doclearpage.
\@mem@extranofeet
2666 \g@addto@macro{\@mem@extranofeet}{%
2667   \def\do#1{\ifvoid\csuse{#1footins}\else\@mem@nofootfalse\fi%
2668   \ifvoid\csuse{footins#1}\else\@mem@nofootfalse\fi%
2669   }
2670   \dolistloop{\@series}%

```

```

2671 \@led@extranofeet}
2672 }{%

```

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

```

\@led@testifnofoot
\@doclearpage 2673 \newcommand*{\@led@testifnofoot}{%
2674 \@led@nofoottrue
2675 \ifvoid\footins\else\@led@nofootfalse\fi
2676 \def\do##1{\ifvoid\csuse{##1footins}\else\@led@nofootfalse\fi%
2677 \ifvoid\csuse{footins##1}\else\@led@nofootfalse\fi}%
2678 \dolistloop{\@series}
2679 \@led@extranofeet}
2680
2681 \renewcommand{\@doclearpage}{%
2682 \@led@testifnofoot
2683 \if@led@nofoot
2684 \setbox\@tempboxa\vsplit\@cclv to\z@ \unvbox\@tempboxa
2685 \setbox\@tempboxa\box\@cclv
2686 \xdef\@deferlist{\@toplist\@botlist\@deferlist}%
2687 \global \let \@toplist \@empty
2688 \global \let \@botlist \@empty
2689 \global \@colroom \@colht
2690 \ifx \@currlist\@empty
2691 \else
2692 \latexerr{Float(s) lost}\@ehb
2693 \global \let \@currlist \@empty
2694 \fi
2695 \@makefcolumn\@deferlist
2696 \@whiles\if@fcolmade \fi{\@opcol\@makefcolumn\@deferlist}%
2697 \if@twocolumn
2698 \if@firstcolumn
2699 \xdef\@dbldeferlist{\@dbltoplist\@dbldeferlist}%
2700 \global \let \@dbltoplist \@empty
2701 \global \@colht \textheight
2702 \begingroup
2703 \dblfloatplacement
2704 \makefcolumn\@dbldeferlist
2705 \@whiles\if@fcolmade \fi{\@outputpage
2706 \makefcolumn\@dbldeferlist}%
2707 \endgroup
2708 \else
2709 \vbox{}\clearpage
2710 \fi
2711 \fi
2712 \else
2713 \setbox\@cclv\vbox{\box\@cclv\vfil}%
2714 \led@makecol\@opcol
2715 \clearpage
2716 \fi}

```

2717 }
2718

28 Cross referencing

Peter Wilson have rewritten portions of the code in this section so that the LaTeX .aux file is used. This will also handle \included files.

Further, I have renamed some of the original EDMAC macros so that they do not clash with the LaTeX label/ref commands (EDMAC and LaTeX use very different mechanisms). In particular, the original EDMAC \label and \pageref have been renamed as \edlabel and \edpageref respectively.

You can mark a place in the text using a command of the form \edlabel{foo}, and later refer to it using the label foo by saying \edpageref{foo}, or \lineref{foo} or \sublineref{foo}. These reference commands will produce, respectively, the page, line and sub-line on which the \edlabel{foo} command occurred.

The reference macros warn you if a reference is made to an undefined label. If foo has been used as a label before, the \edlabel{foo} command will issue a complaint; subsequent \edpageref and \lineref commands will refer to the latest occurrence of \label{foo}.

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

2719 \list@create{\labelref@list}

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

2720 %% \newcommand*\zz@@@{000|000|000} % set three counters to zero in one go
2721 \newcommand*\zz@@@{000|000} % set two counters to zero in one go
2722

\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.²⁹

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 Billett³⁰ found that the original code could be off by several pages. This version, hopefully cures that, and also allows for non-arabic page numbering.

2723 \newcommand*\edlabel}[1]{\@bsphack
2724 \write\linenum@out{\string\@lab}%
2725 \ifx\labelref@list\empty

²⁹The remaining macros in this section were kindly revised by Wayne Sullivan, who substantially improved their efficiency and flexibility.

³⁰(jdb43@cam.ac.uk) via the ctt thread 'ledmac cross referencing', 25 August 2003.

```

2726 \xdef\label@refs{\zz@@}%
2727 \else
2728 \gl@p\labelref@list\to\label@refs
2729 \ifvmode
2730 \advancelabel@refs
2731 \fi
2732 \fi
2733 % \edef\next{\write\@aux{\string\l@dmake@labels\label@refs|{#1}}}%
2734 % \next}

```

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).

```

2735 \protected@write\@auxout{}%
2736 {\string\l@dmake@labels\space\thepage|\label@refs|{#1}}%
2737 \@esphack}
2738

```

`\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.

```

2739 \newcounter{line}%
2740 \newcounter{subline}%
2741 \newcommand{\advancelabel@refs}{%
2742 \setcounter{line}{\expandafter\labelrefsparseline\label@refs}%
2743 \stepcounter{line}%
2744 \ifsublines%
2745 \setcounter{subline}{\expandafter\labelrefsparsesubline\label@refs}%
2746 \stepcounter{subline}{1}%
2747 \def\label@refs{\theline|\thesubline}%
2748 \else%
2749 \def\label@refs{\theline|0}%
2750 \fi%
2751 }
2752 \def\labelrefsparseline#1|#2{#1}
2753 \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).

```

2754 \newcommand*{\l@dmake@labels}{%
2755 \def\l@dmake@labels#1|#2|#3|#4{%
2756 \expandafter\ifx\csname the@label#4\endcsname \relax\else

```

```

2757 \led@warn@DuplicateLabel{#4}%
2758 \fi
2759 \expandafter\gdef\csname the@label#4\endcsname{#1|#2|#3}%
2760 \ignorespaces}
2761

```

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.

```

2762 \AtBeginDocument{%
2763 \def\l@dmake@labels#1|#2|#3|#4{%
2764 }
2765

```

\@lab The \@lab command, which appears in the \linenum@out file, appends the current values of page, line and sub-line to the \labelref@list. These values are defined by the earlier \@page, \@l, and the \sub@on and \sub@off commands appearing in the \linenum@out file.

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.

```

2766 \newcommand*{\@lab}{\xright@appenditem
2767 {\linenumrep{\line@num}|}%
2768 \ifsublines@ \sublinenumrep{\subline@num}\else 0\fi}\to\labelref@list}
2769

```

\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.

```

2770 \newcommand*{\edpageref}[1]{\l@dref@undefined{#1}\l@dgetref@num{1}{#1}}
2771 \newcommand*{\xpageref}[1]{\l@dgetref@num{1}{#1}}
2772

```

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

```

\lineref 2773 \newcommand*{\lineref}[1]{\l@dref@undefined{#1}\l@dgetref@num{2}{#1}}
2774 \newcommand*{\xlineref}[1]{\l@dgetref@num{2}{#1}}
2775

```

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

```

\sublineref 2776 \newcommand*{\sublineref}[1]{\l@dref@undefined{#1}\l@dgetref@num{3}{#1}}
2777 \newcommand*{\xsublineref}[1]{\l@dgetref@num{3}{#1}}
2778

```

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.

```
2779 \newcommand*{\l@dref@undefined}[1]{%
2780   \expandafter\ifx\csname the@label#1\endcsname\relax
2781     \led@warn@RefUndefined{#1}%
2782   \fi}
2783
```

`\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.

```
2784 \newcommand*{\l@dgetref@num}[2]{%
2785   \expandafter
2786   \ifx\csname the@label#2\endcsname \relax
2787     000%
2788   \else
2789     \expandafter\expandafter\expandafter
2790     \l@dlabel@parse\csname the@label#2\endcsname| #1%
2791   \fi}
2792
```

`\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`.)

```
2793 \newcommand*{\l@dlabel@parse}{%
2794   \def\l@dlabel@parse#1|#2|#3|#4{%
2795     \ifcase #4\relax
2796     \or #1%
2797     \or #2%
2798     \or #3%
2799   \fi}
2800
```

`\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.

```
2801 \newcommand*{\xxref}[2]{%
2802   {\expandafter\ifx\csname the@label#1\endcsname
2803     \relax \expandafter\let\csname the@label#1\endcsname\zz@@@fi
2804     \expandafter\ifx\csname the@label#2\endcsname \relax
2805     \expandafter\let\csname the@label#2\endcsname\zz@@@fi
2806     \linenum{\csname the@label#1\endcsname|}%
2807     \csname the@label#2\endcsname}}
2808
```

`\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`.

```
2809 \newcommand*{\edmakelabel}[2]{\expandafter\xdef\csname the@label#1\endcsname{#2}}
2810
```

(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. 79 and 58), since `\xxref` makes a call to `\linenum` in order to do its work.)

29 Endnotes

```
\ld@end Endnotes of all varieties are saved up in a file, typically named <jobname>.end.
\ifld@end@ \ld@end is the output stream number for this file, and \ifld@end@ is a flag that's
\ld@end@true true when the file is open.
\ld@end@false 2811 \newwrite\ld@end
                2812 \newif\ifld@end@
```

```
\ld@end@open \ld@end@open and \ld@end@close are the macros that are used to open and close
\ld@end@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.
```

```
2813 \newcommand{\ld@end@open}[1]{\global\ld@end@true\immediate\openout\ld@end=#1\relax}
2814 \newcommand{\ld@end@close}{\global\ld@end@false\immediate\closeout\ld@end}
2815
```


`\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.

```
2816 \newcommand{\l@dend@stuff}{%
2817   \ifl@dend@relax\else
2818     \l@dend@open{\jobname.end}%
2819   \fi
2820   \immediate\write\l@d@end{\string\l@d@section{\the\section@num}}
2821 }
```

`\endprint` The `\endprint` here is nearly identical in its functioning to `\normalfootfmt`.

`\@gobblethree` The endnote file also contains `\l@d@section` commands, which supply the `\l@d@section` 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.

```
2822 \def\endprint#1#2#3#4{{\csuse{bhookXendnote@#4}\csuse{Xendnotefontsize@#4}{\csuse{Xendnotenumfont@#4}
2823   \enspace{\select@lemmfont#1|#2}\enskip#3\par}}
2824 \providecommand*\@gobblethree}[3]{}
2825
2826 \let\l@d@section=\@gobble
2827
```

`\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.

```
2828 \newcommand*\setprintendlines}[6]{%
2829   \l@d@pnumfalse \l@d@dashfalse
2830   \ifnum#4=#1 \else
2831     \l@d@pnumtrue
2832     \l@d@dashtrue
2833   \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.

```
2834   \ifl@d@pnum \l@d@elintrue \else \l@d@elinfalse \fi
2835   \ifnum#2=#5 \else
2836     \l@d@elintrue
2837     \l@d@dashtrue
2838   \fi
```

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

```
2839   \l@d@ssubfalse
2840   \ifnum#3=0 \else
```

```

2841      \l@d@ssubtrue
2842 \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.

```

2843 \l@d@eslfalse
2844 \ifnum#6=0 \else
2845     \ifnum#6=#3
2846         \ifl@d@elin \l@d@esltrue \else \l@d@eslfalse \fi
2847     \else
2848         \l@d@esltrue
2849         \l@d@dashtrue
2850     \fi
2851 \fi}

```

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

```

2852 \def\printendlines#1|#2|#3|#4|#5|#6|#7|{\begingroup
2853 \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).

```

2854 \printnpnum{#1} \linenumrep{#2}%
2855 \ifl@d@ssub \fullstop \sublinenumrep{#3}\fi
2856 \ifl@d@dash \endashchar\fi
2857 \ifl@d@pnum \printnpnum{#4}\fi
2858 \ifl@d@elin \linenumrep{#5}\fi
2859 \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumrep{#6}\fi
2860 \endgroup}
2861

```

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

```

2862 \newcommand*{\printnpnum}[1]{p.#1} }
2863

```

`\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.

```

2864 \newcommand*{\doendnotes}[1]{\l@dend@close
2865 \begingroup
2866 \makeatletter
2867 \expandafter\let\csname #1end\endcsname=\endprint
2868 \input\jobname.end
2869 \endgroup}

```

`\noendnotes` You can say `\noendnotes` before the first `\beginnumbering` in your file if you aren't going to be using 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.

```

2870 \newcommand*{\noendnotes}{\global\let\l@dend@stuff=\relax
2871 \global\chardef\l@dend=16 }

```

30 Side notes

Regular `\marginpars` do not work inside numbered text — they don't produce any note but do put an extra unnumbered blank line into the text.

`\l@dold@xympar` Changing `\@xympar` a little at least ensures that `\marginpars` in numbered text
`\@xympar` do not disturb the flow.

```
2872 \let\l@dold@xympar\@xympar
2873 \renewcommand{\@xympar}{%
2874   \ifnumberedpar@
2875     \led@warn@NoMarginpars
2876     \esphack
2877   \else
2878     \l@dold@xympar
2879   \fi}
2880
```

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).

```
2881 \newcount\sidenote@margin
2882 \newcommand*{\sidenotemargin}[1]{%
2883   \l@dgetsidenote@margin{#1}%
2884   \ifnum\@l@dttempcntb>\m@ne
2885     \global\sidenote@margin=\@l@dttempcntb
2886   \fi}%
2887 \newcommand*{\l@dgetsidenote@margin}[1]{%
2888   \def\@tempa{#1}\def\@tempb{left}%
2889   \ifx\@tempa\@tempb
2890     \@l@dttempcntb \z@
2891   \else
2892     \def\@tempb{right}%
2893     \ifx\@tempa\@tempb
2894       \@l@dttempcntb \@ne
2895     \else
2896       \def\@tempb{outer}%
2897       \ifx\@tempa\@tempb
2898         \@l@dttempcntb \tw@
2899       \else
2900         \def\@tempb{inner}%
2901         \ifx\@tempa\@tempb
2902           \@l@dttempcntb \thr@@
2903         \else
2904           \led@warn@BadSidenotemargin
2905           \@l@dttempcntb \m@ne
2906         \fi
2907       \fi

```

```

2908     \fi
2909   \fi}
2910 \sidenotemargin{right}
2911

\l@dlp@rbox We need two boxes to store sidenote texts.
\l@drp@rbox 2912 \newbox\l@dlp@rbox
2913 \newbox\l@drp@rbox
2914

\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 2915 \newdimen\ledlsnotewidth \ledlsnotewidth=\marginparwidth
\ledrsnotesep 2916 \newdimen\ledrsnotewidth \ledrsnotewidth=\marginparwidth
\ledlsnotefontsetup 2917 \newdimen\ledlsnotesep \ledlsnotesep=\linenumsep
\ledrsnotefontsetup 2918 \newdimen\ledrsnotesep \ledrsnotesep=\linenumsep
2919 \newcommand*{\ledlsnotefontsetup}{\raggedleft\footnotesize}
2920 \newcommand*{\ledrsnotefontsetup}{\raggedright\footnotesize}
2921

\ledleftnote \ledleftnote{<text>} and \ledrightnote{<text>} are the user commands for
\ledrightnote left and right sidenotes. \ledsidenote{<text>} is the command for a moveable
\ledsidenote sidenote.

2922 \newcommand*{\ledleftnote}[1]{\edtext{}{\l@dlsnote{#1}}}
2923 \newcommand*{\ledrightnote}[1]{\edtext{}{\l@drsnote{#1}}}
2924 \newcommand*{\ledsidenote}[1]{\edtext{}{\l@dcsnote{#1}}}
2925
2926

\l@dlsnote The ‘footnotes’ for left, right, and moveable sidenotes. The whole scheme is rem-
\l@drsnote iniscent of the critical footnotes code.
\l@dcsnote 2927 \newif\ifrighnoteup
2928 \righnoteuptrue
2929 \newcommand*{\l@dlsnote}[1]{%
2930   \begingroup%
2931   \newcommand{\content}{#1}%
2932   \ifnumberedpar@
2933     \xright@appenditem{\noexpand\l@dlsnote{\csexpandonce{content}}}%
2934     \to\inserts@list
2935     \global\advance\insert@count \@ne
2936   \fi\ignorespaces\endgroup}
2937 \newcommand*{\l@drsnote}[1]{%
2938   \begingroup%
2939   \newcommand{\content}{#1}%
2940   \ifnumberedpar@
2941     \xright@appenditem{\noexpand\l@drsnote{\csexpandonce{content}}}%
2942     \to\inserts@list
2943     \global\advance\insert@count \@ne
2944   \fi\ignorespaces\endgroup}

```

```

2945 \newcommand*{\l@dcsnote}[1]{\begingroup%
2946   \newcommand{\content}{#1}%
2947   \ifnumberedpar@
2948     \xright@appenditem{\noexpand\l@dcsnote{\csexpandonce{content}}}%
2949     \to\inserts@list
2950     \global\advance\insert@count \@ne
2951   \fi\ignorespaces\endgroup}
2952

```

`\vl@dlsnote` Put the left/right text into boxes, but just save the moveable text. `\l@dcsnote`text

`\vl@drsnote` is a etoolbox list (comma separated)

```

\vl@dcsnote 2953 \newcommand*{\vl@dlsnote}[1]{\setl@dlp@rbox{#1}}
2954 \newcommand*{\vl@drsnote}[1]{\setl@drp@rbox{#1}}
2955 \newcommand*{\vl@dcsnote}[1]{\listgadd{\l@dcsnote}{#1}}
2956

```

`\setl@dlp@rbox` `\setl@dlp@rbox{<lednums>}{<tag>}{<text>}` puts `<text>` into the `\l@dlp@rbox` box.

`\setl@drp@rbox` And similarly for the right side box. It is these boxes that finally get displayed in the margins.

```

2957 \newcommand*{\setl@dlp@rbox}[1]{%
2958   {\parindent\z@\hspace=\ledlsnotewidth\ledlsnotefontsetup
2959     \global\setbox\l@dlp@rbox
2960     \ifleftnoteup
2961       =\vbox to\z@{\vss #1}%
2962     \else
2963       =\vbox to 0.70\baselineskip{\strut#1\vss}%
2964     \fi}}
2965 \newcommand*{\setl@drp@rbox}[1]{%
2966   {\parindent\z@\hspace=\ledrsnotewidth\ledrsnotefontsetup
2967     \global\setbox\l@drp@rbox
2968     \ifrightnoteup
2969       =\vbox to\z@{\vss#1}%
2970     \else
2971       =\vbox to0.7\baselineskip{\strut#1\vss}%
2972     \fi}}
2973 \newif\ifleftnoteup
2974 \leftnoteuptrue

```

`\sidenotesep` This macro is used to separate sidenotes of the same line.

```

2975 \newcommand{\sidenotesep}{, }
2976 % \end{macrocode}
2977 % \end{macro}
2978 % \begin{macro}{\affixside@note}
2979 % This macro puts any moveable sidenote text into the left or right sidenote
2980 % box, depending on which margin it is meant to go in. It's a very much
2981 % stripped down version of \cs{affixlin@num}.
2982 %
2983 % Before do it, we concatenate all moveable sidenotes of the line, using \cs{sidenotesep} as separator
2984 % It's the result that we put on the sidenote.

```

```

2985 % \changes{v1.4.1}{2012/11/16}{Remove spurious spaces.}
2986 % \begin{macrocode}
2987 \newcommand*{\affixside@note}{%
2988   \def\sidenotecontent@{}%
2989   \numdef\itemcount@{0}%
2990   \def\do##1{%
2991     \ifnumequal{\itemcount@}{0}%
2992       {%
2993         \appto\sidenotecontent@{##1}}% Not print not separator before the 1st note
2994         {\appto\sidenotecontent@{\sidenotesep ##1}}%
2995       }%
2996       \numdef\itemcount@{\itemcount@+1}%
2997   }%
2998   \dolistloop{\l@dcstotetext}%
2999   \ifnumgreater{\itemcount@}{1}{\eledmac@warning{\itemcount@\space sidenotes on line \tl

```

And now, the main part of the macro

```

3000 \gdef\@templ@d{%
3001 \ifx\@templ@d\l@dcstotetext \else%
3002 \if@twocolumn%
3003   \if@firstcolumn%
3004     \setl@dlp@rbox{##1}{\sidenotecontent@}%
3005   \else%
3006     \setl@drp@rbox{\sidenotecontent@}%
3007   \fi%
3008 \else%
3009   \l@dttempcntb=\sidenote@margin%
3010   \ifnum\l@dttempcntb>\@ne%
3011     \advance\l@dttempcntb by\page@num%
3012   \fi%
3013   \ifodd\l@dttempcntb%
3014     \setl@drp@rbox{\sidenotecontent@}%
3015   \else%
3016     \setl@dlp@rbox{\sidenotecontent@}%
3017   \fi%
3018 \fi%
3019 \fi}

```

31 Minipages and such

We can put footnotes into minipages. The preparatory code has been set up earlier, all that remains is to ensure that it is available inside a minipage box. This requires some alteration to the kernel code, specifically the `\@iiminipage` and `\endminipage` macros. We'll arrange this so that additional series can be easily added.

