`reledmac`
Typeset scholarly editions with LaTeX[*]

Maïeul Rouquette[†]

based on the original `ledmac` by
Peter Wilson
Herries Press

which was based on the original `edmac`, `tabmac` and `edstanza` by
John Lavagnino, Dominik Wujastyk, Herbert Breger and Wayne Sullivan.

Abstract

The `reledmac` provides many tool in order to typeset scholarly edition. It is base on the `eledmac` package, which was was based on `ledmac` package, which was based on `edmac` TeX package.

It can be use in combination with `reledpar` in order to typeset two texts in parallel, like an original text and its translation in modern language.

`reledmac` provides many tools and options. Normally, they are all documented in this file. Also provided is a help folder, "examples". The folder contains additional examples (although not for all cases). Example starting by "1-" are for basic uses, those starting by "2-" are for advanced uses.

To report bugs or request a new feature, please go to ledmac GitHub page and click on "New Issue": `https://github.com/maieul/ledmac/issues/`. You must create an account on github.com to access my page (maieul/ledmac). GitHub accounts are free for open-source users. You can post messages in English or in French (preferred).

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

Contents

---

[*]This file (`reledmac.dtx`) has version number v2.0.1, last revised 2015/07/27.
[†]`maieul at maieul dot net`

1

XXXI Page breaking or no page breaking depending of specific lines   **277**

XXXII Long verse: prevents being separated by a page break   **279**

XXXIII Compatibility with `eledmac`   **279**

1   Introduction

1.1   Aim of the package

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, section or paragraph;

- 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.

Apart from `reledmac` there are some other LaTeX packages for critical edition type-setting. However, the am of `reledmac` is to provide a "all in one" and flexible tool in the field of critical edition.

Any suggestion for new features are welcome.

This manual contains a general description of how to use `reledmac` and the complete source code for the package, with extensive documentation (in sections I and following, numbered in Roman numeric) It finish by a list of action to do when migrating from some version to other, a historical of the change 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 `reledmac`.

1.2   History

1.2.1   `edmac`

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

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

The present version was developed in the summer of 1990, with the intent of adding necessary features, streamlining and documenting the code, and further generalizing it to make it easily adaptable to the needs of editors in different disciplines. John did most of the general reworking and documentation, with the financial assistance of the Division of the Humanities and Social Sciences, California Institute of Technology. Dominik adapted the code to the conventions of Frank Mittelbach's `doc` option, and added some

documentation, multiple-column footnotes, cross-references, and crop marks.[1] A description by John and Dominik of this version of `edmac` was published as 'An overview of `edmac`: a Plain TEX format for critical editions', *TUGboat 11* (1990), pp. 623–643.

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

At the time of writing (1994), we are pleased to be able to say that `edmac` is being used for real-life book production of several interesting editions, such as the Latin texts of Euclid's *Elements*,[2] an edition of the letters of Nicolaus Copernicus,[3] Simon Bredon's *Arithmetica*,[4] a Latin translation by Plato of Tivoli of an Arabic astrolabe text,[5] a Latin translation of part II of the Arabic *Algebra* by Abū Kāmil Shujā' b. Aslam,[6] the Latin *Rithmachia* of Werinher von Tegernsee,[7] a middle-Dutch romance epic on the Crusades,[8] a seventeenth-century Hungarian politico-philosophical tract,[9] an anonymous Latin compilation from Hungary entitled *Sermones Compilati in Studio Gererali Quinqeecclesiensi in Regno Ungarie*,[10] the collected letters and papers of Leibniz,[11] Theodosius's *Spherics*, the German *Algorismus* of Sacrobosco, the Sanskrit text of the *Kāśikāvṛtti* of Vāmana and Jayāditya,[12] and the English texts of Thomas Middleton's collected works.

---

[1]This 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).

[2]Gerhard Brey used `edmac` in the production of Hubert L. L. Busard and Menso Folkerts, *Robert of Chester's (?) Redaction of Euclid's* Elements, *the so-called Adelard II Version*, 2 vols., (Basel, Boston, Berlin: Birkhäuser, 1992).

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

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

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

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

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

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

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

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

[11]Leibniz, *Sämtliche Schriften und Briefe*, series I, III, VII, being edited by Dr. H. Breger, Dr. N. Gädeke and others, at the Leibniz-Archiv, Niedersächsische Landesbibliothek, Hannover. (see `http://www.nlb-hannover.de/Leibniz`)

[12]Being prepared at Poona and Lausanne Universities.

1.2.2 `ledmac`

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 Ocober 1996. The `edstanza` (v0.01) functions were added in February 2004. Sidenotes and regular footnotes in numbered text were added in April 2004.

This port was called `ledmac` (LaTeX `edmac`).

Since July 2011, ledmac is maintained by Maïeul Rouquette. Is it more and more powerful and flexible, but also more and more divergent from original TeX macro.

1.2.3 `eledmac`

Important changes were put in version 1.0, to make `ledmac` more easily extensible (see 6 p. 28). These changes can trigger small problems with the old customization.

That is why a new name was selected: `eledmac` (extended `ledmac`).

To migrate from `ledmac` to `eledmac`, please read Appendix Appendix A.2 (p. 283).

1.2.4 `reledmac`

`eledmac` has facilitated the creation of customized critical edition. However, this was made in a non systematic way in method to customize them.

In other side, many deprecated commands were keept, and many technical debt was accumulated, blocking the futur evolution.

For all these reasons, Maïeul Rouquette decided to make a spring cleaning. As some commands name were changed, the ascendant compatibility was (a little) broken.

A new name was selected: `reledmac` (extended renewed `eledmac`).

To migrate from `eledmac` to `reledmac`, please read Appendix **??** (p. **??**).

1.3   List of works edited with (r)(e)ledmac

A collaborative list of works edited with (r)(e)ledmac is available on `https://www.zotero.org/groups/critical_editions_typeset_with_edmac_ledmac_and_eledmac/items`. Please add your own edition made with (r)(e)ledmac.

2   How the package works

The `reledmac` package 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. `reledmac` 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 will want to print the text that you are editing.

Unnumbered text is not printed with line numbers, and you can't use `reledmac`'s note commands with it: this is appropriate for introductions and other material added by the editor around the edited text.

3   Options

The package can be loaded with a number of global options which are listed here. There are two types of options: options which provide specific features and option which optimize package's performance. It is advised to read the relevant parts of the handbook before reading the first type of option, but you can look at the second type in you first reading of handbook.

3.1   Specific features

draft underlines lemmas in the main text.

eledmac-compat help to migrate from `eledmac` to `reledmac` (see Appendix A.9.5 p. 287).

nopbinverse prevents page break inside verses.

noquotation by default, the quotation environment is redefined inside numbered text. You can disable this redefinition with `noquotation` (see 15 p. 53).

parapparatus by default, the apparatus cannot contain paragraph breaks; this option enables paragraphing inside the apparatus.

xindy and `xindy+hyperref` are for selecting `xindy` as the index processor (12.3 p. 47).

widthliketwocolumns set the width of the text disposed on one column to be the same as the width of the text disposed on two parallel columns with `reledpar`. This is useful when alternating between normal and parallel typesetting.

3.2   Optimize package's performance

nocritical disables tools for critical footnotes (`\Afootnote`, `\Bfootnote` etc.). If you do not need critical footnotes, this option lets `eledmac` run faster. It will also preserve room for other packages.

noeledsec disables tools for `\eledsection` and related commands (14.2 p. 52).

noend disables tools for endnotes (`\Aendnote`, `\Bendnote` etc.). If you do not need endnotes, this option lets `eledmac` run faster. It will also preserve room for other packages.

nofamiliar disables tools for familiar footnotes (`\footnoteA`, `\footnoteB` etc.). If you do not need familiar footnotes, this option lets `eledmac` run faster. It will also preserve room for other packages.

noledgroup `reledmac` allows to use of series of critical notes and new series of normal notes inside minipage and ledgroup environments (see 9 p. 42). However, such features use up computer memory, at the expense of other processing needs. So if you do not need this feature, use `noledgroup` option. This should make `reledmac` faster.

series `reledmac` defines five levels of notes: A, B, C, D, E. Using all these levels consumes memory space and processing speed. This is why, if your work does not require all of the A-E series, you can narrow down the available number of series. For example, if you only need A and B series, call the package with `series={A,B}` option.

4   Text lines and paragraphs numbering

4.1   Text lines numbering

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

```
\beginnumbering
text
\endnumbering
```

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

`reledmac` 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).

4.2   Paragraphs

4.2.1   Basis

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

```
\pstart
Paragraph of text.
\pend
```

Text that appears within a numbered section but is not 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
```

4.2.2   Producting automatically \pstart…\pend

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

```
\begingroup
  \beginnumbering
  \autopar

  A paragraph of numbered text.

  Another paragraph of numbered
  text.