```

\l@dfeetbeginmini These will be the hooks in \@iiminipage and \endminipage They can be extended
\l@dfeetendmini   to handle other things if necessary.
3020 \newcommand*{\l@dfeetbeginmini}{\l@ddebbeginmini\l@dfambeginmini}

```

```

3021 \newcommand*{\l@dfetendmini}{\l@dedendmini\l@dfamendmini}
3022

```

`\l@dedbeginmini` These handle the initiation and closure of critical footnotes in a minipage environment.
`\l@dedendmini`

```

3023 \newcommand*{\l@dedbeginmini}{%
3024   \def\do##1{\csletcs{v##1footnote}{mpv##1footnote}}%
3025   \dolistloop{\@series}%
3026 }
3027 \newcommand*{\l@dedendmini}{%
3028   \ifl@dpairing
3029     \ifl@Rcol
3030       \flush@notesR
3031     \else
3032       \flush@notes
3033     \fi
3034   \fi
3035   \def\do##1{\ifvoid\csuse{mp##1footins}\else\csuse{mp##1footgroup}{##1}\fi}%
3036   \dolistloop{\@series}%
3037 }
3038

```

`\l@dfambeginmini` These handle the initiation and closure of familiar footnotes in a minipage environment.
`\l@dfamendmini`

```

3039 \newcommand*{\l@dfambeginmini}{%
3040   \def\do##1{\csletcs{vfootnote##1}{mpvfootnote##1}}%
3041   \dolistloop{\@series}}
3042 \newcommand*{\l@dfamendmini}{%
3043   \def\do##1{\ifvoid\csuse{mpfootins##1}\else\csuse{mpfootgroup##1}{##1}\fi}%
3044   \dolistloop{\@series}}

```

`\@iiiminipage` This is our extended form of the kernel `\@iiiminipage` defined in `ltboxes.dtx`.

```

3045 \def\@iiiminipage#1#2[#3]#4{%
3046   \leavevmode
3047   \@pboxswfalse
3048   \setlength\@tempdima{#4}%
3049   \def\@mpargs{#1}#2[#3]{#4}}%
3050   \setbox\@tempboxa\vbox\bgroup
3051     \color@begingroup
3052     \hsize\@tempdima
3053     \textwidth\hsize \columnwidth\hsize
3054     \@parboxrestore
3055     \def\@mpfn{mpfootnote}\def\thempfn{\thempfootnote}\c@mpfootnote\z@
3056     \let\@footnotetext\@mpfootnotetext

```

The next line is our addition to the original.

```

3057     \l@dfetbeginmini% added
3058     \let\@listdepth\@mplistdepth \@mplistdepth\z@
3059     \@minipagerestore

```

```

3060     \@setminipage}
3061

```

`\endminipage` This is our extended form of the kernel `\endminipage` defined in `ltboxes.dtx`.

```

3062 \def\endminipage{%
3063   \par
3064   \unskip
3065   \ifvoid\@mpfootins\else
3066     \l@dunboxmpfoot
3067   \fi

```

The next line is our addition to the original.

```

3068   \l@dfeetendmini%           added
3069   \@minipagefalse
3070   \color@endgroup
3071   \egroup
3072   \expandafter\@iiiparbox\@mpargs{\unvbox\@tempboxa}}
3073

```

`\l@dunboxmpfoot`

```

3074 \newcommand*\l@dunboxmpfoot{%
3075   \vskip\skip\@mpfootins
3076   \normalcolor
3077   \footnoterule
3078   \unvbox\@mpfootins}
3079

```

`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.

```

3080 \newif\if@ledgroup
3081 \newenvironment{ledgroup}{%
3082   \@ledgrouptrue\def\@mpfn{mpfootnote}\def\thempfn{\thempfootnote}\c@mpfootnote\z@
3083   \let\@footnotetext\@mpfootnotetext
3084   \l@dfeetbeginmini%
3085 }{%
3086   \par
3087   \unskip
3088   \ifvoid\@mpfootins\else
3089     \l@dunboxmpfoot
3090   \fi
3091   \l@dfeetendmini%
3092   \@ledgroupfalse%
3093 }
3094

```

`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.

```

3095 \newenvironment{ledgroupsize}[2][1]{%
    Set the various text measures.
3096   \hsize #2\relax
3097   %% \textwidth #2\relax
3098   %% \columnwidth #2\relax
    Initialize fills for centering.
3099   \let\ledllfill\hfil
3100   \let\ledrlfill\hfil
3101   \def\@tempa{#1}\def\@tempb{1}%
    Left adjusted numbered lines
3102       \ifx\@tempa\@tempb
3103       \let\ledllfill\relax
3104   \else
3105       \def\@tempb{r}%
3106       \ifx\@tempa\@tempb
    Right adjusted numbered lines
3107       \let\ledrlfill\relax
3108       \fi
3109   \fi
    Set up the footnoting.
3110   \@ledgrouptrue%
3111   \def\@mpfn{mpfootnote}\def\thempfn{\thempfootnote}\c@mpfootnote\z@
3112   \let\@footnotetext\@mpfootnotetext
3113   \l@dfetbeginmini%
3114 }{%
3115   \par
3116   \unskip
3117   \ifvoid\@mpfootins\else
3118       \l@dunboxmpfoot
3119   \fi
3120   \l@dfetendmini%
3121   \@ledgroupfalse%
3122 }
3123

```

32 Indexing

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

```

\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   3124 \newcommand{\pagelinesep}{-}
              3125 \newcommand{\edindexlab}{\&\&}

```

```

3126 \newcounter{labidx}
3127 \setcounter{labidx}{0}
3128

\doedindexlabel This macro sets an \edlabel.
3129 \newcommand{\doedindexlabel}{\stepcounter{labidx}%
3130 \edlabel{\edindexlab\thelabidx}}
3131

\thepageline This macro makes up the page/line number combo from the label/ref.
3132 \newcommand{\thepageline}{%
3133 \thepage\pagelinesep\lineref{\edindexlab\thelabidx}}

\thestartpageline These macros make up the page/line start/end number when the \edindex com-
\theendpageline mand is called in critical notes.
3134 \newcommand{\thestartpageline}{\l@dparsedstartpage\pagelinesep\l@dparsedstartline}
3135 \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.
3136 \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.
3137 \newcommand{\prepare@edindex@fornote}[1]{%
3138 \l@dp@rsefootspec#1|%
3139 \@edindex@fornote@true
3140 }

\get@index@command This macro is used to analyse if a text to be indexed has a command after a |.
3141 \def\get@index@command#1|#2{\gdef\@index@command{#2}\gdef\@index@txt{#1}}

\ledinnote These macros are used to specify that an index reference points to a note.
\ledinnotehyperpage 3142 \newcommand{\ledinnote}[2]{\csuse{#1}{#2}\emph{n}}
3143 \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.
3144 \ifclassloaded{memoir}{%
memoir is being used.

\makeindex Need to add the definition of \edindex to \makeindex, and initialise \edindex
\edindex to do nothing. In this case \edindex has an optional argument. We use the hook
provided in memoir v1.61.
3145 \g@addto@macro{\makememindexhook}{%
3146 \def\edindex{\@bsphack%
3147 \ifnextchar [{\l@d@index}{\l@d@index[\jobname]}}
3148 \newcommand{\edindex}[2][\jobname]{\@bsphack\@esphack}

```

`\l@d@index` `\l@d@index[file]` is the first stage of `\edindex`, handling the `idx` file. This is a virtually a verbatim copy of memoir's `\@index`, the change being calling `\l@dwrindexm@m` instead of `\@wrindexm@m`.

```

3149 \def\l@d@index[#1]{%
3150   \@ifundefined{#1idxfile}%
3151   {\ifreportnoidxfile
3152     \led@warn@NoIndexFile{#1}%
3153     \fi
3154     \begingroup
3155     \@sanitize
3156     \@nowrindex}%
3157   {\def\@idxfile{#1}%
3158     \doedindexlabel
3159     \begingroup
3160     \@sanitize
3161     \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`.

```

3162 \newcommand{\l@d@wrindexm@m}[1]{\l@d@@wrindexhyp#1||\}
3163 \def\l@d@@wrindexhyp#1|#2|#3\{\%
3164   \ifshowindexmark\@showidx{#1}\fi
3165   \ifx\#2\%
3166     \if@edindex@fornote@%
3167       \protected@write\@auxout{%
3168         {\string\@@wrindexm@m{\@idxfile}{#1}(\ledinnotehyperpage){\thestartpageline}}%
3169       \protected@write\@auxout{%
3170         {\string\@@wrindexm@m{\@idxfile}{#1})ledinnotehyperpage}{\theendpageline}}%
3171       \else%
3172       \protected@write\@auxout{%
3173         {\string\@@wrindexm@m{\@idxfile}{#1}hyperpage}{\thepageline}}%
3174       \fi%
3175     \else
3176       \def\Hy@temp@A{#2}%
3177       \ifx\Hy@temp@A\HyInd@ParenLeft
3178         \if@edindex@fornote@%
3179           \protected@write\@auxout{%
3180             {\string\@@wrindexm@m{\@idxfile}{#1}(\ledinnotehyperpage{#2}){\thestartpageline}}%
3181           \protected@write\@auxout{%
3182             {\string\@@wrindexm@m{\@idxfile}{#1})ledinnotehyperpage{#2}}{\theendpageline}}%
3183           \else%
3184           \protected@write\@auxout{%
3185             {\string\@@wrindexm@m{\@idxfile}{#1}#2hyperpage}{\thepageline}}%
3186           \fi%
3187         \else
3188           \if@edindex@fornote@%
3189             \protected@write\@auxout{%
3190               {\string\@@wrindexm@m{\@idxfile}{#1}(\ledinnote{#2}){\thestartpageline}}%

```

```

3191         \protected@write\@auxout{}%
3192         {\string\@wrindexm@m{\@idxfile}{#1|)}ledinnote{#2}}{\theendpageline}}%
3193     \else%
3194         \protected@write\@auxout{}%
3195         {\string\@wrindexm@m{\@idxfile}{#1|#2}}{\thepageline}}%
3196     \fi%
3197 \fi
3198 \fi
3199 \endgroup
3200 \@esphack}

```

That finishes the memoir-specific code.

```
3201 }{%
```

memoir is not being used, which makes life somewhat simpler.

`\makeindex` Need to add the definition of `\edindex` to `\makeindex`, and initialise `\edindex` to
`\edindex` do nothing.

```

3202 \pretocmd{\makeindex}{%
3203     \def\edindex{\@bsphack
3204     \doedindexlabel
3205     \begingroup
3206     \@sanitize
3207     \@wredindex}}{}{}
3208 \newcommand{\edindex}[1]{\@bsphack\@esphack}

```

`\@wredindex` Write the index information to the `idx` file.

```

3209 \newcommand{\@wredindex}[2][1=\jobname,usedefault]{%#1 = the index name, #2 = the text
3210     \ifl@makeidx%
3211         \if@edindex@fornote%
3212             \IfSubStr[1]{#2}{|}{\get@index@command#2}{\get@index@command#2}}%
3213             \expandafter\imki@wrindexentry{#1}{\@index@txt|ledinnote{\@index@command}}{\thestartpageline}
3214             \expandafter\imki@wrindexentry{#1}{\@index@txt|ledinnote{\@index@command}}{\theendpageline}
3215         \else%
3216             \imki@wrindexentry{#1}{#2}{\thepageline}%
3217         \fi%
3218     \else%
3219         \if@edindex@fornote%
3220             \IfSubStr[1]{#2}{|}{\get@index@command#2}{\get@index@command#2}}%
3221             \expandafter\protected@write\@indexfile{}%
3222             {\string\indexentry{\@index@txt|ledinnote{\@index@command}}{\thestartpageline}
3223             }%
3224             \expandafter\protected@write\@indexfile{}%
3225             {\string\indexentry{\@index@txt|ledinnote{\@index@command}}{\theendpageline}
3226             }%
3227         \else%
3228             \protected@write\@indexfile{}%
3229             {\string\indexentry{#2}{\thepageline}
3230             }%
3231         \fi%

```

```

3232     \fi%
3233 \endgroup
3234 \@esphack}

```

That finishes the non-memoir index code.

```

3235 }
3236

```

`\l@d@@wrindexhyp` If the `hyperref` package is not loaded, it doesn't make sense to clutter up the index with hyperreffing things.

```

3237 \AtBeginDocument{\@ifpackageloaded{hyperref}{}{%
3238   \def\l@d@@wrindexhyp#1||\{\%
3239     \ifshowindexmark\@showidx{#1}\fi
3240     \IfSubStr[1]{#1}{|}{\get@index@command#1}{\get@index@command#1|}%
3241     \if@edindex@fornote{%
3242       \protected@write\@auxout{%
3243         {\string\@wrindexm@{\@idxfile}{\@index@txt|(ledinnote{\@index@command})}{\thestartpageli
3244         \protected@write\@auxout{%
3245           {\string\@wrindexm@{\@idxfile}{\@index@txt|)ledinnote{\@index@command})}{\theendpageline
3246       }else%
3247       \protected@write\@auxout{%
3248         {\string\@wrindexm@{\@idxfile}{#1}{\thepageline}}%
3249     \fi%
3250 \endgroup
3251 \@esphack}}}}
3252
3253

```

33 Macro as environment

The following is borrowed, and renamed, from the `amsmath` package. See also the CTT thread ‘`eq and amstex`’, 1995/08/31, started by Keith Reckdahl and ended definitively by David M. Jones.

Several of the `[math]` macros scan their body twice. This means we must collect all text in the body of an environment form before calling the macro.

`\@emptytoks` This is actually defined in the `amsgen` package.

```

3254 \newtoks\@emptytoks
3255

```

The rest is from `amsmath`.

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

```

3256 \newtoks\l@denvbody
3257

```

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

```

3258 \newcommand{\addtol@denvbody}[1]{%

```

```
3259 \global\l@denbody\expandafter{\the\l@denbody#1}}
3260
```

`\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`.

```
3261 \newcommand{\l@dcollect@body}[1]{%
3262   \l@denbody{\expandafter#1\expandafter{\the\l@denbody}}%
3263   \edef\processl@denbody{\the\l@denbody\noexpand\end{\@currenvir}}%
3264   \l@denbody\@emptytoks \def\l@dbegin@stack{b}%
3265   \begingroup
3266     \expandafter\let\csname\@currenvir\endcsname\l@dcollect@@body
3267     \edef\processl@denbody{\expandafter\noexpand\csname\@currenvir\endcsname}%
3268     \processl@denbody}
3269
```

`\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.

```
3270 \def\l@dpush@begins#1\begin#2{%
3271   \ifx\end#2\else b\expandafter\l@dpush@begins\fi}
3272
```

`\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.

```
3273 \def\l@dcollect@@body#1\end#2{%
3274   \edef\l@dbegin@stack{\l@dpush@begins#1\begin\end
3275     \expandafter\@gobble\l@dbegin@stack}%
3276   \ifx\@empty\l@dbegin@stack
3277     \endgroup
3278     \@checkend{#2}%
3279     \addtol@denbody{#1}%
3280   \else
3281     \addtol@denbody{#1\end{#2}}%
3282   \fi
3283   \processl@denbody % A little tricky! Note the grouping
3284 }
3285
```

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>

```

34 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 3286 \newcommand*{\hangingsymbol}{%
3287 \newif\ifinstanza
3288 \instanzafalse
```

```
\inserthangingsymbol  The boolean \inserthangingsymbol is set to TRUE when \@lock is greater
\inserthangingsymbol  than 1, i.e. when we are not in the first line of a verse. The switch of
\inserthangingsymbol  \inserthangingsymbol is made in \do@line before the printing of line but
\inserthangingsymbol  after the line number calculation.
```

```
3289 \newif\ifinserthangingsymbol
3290 \newcommand{\inserthangingsymbol}{%
3291 \ifinserthangingsymbol%
3292   \ifinstanza%
3293     \hangingsymbol%
3294   \fi%
3295 \fi%
3296 }
```

```
\ampersand  Within a stanza the \& macro is going to be usurped. We need an alias in case
\ampersand  an & needs to be typeset in a stanza. Define it rather than letting it in case some
\ampersand  other package has already defined it.
```

```
3297 \newcommand*{\ampersand}{\char'\&}
3298
```

```
\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.
```

```
3299 \chardef\body=\catcode'\@
3300 \catcode'\@=11
3301 \chardef\next=\catcode'\&
3302 \catcode'\&=\active
3303
```

A count register is allocated for counting lines in a stanza; also allocated is a dimension register which is used to specify the base value for line indentation; all stanza indentations are multiples of this value. The default value of `\stanzaindentbase` is 20pt.

```
3304 \newcount\stanza@count
3305 \newlength{\stanzaindentbase}
3306 \setlength{\stanzaindentbase}{20pt}
3307
```

```
\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-
\setstanzavalues bers, some list macros are defined. These take numerical values in a list separated
\setstanzavalues by commas and assign the values to special control sequences using \mathchardef.
```

Though this does limit the range from 0 to 32767, it should suffice for most applications, including *penalties*, which will be discussed below.

```

3308 \def\strip@szacnt#1,#2|{\def\@tempb{#1}\def\@tempa{#2|}}
3309 \newcommand*{\setstanzavalues}[2]{\def\@tempa{#2,,|}%
3310     \stanza@count\z@
3311     \def\next{\expandafter\strip@szacnt\@tempa
3312         \ifx\@tempb\empty\let\next\relax\else
3313         \expandafter\mathchardef\csname #1@\number\stanza@count
3314         \@endcsname\@tempb\relax
3315         \advance\stanza@count\@ne\fi\next}%
3316     \next}
3317

```

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

```

3318 \newcommand*{\setstanzaindents}[1]{\setstanzavalues{sza}{#1}}
3319 \newcommand*{\setstanzapenalties}[1]{\setstanzavalues{szp}{#1}}
3320
3321 \newcounter{stanzaindentsrepetition}
3322 \newcount\stanza@modulo
3323
3324 \newcommand*{\managestanza@modulo}[0]{
3325     \advance\stanza@modulo\@ne
3326     \ifnum\stanza@modulo>\value{stanzaindentsrepetition}
3327         \stanza@modulo\@ne
3328     \fi
3329 }

```

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

`\stanza@hang` `\stanza@hang` sets the hanging indentation to be used if the stanza line requires more than one print line. If it is known that each stanza line will fit on one print line, it is advisable to set the hanging indentation to zero. `\sza@penalty` places the specified penalty following each stanza line. By default, this facility is turned off so that no penalty is included. However, the user may initiate these penalties to indicate good and bad places in the stanza for page breaking.

```

3330 \def\stanza@line{
3331     \ifnum\value{stanzaindentsrepetition}=0
3332         \parindent=\csname sza@\number\stanza@count
3333         \@endcsname\stanzaindentbase
3334     \else
3335         \parindent=\csname sza@\number\stanza@modulo

```

```

3336             @\endcsname\stanzaindentbase
3337         \managestanza@modulo
3338     \fi
3339     \pstart\stanza@hang\ignorespaces}
3340 \xdef\stanza@hang{\noexpand\leavevmode\noexpand\startlock
3341             \hangindent\expandafter
3342             \noexpand\csname sza@0@\endcsname\stanzaindentbase
3343             \hangafter\@ne}
3344 \def\sza@penalty{\count@\csname szp@\number\stanza@count @\endcsname
3345             \ifnum\count@>\@M\advance\count@-\@M\penalty-\else
3346             \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 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 macro **\falseverse** can be used to add stanza not numbered and with no impact on the next indent. 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.

```

3347 \let\startstanzahook\relax
3348 \let\endstanzaextra\relax
3349 \xdef\stanza{\noexpand\instanzatrue\expandafter
3350             \begingroup\startstanzahook%
3351             \catcode'\&\active\global\stanza@count\@ne\stanza@modulo\@ne
3352             \noexpand\ifnum\expandafter\noexpand
3353             \csname sza@0@\endcsname=\z@\let\noexpand\stanza@hang\relax
3354             \let\noexpand\endlock\relax\noexpand\else\interlinepenalty
3355             \@M\rightskip\z@ plus 1fil\relax\noexpand\fi\noexpand\ifnum
3356             \expandafter\noexpand\csname szp@0@\endcsname=\z@
3357             \let\noexpand\sza@penalty\relax\noexpand\fi%
3358 \def\noexpand\falseverse{%
3359     \global\advance\stanza@modulo-\@ne%
3360     \global\advance\stanza@count-\@ne%
3361     \relax\noexpand&\leavevmode\skipnumbering}
3362     \def\noexpand&{%
3363         \noexpand\endlock\noexpand\pend\noexpand\sza@penalty\global%
3364         \advance\stanza@count\@ne\noexpand\stanza@line}%
3365     \def\noexpand%
3366     &{\noexpand\endlock\noexpand\pend\endgroup\noexpand\instanzafalse\expandafter\endstanzaextra
3367     \noexpand\stanza@line}
3368

```

`\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`.

```
3369 \newcommand*{\flagstanza}[2][\stanzaindentbase]{%
3370   \hskip -#1\llap{#2}\hskip #1\ignorespaces}
3371
```

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.

```
3372 \catcode'\&=\next
3373 \catcode'\@=\body
3374 %% \let\ampersand=\&
3375 \setstanzavalues{szp}{0}
3376
```

35 Arrays and tables

This is based on the work by Herbert Breger in developing `tabmac.tex`.

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

The original `tabmac.tex` file was void of comments or any explanatory text other than the above notice. The algorithm is Breger's. I have made some cosmetic changes to the original code and reimplemented some things so they are more LaTeX-like. All the commentary is mine, as are any mistakes or errors.

`\l@dtabnoexpands` An extended and modified version of the original additional no expansions..