  \endnumbering
\endgroup
```

\autopar fails, however, on paragraphs that start with a { or with any other command that starts a new group before it generates any text. Such paragraphs need to be started explicitly, before the new group is opened, using \indent, \noindent, or \leavevmode, or using \pstart itself.[13]

4.2.3   Content before specific \pstart and after specific \pend

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

. This feature is also useful when typesetting verses (see 8 p. 38) or eledpar (see 17.3 p. 56).

A \noindent is automatically added before this argument.

4.2.4   Content before every \pstart and after every \pend

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

4.2.5   Numbering paragraphs (\pstart)

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

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

On each \beginnumbering the numbering restarts.

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

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

---

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

4.2.6 Languages written in Right to Left

If you use languages written in right to left, we LuaLaTeX or XƎLaTeX, so you must switch text direction \before the \pstart command.

4.2.7 Memory limits

\pausenumbering
\resumenumbering

This paragraph is kept for history, but problem described below should not appear with recent version of LaTeX. reledmac 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

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

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

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

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

4.3 Lineation commands

4.3.1 Disabling lineation

\numberlinefalse
\numberlinetrue

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

4.3.2   Setting lineation start and step

\firstlinenum   By default, `reledmac` numbers every 5th line. There are two counters, `firstlinenum`
\linenumincrement   and `linenumincrement`, that control this behaviour; they can be changed using
`\firstlinenum{`⟨*num*⟩`}` and `\linenumincrement{`⟨*num*⟩`}`. `\firstlinenum` speci-
fies the first line that will have a printed number, and `\linenumincrement` is the dif-
ference between succesive numbered lines. For example, to start printing numbers at
the first line and to have every other line numbered:
`\firstlinenum{1} \linenumincrement{2}`

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

4.3.3   Setting lineation reset

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

4.3.4   Setting line number margin

\linenummargin   The command `\linenummargin`⟨*location*⟩ specifies the margin where the line (or pstart)
numbers will be printed. The permissable value for ⟨*location*⟩ is one out of the list `left`,
`right`, `inner`, or `outer`, for example `\linenummargin{inner}`. The package's default
setting is
`\linenummargin{left}`
to typeset the numbers in the left hand margin. You can change this whenever you're
not in the middle of making a paragraph.
More precisely, the value of `\linenummargin` used is that in effect at the `\pend` of
a numbered paragraph. Apart from an initial setting for `\linenummargin`, only change
it after a `\pend`, whereupon it will apply to all following numbered paragraphs, until
changed again (changing it between a `\pstart` and `\pend` pair will apply the change
to all the current paragraph).

4.3.5   Other settings

\leftlinenum
\rightlinenum
\linenumsep

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

4.4   Changing the line numbers

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

4.4.1   Sublineation

\startsub
\endsub

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

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

4.4.2   Locking lineation

\startlock
\endlock

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

\lockdisp

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

4.4.3   Setting and changing line number

\setline
\advanceline

In some cases you may want to modify the line numbers that are automatically calculated: if you are printing only fragments of a work but want to print line numbers appropriate to a complete version, for example. The `\setline{⟨num⟩}` and `\advanceline{⟨num⟩}` commands may be used to change the current line's number (or the sub-line number, if sub-lineation is currently on). They change both the marginal

line numbers and the line numbers passed to the notes. \setline takes one argument, the value to which you want the line number set; it must be 0 or greater. \advanceline takes one argument, an amount that should be added to the current line number; it may be positive or negative.

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

4.4.4   Line number style

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

Alph Uppercase letters (A… Z).

alph Lowercase letters (a… z).

arabic Arabic numerals (1, 2, …)

Roman Uppercase Roman numerals (I, II, …)

roman Lowercase Roman numerals (i, ii, …)

Note that with the Alph or alph styles, 'numbers' must be between 1 and 26 inclusive.
Similarly \sublinenumberstyle{⟨*style*⟩} can be used to change the numbering style of sub-line numbers, which is normally arabic numerals.

4.4.5   Skipping and hiding number

\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. Note that if you use it in \stanza, you must call it at the beginning of the verse.
\hidenumbering   When inserted into a numbered line the macro \hidenumbering causes the number for that particular line to be hidden; namely, no line number will print. Note that if you use it in \stanza, you must call it at the beginning of the verse.

4.4.6   Execute code at each line

dolinehook   reledmac provides to advanced feature for user. The argument passed to \dolinehook{⟨*arg*⟩}
doinsidelinehook   will be executed before slicing a new line in the paragraph. The argument passed to \doinsidelinehook{⟨*arg*⟩} will be executed before printing a new line. In many case, the second of two is more useful than the first. The file examples/2-line_numbers_in_header.tex provides an example of use in order to print the first and last line number of a page in the header.

5 Apparatus commands

5.1 Terminology

We call "critical notes" notes which refer to both a lemma, that is a part of text and a line number. Critical notes are subdivided in critical footnotes and critical endnotes.

We call "familiar notes" notes which refer to a footnote mark in the main text.

`reledmac` manages many series of notes of each category. A series of notes is identified by an uppercase letter. When the series letter is at the beginning of a command name, it refers to a critical footnote. When the series letter is at the end of a command name, it refers to a critical endnote.

So :

- `\Afootnote` is a critical footnote of the series A.

- `\Bendnote` is a critical footnote of the series B.

- `\footnoteC` is a critical footnote of the series C.

5.2 Critical notes

5.2.1 The lemma

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

> `\edtext{⟨lemma⟩}{⟨commands⟩}`

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

For example:

```
I am happy :
I saw my friend \edtext{Smith}{
\Afootnote{Jones C, D.}}
on Tuesday.
```

1 I am happy : I saw my friend Smith on
2 Tuesday.

————————
1 Smith] Jones C, D.

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

The ⟨*lemma*⟩ may contain further `\edtext` commands. Nesting makes it possible to print an explanatory note on a long passage together with notes on variants for individual words within the passage. For example:

```
I am happy : \edtext{I saw my friend
  \edtext{Smith}{\Afootnote{Jones
  C, D.}} on Tuesday.}{
  \Bfootnote{The date was
  July 16, 1954.}
}
```

1 I am happy : I saw my friend Smith on
2 Tuesday.

————————
1 Smith] Jones C, D.

————————
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.)

5.2.2   Footnotes

The second argument of the \edtext macro, ⟨*commands*⟩, may contain a series of subsidiary commands that generate various kinds of notes.

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

If you need more series of critical notes, please look at 5.5.1 p. 28.

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:

- fulllines to disable \Xtwolines and \Xmorethantwolines features for this note (cf. 6.2.2 p. 30).

- nonum to disable line numbering for this note.

- nosep to disable the lemma separator for this note.

Example: \Afootnote[nonum]{⟨*text*⟩}.

5.2.3   Endnotes

\Aendnote   The package also maintains five separate series of endnotes.
\Bendnote   If you do not need the endnotes facility, you should use noend option when loading
\Cendnote   eledmac.
\Dendnote   The mechanism is similar to the one for footnotes: each macro takes one or more
\Eendnote   optional arguments and one single argument, like:
\Aendnote[⟨*option*⟩]{⟨*text*⟩}.

[<option>] can contain a comma separated list of values. Allowed values are:

- fulllines to disable \Xendtwolines and \Xendmorethantwolines features for this particular note (cf. 6.2.2 p. 30).

- nonum to disable line number for this particular note.

- nosep to disable the lemma separator for this particular note.

Normally, endnotes are not printed: you must use the \doendnotes{⟨*s*⟩}, where ⟨*s*⟩ is the letter of the series to be printed. Put this command where you want the corresponding set of endnotes printed. In this case, all the endnotes of the ⟨*s*⟩ series are printed, for all numbered section.

\doendnotesbysection   However, you may want to print the endnotes of one given series covering the first

numbered section, then the endnotes of another given series covering the first numbered section, then the endnotes of the first given series covering the second numbered section, then the endnotes of the second given series covering the second numbered section, and so forth. In this case, use \doendnotesbysection{⟨*s*⟩}. For each value of ⟨*s*⟩, the first call of the command will print the notes for the first series, the second call will print the notes for the second series etc. For example, do:

```
\section{Endnotes}
\subsection{First text}
\doendnotesbysection{A}
\doendnotesbysection{B}
\subsection{Second text}
\doendnotesbysection{A}
\doendnotesbysection{B}
```

Note that by default inside endnotes no separator is used between the lemma and the content. However you can use the \Xendlemmaseparator macro to define one (6.3.2 p. 33).

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

5.2.4 Paragraph in critical apparatus

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}
```

5.2.5 Change lemma and line number

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

```
I am happy :
\edtext{I saw my friend
  \edtext{Smith}{\Afootnote{Jones
  C, D.}} on Tuesday.}
  {\lemma{I \dots\ Tuesday.}
  \Bfootnote{The date was
  July 16, 1954.}
}
```

1 I am happy : I saw my friend Smith on
2 Tuesday.

————————
1 Smith] Jones C, D.