```

3377 \newcommand*{\l@dtabnoexpands}{%
3378   \let\rtab=0%
3379   \let\ctab=0%
3380   \let\ltab=0%
3381   \let\rtabtext=0%
3382   \let\ltabtext=0%
3383   \let\ctabtext=0%
3384   \let\edbeforetab=0%
3385   \let\edaftertab=0%
3386   \let\edatab=0%
3387   \let\edatabell=0%
3388   \let\edatleft=0%
3389   \let\edatright=0%
3390   \let\edvertline=0%
3391   \let\edvertdots=0%
3392   \let\edrowfill=0%
3393 }
3394
```

`\l@dampcount` `\l@dampcount` is a counter for the & column dividers and `\l@dcolcount` is a
`\l@dcolcount` counter for the columns. These were `\Undcount` and `\stellencount` respectively.

```

3395 \newcount\l@dampcount
3396 \l@dampcount=1\relax
3397 \newcount\l@dcolcount
3398 \l@dcolcount=0\relax
3399
```

`\hilfsbox` Some (temporary) helper items.

```

\hilfsskip 3400 \newbox\hilfsbox
\Hilfsbox 3401 \newskip\hilfsskip
\hilfscount 3402 \newbox\Hilfsbox
3403 \newcount\hilfscount
3404
```

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).

```

3405 \newdimen\dcoli
3406 \newdimen\dcolii
3407 \newdimen\dcoliii
3408 \newdimen\dcoliv
3409 \newdimen\dcolv
3410 \newdimen\dcolvi
3411 \newdimen\dcolvii
3412 \newdimen\dcolviii
3413 \newdimen\dcolix
3414 \newdimen\dcolx
3415 \newdimen\dcolxi
```

```

3416 \newdimen\dcplxii
3417 \newdimen\dcplxiii
3418 \newdimen\dcplxiv
3419 \newdimen\dcpl xv
3420 \newdimen\dcplxvi
3421 \newdimen\dcplxvii
3422 \newdimen\dcplxviii
3423 \newdimen\dcplxix
3424 \newdimen\dcplx x
3425 \newdimen\dcplx xi
3426 \newdimen\dcplx xii
3427 \newdimen\dcplx xiii
3428 \newdimen\dcplx xiv
3429 \newdimen\dcplx xv
3430 \newdimen\dcplx xvi
3431 \newdimen\dcplx xvii
3432 \newdimen\dcplx xviii
3433 \newdimen\dcplx xix
3434 \newdimen\dcplx x x
3435 \newdimen\dcplerr % added for error handling
3436

```

`\l@dcplwidth` This is a cunning way of storing the column widths indexed by the column number
`\l@dcplcount`, like an array. (was `\Dimenzuordnung`)

```

3437 \newcommand{\l@dcplwidth}{\ifcase \the\l@dcplcount \dcoli %???
3438 \or \dcoli \or \dcolii \or \dcoliii
3439 \or \dcoliv \or \dcolv \or \dcolvi
3440 \or \dcolvii \or \dcolviii \or \dcolix \or \dcolx
3441 \or \dcolxi \or \dcolxii \or \dcolxiii
3442 \or \dcolxiv \or \dcolxv \or \dcolxvi
3443 \or \dcolxvii \or \dcolxviii \or \dcolxix \or \dcolxx
3444 \or \dcolxxi \or \dcolxxii \or \dcolxxiii
3445 \or \dcolxxiv \or \dcolxxv \or \dcolxxvi
3446 \or \dcolxxvii \or \dcolxxviii \or \dcolxxix \or \dcolxxx
3447 \else \dcplerr \fi}
3448

```

`\step1@dcplcount` This increments the column counter, and issues an error message if it is too large.

```

3449 \newcommand*{\step1@dcplcount}{\advance\l@dcplcount\@ne
3450 \ifnum\l@dcplcount>30\relax
3451 \led@err@TooManyColumns
3452 \fi}
3453

```

`\l@dsetmaxcolwidth` Sets the column width to the maximum value seen so far. (was `\dimenzuordnung`)

```

3454 \newcommand{\l@dsetmaxcolwidth}{%
3455 \ifdim\l@dcplwidth < \wd\hilfsbox
3456 \l@dcplwidth = \wd\hilfsbox
3457 \else \relax \fi}
3458

```

`\EDTEXT` We need to be able to modify the `\edtext` and `\critext` macros and also restore `\xedtext` their original definitions.

```
\CRITEXT 3459 \let\EDTEXT=\edtext
\xcritext 3460 \newcommand{\xedtext}[2]{\EDTEXT{#1}{#2}}
          3461 \let\CRITEXT=\critext
          3462 \long\def\xcritext #1#2/{\CRITEXT{#1}{#2}/}
```

`\EDLABEL` We need to be able to modify and restore the `\edlabel` macro.

```
\xedlabel 3463 \let\EDLABEL=\edlabel
          3464 \newcommand*\{\xedlabel}[1]{\EDLABEL{#1}}
```

`\EDINDEX` Macros supporting modification and restoration of `\edindex`.

```
\xedindex 3465 \let\EDINDEX=\edindex
\nulledindex 3466 \ifl@dmemoir
          3467   \newcommand{\xedindex}{\@bsphack%
          3468     \ifnextchar [{\l@d@index}{\l@d@index[\jobname]}}
          3469   \newcommand{\nulledindex}[2][\jobname]{\@bsphack\@esphack}
          3470 \else
          3471   \newcommand{\xedindex}{\@bsphack%
          3472     \doedindexlabel
          3473     \begingroup
          3474     \@sanitize
          3475     \@wredindex}
          3476   \newcommand{\nulledindex}[1]{\@bsphack\@esphack}
          3477 \fi
          3478
```

`\@line@num` Macro supporting restoration of `\linenum`.

```
3479 \let\@line@num=\linenum
```

`\l@dgobbledarg` `\l@dgobbledarg` replaces its delineated argument by `\relax` (was `\verschwinden`).

`\l@dgobblearg` `\l@dgobblearg{<arg>}` replaces its argument by `\relax`.

```
3480 \def\l@dgobbledarg #1/{\relax}
3481 \newcommand*\{\l@dgobblearg}[1]{\relax}
3482
```

`\Relax`

```
\NEXT 3483 \let\Relax=\relax
```

`\@hilfs@count` 3484 `\let\NEXT=\next`

```
3485 \newcount\@hilfs@count
3486
```

`\measuremcell` Measure (recursively) the width required for a math cell. (was `\messen`)

```
3487 \def\measuremcell #1&{%
3488   \ifx #1\ \ifnum\l@dc@count=0\let\NEXT\relax%
3489   \else\l@dcheckcols%
3490     \l@dc@count=0%
3491   \let\NEXT\measuremcell%
```

```

3492         \fi%
3493     \else\setbox\hilfsbox=\hbox{$\displaystyle{#1}$}%
3494         \step1@dcolcount%
3495         \l@dsetmaxcolwidth%
3496         \let\NEXT\measuremcell%
3497     \fi\NEXT}
3498

```

`\measuretcell` Measure (recursively) the width required for a text cell. (was `\messentext`)

```

3499 \def\measuretcell #1\{%
3500     \ifx #1\ \ifnum\l@dcolcount=0\let\NEXT\relax%
3501         \else\l@dcheckcols%
3502             \l@dcolcount=0%
3503             \let\NEXT\measuretcell%
3504         \fi%
3505     \else\setbox\hilfsbox=\hbox{#1}%
3506         \step1@dcolcount%
3507         \l@dsetmaxcolwidth%
3508         \let\NEXT\measuretcell%
3509     \fi\NEXT}
3510

```

`\measuremrow` Measure (recursively) the width required for a math row. (was `\Messen`)

```

3511 \def\measuremrow #1\{%
3512     \ifx #1\let\NEXT\relax%
3513     \else\measuremcell #1\&\&\&%
3514         \let\NEXT\measuremrow%
3515     \fi\NEXT}

```

`\measuretrow` Measure (recursively) the width required for a text row. (was `\Messentext`)

```

3516 \def\measuretrow #1\{%
3517     \ifx #1\let\NEXT\relax%
3518     \else\measuretcell #1\&\&\&%
3519         \let\NEXT\measuretrow%
3520     \fi\NEXT}
3521

```

`\edtabcolsep` The length `\edtabcolsep` controls the distance between columns. (was `\abstand`)

```

3522 \newskip\edtabcolsep
3523 \global\edtabcolsep=10pt
3524

```

`\NEXT`

```

\Next 3525 \let\NEXT\relax
3526 \let\Next=\next

```

`\variab`

```

3527 \newcommand{\variab}{\relax}
3528

```


`\l@dcheckcols` Check that the number of columns is consistent. (was `\tabfehlermeldung`)

```

3529 \newcommand*\l@dcheckcols}{%
3530   \ifnum\l@dcolcount=1\relax
3531   \else
3532     \ifnum\l@dampcount=1\relax
3533     \else
3534       \ifnum\l@dcolcount=\l@dampcount\relax
3535       \else
3536         \l@d@err@UnequalColumns
3537       \fi
3538     \fi
3539     \l@dampcount=\l@dcolcount
3540   \fi}
3541
```

`\l@dmodforcritext` Modify and restore various macros for when `\critext` is used.

```

\l@drestoreforcritext 3542 \newcommand{\l@dmodforcritext}{%
3543   \let\critext\relax%
3544   \def\do##1{\global\csletcs{##1footnote}{\l@dgobbledarg}}
3545   \dolistloop{\@series}%
3546   \let\edindex\nulledindex%
3547   \let\linenum@gobble}
3548 \newcommand{\l@drestoreforcritext}{%
3549   \def\do##1{\csdef{##1footnote}##1##2/{\csuse{##1@footnote}{##1}{##2}}}
3550   \dolistloop{\@series}%
3551   \let\edindex\xedndex}
3552
```

`\l@dmodforedtext` Modify and restore various macros for when `\edtext` is used.

```

\l@drestoreforedtext 3553 \newcommand{\l@dmodforedtext}{%
3554   \let\edtext\relax
3555   \def\do##1{\global\csletcs{##1footnote}{\l@dgobblearg}}
3556   \dolistloop{\@series}%
3557   \let\edindex\nulledindex
3558   \let\linenum@gobble}
3559 \newcommand{\l@drestoreforedtext}{%
3560   \def\do##1{\csgdef{##1footnote}##1{\csuse{##1@footnote}{##1}}}
3561   \dolistloop{\@series}%
3562   \let\edindex\xedndex}

```

`\l@dnullfills` Nullify and restore some column fillers, etc.

```

\l@drestorefills 3563 \newcommand{\l@dnullfills}{%
3564   \def\edlabel##1{}%
3565   \def\edrowfill##1##2##3{}%
3566 }
3567 \newcommand{\l@drestorefills}{%
3568   \def\edrowfill##1##2##3{\@EDROWFILL@{##1}{##2}{##3}}%
3569 }
3570
```

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@dc@lcount=0%\removelastskip
    \let\Next\relax%
  \else\l@dc@lcount=0%
    \let\Next=\rverteilen%
  \fi%
\else%
  \footnoteverschw%
  \step\l@dc@lcount%
  \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.

```
3571 \newcommand{\letsforverteilen}{%
3572   \let\critext\xcritext
3573   \let\edtext\xedtext
3574   \let\edindex\xedindex}
```

```

3575 \def\do##1{\global\csletcs{##1footnote}{##1@footnote}}
3576 \dolistloop{\@series}%
3577 \let\linenum\@line@num
3578 \hilfsskip=\l@dcwidth%
3579 \advance\hilfsskip by -\wd\hilfsbox
3580 \def\edlabel##1{\xedlabel{##1}}
3581

```

`\setmcellright` Typeset (recursively) cells of display math right justified. (was `\rverteilen`)

```

3582 \def\setmcellright #1&{\def\edlabel##1{%
3583     \let\edindex\nulledindex
3584     \ifx #1\\ \ifnum\l@dcwidth=0%\removelastskip
3585         \let\Next\relax%
3586     \else\l@dcwidth=0%
3587         \let\Next=\setmcellright%
3588     \fi%
3589 \else%
3590     \disablel@dtabfeet%
3591     \step1@dcwidth%
3592     \setbox\hilfsbox=\hbox{$\displaystyle{#1}$}%
3593     \letsforverteilen%
3594     \hskip\hilfsskip$\displaystyle{#1}$%
3595     \hskip\edtabcolsep%
3596     \let\Next=\setmcellright%
3597 \fi\Next}
3598

```

`\settcclright` Typeset (recursively) cells of text right justified. (was `\rverteilentext`)

```

3599 \def\settcclright #1&{\def\edlabel##1{%
3600     \let\edindex\nulledindex
3601     \ifx #1\\ \ifnum\l@dcwidth=0%\removelastskip
3602         \let\Next\relax%
3603     \else\l@dcwidth=0%
3604         \let\Next=\settcclright%
3605     \fi%
3606 \else%
3607     \disablel@dtabfeet%
3608     \step1@dcwidth%
3609     \setbox\hilfsbox=\hbox{#1}%
3610     \letsforverteilen%
3611     \hskip\hilfsskip#1%
3612     \hskip\edtabcolsep%
3613     \let\Next=\settcclright%
3614 \fi\Next}

```

`\setmcellleft` Typeset (recursively) cells of display math left justified. (was `\lverteilen`)

```

3615 \def\setmcellleft #1&{\def\edlabel##1{%
3616     \let\edindex\nulledindex
3617     \ifx #1\\ \ifnum\l@dcwidth=0 \let\Next\relax%

```

```

3618         \else\l@dcolcount=0%
3619             \let\Next=\setmcellleft%
3620         \fi%
3621     \else \disablel@dtabfeet%
3622         \stepl@dcolcount%
3623         \setbox\hilfsbox=\hbox{$\displaystyle{#1}$}%
3624         \letsforverteilen
3625         $\displaystyle{#1}$\hskip\hilfsskip\hskip\edtabcolsep%
3626         \let\Next=\setmcellleft%
3627     \fi\Next}
3628

```

`\settcellleft` Typeset (recursively) cells of text left justified. (was `\lverteilentext`)

```

3629 \def\settcellleft #1{\def\edlabel##1{%
3630     \let\edindex\nulledindex
3631     \ifx #1\ \ifnum\l@dcolcount=0 \let\Next\relax%
3632         \else\l@dcolcount=0%
3633         \let\Next=\settcellleft%
3634     \fi%
3635     \else \disablel@dtabfeet%
3636         \stepl@dcolcount%
3637         \setbox\hilfsbox=\hbox{#1}%
3638         \letsforverteilen
3639         #1\hskip\hilfsskip\hskip\edtabcolsep%
3640         \let\Next=\settcellleft%
3641     \fi\Next}

```

`\setmcellcenter` Typeset (recursively) cells of display math centered. (was `\zverteilen`)

```

3642 \def\setmcellcenter #1{\def\edlabel##1{%
3643     \let\edindex\nulledindex
3644     \ifx #1\ \ifnum\l@dcolcount=0\let\Next\relax%
3645         \else\l@dcolcount=0%
3646         \let\Next=\setmcellcenter%
3647     \fi%
3648     \else \disablel@dtabfeet%
3649         \stepl@dcolcount%
3650         \setbox\hilfsbox=\hbox{$\displaystyle{#1}$}%
3651         \letsforverteilen%
3652         \hskip 0.5\hilfsskip$\displaystyle{#1}$\hskip0.5\hilfsskip%
3653         \hskip\edtabcolsep%
3654         \let\Next=\setmcellcenter%
3655     \fi\Next}
3656

```

`\settcellcenter` Typeset (recursively) cells of text centered. (new)

```

3657 \def\settcellcenter #1{\def\edlabel##1{%
3658     \let\edindex\nulledindex
3659     \ifx #1\ \ifnum\l@dcolcount=0 \let\Next\relax%
3660         \else\l@dcolcount=0%

```

```

3661             \let\Next=\settcellcenter%
3662         \fi%
3663     \else \disablel@dtabfeet%
3664         \step1@dcolcount%
3665         \setbox\hilfsbox=\hbox{#1}%
3666         \letsforverteilen%
3667         \hskip 0.5\hilfsskip #1\hskip 0.5\hilfsskip%
3668         \hskip\edtabcolsep%
3669         \let\Next=\settcellcenter%
3670     \fi\Next}
3671

```

\NEXT

```

3672 \let\NEXT=\relax
3673

```

\setmrowright Typeset (recursively) rows of right justified math. (was \rsetzen)

```

3674 \def\setmrowright #1\{%
3675     \ifx #1& \let\NEXT\relax
3676     \else \centerline{\setmcellright #1&\&\&}
3677         \let\NEXT=\setmrowright
3678     \fi\NEXT}

```

\settroright Typeset (recursively) rows of right justified text. (was \rsetzentext)

```

3679 \def\settroright #1\{%
3680     \ifx #1& \let\NEXT\relax
3681     \else \centerline{\settcellright #1&\&\&}
3682         \let\NEXT=\settroright
3683     \fi\NEXT}
3684

```

\setmrowleft Typeset (recursively) rows of left justified math. (was \lsetzen)

```

3685 \def\setmrowleft #1\{%
3686     \ifx #1& \let\NEXT\relax
3687     \else \centerline{\setmcellleft #1&\&\&}
3688         \let\NEXT=\setmrowleft
3689     \fi\NEXT}

```

\settrorleft Typeset (recursively) rows of left justified text. (was \lsetzentext)

```

3690 \def\settrorleft #1\{%
3691     \ifx #1& \let\NEXT\relax
3692     \else \centerline{\settcellleft #1&\&\&}
3693         \let\NEXT=\settrorleft
3694     \fi\NEXT}
3695

```

\setmrowcenter Typeset (recursively) rows of centered math. (was \zsetzen)

```

3696 \def\setmrowcenter #1\{%
3697     \ifx #1& \let\NEXT\relax%

```

```

3698 \else \centerline{\setmcellcenter #1\\&\\&}
3699 \let\NEXT=\setmrowcenter
3700 \fi\NEXT}

```

\settrowcenter Typeset (recursively) rows of centered text. (new)

```

3701 \def\settrowcenter #1\\{%
3702 \ifx #1& \let\NEXT\relax
3703 \else \centerline{\settcclcenter #1\\&\\&}
3704 \let\NEXT=\settrowcenter
3705 \fi\NEXT}
3706

```

\nullsetzen (was \nullsetzen)

```

3707 \newcommand{\nullsetzen}{%
3708 \step1@dcolcount%
3709 \l@dcolwidth=0pt%
3710 \ifnum\l@dcolcount=30\let\NEXT\relax%
3711 \l@dcolcount=0\relax
3712 \else\let\NEXT\nullsetzen%
3713 \fi\NEXT}
3714

```

\edatleft **\edatleft**[$\langle math \rangle$]{ $\langle symbol \rangle$ }{ $\langle len \rangle$ } (combination and generalisation of original **\Seklam** and **\Seklamgl**). Left $\langle symbol \rangle$, $2\langle len \rangle$ high with prepended $\langle math \rangle$ vertically centered.

```

3715 \newcommand{\edatleft}[3][\@empty]{%
3716 \ifx#1\@empty
3717 \vbox to 10pt{\vss\hbox{$\left#2\vrule width0pt height #3
3718 depth 0pt \right. $\hss}\vfil}
3719 \else
3720 \vbox to 4pt{\vss\hbox{$#1\left#2\vrule width0pt height #3
3721 depth 0pt \right. $\}\vfil}
3722 \fi}

```

\edatright **\edatright**[$\langle math \rangle$]{ $\langle symbol \rangle$ }{ $\langle len \rangle$ } (combination and generalisation of original **\sekla** and **\sekla**gl). Right $\langle symbol \rangle$, $2\langle len \rangle$ high with appended $\langle math \rangle$ vertically centered.