————————
1–2 I ... Tuesday.] The date was July 16, 1954.

\linenum   You can use \linenum{⟨*arg*⟩} to change the line numbers passed to the notes. The notes are actually given seven parameters: the page, line, and sub-line number for the

start of the lemma; the same three numbers for the end of the lemma; and the font specifier for the lemma. As the argument to `\linenum`, you specify those seven parameters in that order, separated by vertical bars (the | character). However, you can retain the value computed by `reledmac` 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 does not 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 (10 p. 42) 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.

5.2.6   Changing the names of commands for critical apparatus

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 does not mean you have to type `\Afootnote` when you would rather say something you find more meaningful, like `\variant`.

We recommend that you create a series of such aliases and use them instead of the names chosen here; all you have to do is put commands of this form at the start of your file:[14]

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

5.3   Disambiguation of identical words in the apparatus

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

---

[14]We use `\newcommand` and `\newcommandx` instead of classical `\let` command because the edtabular environments have to modify the notes definition, and we need to use the newest definition of notes. Read the handbook of `xargs` to know more about `\newcommandx`.

5.3.1   Basic use

\sameword   To use this tool, you have to mark every occurrence of the potentially ambiguous term
with the \sameword command:

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

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

5.3.2   Notes about input encoding with UTF-8 processor

If you use UTF-8 processor, like X∃LATEX or LuaLATEX, there should not be any glitches.
However, pay attention to how characters are encoded. Similar-looking characters may
be represented differently in unicode numbering.

For instance, in Greek, "α" has two possible unicode numbers:

- GREEK SMALL LETTER ALPHA (U+03B1) + COMBINING GREEK YPOGE-
  GRAMMENI (U+0345)

- GREEK SMALL LETTER ALPHA WITH YPOGEGRAMMENI (U+1FB3)

Which unicode number you use depends, many times, on your keyboard configura-
tion (the computer-input system).

Inside reledmac, the \sameword command considers these two unicodes options
as different characters. If you use only one unicode number consistently, the distinction
will probably make no difference to how your text looks, but \sameword will process
the text inaccurately, based on the unicode numbers. To prevent this, do the following:

- If you use X∃LATEX, add this line in your preamble: \XeTeXinputnormalization 1.

- If you use LuaLATEX, use the uninormalize package of Michal Hoftich[15] with the
  buffer option set to true.

With these tools, X∃TEX / LuaTEX will dynamicaly normalize unicode input when
reading the file. Consequently, you will have no problems with the \sameword com-
mand.

5.3.3   Use with \lemma command

If you use the \lemma command, eledmac cannot know to which occurence of
\sameword in the first argument of \edtext a word marked with \sameword in \lemma
should refer.

For example in the following example:

```
some thing
    \edtext{\sameword{sw}
```

---

[15] https://github.com/michal-h21/uninormalize.

```
              and other \sameword{sw}
              and again \sameword{sw}
              it is all}%
    }{\lemma{\sameword{sw} \ldots all}\Afootnote{critical note}}.%
```

eledmac cannot know if the "sw" in \lemma refers to the word after "thing", after "other", or after "again".

Consequently, you have to tell to eledmac which instance of \sameword in the first argument of \edtext you want to reference:

- In the content of \lemma, use \sameword with no optional argument.

- In the first argument of \edtext, use \sameword with the optional argument [$\langle X \rangle$]. $\langle X \rangle$ is the depth of the \edtext where the \lemma is used. So if the \lemma is called in a \edtext inside another \edtext, $\langle X \rangle$ is equal to 2. If the \lemma is called in a \edtext "of first level", $\langle X \rangle$ is equal to 1. If the lemma is called in both 1 and 2 \edtext depth, $\langle X \rangle$ is 1,2. If that word is referenced in the lemma of every \edtext depth, $\langle X \rangle$ can also be set to inlemma.

Note that only words that are actually referenced in a \lemma need the optional argument. Therefore, the first \sameword in the example above should have "1" as its optional argument, to be referenced correctly in the lemma.
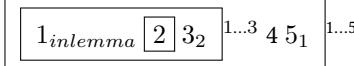
Note also that the $\langle X \rangle$ does not refer to the level where the \sameword occurs, but to the level of the \lemma that refers to that \sameword. For example:

```
\edtext{some \edtext{\sameword[1]{word}}{\Afootnote{om. M}}
        and other \sameword{word}
        and again a \sameword{word}
        it is all}%
}{\lemma{some \sameword{word} \ldots all}\Afootnote{critical note}}.%
```

Here the \sameword occurs in an \edtext of level 2, but since it is referenced by \lemma on level 1, it has "1" in the optional argument.