```

3723 \newcommand{\edatright}[3][\@empty]{%
3724 \ifx#1\@empty
3725 \vbox to 10pt{\vss\hbox{$\left.\vrule width0pt height #3
3726 depth 0pt \right#2 $\hss}\vfil}
3727 \else
3728 \vbox to 4pt{\vss\hbox{$\left.\vrule width0pt height #3
3729 depth 0pt \right#2 #1 $\}\vfil}
3730 \fi}
3731

```

\edvertline **\edvertline**{ $\langle len \rangle$ } vertical line $\langle len \rangle$ high. (was **\sestrich**)

```

3732 \newcommand{\edvertline}[1]{\vbox to 8pt{\vss\hbox{\vrule height #1}\vfil}}
3733

```

`\edvertdots` `\edvertdots{<len>}` vertical dotted line *<len>* high. (was `\sepunkte`)

```
3734 \newcommand{\edvertdots}[1]{\vbox to 1pt{\vss\vbox to #1%
3735         {\cleaders\hbox{$\m@th\hbox{.}\vbox to 0.5em{ }$}\vfil}}}
3736
```

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`)

```
3737 \newdimen\edfilldimen
3738 \edfilldimen=0pt
3739
```

`\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....`

```
3740 \newcounter{addcolcount}
3741 \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@`.

```

3742 \newcommand{\l@dtabaddcols}[2]{%
3743   \l@dcheckstartend{#1}{#2}%
3744   \ifl@dstartendok
3745     \setcounter{addcolcount}{#1}%
3746     \@whilenum \value{addcolcount}<#2\relax \do
3747       {\advance\edfilldimen by \the\csname dcol\theaddcolcount\endcsname
3748        \advance\edfilldimen by \edtabcolsep
3749        \stepcounter{addcolcount}}%
3750     \advance\edfilldimen by \the\csname dcol\theaddcolcount\endcsname
3751     \fi
3752 }
3753

```

`\ifl@dstartendok` `\l@dcheckstartend{ $\langle startcol \rangle$ }{ $\langle endcol \rangle$ }` checks that the values of $\langle startcol \rangle$ and $\langle endcol \rangle$ are sensible. If they are then `\ifl@dstartendok` is set TRUE, otherwise it is set FALSE.

```

3754 \newif\ifl@dstartendok
3755 \newcommand{\l@dcheckstartend}[2]{%
3756   \l@dstartendoktrue
3757   \ifnum #1<\@ne
3758     \l@dstartendokfalse
3759     \led@err@LowStartColumn
3760   \fi
3761   \ifnum #2>30\relax
3762     \l@dstartendokfalse
3763     \led@err@HighEndColumn
3764   \fi
3765   \ifnum #1>#2\relax
3766     \l@dstartendokfalse
3767     \led@err@ReverseColumns
3768   %% \eledmac@error{Start column is greater than end column}{\@ehc}%
3769   \fi
3770 }
3771

```

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

```

3772 \newcommand*{\edrowfill}[3]{%
3773   \l@dtabaddcols{#1}{#2}%
3774   \hb@xt@ \the\l@dcolwidth{\hb@xt@ \the\edfilldimen{#3}\hss}}
3775 \let\@edrowfill@=\edrowfill
3776 \def\@EDROWFILL@#1#2#3{\@edrowfill@{#1}{#2}{#3}}
3777

```

`\edbeforetab` The macro `\edbeforetab{ $\langle text \rangle$ }{ $\langle math \rangle$ }` puts $\langle text \rangle$ at the left margin before array cell entry $\langle math \rangle$. Conversely, the macro `\edaftertab{ $\langle math \rangle$ }{ $\langle text \rangle$ }` puts $\langle text \rangle$ at the right margin after array cell entry $\langle math \rangle$. `\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)

```
3778 \newcommand{\leftltab}[1]{%
3779   \hb@xt@{\z@\vbox{\edtabindent%
3780     \moveleft\Hilfsskip\hbox{\ #1}}\hss}}
3781
```

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

```
3782 \newcommand{\leftrtab}[2]{%
3783   #2\hb@xt@{\z@\vbox{\edtabindent%
3784     \advance\Hilfsskip by\dcoli%
3785     \moveleft\Hilfsskip\hbox{\ #1}}\hss}}
3786
```

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

```
3787 \newcommand{\leftctab}[2]{%
3788   \hb@xt@{\z@\vbox{\edtabindent\l@dcolcount=\l@dampcount%
3789     \advance\Hilfsskip by 0.5\dcoli%
3790     \setbox\hilfsbox=\hbox{\def\edlabel##1}{%
3791       \disablel@dtabfeet$\displaystyle{#2}$}%
3792     \advance\Hilfsskip by -0.5\wd\hilfsbox%
3793     \moveleft\Hilfsskip\hbox{\ #1}}\hss}%
3794   #2}
3795
```

\rightctab \rightctab{<math>}{<text>} for \edaftertab in \ctab. (was \rechtsztab)

```
3796 \newcommand{\rightctab}[2]{%
3797   \setbox\hilfsbox=\hbox{\def\edlabel##1}{%
3798     \disablel@dtabfeet#2}\l@dampcount=\l@dcolcount%
3799   #1\hb@xt@{\z@\vbox{\edtabindent\l@dcolcount=\l@dampcount%
3800     \advance\Hilfsskip by 0.5\l@dcolwidth%
3801     \advance\Hilfsskip by -\wd\hilfsbox%
3802     \setbox\hilfsbox=\hbox{\def\edlabel##1}{%
3803       \disablel@dtabfeet$\displaystyle{#1}$}%
3804     \advance\Hilfsskip by -0.5\wd\hilfsbox%
3805     \advance\Hilfsskip by \edtabcolsep%
3806     \moveright\Hilfsskip\hbox{\ #2}}\hss}%
3807   }
3808
```

\rightltab \rightltab{<math>}{<text>} for \edaftertab in \ltab. (was \rechtsltab)

```
3809 \newcommand{\rightltab}[2]{%
3810   \setbox\hilfsbox=\hbox{\def\edlabel##1}{%
3811     \disablel@dtabfeet#2}\l@dampcount=\l@dcolcount%
3812   #1\hb@xt@{\z@\vbox{\edtabindent\l@dcolcount=\l@dampcount%
3813     \advance\Hilfsskip by\l@dcolwidth%
3814     \advance\Hilfsskip by-\wd\hilfsbox%
```

```

3815      \setbox\hilfsbox=\hbox{\def\edlabel##1}{%
3816      \disablel@dtabfeet$\displaystyle{#1}$}%
3817      \advance\Hilfsskip by-\wd\hilfsbox%
3818      \advance\Hilfsskip by\edtabcolsep%
3819      \moveright\Hilfsskip\hbox{ #2}}\hss}%
3820      }
3821

```

`\righttrtab` `\righttrtab{<math>}<{<text>}` for `\edaftertab` in `\rtab`. (was `\rechtsrtab`)

```

3822 \newcommand{\righttrtab}[2]{%
3823     \setbox\hilfsbox=\hbox{\def\edlabel##1}{%
3824     \disablel@dtabfeet#2}%
3825     #1\hb@xt@#2{\vbox{\edtabindent%
3826     \advance\Hilfsskip by-\wd\hilfsbox%
3827     \advance\Hilfsskip by\edtabcolsep%
3828     \moveright\Hilfsskip\hbox{ #2}}\hss}%
3829     }
3830

```

`\rtab` `\rtab{<body>}` typesets `<body>` 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 `<body>` to get the column widths, and then in a second pass to typeset the body.

```

3831 \newcommand{\rtab}[1]{%
3832     \l@dnnullfills
3833     \def\edbeforetab##1##2{\lefttrtab{##1}{##2}}%
3834     \def\edaftertab##1##2{\righttrtab{##1}{##2}}%
3835     \measurembody{#1}%
3836     \l@drestorefills
3837     \variab
3838     \setmrowright #1\&\%
3839     \enablel@dtabfeet}
3840

```

`\measurembody` `\measurembody{<body>}` measures the array `<body>`.

```

3841 \newcommand{\measurembody}[1]{%
3842     \disablel@dtabfeet%
3843     \l@dcolcount=0%
3844     \nullsetzen%
3845     \l@dcolcount=0
3846     \measuremrow #1\&\%
3847     \global\l@dampcount=1}
3848

```

`\rtabtext` `\rtabtext{<body>}` typesets `<body>` as a tabular with the entries right justified. (was `\rtabtext`)

```

3849 \newcommand{\rtabtext}[1]{%

```

```

3850 \l@dnnullfills
3851 \measuretbody{#1}%
3852 \l@drestorefills
3853 \variab
3854 \setthrowright #1\\&\\%
3855 \enablel@dtabfeet}
3856

```

`\measuretbody` `\measuretbody{<body>}` measures the tabular `<body>`.

```

3857 \newcommand{\measuretbody}[1]{%
3858 \disablel@dtabfeet%
3859 \l@dcolcount=0%
3860 \nullsetzen%
3861 \l@dcolcount=0
3862 \measuretrrow #1\\&\\%
3863 \global\l@dampcount=1}
3864

```

`\ltab` Array with entries left justified. (was `\ltab`)

```

\edbeforetab 3865 \newcommand{\ltab}[1]{%
\edaftertab 3866 \l@dnnullfills
3867 \def\edbeforetab##1##2{\leftltab{##1}{##2}}%
3868 \def\edaftertab##1##2{\rightltab{##1}{##2}}%
3869 \measuretbody{#1}%
3870 \l@drestorefills
3871 \variab
3872 \setmrowleft #1\\&\\%
3873 \enablel@dtabfeet}
3874

```

`\ltabtext` Tabular with entries left justified. (was `\ltabtext`)

```

3875 \newcommand{\ltabtext}[1]{%
3876 \l@dnnullfills
3877 \measuretbody{#1}%
3878 \l@drestorefills
3879 \variab
3880 \setmrowleft #1\\&\\%
3881 \enablel@dtabfeet}
3882

```

`\ctab` Array with centered entries. (was `\ztab`)

```

\edbeforetab 3883 \newcommand{\ctab}[1]{%
\edaftertab 3884 \l@dnnullfills
3885 \def\edbeforetab##1##2{\leftctab{##1}{##2}}%
3886 \def\edaftertab##1##2{\rightctab{##1}{##2}}%
3887 \measuretbody{#1}%
3888 \l@drestorefills
3889 \variab
3890 \setmrowcenter #1\\&\\%

```

```
3891 \enablel@dtabfeet}
3892
```

`\ctabtext` Tabular with entries centered. (new)

```
3893 \newcommand{\ctabtext}[1]{%
3894 \l@dnnullfills
3895 \measuretbody{#1}%
3896 \l@drestorefills
3897 \variab
3898 \settrrowcenter #1\\&\\%
3899 \enablel@dtabfeet}
3900
```

`\spreadtext` (was `\breitertext`)

```
3901 \newcommand{\spreadtext}[1]{%\l@dcolcount=\l@dampcount%
3902 \hb@xt@ \the\l@dcolwidth{\hbox{#1}\hss}}
```

`\spreadmath` (was `\breiter`, ‘breiter’ = ‘broadly’)

```
3903 \newcommand{\spreadmath}[1]{%
3904 \hb@xt@ \the\l@dcolwidth{\hbox{$\displaystyle{#1}$}\hss}}
3905
```

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`)

```
3906 \def\tabellzwischen #1&{%
3907 \ifx #1\ \let\NEXT\relax \l@dcolcount=0
3908 \else \stepl@dcolcount%
3909 \l@dcolwidth = #1 mm
3910 \let\NEXT=\tabellzwischen
3911 \fi \NEXT }
3912
```

`\edatabell` For example `\edatabell 4 & 19 & 8 \` specifies 3 columns with widths of 4, 19, and 8mm. (was `\atabell`)

```
3913 \def\edatabell #1\\{%
3914 \tabellzwischen #1&\\&}
```

`\Setzen` (was `\Setzen`, ‘setzen’ = ‘set’)

```
3915 \def\Setzen #1&{%
3916 \ifx #1\relax \let\NEXT=\relax
3917 \else \stepl@dcolcount%
3918 \let\tabelskip=\l@dcolwidth
3919 \EDTAB #1|
3920 \let\NEXT=\Setzen
3921 \fi\NEXT}
3922
```

\EDATAB (was \ATAB)

```

3923 \def\EDATAB #1\{%
3924     \ifx #1\Relax \centerline{\Setzen #1\relax&}
3925         \let\Next\relax
3926     \else \centerline{\Setzen #1&\relax&}
3927         \let\Next=\EDATAB
3928     \fi\Next}

```

\edatab (was \atab)

```

3929 \newcommand{\edatab}[1]{%
3930     \variab%
3931     \EDATAB #1\\Relax\\}
3932

```

\HILFSskip More helpers.

```

\Hilfsskip 3933 \newskip\HILFSskip
3934 \newskip\Hilfsskip
3935

```

\EDTABINDENT (was \TABINDENT)

```

3936 \newcommand{\EDTABINDENT}{%
3937     \ifnum\l@dcolcount=30\let\NEXT\relax\l@dcolcount=0%
3938     \else\step\l@dcolcount%
3939         \advance\Hilfsskip by\l@dcolwidth%
3940         \ifdim\l@dcolwidth=0pt\advance\hilfscount\@ne
3941         \else\advance\Hilfsskip by \the\hilfscount\edtabcolsep%
3942         \hilfscount=1\fi%
3943         \let\NEXT=\EDTABINDENT%
3944     \fi\NEXT}%

```

\edtabindent (was \tabindent)

```

3945 \newcommand{\edtabindent}{%
3946     \l@dcolcount=0\relax
3947     \Hilfsskip=0pt%
3948     \hilfscount=1\relax
3949     \EDTABINDENT%
3950     \hilfsskip=\hsize%
3951     \advance\hilfsskip -\Hilfsskip%
3952     \Hilfsskip=0.5\hilfsskip%
3953     }%
3954

```

\EDTAB (was \TAB)

```

3955 \def\EDTAB #1|#2|{%
3956     \setbox\tabhilfbox=\hbox{$\displaystyle{#1}$}%
3957     \setbox\tabHilfbox=\hbox{$\displaystyle{#2}$}%
3958     \advance\tabelskip -\wd\tabhilfbox%
3959     \advance\tabelskip -\wd\tabHilfbox%
3960     \unhbox\tabhilfbox\hskip\tabelskip%

```

\tabhilfbox Further helpers.

`\edarrayl` The ‘environment’ forms for `\ltab`, `\ctab` and `\rtab`.

`edtabular` The ‘environment’ forms for `\ltabtext`, `\ctabtext` and `\rtabtext`.

Here's the code for enabling `\edtext` (instead of `\critext`).

```

\enablel@dtabfeet 3981 \newcommand{\usingcritext}{%
    \usingedtext 3982 \def\disablel@dtabfeet{\l@dmodforcritext}%
    3983 \def\enablel@dtabfeet{\l@drestoreforcritext}}
    3984 \newcommand{\usingedtext}{%
    3985 \def\disablel@dtabfeet{\l@dmodforedtext}%
    3986 \def\enablel@dtabfeet{\l@drestoreforedtext}}
    3987
    3988 \usingedtext
    3989

```

36 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.

```
3990 \def\normal@page@break{}
```

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

```
3991 \def\l@prev@pb{}
```

```
3992 \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.

```
3993 \newcommand{\ledpb}{\write\linenum@out{\string\led@pb}}
```

```
3994 \newcommand{\ledpbnum}[1]{\write\linenum@out{\string\led@pbnum{#1}}}
```

```
3995 \newcommand{\lednopb}{\write\linenum@out{\string\led@nopb}}
```

```
3996 \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.

```
3997 \newcommand{\led@pb}{\listcsxadd{l@prev@pb}{\the\absline@num}}
```

```
3998 \newcommand{\led@pbnum}[1]{\listcsxadd{l@prev@pb}{#1}}
```

```
3999 \newcommand{\led@nopb}{\listcsxadd{l@prev@nopb}{\the\absline@num}}
```

```
4000 \newcommand{\led@nopbnum}[1]{\listcsxadd{l@prev@nopb}{#1}}
```

\ledpbsetting The **\ledpbsetting** macro only changes the value of **\led@pb@macro**, for which the default value is **before**.

```
4001 \def\led@pb@setting{before}
```

```
4002 \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**.

```
4003 \newcommand{\led@check@pb}{\xifinlistcs{\the\absline@num}{l@prev@pb}{\pagebreak[4]}{}}
```

```

4004 \newcommand{\led@check@nopb}{%
4005   \IfStrEq{\led@pb@setting}{before}{%
4006     \xifinlistcs{\the\absline@num}{l@prev@nopb}%
4007     {\numdef{\abs@prevline}{\the\absline@num-1}%
4008     \xifinlistcs{\abs@prevline}{normal@page@break}%
4009     {\nopagebreak[4]\enlargethispage{\baselineskip}}%
4010     {}}%
4011   }%
4012 }%
4013 }%
4014 \IfStrEq{\led@pb@setting}{after}{%
4015   \xifinlistcs{\the\absline@num}{l@prev@nopb}{%
4016     \xifinlistcs{\the\absline@num}{normal@page@break}%
4017     {\nopagebreak[4]\enlargethispage{\baselineskip}}%
4018     {}}%
4019 }%
4020 }%
4021 }%
4022 }%
4023 }

```

37 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.

```

4024 \newcommand{\check@pb@in@verse}{%
4025   \ifinstanza\iflednopbinverse\ifinserthangingsymbol% Using stanzas and enabling page br
4026     \ifnum\page@num=\last@page@num\else%If we have change page
4027       \IfStrEq{\led@pb@setting}{before}{%
4028         \numgdef{\abs@line@verse}{\the\absline@num-1}%
4029         \ledpbnum{\abs@line@verse}%
4030       }%
4031       \IfStrEq{\led@pb@setting}{after}{%
4032         \numgdef{\abs@line@verse}{\the\absline@num-1}%
4033         \lednopbnum{\abs@line@verse}%
4034       }%
4035     \fi%
4036   \fi\fi\fi%
4037 }

```


38 The End

i/code_i

Appendix A Some things to do when changing version

Appendix A.1 Migration from ledmac to eledmac

In eledmac, some changes were made in the code to allow for easy customization. This can cause problems for people who have made their own customizations. The next sections explain how to correct this.

If you created your own series using `\addfootins` and `\addfootinsX`, you should instead use the `\newseries` command (see 4.6 p.23). You must delete your `\Xfootnote` command.

If you customized the `\XXXXXXfmt` command, you should see if commands for display options (4.3 p.17) and options in `\Xfootnote` (4.1 p.15) 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 `\XXXXXfmt` command, but you must replace:

```
\renewcommand*{XXXXXfmt}[3]
```

with

```
\renewcommandx*{XXXXXfmt}[4][4=Z]
```

If you don't do that, you will see a spurious `[X]`, where `X` is series letter.

If you used a `\protect` command inside a `\footnote` command inside a numbered section, you must change the `\protect` to `\noexpand`. If you don't, the command after the `\protect` won't be displayed.

Appendix A.2 Migration to eledmac 1.5.1

The version 1.5.1 corrects a bug with `stanzaindentrepitition` (cf. p. 23). This bug had two consequences:

1. `stanzaindentrepitition` didn't work when its value was greater than 2.
2. `stanzaindentrepitition` worked wrong when its value was equal to 2.

So, if you used `stanzaindentrepitition` with value equal to 2, you must change your `\setstanzaindent`. Explanation:

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

³¹<https://github.com/maieul/ledmac/issues>

This code, in a version older than 1.5.1, made that the first verse had an indent of 0, the second verse of 1, the third verse of 0, the fourth verse of 1 etc.

But instead the code should have assigned the reverse: the first verse had an indent of 1, the second verse of 0, the third verse of 1, the fourth verse of 0 etc.

So version 1.5.1 corrected this bug. If you want to keep the older presentation, you must change:

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

by:

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

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 LaTeX*. 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>\@@line</code> 1706
<code>\&</code> 23, 3297, 3301, 3302, 3351, 3366, 3372, 3374	<code>\@@wrindexm@m</code> 3168, 3170, 3173, 3180, 3182, 3185,

- 3190, 3192, 3195, 3243, 3245, 3248
\@EDROWFILL@ 3568, 3772
\@M 1706, 3345, 3355
\@MM 1418
\@adv 584, 782
\@arabic 922
\@aux 2733
\@auxout 2735, 3167,
3169, 3172, 3179, 3181, 3184,
3189, 3191, 3194, 3242, 3244, 3247
\@botlist 2686, 2688
\@cclv 2597, 2601, 2602, 2684, 2685, 2713
\@chapter 213
\@checkend 3278
\@colht 2578, 2689, 2701
\@colroom 2689
\@combinefloats 2573
\@currentlabel
..... 956, 1906, 2005, 2055, 2146
\@currentenvir 3263, 3266, 3267
\@currlist 2690, 2693
\@dbldeferlist 2699, 2704, 2706
\@dblfloatplacement 2703
\@dbltoplist 2699, 2700
\@deferlist 2686, 2695, 2696
\@docclearpage 2673
\@edindex@fornote@true 3139
\@edrowfill@ 3772
\@ehb 2692
\@emptytoks 3254, 3264
\@footnotemark 1839
\@footnotetext
..... 1835, 1852, 3056, 3083, 3112
\@freelist 2571
\@gobble 814, 815, 2826, 3275, 3547, 3558
\@gobblethree 265, 2822
\@h 1704
\@hilfs@count 3483
\@idxfile 3157, 3168,
3170, 3173, 3180, 3182, 3185,
3190, 3192, 3195, 3243, 3245, 3248
\@ifclassloaded
..... 34, 1834, 2640, 2665, 3144
\@ifnextchar 3147, 3468
\@ifpackageloaded 37, 2593, 3237
\@iiiminipage 3045
\@iiiparbox 3072
\@index@command 3141,
3213, 3214, 3222, 3225, 3243, 3245
\@index@txt 3141,
3213, 3214, 3222, 3225, 3243, 3245
\@indexfile 3221, 3224, 3228
\@inputcheck 471
\@insert 1308–1310, 1344–1346
\@k 1704
\@kludgeins 2575, 2637
\@l 500, 743, 745, 747, 756, 758, 760, 763
\@l@dttempcnta ... 32, 618, 620, 622,
623, 1092, 1093, 1095, 1097,
1100, 1101, 1116, 1152–1156,
1158, 1165–1169, 1171, 1174,
1177, 1179, 1183, 1213, 1217,
1221, 1228, 1232, 1236, 1317,
1321, 1325, 1328, 1331, 1334, 1335
\@l@dttempcntb 32, 334, 335, 340, 344,
348, 352, 355, 378, 379, 386,
390, 394, 396, 404, 405, 1150,
1162, 1183, 1191–1193, 1195,
1213, 1217, 1221, 1228, 1232,
1236, 1266–1268, 1270, 1323,
1324, 2884, 2885, 2890, 2894,
2898, 2902, 2905, 3009–3011, 3013
\@l@reg 500
\@lab 701, 2724, 2766
\@latexerr 2692
\@led@extranofeet .. 2662, 2671, 2679
\@led@nofootfalse 2675–2677
\@led@nofoottrue 2674
\@led@testifnofoot 2673
\@ledgroupfalse 3092, 3121
\@ledgrouptrue 3082, 3110
\@line@num 3479, 3577
\@listdepth 3058
\@lock . 152, 455, 525, 527, 529, 542,
651, 652, 654, 655, 671, 672,
674, 1022, 1062, 1122, 1124,
1125, 1127, 1225, 1240, 1242, 1244
\@lopL 568
\@lopR 568
\@makechapterhead 209–212
\@makecol 2644
\@makefcolumn .. 2695, 2696, 2704, 2706
\@makeschapterhead 205–208
\@makespecialcolbox 2576
\@maxdepth 2591, 2600
\@mem@extranofeet 2666
\@mem@nofootfalse 2667, 2668
\@midlist 2571, 2572
\@minipagefalse 3069

- \@minipagerestore 3059
 - \@minus 162,
169, 176, 183, 190, 197, 2242, 2243
 - \@mpargs 3049, 3072
 - \@mpfn 3055, 3082, 3111
 - \@mpfootins 3065, 3075, 3078, 3088, 3117
 - \@mpfootnotetext ... 3056, 3083, 3112
 - \@mplistdepth 3058
 - \@nameuse 411,
413, 1423, 1424, 1550, 1552,
1631, 1632, 1672, 1674, 1761,
1763, 1812, 1814, 1882, 1886,
1888, 1893, 1896, 1897, 1903,
1907, 1911, 1915, 1927, 1932,
1935, 1937, 1938, 1950, 1956,
2002, 2006, 2016, 2024, 2026,
2052, 2056, 2066, 2074, 2076,
2114, 2115, 2124, 2132, 2133,
2138, 2147, 2151, 2161, 2163,
2188, 2189, 2191, 2192, 2197,
2652, 2653, 2655, 2656, 2658, 2660
 - \@next@page 742, 743
 - \@nobreakfalse 931
 - \@nobreaktrue 929, 933
 - \@nowrindex 3156
 - \@oldnobreak 929, 931, 982
 - \@opcol 2696, 2714
 - \@opxtrafeetii 2621, 2622, 2651
 - \@outputbox
. 2171, 2172, 2186, 2187, 2578,
2580, 2581, 2597, 2599, 2619, 2620
 - \@outputpage 2705
 - \@page 547
 - \@parboxrestore 1428, 1901, 3054
 - \@pboxswfalse 3047
 - \@pend 568
 - \@pendR 568
 - \@plus 162, 164, 169, 171, 176,
178, 183, 185, 190, 192, 197,
199, 1439, 2012, 2062, 2242, 2243
 - \@ref 686, 768
 - \@ref@reg 688
 - \@reinserts 2645
 - \@schapter 214
 - \@series ... 469, 2174, 2179, 2203,
2205, 2325, 2334, 2335, 2346,
2348, 2362, 2373, 2624, 2631,
2670, 2678, 3025, 3036, 3041,
3044, 3545, 3550, 3556, 3561, 3576
 - \@set 599, 787
 - \@setminipage 3060
 - \@showidx 3164, 3239
 - \@tag 822, 839, 862, 897, 1861
 - \@tempboxa 2684, 2685, 3050, 3072
 - \@tempdima 2601, 3048, 3052
 - \@templ@d 3000, 3001
 - \@textbottom 2583
 - \@texttop 2579
 - \@toksa 438, 446
 - \@toksb 438, 445–447
 - \@toplist 2686, 2687
 - \@whilenum 3746
 - \@whilesw 2696, 2705
 - \@wredindex 3207, 3209, 3475
 - \@x@sf 1828, 1831, 1842, 1848, 1872, 1878
 - \@xloop 1342, 1349
 - \@xympar 2872
 - \^ 494
 - _ 3780, 3785, 3793
- A**
- \abs@line@verse 4028, 4029, 4032, 4033
 - \abs@prevline 4007, 4008
 - \absline@num 146, 454,
505, 508, 511, 565, 613, 616,
625, 639, 661, 683, 693, 740,
741, 751, 753, 1053, 1074, 1075,
1083, 1307, 3997, 3999, 4003,
4006, 4007, 4015, 4016, 4028, 4032
 - Abu Kamil Shuja' b. Aslam 8
 - \actionlines@list
..... 457, 477, 480, 487, 613,
616, 625, 639, 661, 683, 1105, 1108
 - \actions@list
. 457, 481, 488, 614, 623, 627,
629, 641, 650, 663, 670, 684, 1109
 - \add@inserts 1030, 1296
 - \add@inserts@next 1296
 - \add@penalties 1317
 - \addcontentsline 265
 - \addfootins 2648
 - \addfootinsX 2182
 - \addtocounter 983
 - \addtol@denbody ... 3258, 3279, 3281
 - Adelard II 8
 - \advancelabel@refs 2730, 2739
 - \advanceline . 13, 70, 73, 782, 805, 815
 - \advancepageno 2566

\Aendnote 15
 \affixline@num 1028, 1144
 \affixpstart@num 1029, 1255
 \affixside@note 1030, 2978, 2987
 \Afootnote 15
 \afterlemmaseparator 19
 \afternote 20
 \afternumberinfootnote 18
 \aftersymmlinenum 18
 \allowbreak 1754, 1801, 2017, 2067
 \ampersand 25, 3297, 3374
 \appto 2993, 2994
 \apptocmd 207, 211, 213, 214
 \AtBeginDocument ... 2593, 2762, 3237
 \autopar 10, 91, 990
 \autoparfalse 286, 991
 \autopartrue 1004

B

\ballast 37
 \ballast@count . 1069, 1072, 1077, 1317
 Beeton, Barbara Ann Neuhaus Friend 11
 \beforeledchapter 215
 \beforelemmaseparator 19
 \beforenotesX 21
 \beforenumberinfootnote 18
 \beforemsymmlinenum 18
 \beforeXnotes 21
 \beginnumbering 9, 128, 299, 936, 1001
 \beginnumberingR 996
 \Bendnote 15
 \Bfootnote 15
 \bfseries 922
 \bhooknoteX 20
 \bhookXendnote 20
 \bhookXnote 20
 \body 1350, 1351, 3299, 3373
 \bodyfootmarkA 30
 \box .. 1045, 1047, 1626, 1641, 1686,
 1705, 2128, 2142, 2597, 2685, 2713
 \boxlinenum 18
 \boxmaxdepth 2600
 \boxsymmlinenum 19
 Bredon, Simon 8
 Breger, Herbert 5, 8, 164
 Brey, Gerhard 8
 \brokenpenalty 986
 Burt, John 6
 Busard, Hubert L. L. 8
 \bypage@false 302, 316, 322

\bypage@true 302, 310
 \bypstart@false 302, 311, 323
 \bypstart@true 302, 317

C

\c@addcolcount 3740
 \c@ballast 1069, 1077
 \c@firstlinenum
 361, 1164, 1166, 1169, 1171
 \c@firstsublinenum
 365, 1151, 1153, 1156, 1158
 \c@labidx 3124
 \c@linenumincrement .. 361, 1167, 1168
 \c@mpfootnote 3055, 3082, 3111
 \c@page ... 745, 747, 755, 758, 760, 763
 \c@pstart 922
 \c@sublinenumincrement 365, 1154, 1155
 \Cendnote 15
 \centerline 3676, 3681,
 3687, 3692, 3698, 3703, 3924, 3926
 \Cfootnote 15
 \ch@ck@l@ck 1181, 1209
 \ch@cksub@l@ck 1160, 1209
 \ch@pt@c 203
 \changes 2353, 2356, 2985
 \chapter 203, 204, 216, 217
 \char 3297
 \chardef 2871, 3299, 3301
 \check@pb@in@verse 1027, 4024
 Chester, Robert of 8
 Claassens, Geert H. M. 8
 class 1 feet 124, 137
 class 2 feet 137
 \cleaders 3735
 \cleardoublepage 215, 216
 \closeout 720, 726, 729, 733, 2814
 \clubpenalty 986, 1321
 \color@begingroup
 1429, 1637, 1902, 2138, 2604, 3051
 \color@endgroup
 1430, 1637, 1903, 2138, 2608, 3070
 \columnwidth
 1427, 1600, 1900, 3053, 3098
 \content
 2255, 2295, 2311, 2931, 2939, 2946
 Copernicus, Nicolaus 8
 \count 1573, 1578, 1590, 1594, 1729,
 1732, 1781, 1809, 1970, 1975,
 1991, 1994, 2042, 2045, 2089, 2092
 \countdef 2566

- `\cr` 1707, 1710
`\CRITEXT` 3459
`\critext`
 39, 816, 825, 838, 3461, 3543, 3572
`\cs` 826, 831, 835, 2322,
 2340, 2356, 2357, 2457, 2981, 2983
`\csdef` 3549
`\csexpandonce`
 . 1394, 1405, 1862, 1943, 2267,
 2280, 2299, 2314, 2933, 2941, 2948
`\csgdef` 1566, 1584, 1718, 1770, 1960,
 1981, 2032, 2082, 2207–2223,
 2228, 2230, 2231, 2233–2235,
 2237–2246, 2290, 2307, 2370, 3560
`\cslet` 2236
`\csletcs` 2317,
 2321, 3024, 3040, 3544, 3555, 3575
`\csnumdef` 968, 970
`\csundef` 468
`\csuse` 1411,
 1412, 1425, 1426, 1437, 1443,
 1446–1448, 1454, 1540, 1547,
 1553, 1574, 1575, 1579, 1580,
 1597, 1610, 1618, 1619, 1633,
 1634, 1652, 1659–1661, 1667,
 1668, 1677, 1678, 1693, 1695–
 1697, 1699, 1737, 1742, 1746,
 1750–1752, 1756, 1764, 1784,
 1789, 1793, 1797–1799, 1802,
 1803, 1815, 1889, 1890, 1893,
 1898, 1899, 1910, 1911, 1921,
 1942, 1971, 1972, 1976, 1977,
 1999, 2009, 2010, 2016, 2019,
 2050, 2058, 2060, 2066, 2069,
 2095, 2107, 2120, 2121, 2134,
 2135, 2151, 2158, 2167, 2173,
 2178, 2233, 2234, 2266, 2279,
 2285, 2297–2299, 2371, 2380,
 2444, 2493, 2499, 2510, 2512–
 2516, 2519, 2520, 2524, 2529,
 2533, 2534, 2538, 2543, 2547,
 2548, 2553, 2558, 2623, 2630,
 2667, 2668, 2676, 2677, 2822,
 3035, 3043, 3142, 3143, 3549, 3560
`\csxdef` 1373, 1375, 1379, 1381, 1388,
 1391, 1394, 1399, 1402, 1405, 2560
`\ctab` 3379, 3883, 3974
`\ctabtext` 3383, 3893, 3978
- D**
- `\dcolerr` 3435, 3447
`\dcoli` ... 3405, 3437, 3438, 3784, 3789
`\dcolii` 3406, 3438
`\dcoliii` 3407, 3438
`\dcoliv` 3408, 3439
`\dcolix` 3413, 3440
`\dcolv` 3409, 3439
`\dcolvi` 3410, 3439
`\dcolvii` 3411, 3440
`\dcolviii` 3412, 3440
`\dcolx` 3414, 3440
`\dcolxi` 3415, 3441
`\dcolxii` 3416, 3441
`\dcolxiii` 3417, 3441
`\dcolxiv` 3418, 3442
`\dcolxix` 3423, 3443
`\dcolxv` 3419, 3442
`\dcolxvi` 3420, 3442
`\dcolxvii` 3421, 3443
`\dcolxviii` 3422, 3443
`\dcolxx` 3424, 3443
`\dcolxxi` 3425, 3444
`\dcolxxii` 3426, 3444
`\dcolxxiii` 3427, 3444
`\dcolxxiv` 3428, 3445
`\dcolxxix` 3433, 3446
`\dcolxxv` 3429, 3445
`\dcolxxvi` 3430, 3445
`\dcolxxvii` 3431, 3446
`\dcolxxviii` 3432, 3446
`\dcolxxx` 3434, 3446
`\DeclareOption` 9–13
Dekker, Dirk-Jan 7, 38
`\Dendnote` 15
`\Dfootnote` 15
`\dimen` 773, 774, 776–778,
 780, 1574, 1579, 1598–1600,
 1603, 1712–1714, 1730, 1733,
 1782, 1810, 1971, 1976, 1992,
 1995, 2043, 2046, 2096–2098, 2101
`\dimen@` 2580, 2582
`\disablel@dtabfeet`
 3590, 3607, 3621, 3635,
 3648, 3663, 3791, 3798, 3803,
 3811, 3816, 3824, 3842, 3858, 3981
`\displaystyle` 3493, 3592,
 3594, 3623, 3625, 3650, 3652,
 3791, 3803, 3816, 3904, 3956, 3957
`\displaywidowpenalty` 987

- \divide .. 1154, 1167, 1600, 1713, 2098
 - \do@actions 1054, 1081
 - \do@actions@fixedcode ... 1102, 1115
 - \do@actions@next 1081
 - \do@ballast 1055, 1069
 - \do@insidelinehook 1030, 1037
 - \do@line 971, 1012
 - \do@linehook 1016, 1037
 - \do@lockoff 660
 - \do@lockoffL 660
 - \do@lockon 631
 - \do@lockonL 631
 - \docsvlist 1369, 2201, 2365, 2376
 - \doedindexlabel 3129, 3158, 3204, 3472
 - \doendnotes 27, 2864
 - \dolistloop
 - .. 469, 2174, 2179, 2334, 2346,
 - 2362, 2373, 2624, 2631, 2670,
 - 2678, 2998, 3025, 3036, 3041,
 - 3044, 3545, 3550, 3556, 3561, 3576
 - \doreinxtrafeeti ... 2170, 2190, 2626
 - \doreinxtrafeetii .. 2627, 2629, 2654
 - \dosplits 1704
 - Downes, Michael 37, 106, 108
 - \doxtrafeet 2614
 - \doxtrafeeti 2170, 2185, 2615
 - \doxtrafeetii 2616, 2618
 - \dp 1419,
 - 1624, 1639, 2126, 2140, 2580, 2601
 - \dummy@edtext 809, 817
 - \dummy@ref 687, 697
 - \dummy@text 808, 816
- E**
- \edaftertab
 - .. 33, 176, 3385, 3831, 3865, 3883
 - edarrayc (environment) 31, 3973
 - edarrayl (environment) 31, 3973
 - edarrayr (environment) 31, 3973
 - \EDATAB 3923, 3931
 - \edatab 3386, 3929
 - \edatabell 3387, 3913
 - \edatleft 33, 3388, 3715
 - \edatright 33, 3389, 3723
 - \edbeforetab
 - .. 33, 176, 3384, 3831, 3865, 3883
 - \edfilldimen
 - 3737, 3747, 3748, 3750, 3774
 - \edfont@info 888, 891, 895
 - \EDINDEX 3465
 - \edindex 30, 3145, 3202, 3465,
 - 3546, 3551, 3557, 3562, 3574,
 - 3583, 3600, 3616, 3630, 3643, 3658
 - \edindexlab 31, 3124, 3130, 3133
 - \EDLABEL 3463
 - \edlabel 27, 814, 2723,
 - 3130, 3463, 3564, 3580, 3582,
 - 3599, 3615, 3629, 3642, 3657,
 - 3790, 3797, 3802, 3810, 3815, 3823
 - \edmakelabel 28, 2809
 - \edpageref 27, 2770
 - \edrowfill . 32, 3392, 3565, 3568, 3772
 - \EDTAB 3919, 3955
 - \edtabcolsep 32, 3522,
 - 3595, 3612, 3625, 3639, 3653,
 - 3668, 3748, 3805, 3818, 3827, 3941
 - \EDTABINDENT 3936, 3949
 - \edtabindent 3779,
 - 3783, 3788, 3799, 3812, 3825, 3945
 - \EDTABtext 3963
 - edtabularc (environment) 31, 3977
 - edtabularl (environment) 31, 3977
 - edtabularr (environment) 31, 3977
 - \EDTEXT 3459
 - \edtext 14, 817, 860, 1855,
 - 1954, 2922–2924, 3459, 3554, 3573
 - \edvertdots 34, 3391, 3734
 - \edvertline 34, 3390, 3732
 - \Eendnote 15
 - \Efootnote 15
 - \Eledmac 2319
 - \eledmac@error 39,
 - 41, 43, 45, 57, 80, 83, 86, 89,
 - 91, 107, 109, 112, 114, 116, 3768
 - \eledmac@warning
 - 38, 60, 62, 64, 66, 68,
 - 70, 73, 76, 78, 94, 96, 98, 101,
 - 103, 105, 2183, 2205, 2649, 2999
 - \emph 3142, 3143
 - \empty 30, 119, 273, 276, 436,
 - 437, 477, 850, 872, 886, 900,
 - 904, 910, 943, 1105, 1163, 1179,
 - 1298–1300, 1311, 1343, 2725, 3312
 - \enablel@dtabfeet 3839,
 - 3855, 3873, 3881, 3891, 3899, 3981
 - \end@lemmas ... 807, 850, 851, 872, 873
 - \endashchar 22, 1451, 1531, 2856
 - \endgraf 966, 1006, 1010
 - \endline@num 462, 704, 710
 - \endlock 13, 797, 813, 3354, 3363, 3366

- `\endminipage` 3062
- `\endnumbering` ... 9, 131, 266, 288, 298
- `\endpage@num` 462, 703, 710
- `\endprint` 27, 2822, 2867
- `\endquotation` 160
- `\endquote` 160
- `\endstanzaextra` 25, 3347
- `\endsub` 13, 773, 812
- `\endsubline@num` 462, 705, 711
- `\enlargethispage` 4009, 4017
- `\enskip` 2823
- `\enspace` . 1911, 2016, 2066, 2151, 2823
- environments:
 - `edarrayc` 31, 3973
 - `edarrayl` 31, 3973
 - `edarrayr` 31, 3973
 - `edtabularc` 31, 3977
 - `edtabularl` 31, 3977
 - `edtabularr` 31, 3977
 - `ledgroup` 26, 3080
 - `ledgroupsize` 26, 3095
 - `minipage` 26
- Euclid 8
- `\ExecuteOptions` 14
- `\extensionchars` 36, 117, 137, 294
- `\extractendline@` 2484, 2487
- `\extractendsubline@` 2485, 2487
- `\extractline@` 2482, 2487, 2490
- `\extractsubline@` ... 2483, 2487, 2490
- F**
- `\f@encoding` 895
- `\f@family` 895
- `\f@series` 895
- `\f@shape` 895
- `\f@x@l@cks` 1204, 1209
- Fairbairns, Robin 29
- `\falseverse` 3347
- `\first@linenum@out@false` ... 715, 721
- `\first@linenum@out@true` 715
- `\firstlinenum` 11, 12, 370
- `\firstseries` 2329
- `\firstsublinenum` 11, 12, 370
- `\fix@page` 501, 554
- `\flag@end`
 - . 766, 846, 856, 857, 868, 878, 879
- `\flag@start`
 - . 766, 845, 846, 857, 867, 868, 879
- `\flagstanza` 25, 3369
- `\floatingpenalty` 1418
- `\flush@notes` 976, 1341, 3032
- `\flush@notesR` 3030
- Folkerts, Menso 8
- `\fontencoding` 1355
- `\fontfamily` 1355
- `\fontseries` 1355
- `\fontshape` 1355
- `\footfootmarkA` 30
- `\footfudgefiddle` 38, 1596, 1600, 2098
- `\footins` . 2596, 2603, 2607, 2635, 2675
- `\footnormal` 1556, 2291, 2650
- `\footnormalX` 30, 1959, 2184, 2308
- `\footnote@luatexpardir` 1435
- `\footnote@luatextextdir` . 1434, 1456
- `\footnoteA` 30
- `\footnoteB` 30
- `\footnoteC` 30
- `\footnoteD` 30
- `\footnoteE` 30
- `\footnotelang@lua` .. 1371, 2259, 2272
- `\footnotelang@poly` .. 1385, 2263, 2276
- `\footnoteoptions@`
 - 1358, 2264, 2268, 2277, 2282
- `\footnoterule` .. 1546, 1929, 2606, 3077
- `\footnotesize` 2469, 2919, 2920
- `\footparagraph` 17, 1583
- `\footparagraphX` 30, 2081
- `\footsplitskips`
 - . 1413, 1416, 1620, 1635, 1738,
 - 1785, 1891, 2000, 2051, 2122, 2136
- `\footthreecol` 17, 1717
- `\footthreecolX` 30, 2031
- `\foottwocol` 17, 1769
- `\foottwocolX` 30, 1980
- `\foottwocolX` 1980
- `\fullstop` 22, 426, 1451,
 - 1528, 1530, 1532, 1534, 2855, 2859
- G**
- `\g@addto@macro` 2185,
 - 2190, 2193, 2196, 2641, 2642,
 - 2651, 2654, 2657, 2659, 2666, 3145
- Gädeke, Nora 8
- `\get@index@command`
 - 3141, 3212, 3220, 3240
- `\get@linelistfile` 474, 489
- `\getline@num` 1020, 1052
- `\gl@p` .. 448, 480, 481, 851, 873, 890,
 - 1108, 1109, 1304, 1308, 1344, 2728
- `\gl@poff` 448, 449

H

`\hangafter` 3343
`\hangindentX` 20
`\hangingsymbol` 25, 26, 3286, 3293
`\hb@xt@` 1030, 1032,
 1045, 1047, 3774, 3779, 3783,
 3788, 3799, 3812, 3825, 3902, 3904
`\hfilneg` 1706
`\Hilfsbox` 3400
`\hilfsbox` 3400, 3455, 3456,
 3493, 3505, 3579, 3592, 3609,
 3623, 3637, 3650, 3665, 3790,
 3792, 3797, 3801, 3802, 3804,
 3810, 3814, 3815, 3817, 3823, 3826
`\hilfscount` 3400, 3940–3942, 3948
`\HILFSskip` 3933
`\Hilfsskip` 3780,
 3784, 3785, 3789, 3792, 3793,
 3800, 3801, 3804–3806, 3813,
 3814, 3817–3819, 3826–3828,
 3933, 3939, 3941, 3947, 3951, 3952
`\hilfsskip`
 . 3400, 3578, 3579, 3594, 3611,
 3625, 3639, 3652, 3667, 3950–3952
`\hsizethreecol` 20
`\hsizethreecolX` 20
`\hsizetwocol` 20
`\hsizetwocolX` 20
`\Hy@temp@A` 3176, 3177
`\HyInd@ParenLeft` 3177
`\hyperpage` 3143

I

`\if@edindex@fornote@` 3136,
 3166, 3178, 3188, 3211, 3219, 3241
`\if@edindex@fornote@true` 3136
`\if@fcolmade` 2696, 2705
`\if@firstcolumn` 1185, 1260, 2698, 3003
`\if@led@nofoot` 2662, 2683
`\if@ledgroup` 725, 3080
`\if@nobreak` 928
`\if@RTL` 23, 846, 857,
 868, 879, 1387, 1398, 1626, 1893
`\ifautopar` 166, 173, 180, 187, 194,
 201, 222, 235, 244, 257, 946, 990
`\ifbypage@` .. 302, 548, 559, 1086, 1502
`\ifbypstart@` 302, 972
`\ifcsdef` 23, 1694, 2504
`\ifcsempy`
 1446, 1659, 1696, 1750, 1797, 2508

`\ifcsequal` 2506
`\ifcsstring` 2333, 2345
`\ifdefstring` 1456
`\ifdim` 774, 776, 778, 780, 1827, 3455, 3940
`\ifdimequal` 1537, 1607,
 1918, 2104, 2513, 2520, 2534, 2548
`\iffirst@linenum@out@` 715, 719
`\ifFN@bottom` 2593, 2603
`\ifhbox` 1685, 1690
`\ifhmode` 1841, 1848, 1871, 1878
`\ifinserthangingsymbol` .. 3289, 4025
`\ifinstanza` 948, 1007, 3286, 3292, 4025
`\ifl@d@dash` 1474, 1531, 2856
`\ifl@d@elin` 1474,
 1520, 1533, 1534, 2846, 2858, 2859
`\ifl@d@esl` 1474, 1534, 2859
`\ifl@d@pnum`
 1474, 1508, 1528, 1532, 2834, 2857
`\ifl@d@ssub` 1474, 1530, 2855
`\ifl@dend@` 2811, 2817
`\ifl@dmemoir` 33, 203, 3466
`\ifl@dpairing`
 ... 121, 140, 270, 286, 1464, 3028
`\ifl@dskipnumber` 800, 1146
`\ifl@dstartendok` 3744, 3754
`\ifl@imakeidx` 36, 3210
`\iflabelpstart` 925, 956
`\ifledfinal` 4, 24, 36
`\iflednopbinverse` 7, 4024, 4025
`\ifledplinenum` 2468
`\ifledRcol` . 121, 993, 1465, 2257, 3029
`\ifleftnoteup` 2960, 2973
`\ifluatex` 1433, 1455, 2258, 2271
`\ifnoquotation@` 6, 218
`\ifnoteschanged@` 280, 466
`\ifnumberedpar@` ... 915, 938, 962,
 1854, 1860, 1941, 1953, 2256,
 2313, 2314, 2874, 2932, 2940, 2947
`\ifnumbering` 120,
 129, 267, 290, 305, 934, 959, 999
`\ifnumberingR` 121, 994
`\ifnumberline` 883, 1056, 1145
`\ifnumberpstart` .. 923, 947, 979, 1007
`\ifnumequal` 973, 1693, 1695, 2991
`\ifnumgreater` 2999
`\ifodd` 1195, 1270, 3013
`\ifparapparatus@` 4
`\ifpst@rtedL` 121
`\ifpstartnum` 1281, 1284, 1289
`\ifreportnoidxfile` 3151

- \ifrightnoteup 2927, 2968
 - \ifshowindexmark 3164, 3239
 - \ifsidepstartnum 949, 1255
 - \ifstrempy 2361, 2372
 - \IfStrEq 739, 749, 1021, 1035, 4005, 4014, 4027, 4031
 - \IfStrEqCase 763
 - \ifstrequal 1360, 1372, 1386, 2371
 - \ifsublines@ 424, 453, 537, 573, 578, 584, 599, 617, 626, 640, 662, 709, 711, 1057, 1094, 1149, 2744, 2768
 - \IfSubStr 3212, 3220, 3240
 - \iftoggle 1446, 1539, 1609, 1659, 1750, 1797, 1920, 2106, 2486, 2492, 2497, 2502, 2521, 2525, 2526, 2529, 2535, 2536, 2539, 2540, 2543, 2549, 2550, 2554, 2555, 2558
 - \ifvbox 969, 2575, 2637
 - \ifvmode 2729
 - \ifvoid 2173, 2178, 2188, 2191, 2197, 2596, 2623, 2630, 2635, 2652, 2655, 2660, 2667, 2668, 2675–2677, 3035, 3043, 3065, 3088, 3117
 - \imki@wrindexentry .. 3213, 3214, 3216
 - \indexentry 3222, 3225, 3229
 - \initnumbering@reg 128
 - \initnumbering@ssectcmd 141, 160
 - \inplaceoflemmaseparator 19
 - \inplaceofnumber 18
 - \InputIfFileExists 490
 - \insert 1410, 1616, 1736, 1783, 1888, 1998, 2049, 2118, 2178, 2192, 2630, 2635, 2637, 2656
 - \insert@count 685, 686, 768, 842, 864, 1362, 1374, 1376, 1389, 1392, 1395, 1864, 1945, 2281, 2935, 2943, 2950
 - \insert@countR 1366, 1380, 1382, 1400, 1403, 1406, 2269
 - \inserthangingsymbol 1032, 3290
 - \inserthangingsymbolfalse 1025
 - \inserthangingsymboltrue 1023
 - \inserthangingymbol 3289
 - \insertlines@list 273, 457, 486, 693, 1300, 1304
 - \insertparafootsep .. 1655, 1692, 2149
 - \inserts@list 942, 1295, 1298, 1308, 1343, 1344, 1361, 1373, 1375, 1388, 1391, 1394, 1863, 1944, 2280, 2934, 2942, 2949
 - \inserts@listR 1365, 1379, 1381, 1399, 1402, 1405, 2267
 - \instanzafalse 3288, 3366
 - \instanzatruer 3349
 - \interfootnotelinepenalty 1417
 - \interlinepenalty 987, 1328, 1417, 3354
 - \interparanoteglue 2476
 - \ipn@skip 2476
 - \itemcount@ 2989, 2991, 2996, 2999
- J**
- Jayaditya 8
- K**
- Kabelschacht, Alois 95
 - Krukov, Alexej 71
- L**
- \l@d@wrindexhyp 3162, 3237
 - \l@d@add 905, 907, 911, 913
 - \l@d@dashfalse 1483, 1501, 2829
 - \l@d@dashtrue 1505, 1511, 1523, 2832, 2837, 2849
 - \l@d@elinfalse 1479, 1508, 2834
 - \l@d@elintrue .. 1508, 1510, 2834, 2836
 - \l@d@end 2312, 2811, 2813, 2814, 2820, 2871
 - \l@d@err@UnequalColumns 3536
 - \l@d@eslfalse 1481, 1517, 1520, 2843, 2846
 - \l@d@esltrue ... 1520, 1522, 2846, 2848
 - \l@d@index 3147, 3149, 3468
 - \l@d@makecol 2587, 2644, 2714
 - \l@d@nums . 841, 888, 891, 899, 900, 913, 2265, 2267, 2278, 2280, 2313
 - \l@d@pnumfalse 1475, 1501, 2829
 - \l@d@pnumtrue 1504, 2831
 - \l@d@reinserts 2634, 2645
 - \l@d@section 2820, 2822
 - \l@d@set 606, 794
 - \l@d@ssubfalse 1477, 1513, 2839
 - \l@d@ssubtrue 1515, 2841
 - \l@d@wrindexm@ 3161, 3162
 - \l@dampcount 3395, 3532, 3534, 3539, 3788, 3798, 3799, 3811, 3812, 3847, 3863, 3901
 - \l@dbegin@stack 3264, 3274–3276
 - \l@dbfnote 1855, 1859

- \l@dcheckcols 3489, 3501, 3529
- \l@dcheckstartend 3743, 3754
- \l@dchset@num 504, 507, 606
- \l@dcolcount 3395, 3437,
3449, 3450, 3488, 3490, 3500,
3502, 3530, 3534, 3539, 3584,
3586, 3601, 3603, 3617, 3618,
3631, 3632, 3644, 3645, 3659,
3660, 3710, 3711, 3788, 3798,
3799, 3811, 3812, 3843, 3845,
3859, 3861, 3901, 3907, 3937, 3946
- \l@dcollect@body 3266, 3273
- \l@dcollect@body
.... 3261, 3973–3975, 3977–3979
- \l@dcolwidth 3437, 3455, 3456,
3578, 3709, 3774, 3800, 3813,
3902, 3904, 3909, 3918, 3939, 3940
- \l@dcsnote 2924, 2927
- \l@dcsnotetext . 1039, 2955, 2998, 3001
- \l@ddodoreinxtrafeet 2625, 2636, 2642
- \l@dfootinsert 2588, 2594
- \l@dextrafeet 2611, 2614, 2641
- \l@dbeginmini 2657, 3020, 3023
- \l@dendmini 2659, 3021, 3023
- \l@demptyd@ta 1017, 1039
- \l@dend@close 2813, 2864
- \l@dend@false 2811, 2814
- \l@dend@open 2813, 2818
- \l@dend@stuff ... 138, 295, 2816, 2870
- \l@dend@true 2811, 2813
- \l@denvbody 3256, 3259, 3262–3264
- \l@dfambeginmini ... 2193, 3020, 3039
- \l@dfamendmini 2196, 3021, 3039
- \l@dfeetbeginmini
..... 3020, 3057, 3084, 3113
- \l@dfeetendmini 3020, 3068, 3091, 3120
- \l@dgetline@margin 331
- \l@dgetlock@disp 375, 403
- \l@dgetref@num 2770,
2771, 2773, 2774, 2776, 2777, 2784
- \l@dgetsidenote@margin 2881
- \l@dobblearg 3480
- \l@dobbledarg 3480
- \l@dlabel@parse 2790, 2793
- \l@dld@ta 1030, 1039, 1184, 1261, 1273
- \l@dldp@rbox 1045, 2912, 2959
- \l@dlsn@te 1031, 1044
- \l@dlsnote 2922, 2927
- \l@dmake@labels 2733, 2736, 2754, 2763
- \l@dmemoirfalse 34
- \l@dmemoirtrue 34
- \l@dmodforcritext 3542, 3982
- \l@dmodforedtext 3553, 3985
- \l@dnullfills 3563,
3832, 3850, 3866, 3876, 3884, 3894
- \l@dnumpstartsL 121, 145
- \l@dold@footnotetext 1835, 1837
- \l@dold@xympar 2872
- \l@doldold@footnotetext . 1852, 1868
- \l@dp@rsefootspec 1484, 3138
- \l@dpairingfalse 121
- \l@dpairingtrue 121
- \l@dparsedendline 1484, 3135
- \l@dparsedendpage 1484, 3135
- \l@dparsedendsub 1484
- \l@dparsedstartline 1484, 3134
- \l@dparsedstartpage 1484, 3134
- \l@dparsedstartsub 1484
- \l@dparsefootspec 1484
- \l@dpush@begins 3270, 3274
- \l@d@rd@ta 1032, 1039, 1184, 1263, 1271
- \l@dref@undefined
..... 2770, 2773, 2776, 2779
- \l@drestorefills 3563,
3836, 3852, 3870, 3878, 3888, 3896
- \l@drestoreforcritext ... 3542, 3983
- \l@dstoreforedtext 3553, 3986
- \l@drp@rbox 1047, 2912, 2967
- \l@d@rsn@te 1033, 1044
- \l@d@rsnote 2923, 2927
- \l@dsetmaxcolwidth .. 3454, 3495, 3507
- \l@dskipnumberfalse 800, 1147
- \l@dskipnumbertrue 800, 1138
- \l@dstartendokfalse . 3758, 3762, 3766
- \l@dstartendoktrue 3756
- \l@dtabaddcols 3742, 3773
- \l@dtabnoexpands 818, 3377
- \l@dunboxmpfoot 3066, 3074, 3089, 3118
- \l@dunhbox@line 1012
- \l@dzeropenalties 965, 985
- \l@imakeidxfalse 37
- \l@imakeidxtrue 37
- \l@prev@nopb 149, 3992
- \l@prev@pb 148, 3991
- Lück, Uwe 6
- \label 28
- \label@refs 2726, 2728,
2733, 2736, 2742, 2745, 2747, 2749
- \labelpstartfalse 920
- \labelpstarttrue 920

- `\labelref@list` . 2719, 2725, 2728, 2768
- `\labelrefsparseline` 2739
- `\labelrefsparsesubline` 2739
- `\last@page@num` 554, 4026
- `\lastbox` . 1005, 1019, 1647, 1684, 1689
- `\lastkern` 1827
- `\lastskip` 773, 777
- Lavagnino, John 5, 7
- Leal, Jeronimo@Leal, Jerónimo 6
- `\led@check@nopb` 1021, 1035, 4003
- `\led@check@pb` 1021, 1035, 4003
- `\led@err@AutoparNotNumbered` 79, 995, 1000
- `\led@err@HighEndColumn` . . . 106, 3763
- `\led@err@LineationInNumbered` . . . 56, 306
- `\led@err@LowStartColumn` . . 106, 3759
- `\led@err@NumberingNotStarted` . . . 40, 284
- `\led@err@NumberingShouldHaveStarted` 40, 297
- `\led@err@NumberingStarted` . . . 40, 130
- `\led@err@PendNoPstart` 79, 963
- `\led@err@PendNotNumbered` . . . 79, 960
- `\led@err@PstartInPstart` . . . 79, 939
- `\led@err@PstartNotNumbered` . . 79, 935
- `\led@err@ReverseColumns` . . 106, 3767
- `\led@err@TooManyColumns` . . 106, 3451
- `\led@err@UnequalColumns` 106
- `\led@mess@NotesChanged` 46, 281
- `\led@mess@SectionContinued` . . 54, 293
- `\led@nopb` 3995, 3997
- `\led@nopbnum` 3996, 3997
- `\led@pb` 3993, 3997
- `\led@pb@setting` 739, 749, 763, 1021, 1035, 4001, 4005, 4014, 4027, 4031
- `\led@pbnum` 3994, 3997
- `\led@warn@BadAction` 93, 1140
- `\led@warn@BadAdvancelineLine` . . . 69, 593
- `\led@warn@BadAdvancelineSubline` 69, 587
- `\led@warn@BadLineation` 59, 326
- `\led@warn@BadLinenummargin` . . 59, 354
- `\led@warn@BadLockdisp` 59, 381
- `\led@warn@BadSetline` 75, 785
- `\led@warn@BadSetlinenum` . . . 75, 792
- `\led@warn@BadSidenotemargin` . . 102, 2904
- `\led@warn@BadSubblockdisp` . . . 59, 407
- `\led@warn@DuplicateLabel` . . . 95, 2757
- `\led@warn@NoIndexFile` 104, 3152
- `\led@warn@NoLineFile` 67, 495
- `\led@warn@NoMarginpars` . . . 100, 2875
- `\led@warn@RefUndefined` 95, 2781
- `\ledchapter` 160
- `\ledchapter*` 160
- `\ledfinalfalse` 11
- `\ledfinaltrue` 10
- `\ledfootinsdim` 1556, 2245, 2246
- `ledgroup` (environment) 26, 3080
- `ledgroupsize` (environment) . . 26, 3095
- `\ledinnote` 3142
- `\ledinnotehyperpage` 3142
- `\ledleftnote` 29, 2922
- `\ledlinenum` 419
- `\ledllfill` 1032, 1049, 3099, 3103
- `\ledlsnotefontsetup` . . . 29, 2915, 2958
- `\ledlsnotesep` 29, 1045, 2915
- `\ledlsnotewidth` 29, 2915, 2958
- `\lednopb` 36, 3993
- `\lednopbinversetrue` 13, 36
- `\lednopbnum` 3993, 4033
- `\ledpb` 36, 3993
- `\ledpbnum` 3993, 4029
- `\ledpbsetting` 36, 4001
- `\ledplinenumtrue` 2471
- `\ledrightnote` 29, 2922
- `\ledrlfill` 1032, 1049, 3100, 3107
- `\ledrsnotefontsetup` . . . 29, 2915, 2966
- `\ledrsnotesep` 29, 1047, 2915
- `\ledrsnotewidth` 29, 2915, 2966
- `\ledsection` 160
- `\ledsection*` 160
- `\ledsectnotoc` 35, 265
- `\ledsetnormalparstuff` 1432, 1656, 1909, 2150
- `\ledsidenote` 29, 2922
- `\ledsubsection` 160
- `\ledsubsection*` 160
- `\ledsubsubsection` 160
- `\ledsubsubsection*` 160
- `\left` 3717, 3720, 3725, 3728
- `\leftctab` 3787, 3885
- `\leftlinenum` 13, 419, 1186, 1198
- `\leftltab` 3778, 3867
- `\leftmargin` 227, 228, 249, 250
- `\leftnoteupfalse` 29
- `\leftnoteuptrue` 2974
- `\leftpstartnum` 1255
- `\leftrtab` 3782, 3833
- Leibniz 8
- `\lemma` 15, 897
- `\lemmaseparator` 19, 2475

\letsforverteilen	3571,	M
3593, 3610, 3624, 3638, 3651, 3666		\m@m@makecolfloats
\line@list	276, <u>457</u> , 485, 711, 886, 890	2570, 2589
\line@list@stuff	137, 294, <u>717</u>	\m@m@makecolintro
\line@margin	<u>331</u> , 1191, 1266	2570, 2590
\line@num	150, 423, <u>451</u> , 509,	\m@m@dooreinextrafeet
543, 549, 560, 591, 592, 594,		2642
602, 607, 608, 620, 704, 708,		\m@m@doextrafeet
1063, 1087, 1097, 1162, 1164,		2641
1165, 1174, 1175, 1695, 2767, 2999		\m@m@mf@check
\line@set	901, <u>902</u>	1826, 1843, 1873
\line@tion	12, 57, 60, <u>304</u>	\m@m@mf@prepare
\line@info@	2487, 2490, 2560	1823, 1838, 1847, 1877, 2299
\linenum	16,	\m@th
898, 2806, 3479, 3547, 3558, 3577		3735
\linenum@out	<u>714</u> , 720, 722, 726,	\makehboxofhboxes
727, 729, 730, 733, 734, 743,		1666, 1676, <u>1681</u> , 2156, 2165
745, 747, 756, 758, 760, 763,		\makeindex
767, 770, 775, 779, 782, 787,		<u>3145</u> , <u>3202</u>
794, 797, 798, 804, 2724, 3993–3996		\makememindexhook
\linenumberlist	12, <u>30</u> , 1163, 1175	3145
\linenumberstyle	14, <u>410</u>	\managestanza@modulo
\linenumberincrement	11, 12, <u>370</u>	<u>3318</u> , 3337
\linenummargin	12, 62, <u>331</u>	\marginparwidth
\linenumr@p	<u>410</u>	2915, 2916
\linenumrep	<u>410</u> ,	\mathchardef
423, 1529, 1533, 2767, 2854, 2858		3313
\linenumsep		\maxdepth
.. 13, <u>419</u> , 1285, 1290, 2917, 2918		2591
\lineref	27, <u>2773</u> , 3133	\maxdimen
\linewidth	1030	1621, 1636, 2123, 2137
\list@clear	<u>437</u> , 485–488, 942	\maxhnotesX
\list@clearing@reg	473, <u>484</u>	21
\list@create		\maxhXnotes
... <u>436</u> , 457–460, 807, 1295, 2719		21
\listadd	2333, 2345	Mayer, Gyula
\listcsxadd	565, 3997–4000	8
\listead	2332, 2347	\measuremb
\listgadd	2955	3835, <u>3841</u> , 3869, 3887
\listxadd	2325	\measuremc
\lock@disp	375, 1227, 1231, 1235	<u>3487</u> , 3513
\lock@off	633, 634, <u>660</u> , 798	\measuremrow
\lock@on	<u>631</u> , 797	<u>3511</u> , 3846
\lockdisp	13, 64, <u>375</u>	\measuretbody
Lorch, Richard	8	3851, <u>3857</u> , 3877, 3895
\ltab	3380, <u>3865</u> , 3973	\measuretbody
\ltabtext	3382, <u>3875</u> , 3977	<u>3499</u> , 3518
\luatexpardir	1375, 1381, 1435	\measuretbody
\luatextextdir	1373, 1379, 1434	<u>3516</u> , 3862
Luecking, Dan	43	\message
		55, 136
		Middleton, Thomas
		8, 58
		minipage (environment)
		26
		Mittelbach, Frank