In the following schema, each framed box represents an \edtext level. Each number is an occurrence of \sameword. After a framed box, the text in superscript represents the content of \lemma for that \edtext level. The text in subscript at the right of a number represents the content of the optional argument of \sameword.

$$\boxed{1_{inlemma} \;\boxed{2}\; 3_2 \;{}^{1...3}\; 4\; 5_1}\;{}^{1...5}$$

The \sameword number 3 is called in a \lemma related to an \edtext of level 2. It must be marked by "2".

The \sameword number 5 is called in a \lemma related to \edtext of level 1. It must be marked by "1".

The \sameword number is called in two \lemmas: one related to a \edtext of level 1, the other related to \edtext of level 2. It must be marked by "1,2". However, as \lemma is called only in level 1 and 2, "1,2" could replaced by "inlemma".

The \sameword number "2" is in the first argument of a \edtext of level 3, but it has no \lemma-command, so there is no need to mark it.

5.3.4   Customizing

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

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

5.4   Familiar notes

5.4.1   Basic use

\footnoteA   As well as the standard LaTeX footnotes generated via \footnote, the package also pro-
\footnoteB   vides five series of additional footnotes called \footnoteA through \footnoteE. These
\footnoteC   have the familiar marker in the text, and the marked text at the foot of the page can be
\footnoteD   formated using any of the styles described for the critical footnotes. Note that the 'reg-
\footnoteE   ular' 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.

5.4.2   Customizing mark

\thefootnoteA   Each series uses a set of macros for styling the marks. The mark numbering scheme of
\bodyfootmarkA   series A is defined by the \thefootnoteA macro; the default is:
\footfootmarkA   \renewcommand*{\thefootnoteA}{\arabic{footnoteA}}
The appearance of the mark in the text is controlled by \bodyfootmarkA which is de-
fined 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.

5.4.3   Separator for multiple 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 defi-
nition is:
\providecommand*{\multfootsep}{^{\normalfont,}}
and can be changed if necessary.

5.5   Changing series

5.5.1   Create a new series

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

5.5.2   Delete series

As the number of series which are defined increases, `reledmac` gets slower. If you do not need all of the six standard series (A, B, C, D, E), you can load the package with the `series` option. For example if you need only series A and B, use:

```
\usepackage[series={A,B}]{eledmac}
```

5.5.3   Series order

The default series order is the one called with the `series` option of the package, or, if this option is not used, A, B, C, D, E. Series order determines footnotes order.

seriesatbegin   However in some specific cases, you need to change the series order at some point
seriesatend   inside the document. You can use `\seriesatbegin{⟨s⟩}` to pull up a given series ⟨s⟩
to the beginning, or `\seriesatend{⟨s⟩}` to push it down to the end.

5.6   Position of critical and familiar footnotes

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

```
\fnpos{critical-familiar}
```

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

```
\mpfnpos{familiar-critical}
```

6   Critical apparatus appearance

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. If the optional argument is omitted or empty, the setting will concern all series.

When a length, noted ⟨l⟩, is used, it can be stretchable: a plus b minus c. The final length m is calculated by LaTeX to have: $a - c \leq m \leq a + b$. If you use some relative unity[16], it will be relative to fontsize of the footnote, except for commands concerning

---

[16]Like em which is the width of a mg.

the place kept by the notes — including blank space.

There is also name convention:

- Names prefixed by X are for setting of critical footnotes.

- Names prefixed by Xend are for setting of critical endnotes.

- Names suffixed by X are for setting of familiar footnotes.

6.1 Notes arrangement in a series

\Xarrangement
\arrangementX

By default, all footnotes are formatted as a series of separate paragraphs in one column. Three other formats are also available for notes.

Use \Xarrangement[⟨s⟩]{⟨a⟩} to change the arrangement of the ⟨s⟩ series of critical footnotes and \arrangementX[⟨s⟩]{⟨a⟩} to change the arrangement of the ⟨s⟩ series of familiar footnotes.

The value of ⟨a⟩ can be one of the following

- paragraph formats all the footnotes of a series as a single paragraph;

- twocol formats them as separate paragraphs, but in two columns;

- threecol, in three columns.