		7
		\morenoexpands
		38, <u>810</u>
		\moveleft
		3780, 3785, 3793
		\moveright
		3806, 3819, 3828
		\mpnormalfootgroup
		<u>1549</u> , 1577
		\mpnormalfootgroupX
		<u>1934</u> , 1974
		\mpnormalvfootnote
		1422, 1576, 1723, 1775
		\mpnormalvfootnoteX
		1895, 1973, 1986, 2037
		\mppara@footgroup
		1593, <u>1671</u>
		\mppara@footgroupX
		2091, 2154
		\mppara@vfootnote
		1592, <u>1630</u>
		\mppara@vfootnoteX
		2090, <u>2117</u>
		\mpthreecolfootgroup
		1724, <u>1760</u>
		\mpthreecolfootgroupX
		2038, <u>2069</u>
		\mpthreecolfootsetup
		1725, <u>1731</u>
		\mpthreecolfootsetupX
		2039, <u>2041</u>
		\mptwocolfootgroup
		1776, 1808

- \mptwocolfootgroupX 1987, 2019
- \mptwocolfootsetup 1777, 1808
- \mptwocolfootsetupX 1988, 1990
- \multfootsep 29, 1820, 1830
- \multiplefootnotemark
 1820, 1824, 1825, 1827
- N**
- \n@num 681, 804
- \n@num@reg 681
- \nc@page 755, 756
- \NeedsTeXFormat 2
- \new 2331–
 2333, 2335, 2344, 2345, 2347, 2348
- \new@line 738, 1032
- \newbox 915, 918,
 2912, 2913, 3400, 3402, 3970, 3971
- \newcommandx 1358,
 1371, 1385, 1441, 1645, 1654,
 1740, 1787, 2368, 2460, 2475, 3209
- \newcounter
 . 361, 363, 365, 367, 921, 1070,
 2302, 2739, 2740, 3126, 3321, 3740
- \newhookcommand@series 2379
- \newhookcommand@series@reload . 2442
- \newhooktoggle@series
 2456, 2459, 2462–2467
- \newif 4–7, 23, 33,
 36, 120, 121, 123, 126, 127, 302,
 303, 453, 466, 715, 800, 883,
 916, 923, 925, 990, 1256, 1281,
 1474, 1476, 1478, 1480, 1482,
 2470, 2593, 2662, 2812, 2927,
 2973, 3080, 3136, 3287, 3289, 3754
- \newinsert 2248–2251
- \newlength 419, 3305
- \newlinechar 2310
- \newread 471
- \newseries 2199, 2327, 2328
- \newseries@ 2200, 2204
- \newtoggle 1557, 1561,
 2224–2227, 2229, 2232, 2473, 2474
- \newwrite 714, 2811
- \NEXT 3483,
 3488, 3491, 3496, 3497, 3500,
 3503, 3508, 3509, 3512, 3514,
 3515, 3517, 3519, 3520, 3525,
 3672, 3675, 3677, 3678, 3680,
 3682, 3683, 3686, 3688, 3689,
 3691, 3693, 3694, 3697, 3699,
 3700, 3702, 3704, 3705, 3710,
 3712, 3713, 3907, 3910, 3911,
 3916, 3920, 3921, 3937, 3943, 3944
- \Next 3525, 3585,
 3587, 3596, 3597, 3602, 3604,
 3613, 3614, 3617, 3619, 3626,
 3627, 3631, 3633, 3640, 3641,
 3644, 3646, 3654, 3655, 3659,
 3661, 3669, 3670, 3925, 3927, 3928
- \next@absline 751, 752
- \next@action 94, 481, 1076,
 1084, 1085, 1091, 1092, 1100, 1109
- \next@actionline
 . 478, 480, 1075, 1083, 1106, 1108
- \next@insert
 943, 1299, 1302, 1304, 1307, 1311
- \next@page@num
 155, 512, 514, 552, 564, 614
- \no@expands 810, 840, 862, 897
- \noalign 1709
- \noendnotes 27, 2870
- \noindent
 . 1007, 1253, 1439, 1545, 1547,
 1553, 1614, 1622, 1626, 1637,
 1669, 1679, 1756, 1764, 1803,
 1815, 1893, 2124, 2138, 2159, 2168
- \nolemmaseparator 19, 2475
- \nonbreakableafternumber 18
- \nonum@ 2473
- \nonumberinfootnote 18
- \noquotation@true 9
- \normal@footnotemarkX 1880, 1962
- \normal@page@break 147, 3990
- \normal@pars 269, 944,
 1010, 1438, 1741, 1788, 2008, 2059
- \normalbfnoteX 1940, 1954
- \normalbodyfootmarkX 1885, 1963
- \normalcolor 1551, 1673, 1762, 1813,
 1936, 2025, 2075, 2162, 2605, 3076
- \normalfont 421, 1821, 1886, 2468
- \normalfootfmt 1432, 1569
- \normalfootfmtX 1905, 1966
- \normalfootfootmarkX 1914, 1967
- \normalfootgroup 1547, 1570
- \normalfootgroupX 1931, 1968
- \normalfootnoterule 1546, 1572
- \normalfootnoteruleX 1929, 1969, 2088
- \normalfootstart 1536, 1567
- \normalfootstartX 1917, 1961
- \normalvfootnote 1409, 1568

- \normalvfootnoteX [1887](#), [1964](#)
 - \nosep@ [2474](#)
 - \notblank [1369](#)
 - \notbool [1409](#), [1422](#),
1441, 1735, 1740, 1783, 1787, 2253
 - \notfontsetup
.... [1755](#), 2216–2218, [2468](#), [2478](#)
 - \notfontsizeX [19](#)
 - \notenumfont 2213–2215, [2468](#)
 - \notenumfontX [19](#)
 - \noteschanged@false [466](#), [491](#)
 - \noteschanged@true
.... [274](#), [277](#), [466](#), [496](#), [887](#), [1301](#)
 - \nulledindex [3465](#), [3546](#), [3557](#),
3583, 3600, 3616, 3630, 3643, 3658
 - \nullsetzen [3707](#), [3844](#), [3860](#)
 - \num@lines . [915](#), [966](#), [1318](#), [1324](#), [1327](#)
 - \numberedpar@false [915](#)
 - \numberedpar@true [915](#), [955](#)
 - \numberingfalse [120](#), [268](#)
 - \numberingtrue [120](#), [133](#), [288](#)
 - \numberlinefalse [12](#)
 - \numberlinetrue [12](#), [884](#)
 - \numberonlyfirstinline [18](#)
 - \numberonlyfirstintwolines [18](#)
 - \numberpstartfalse [12](#), [920](#)
 - \numberpstarttrue [11](#), [920](#)
 - \numdef [751](#), [2989](#), [2996](#), [4007](#)
 - \numgdef [742](#), [755](#), [4028](#), [4032](#)
 - \numlabfont [22](#), [419](#)
- O**
- \one@line [915](#), [1018](#), [1019](#), [1032](#)
 - \onlypstartinfootnote [18](#)
 - \openout [722](#), [727](#), [730](#), [734](#), [2813](#)
- P**
- \p@pstart [957](#)
 - \PackageError [39](#)
 - \PackageWarning [38](#)
 - \page@action [513](#), [612](#), [698](#)
 - \page@num [462](#), [476](#),
551, 562, 703, 708, 1085, 1193,
1268, 1693, 1702, 2999, 3011, 4026
 - \page@start [771](#), [2595](#)
 - \pagebreak [4003](#)
 - \pagelinesep [31](#), [3124](#), 3133–3135
 - \pageno [96](#), [98](#), [2566](#)
 - \pageparbreak [37](#), [1253](#)
 - \pageref [28](#)
 - \par@line
.. [915](#), [967](#), [1319](#), [1320](#), [1323](#), [1327](#)
 - \para@footgroup [1589](#), [1664](#)
 - \para@footgroupX [2087](#), [2154](#)
 - \para@footsetup [1591](#), [1597](#)
 - \para@footsetupX [2093](#), [2095](#)
 - \para@vfootnote [1587](#), [1615](#)
 - \para@vfootnoteX [2085](#), [2117](#)
 - \parafootfmt [1588](#), [1654](#)
 - \parafootfmtX [2086](#), [2145](#)
 - \parafootftmsep [2241](#), [2480](#)
 - \parafootsep [20](#)
 - \parafootstart [1586](#), [1605](#)
 - \parafootstartX [2084](#), [2103](#)
 - \parapparatus@false [8](#)
 - \parapparatus@true [12](#)
 - \patchcmd . [205](#), [208](#), [209](#), [212](#), [216](#), [217](#)
 - \pausenumbering [11](#), [287](#)
 - \pend [10](#), [86](#), [89](#), [162](#), [164](#), [169](#),
171, 176, 178, 183, 185, 190,
192, 197, 199, 203, 204, 215,
940, 959, 1008, 1253, 3363, 3366
 - Plato of Tivoli [8](#)
 - \postbodyfootmark [1869](#), [1883](#)
 - \postdisplaypenalty [988](#)
 - \prebodyfootmark [1869](#), [1881](#)
 - \predisdisplaypenalty [987](#)
 - \prenotesX [21](#), [1564](#)
 - \prenotesX@ [1563](#), [1564](#), [1918](#), [2104](#)
 - \prepare@edindex@fornote
..... [2265](#), [2278](#), [3137](#)
 - \pretocmd [206](#), [210](#), [3202](#)
 - \prev@nopb [3991](#)
 - \prev@pb [3991](#)
 - \prevgraf [966](#)
 - \prevline [1694](#), [2505](#)
 - \prevpage@num [1692](#)
 - \preXnotes [21](#), [1557](#), [1561](#)
 - \preXnotes@ [1537](#), [1557](#), [1561](#), [1607](#)
 - \printendlines [2822](#), [2852](#)
 - \printlinefootnote
.... [1444](#), [1657](#), [1748](#), [1795](#), [2481](#)
 - \printlines [1526](#),
2522, 2526, 2536, 2540, 2550, 2555
 - \printnpnum [27](#), [2854](#), [2857](#), [2862](#)
 - \printpstart [1463](#),
2521, 2525, 2535, 2539, 2549, 2554
 - \processl@denvbody
..... [3263](#), [3267](#), [3268](#), [3283](#)
 - \ProcessOptions [15](#)

- `\protected@csxdef` 1942, 2297
`\protected@edef`
 956, 1906, 2005, 2055, 2146
`\protected@write`
 . 2735, 3167, 3169, 3172, 3179,
 3181, 3184, 3189, 3191, 3194,
 3221, 3224, 3228, 3242, 3244, 3247
`\ProvidesPackage` 3
`\pst@rtedLfalse` 121, 144, 271
`\pst@rtedLtrue` 121, 291
`\pstart` 10, 80, 83, 84,
 89, 163, 166, 170, 173, 177, 180,
 184, 187, 191, 194, 198, 201,
 203, 204, 215, 920, 1007, 1253, 3339
`\pstartinfootnote` .. 18, 312, 318, 324
`\pstartline` 970, 973
`\pstartnum` 1255
`\pstartnumfalse` 1286, 1293
`\pstartnumtrue` 980, 1282
- Q**
- `\quotation` 160
`\quote` 160
- R**
- `\raggedright`
 1745, 1792, 2014, 2064, 2920
`\raw@text` 915, 945, 969, 1018
`\rbracket` 22, 1451, 2236
`\read@linelist` 471, 718
`\ref` 28
`\Relax` 3483, 3924, 3931
`\rem@inder` 1175, 1177–1179
`\removehboxes`
 1667, 1677, 1681, 2157, 2166
`\removelastskip` 3584, 3601
`\RequirePackage` 17, 18, 20–22
`\reserveinserts` 19
`\resetprevline@` 61, 157, 467, 973, 1088
`\resumenumbering` 11, 287
`\right` 3718, 3721, 3726, 3729
`\rightctab` 3796, 3886
`\rightlinenum` 13, 419, 1188, 1196
`\rightltab` 3809, 3868
`\rightnoteupfalse` 29
`\rightnoteuptrue` 2928
`\rightpstartnum` 1263, 1271, 1288
`\rightrtab` 3822, 3834
`\rightstartnum` 1255
- `\rigidbalance` ... 1704, 1759, 1767,
 1806, 1818, 2022, 2029, 2072, 2079
`\rlap` 1188, 1196, 1263, 1271
 Robinson, Peter 6
`\roman` 3741
`\rtab` 3378, 3831, 3975
`\rtabtext` 3381, 3849, 3979
- S**
- Sacrobosco 8
`\sc@n@list` 1176, 1178
 Schöpf, Rainer 7
`\section@num`
 117, 134, 136, 137, 292–294, 2820
`\select@lemmafонт` 811, 1353
`\select@lemmafont` 22,
 1353, 1445, 1658, 1749, 1796, 2823
`\series` .. 2199, 2330, 2332, 2343, 2347
`\seriesatbegin` 2329
`\seriesatend` 2339, 2342
`\set@line` 841, 863, 885
`\set@line@action` 506, 597, 604, 615, 700
`\setcommand@series` 2368
`\setldlp@rbox` . 2953, 2957, 3004, 3016
`\setldrpr@rbox` . 2954, 2965, 3006, 3014
`\setldrpr@box` 2957
`\setline` 13, 76, 783, 815, 973
`\setlinenum` 13, 78, 790
`\setmcellcenter` 3642, 3698
`\setmcellleft` 3615, 3687
`\setmcellright` 3582, 3676
`\setmrowcenter` 3696, 3890
`\setmrowleft` 3685, 3872
`\setmrowright` 3674, 3838
`\setprintendlines` 2828, 2853
`\setprintlines` 1500, 1527
`\setstanzaindents` 23, 3318
`\setstanzapenalties` 24, 3318
`\setstanzavalues` 3308, 3318, 3319, 3375
`\settccllcenter` 3657, 3703
`\settccllleft` 3629, 3692
`\settccllright` 3599, 3681
`\settoggle` 1361, 1365, 2360
`\settoggle@series` .. 2355, 2359, 2460
`\setthrowcenter` 3701, 3898
`\setthrowleft` 3690, 3880
`\setthrowright` 3679, 3854
`\Setzen` 3915, 3924, 3926
`\showlemma` 24, 36, 849, 871
`\sidenote@margin` 2881, 3009

- \sidenotecontent@ 2988,
2993, 2994, 3004, 3006, 3014, 3016
 - \sidenotemargin 29, 2881
 - \sidenotemmargin 103
 - \sidenotesep 29, 2975
 - \skip 1540, 1543,
1550, 1575, 1580, 1610, 1613,
1672, 1761, 1812, 1921, 1924,
1935, 1972, 1977, 2024, 2074,
2107, 2110, 2114, 2161, 2603, 3075
 - \skip@lockoff 634, 660
 - \skipnumbering . . 14, 162, 164, 169,
171, 176, 178, 183, 185, 190,
192, 197, 199, 203, 204, 213,
214, 221, 234, 243, 256, 800, 3361
 - \skipnumbering@reg 800
 - \spacefactor
1828, 1831, 1842, 1848, 1872, 1878
 - \spaceskip 1414, 1892, 2001
 - \splitmaxdepth 1419
 - \splitoff 1704
 - \splittopskip . . . 1015, 1419, 1706,
1757, 1759, 1765, 1767, 1804,
1806, 1816, 1818, 2020, 2022,
2027, 2029, 2070, 2072, 2077, 2079
 - \spreadmath 32, 3903
 - \spreadtext 32, 3901
 - \stanza 23, 3347
 - \stanza@count . . . 3299, 3310, 3313,
3315, 3332, 3344, 3351, 3360, 3364
 - \stanza@hang 3330, 3353
 - \stanza@line 3330, 3364, 3367
 - \stanza@modulo
3322, 3325–3327, 3335, 3351, 3359
 - \stanzaindentbase
. 23, 3299, 3333, 3336, 3342, 3369
 - \startlock 13, 797, 813, 3340
 - \startstanzahook 25, 3347
 - \startsub 13, 773, 812
 - \stepcounter
. 2296, 2743, 2746, 3129, 3749
 - \step1@dcolcount 3449, 3494,
3506, 3591, 3608, 3622, 3636,
3649, 3664, 3708, 3908, 3917, 3938
 - \strip@pt 1603, 2101
 - \strip@szacnt 3308
 - \sub@action 522, 624, 699
 - \sub@change 156,
516, 517, 523, 574, 576, 579, 581
 - \sub@lock 153, 455, 531, 533,
535, 538, 642, 643, 645, 646,
664, 665, 667, 1058, 1130, 1132,
1133, 1135, 1210, 1246, 1248, 1250
 - \sub@off 573, 779
 - \sub@on 573, 775
 - \subline@num . . . 151, 425, 426, 452,
539, 543, 549, 560, 585, 586,
588, 600, 618, 705, 709, 1059,
1064, 1087, 1095, 1150–1152, 2768
 - \sublinenumberstyle 14, 410
 - \sublinenumincrement 11, 12, 370
 - \sublinenumr@p 410
 - \sublinenumrep 410,
426, 1530, 1534, 2768, 2855, 2859
 - \sublineref 27, 2776
 - \sublines@false . . . 154, 453, 520, 1120
 - \sublines@true 453, 518, 1118
 - \sublock@disp . . . 401, 1212, 1216, 1220
 - \sublockdisp 66, 401
 - \subsection 178, 185, 2352
 - \subsubsection 192, 199, 2354
 - Sullivan, Wayne 7,
8, 23, 37, 44, 51, 106, 107, 140, 160
 - \symlinenumber 18
 - \symlinenum 2228, 2468
 - \sza@penalty 3330, 3357, 3363
- T**
- \tabellzwischen 3906, 3914
 - \tabelskip 3918, 3958–3960, 3966–3968
 - \tabHilfbox 3957,
3959, 3961, 3965, 3967, 3969, 3970
 - \tabhilfbox 3956,
3958, 3960, 3964, 3966, 3968, 3970
 - Tapp, Christian 6
 - \textheight 2701
 - \textnormal 1451–1453
 - \textsuperscript . . . 1821, 1886, 1915
 - \textwidth 3053, 3097
 - \theadcolcount 3740, 3747, 3750
 - \theendpageline 3134,
3170, 3182, 3192, 3214, 3225, 3245
 - \thefootnoteA 30
 - \thelabidx 3130, 3133
 - \theline 2747, 2749
 - \thempfn 3055, 3082, 3111
 - \thempfootnote 3055, 3082, 3111
 - Theodosius 8

- `\thepage` 742,
 745, 747, 758, 760, 763, 2736, 3133
`\thepageline` 3132,
 3173, 3185, 3195, 3216, 3229, 3248
`\thepstart` 12, 920, 922,
 950, 957, 1007, 1284, 1291, 1471
`\thepstartL` 1468
`\thepstartR` 1466
`\thestartpageline` 3134,
 3168, 3180, 3190, 3213, 3222, 3243
`\thesubline` 2747
`\thinspace` 1456, 1458
`\thr@@` 352, 645, 654, 665, 672,
 1125, 1133, 1730, 1733, 1759,
 1767, 2043, 2046, 2072, 2079, 2902
`\threecolfootfmt` 1720, 1740
`\threecolfootfmtX` 2034, 2054
`\threecolfootgroup` 1721, 1755
`\threecolfootgroupX` 2035, 2069
`\threecolfootsetup` 1722, 1728
`\threecolfootsetupX` 2036, 2041
`\threecolvfootnote` 1719, 1735
`\threecolvfootnoteX` 2033, 2048
`\togglefalse` ... 1540, 1610, 1921, 2107
`\toggletrue` 1558, 1562
`\tolerance` 1744, 1791, 2013, 2063
`\twocolfootfmt` 1772, 1780
`\twocolfootfmtX` 1983, 2004
`\twocolfootgroup` 1773, 1780
`\twocolfootgroupX` 1984, 2019
`\twocolfootsetup` 1774, 1780
`\twocolfootsetupX` 1985, 1990
`\twocolvfootnote` 1771, 1780
`\twocolvfootnoteX` 1982, 1997
`\txtbeforeXnotes` 21
- U**
- `\unhbox` 1012, 1648, 1667, 1669, 1677,
 1679, 1686, 1690, 2157, 2159,
 2166, 2168, 3960, 3961, 3968, 3969
`\unkern` 1829
`\unpenalty` 1651, 1683
`\unvbox` ... 1019, 1424, 1547, 1554,
 1632, 1646, 1665, 1675, 1715,
 1897, 1932, 1938, 2133, 2155,
 2164, 2172, 2178, 2187, 2192,
 2581, 2602, 2607, 2620, 2630,
 2635, 2637, 2656, 2684, 3072, 3078
`\unvxh` ... 1623, 1638, 1645, 2125, 2139
`\usingcritext` 3981
- `\usingedtext` 3981
- V**
- `\valign` 1707
`\value` 3326, 3331, 3746
 Vamana 8
`\variab` 3527, 3837,
 3853, 3871, 3879, 3889, 3897, 3930
`\vbadness` 1014, 1706
`\vbfnoteX` 1943, 1948
`\vbox` 945, 1423, 1621, 1631,
 1636, 1646, 1896, 2123, 2132,
 2137, 2171, 2186, 2578, 2599,
 2619, 2709, 2713, 2961, 2963,
 2969, 2971, 3050, 3717, 3720,
 3725, 3728, 3732, 3734, 3735,
 3779, 3783, 3788, 3799, 3812, 3825
`\vfil` 1707, 2713,
 3718, 3721, 3726, 3729, 3732, 3735
`\vfill` 2603
`\vl@dbfnote` 1859
`\vl@dcsnote` 2948, 2953
`\vl@dlsnote` 2933, 2953
`\vl@drsnote` 2941, 2953
`\vnumfootnoteX` 1952, 1965
`\vrule` ... 3717, 3720, 3725, 3728, 3732
`\vsize` 1556
`\vsplit` 1018, 1714, 2684
- W**
- `\wd` 1007, 1032, 1625,
 1640, 2127, 2141, 3455, 3456,
 3579, 3792, 3801, 3804, 3814,
 3817, 3826, 3958, 3959, 3966, 3967
 Whitney, Ron 7
`\widowpenalty` 988, 1325
`\WithSuffix` 168, 182, 196, 204
 Wujastyk, Dominik 5, 7
- X**
- `\x@lemma` 851–853, 873–875
`\xcritext` 3459, 3572
`\xedindex` 3465, 3551, 3562, 3574
`\xedlabel` 3463, 3580
`\xedtext` 3459, 3573
`\Xendnotefontsize` 20
`\Xendnotenumfont` 19
`\Xhangindent` 20
`\xifinlist` 2205

<code>\xifinlistcs</code>	740, 741, 752,	707, 1361, 1365, 1373, 1375,
	753, 4003, 4006, 4008, 4015, 4016	1379, 1381, 1388, 1391, 1394,
<code>\xleft@appenditem</code>	<u>444</u>	1399, 1402, 1405, 1862, 1943,
<code>\xlineref</code>	28, <u>2773</u>	2265, 2278, 2766, 2933, 2941, 2948
<code>\Xnotefontsize</code>	19	<code>\xspaceskip</code> 1414, 1892, 2001
<code>\Xnotenunfont</code>	19	<code>\xsublineref</code> 28, <u>2776</u>
<code>\xpageref</code>	28, <u>2770</u>	<code>\xxref</code> 28, <u>2801</u>
<code>\xright@appenditem</code>		
 <u>438</u> , 613, 614, 616, 623,	