- normal, restore normal arrangement.

You should set up the page layout parameters, and in particular the \baselineskip of the footnotes, before you call this macro because its 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]

The notes arrangement must be called after having defined the document geometry setting. If you must change geometry setting inside your document, do not forget to call again not arrangement.

\hsize has been set for the pages that use this series of notes; otherwise TeX will try to put too many or too few of these notes on each page. If you need to change the \hsize within the document, call \footparagraph again afterwards to take account of the new value.

6.2 Control line number printing

6.2.1 Print line number only at first time

\Xnumberonlyfirstinline

By default, the line number is printed in every note. If you want to print it only the first time for a given line number (i.e one time for line 1, one time for line 2 etc.), you can use \Xnumberonlyfirstinline[⟨s⟩].

---

[17]There is one tiny proviso about using paragraphed notes: you should not 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. XII.6.3 p. 146 explains why this restriction is necessary.

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

\Xnumberonlyfirstintwolines   Suppose you have a lemma on line 2 and a lemma between line 2 and line 3. With \Xnumberonlyfirstinline, the second lemma is considered to be on the same line as the first lemma. But if you use both \Xnumberonlyfirstinline[⟨s⟩] and \Xnumberonlyfirstintwolines[⟨s⟩], the distinction is made. Use \Xnumberonlyfirstintwolines[⟨s⟩]

\Xsymlinenum to disable this (⟨s⟩ can be empty if you want to disable it for every series). For setting a particular symbol in place of the line number, you can use \Xsymlinenum[⟨s⟩]{⟨symbol⟩} in combination with \Xnumberonlyfirstinline[⟨s⟩]. From the second lemma of the same line, the symbol will be used instead of the line number. Note that any command called in ⟨symbol⟩ must be robust. Use \robustify to robustify a not robust command.

6.2.2   Abbreviate line range

\Xtwolines If a lemma is printed on two subsequent lines, Eledmac will print the first and the last
\Xmorethantwolines line numbers. Instead of this, it is also possible to print an abbreviation which stands for "line 1 and subsequent line(s)".

  To achieve this, use \Xtwolines[⟨s⟩]{⟨text⟩} and \Xmorethantwolines[⟨s⟩]{⟨text⟩}. The ⟨text⟩ argument of \Xtwolines will be printed if the lemma is on two lines, and the ⟨text⟩ argument of \Xmorethantwolines will be printed if the lemma is on three or more lines. For example:

```
\Xtwolines{sq.}
\Xmorethantwolines{sqq.}
```

Will print "1sq." for a lemma which falls on lines 1-2 and "1sqq." for a lemma which falls on lines 1-4.

\Xmorethantwolines   If you use \Xtwolines without setting \Xmorethantwolines, the ⟨text⟩ argument of \Xtwolines will be used for lemmas which fall on three or more lines.

  However, if you want to use a short form (when the lemma overlaps two lines, but not more than two), use \Xtwolinesbutnotmore[⟨series⟩].

  It is possible to disable \Xtwolinesbutnotmore[⟨series⟩] with \Xtwolinesbutnotmore[⟨series⟩][false].

  When you use lineation by page, the final page number, if different from the initial page number, will not be printed, because the final page number is included in the \Xendtwolines symbol.

\Xtwolinesonlyinsamepage   However, you can force print the final page number with \Xtwolinesonlyinsamepage[⟨series⟩].

  Use \Xtwolinesonlyinsamepage[⟨series⟩][false] to disable this.

  You can disable \Xtwolines and related for a specific note by using the '[fulllines]' argument in the note macro cf. 5.2.2 p. 22.

\Xendtwolines   For endnotes, use \Xendtwolines; \Xendmorethantwolines; \Xendtwolinesbutnotmore;
\Xendmorethantwolines \Xendtwolinesonlyinsamepage instead of \Xtwolines; \Xmorethantwolines; \Xtwolinesbutnotmore;
\Xendtwolinesbutnotmore \Xtwolinesonlyinsamepage.

6.2.3   Disable line number

\Xnonumber You can use \Xnonumber[⟨s⟩] if you do not want to have the line number in a foot-

\Xendnonumber   note. To cancel it, use \Xnonumber[⟨*s*⟩][false]. \Xendnonumber[⟨*s*⟩] is the same
for endnote.

6.2.4   Printing pstart number

\Xpstart   You can use \Xpstart[⟨*s*⟩] if you want to print the pstart number in the footnote,
before the line and subline number. Use \Xpstart[⟨*s*⟩][false] to disable this (⟨*s*⟩ can
be empty if you want to disable it for every series). Note that when you change the
lineation system, the option is automatically switched :

- If you use lineation by pstart, the option is enabled.