	625, 627, 629, 639, 641, 650,	
	661, 663, 670, 683, 684, 693,	
		Z
		<code>\z@skip</code> 1414, 1420, 1892, 2001
		<code>\zz@@@</code> <u>2720</u> , 2726, 2803, 2805

Change History

v0.1.		<code>\l@ddofootinsert</code> : Renamed
General: First public release	1	<code>\dofootinsert</code> as <code>\l@ddofootinsert</code>
v0.2.	 136
General: Added tabmac code, and		<code>\m@m@makecolintro</code> : Added
extended indexing	1	<code>\m@m@makecolfloats</code> , <code>\m@m@makecoltext</code>
<code>\eledmac@error</code> : Added <code>\eledmac@error</code>		and <code>\m@m@makecolintro</code> . . . 135
and replaced error messages . .	44	<code>\morenoexpands</code> : Removed some
<code>\ifl@dmemoir</code> : Added <code>\ifl@dmemoir</code>		<code>\lets</code> from <code>\no@expands</code> . These
for memoir class having been		were in EDMAC but I feel that
used	44	they should not have been as
<code>\morenoexpands</code> : Added <code>\l@dtabnoexpands</code>		they disabled page/line refs in
to <code>\no@expands</code>	76	a footnotes 76
v0.2.1.		<code>\zz@@@</code> : Minor change to <code>\zz@@@</code> . 140
<code>\@lab</code> : Removed page setting from		v0.2.2.
<code>\@lab</code>	142	General: Improved paragraph foot-
General: Added text about normal		notes 1
labeling	28	New Dekker example 1
Bug fixes and match with mem-		<code>\footfudgefiddle</code> : Added
patch v1.8	1	<code>\footfudgefiddle</code> 105
Major changes to insert code		<code>\l@d@section</code> : Used <code>\providecommand</code>
when memoir is loaded	138	for <code>\@gobblethree</code> to avoid
<code>\doxtrafeet</code> : Renamed <code>\doxtrafeet</code>		clash with the amsfonts pack-
to <code>\l@ddoxtrafeet</code>	137	age 145
<code>\edlabel</code> : Tweaked <code>\edlabel</code> to		<code>\line@list@stuff</code> : Added ini-
get correct page numbers . .	140	tial write of page number in
<code>\l@d@makecol</code> : Rewrote <code>\@makecol</code> ,		<code>\line@list@stuff</code> 70
calling it <code>\l@d@makecol</code> . . .	136	<code>\para@footsetup</code> : Added
<code>\l@ddoreinxtrafeet</code> : Re-		<code>\footfudgefiddle</code> to
named <code>\dodoreinxtrafeet</code> to		<code>\para@footsetup</code> 105
<code>\l@ddoreinxtrafeet</code>	137	<code>\para@footsetupX</code> : Added

<code>\footfudgefiddle</code>	to	<code>\mpnormalvfootnote:</code>	Added
<code>\para@footsetupX</code> 122	<code>\mpnormalvfootnote</code> 98
v0.3.		<code>\showlemma:</code>	Added <code>\showlemma</code> . 43
<code>\@l@reg:</code>	Added a bunch of code to	v0.4.1.	
<code>\@l</code>	for handling <code>\setlinenum</code>	<code>\@opxtrafeetii:</code>	Added <code>\@opxtrafeetii</code>
<code>\@lab:</code>	Replaced <code>\the\line@num</code> 137	
	by <code>\linenumr@p\line@num</code> in	General:	Added code for changing
<code>\@lab</code> ,	and similar for sub-lines	<code>\@doclearpage</code> 138
General:	Includes <code>edstanza</code> and	Let <code>eledmac</code> take advantage of	
more 1	memoir's indexing 154
<code>\ledlinenum:</code>	Added <code>\linenumr@p</code>	Not released. Minor editorial im-	
and	<code>\sublinenum@rep</code>	provements and code tweaks	.. 1
to	<code>\leftlinenum</code> and	Only change <code>\@footnotetext</code>	
<code>\rightlinenum</code> 55	and <code>\@footnotemark</code> if memoir	
<code>\linenumberlist:</code>	Added	not used 115
<code>\linenumberlist</code>	mechanism . 44	<code>\doxtrafeetii:</code>	Changed
<code>\printendlines:</code>	Added <code>\linenumr@p</code>	<code>\doxtrafeetii</code>	code for easier
and	<code>\sublinenumr@p</code> to	extensions 137
<code>\printendlines</code> 146	v0.5.	
<code>\printlines:</code>	Added <code>\linenumr@p</code>	<code>\@footnotetext:</code>	Enabled regular
and	<code>\sublinenumr@p</code> to	<code>\footnote</code>	in numbered text 115
<code>\printlines</code> 101	<code>\@xympar:</code>	Eliminated <code>\marginpar</code>
<code>\sublinenumr@p:</code>	Added <code>\linenumberstyle</code>	disturbance 147
and <code>\sublinenumberstyle</code>	... 55	General:	Added left and right side
v0.3.1.		notes 147
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 147
<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 151	<code>\affixline@num</code>	to cater for
General:	Added <code>\showlemma</code> to	sidenotes 88
<code>\edtext</code> (and <code>\critext</code>)	... 78	<code>ledgroupsize:</code>	Only change
Added <code>minipage</code> , etc., support	. 1	<code>\hsize</code> in <code>ledgroupsize</code> envi-	
<code>ledgroupsize:</code>	Added <code>ledgroup-</code>	ronment otherwise page number	
sized environment 153	can be in wrong place 153
<code>\footnormal:</code>	Added <code>minpage</code> foot-	<code>\l@dgetsidenote@margin:</code>	Added
note setup to <code>\footnormal</code>	. 104	<code>\sidenotemargin</code>	and
<code>\if@ledgroup:</code>	Added <code>ledgroup</code> en-	<code>\sidenote@margin</code> 147
vironment 152	v0.6.	
<code>\ifparapparatus@:</code>	Added fi-	<code>\@l@reg:</code>	Added <code>\fix@page</code> to <code>\@l</code> 63
nal/draft options 43	Extended <code>\@l</code> to include the page	
<code>\l@dfeetendmini:</code>	Added	number 63
<code>\l@dfeetbeginmini</code> , <code>\l@dfeetendmini</code>		<code>\@lopR:</code>	Added <code>\@pend</code> , <code>\@pendR</code> ,
and all their supporting code	150	<code>\@lopL</code> and <code>\@lopR</code> in anticipa-	
<code>\mpnormalfootgroup:</code>	Added	tion of parallel processing	... 65
<code>\mpnormalfootgroup</code> 102	General:	Fixed long paragraphs
		looping 1

Fixed minor typos	1	<code>\l@dcsnotetext</code> : Added <code>\l@emptyd@ta</code>	
Prepared for eledpar package	1	<code>\l@ddofootinsert</code> : Deleted	84
<code>\fix@page</code> : Added <code>\last@page@num</code> and <code>\fix@page</code>	64	<code>\page@start</code> from <code>\l@ddofootinsert</code>	136
<code>\new@line</code> : Extended <code>\new@line</code> to output page numbers	70	<code>\l@dgetline@margin</code> : Added	
<code>\page@start</code> : Made <code>\page@start</code> a no-op	71	<code>\l@dgetline@margin</code>	53
<code>\vl@dbfnote</code> : Changed <code>\l@dbfnote</code> and <code>\vl@dbfnote</code> as originals could give incorrect markers in the footnotes	115	<code>\l@dgetlock@disp</code> : Added	
v0.7.		<code>\l@dgetlock@disp</code>	54
<code>\l@reg</code> : Added <code>\l@reg</code>	63	<code>\l@dgetsidenote@margin</code> : Added	
<code>\@ref@reg</code> : Added <code>\@ref@reg</code>	68	<code>\l@dgetsidenote@margin</code>	147
General: eledmac having been avail- able for 2 years, deleted the commented out original edmac texts	1	<code>\l@drsn@te</code> : Added <code>\l@dlsn@te</code> and <code>\l@drsn@te</code> for use in	84
Maïeul Rouquette new main- tainer	1	<code>\do@line</code>	
Made macros of all messages	44	<code>\l@dunboxmpfoot</code> : Added	
Replaced all <code>\interfootnotelinepenalty</code> etc., by just <code>\interfootnotelinepenalty</code>	1	<code>\l@dunboxmpfoot</code> containing some common code	152
Tidying up for eledpar and ledarab packages	1	<code>\l@dzeropenalties</code> : Added	
<code>\affixline@num</code> : Added skipnum- mering to <code>\affixline@num</code>	88	<code>\l@dzeropenalties</code>	82
<code>\do@actions@fixedcode</code> : Added		<code>\ledlinenum</code> : Added <code>\ledlinenum</code> for use by <code>\leftlinenum</code> and	
<code>\do@actions@next</code> : Added number skipping to <code>\do@actions</code>	86	<code>\rightlinenum</code>	55
<code>\do@insidelinehook</code> : Added		<code>\line@list@stuff</code> : Deleted	
<code>\do@linehook</code> for use in		<code>\page@start</code> from <code>\line@list@stuff</code>	70
<code>\do@line</code>	84	<code>\list@clearing@reg</code> : Added	
<code>\endnumbering</code> : Changed		<code>\list@clearing@reg</code>	62
<code>\endnumbering</code> for eledpar	50	<code>\n@num@reg</code> : Added <code>\n@num</code>	68
<code>\f@x@l@cks</code> : Added <code>\ch@cksub@l@ck</code> , <code>\ch@ck@l@ck</code> and <code>\f@x@l@cks</code>	90	<code>\normalbfnoteX</code> : Removed extraneous space from	
<code>\footplitskips</code> : Added		<code>\normalbfnoteX</code>	118
<code>\footplitskips</code> for use in many footnote styles	97	<code>\resumenumbering</code> : Changed	
<code>\get@linelistfile</code> : Added		<code>\resumenumbering</code> for eledpar	51
<code>\get@linelistfile</code>	62	<code>\setprintendlines</code> : Added	
<code>\ifledRcol</code> : Added <code>\l@dnumpstartsL</code> , <code>\ifl@dpairing</code> and <code>\ifpst@rted</code> for/from eledpar	47	<code>\setprintendlines</code> for use by	
<code>\initnumbering@reg</code> : Added		<code>\printendlines</code>	145
<code>\initnumbering@reg</code>	47	<code>\setprintlines</code> : Added <code>\setprintlines</code> for use by <code>\printlines</code>	101
		<code>\skipnumbering@reg</code> : Added	
		<code>\skipnumbering</code> and supports	72
		<code>\sublinenumincrement</code> : Added	
		<code>\firstlinenum</code> , <code>\linenumincrement</code> , <code>\firstsublinenum</code> and	
		<code>\linenumincrement</code>	54
		<code>\sublinenumr@p</code> : Using <code>\linenumrep</code> instead of <code>\linenumr@p</code>	55
		Using <code>\sublinenumrep</code> instead of	
		<code>\sublinenumr@p</code>	55
		<code>\vnumfootnoteX</code> : Removed extraneous space from	

<code>\vnumfootnoteX</code>	118	New stanzaindentrepetition counter: to repeat stanza indents every n verses.	1
v0.8.		<code>\managestanza@modulo</code> : New stanzaindentrepetition counter to repeat stanza indents every n verses.	162
General: Bug on endnotes fixed: in a // text, all endnotes will print and be placed at the ends of columns ()	1	v0.13.1.	
v0.8.1.		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
General: Bug on <code>\edtext</code> ; <code>\critex</code> ; <code>\lemma</code> fixed: we can now use non switching commands	1	v0.14.	
v0.9.		General: Tweaked <code>\edlabel</code> to get correct line number if the command is first element of a paragraph.	1
General: No more ledpatch. All patches are now in the main file.	1	<code>\edlabel</code> : Tweaked <code>\edlabel</code> to get correct line number if the command is first element of a paragraph.	140
v0.9.1.		v0.15.	
General: Fix some bugs linked to integrating ledpatch on the main file.	1	General: Line numbering can be reset at each pstart.	51
v0.10.		Possibility to print <code>\pstart</code> number in side.	12
General: Corrections to <code>\section</code> and other titles in numbered sections	1	<code>\affixline@num</code> : Line numbering can be disabled.	88
v0.11.		<code>\ifinserthangingsymbol</code> : New management of hangingsymbol insertion, preventing undesirable insertions.	161
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	<code>\printlines</code> : Line numbering can be reset at each pstart.	101
v0.12.		v0.17.	
General: For compatibility with eldpar, possibility to use <code>\autopar</code> on the right side.	1	<code>\ifinserthangingsymbol</code> : New new management of hangingsymbol insertion, preventing undesirable insertions.	161
Possibility to number <code>\pstart</code>	11		
Possibility to number the pstart with the commands <code>\numberpstarttrue</code>	1	v1.0.	
<code>\ifledRcol</code> : Added <code>\ifledRcol</code> and <code>\ifnumberingR</code> for/from eldpar	47	General: <code>\lemma</code> can contain commands.	15
v0.12.1.		Debug in lineation command	12
General: Don't number <code>\pstarts</code> of stanza.	1	New generic commands to customize footnote display.	17
The numbering of <code>\pstarts</code> restarts on each <code>\beginnumbering</code>	1	Options nonum and nosepp in <code>\Xfootnote</code>	15
v0.13.		Options of <code>\Xfootnotes</code>	96
General: New stanzaindentrepetition counter to repeat stanza indents every n verses.	23		

Possibility to have commands in sidenotes.	29	create a correct direction for the footnote separator.	126
Some compatibility break with eledmac. Change of name: eledmac.	1	<code>\normalfootfmt</code> : Direction of footnotes with polyglossia.	98
<code>\morenoexpands</code> : Change to be compatible with new features	76	<code>\rbracket</code> : Switch the right bracket to a left bracket when the lemma is RTL (needs polyglossia or LuaTeX).	99
v1.0.1.			
General: Correction on <code>\numberonlyfirstinlineline</code> with lineation by pstart or by page.	18	<code>\endquote</code> : New option <i>noquotation</i>	48
v1.1.		<code>\labelrefsparsesubline</code> : Fix bug with <code>\edlabel</code>	141
General: Add <code>\labelpstarttrue</code>	12	v1.4.2.	
Add <code>\numberonlyfirstintwolines</code>	18	General: Debug with some special classes.	1
Add <code>\pstartinfootnote</code> and <code>\onlypstartinfootnote</code>	18	v1.4.3.	
New hook to add arbitrary code at the beginning of the notes	20	General: Add <code>\nonbreakableafternumber</code>	18
New options for block of notes.	21	Spurious space after familiar footnotes.	1
New package option: parapparat.	1	v1.4.4.	
New tools to change order of series	128	General: Label inside familiar footnotes.	1
sectioning commands	34	v1.4.5.	
<code>\ledfootinsdim</code> : Deprecated		General: Bug with komascript + eledpar + chapter.	1
<code>\ledfootinsdim</code>	103	v1.4.6.	
<code>\preXnotes</code> : New skip <code>\preXnotes@</code>	103	General: Bug with memoir class introduced by 1.4.5.	1
v1.2.		v1.4.7.	
General: Add <code>\ledsectnotoc</code> command.	35	<code>\endquote</code> : Compatibility of sectionning commands with <code>\autopar</code>	48
<code>\endquote</code> : Compatibility of <code>\ledchapter</code> with the <i>memoir</i> class.	48	v1.4.8.	
<code>\preXnotes</code> : Debug in familiar footnotes (but introduced by v1.1). . . .	103	General: Corrects a bug with parallel texts introduced by 1.1. . . .	1
v1.3.		v1.4.9.	
<code>\endquote</code> : <i>Quotation</i> and quote environment inside the numbering sections.	48	<code>\normalbfnoteX</code> : Allow to redefine <code>\thefootnoteX</code> with <code>alph</code> when some packages are loaded. . . .	118
v1.4.		v1.5.	
General: Compatibility of <code>\edtext</code> (and <code>\critext</code>) with the right-to-left direction (with Polyglossia).	78	General: Correct indexing when the call is made in critical notes. . . .	153
Compatibility with LuaTeX of RTL notes.	43	<code>\do@insidelinehook</code> : Added <code>\do@insidelinehook</code> for use in <code>\do@line</code>	84
<code>\newseries@</code> : Remembers the language of the lemma, in order to		<code>\edindex</code> : Compatibility with <code>imakeidx</code> package, and possibil-	

ity to use multiple index with <code>\edindex</code>	154		macro.	163
<code>\iffN@bottom</code> : Use the bottom op- tion of <code>footmisc</code> package. . . .	136	v1.6.1.	General: Corrects a false hanging verse when a verse is exactly the length of a line.	1
v1.5.1.			<code>\ifinserthangingsymbol</code> : Hang verse is now not automatically flush right.	161
<code>\managestanza@modulo</code> : Cor- rect <code>stanzaindentsrepetition</code> counter	162		<code>\l@dunhbox@line</code> : Move the call to <code>\inserthangingsymbol</code> to al- low use <code>\hfill</code> inside.	84
<code>\normalvfootnoteX</code> : Fix bug with normal familiar footnotes when mixing RTL and LTR text. . .	116		<code>\pend</code> : Spurious space in <code>\pend</code> . .	81
v1.5.2.			<code>\pstart</code> : Spurious space in <code>\pstart</code>	80
<code>\line@list@stuff</code> : Open / close immediatly the line-list file when in minipage, except if the minipage is a ledgroup.	70	v1.7.0.	General: New features for managing page breaks.	36
v1.6.0.				
<code>\falseverse</code> : Add <code>\falseverse</code>				