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

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

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

6.2.5   Space around number

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

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

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

6.2.6   Space around line symbol

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

\Xaftersymlinenum   With \Xaftersymlinenum[⟨*s*⟩]{⟨*l*⟩} you can add some space after the line symbol
in a footnote. The default value is value set by \Xafternumber.

6.2.7   Space in place of number

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

\Xendinplaceofnumber   . \Xendinplaceofnumber[⟨*s*⟩]{⟨*l*⟩} is the same, for critical endnotes.

6.2.8   Boxing line number and line symbol

\Xboxlinenum It could be useful to put the line number inside a fixed box: the content of the note will be printed after this box. You can use \Xboxlinenum[⟨*s*⟩]{⟨*l*⟩} to do that. To subsequently disable this feature, use \Xboxlinenum 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}
\Xafternumber{0em}
\Xboxlinenum{1em}
```

\Xboxsymlinenum   \Xboxsymlinenum[⟨*s*⟩]{⟨*l*⟩} is the same as \Xboxlinenum but for the line number symbol.

Xboxlinenumalign   If you put line number in box, it will be aligned left inside the box. However, you can change it using \Xboxlinenumalign[⟨*s*⟩]{⟨*text*⟩} where ⟨*text*⟩ can be the following:

L to align left (default value);

R to align right;

C to center.

When using \Xboxlinenum, reledmac put all the line number description in the same box. That is, the same box will contain: the start line number, the dash, and either the end line number or the range symbol (like ff .). However, it is possible to box them in two different boxes.

- \Xboxstartlinenum[⟨*s*⟩]{⟨*l*⟩} will box the start line number in a box of length ⟨*l*⟩. The content will be put at the right of the box.

- \Xboxendlinenum[⟨*s*⟩]{⟨*l*⟩} will box the dash plus the end line number or the range symbol in a box of length ⟨*l*⟩. The content will be put at the left of the box.

With these two commands, it is possible to horizontaly align the dash of line number when using critical notes, to obtain something like:

```
 1
12-23
24ff.
```

\Xendboxlinenum   \Xendboxlinenum[⟨*s*⟩]{⟨*l*⟩}, \Xendboxlinenumalign[⟨*s*⟩]{⟨*text*⟩}, \Xendboxstartlinenum[⟨*s*⟩]{⟨
\Xendboxlinenumalign   \Xendboxendlinenum[⟨*s*⟩]{⟨*l*⟩} are the same as, respectively, \Xboxlinenum and
\Xendboxstartlinenumalign   \Xboxlinenumalign, \Xboxstartlinenum, \Xboxendlinenum except in endnotes.
\Xendboxendlinenumalign

6.3   Separator between the lemma and the note

6.3.1   For footnotes

\Xlemmaseparator By default, in a footnote, the separator between the lemma and the note is a right bracket

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

\Xbeforelemmaseparator   Using \Xbeforelemmaseparator[⟨*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.

\Xafterlemmaseparator   Using \Xafterlemmaseparator[⟨*s*⟩]{⟨*l*⟩} you can add some space between separator and note. If your lemma separator is empty, this space will not be printed. The default value is 0.5 em.

\Xnolemmaseparator   You can suppress the lemma separator, using \Xnolemmaseparator[⟨*s*⟩], which is simply a alias of \Xlemmaseparator[⟨*s*⟩]{}.

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

6.3.2   For endnotes

\Xendlemmaseparator   By default, there is no separator inside endnotes between the lemma and the content of the note. You can use \Xendlemmaseparator[⟨*s*⟩]{⟨*Xendlemmaseparator*⟩} to change this. The optional argument can be used to specify the series in which it is used. An common value of <Xendlemmaseparator> is \rbracket.

Note that there is a non-breakable space between the lemma and the separator, but a **breakable** space between the separator and the lemma.

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

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

\Xendinplaceoflemmaseparator   With \Xendinplaceoflemmaseparator[⟨*s*⟩]{⟨*l*⟩} you can add some space if you chose to remove the lemma separator. The default value is 0.5 em.

6.4   Font style

6.4.1   For line number

\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 foot-

notes 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 foot-
notes 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[⟨*s*⟩]{⟨*l*⟩} is used to define the font size of end critical foot-
notes of the series. The default value is \footnotesize. The ⟨*command*⟩ must not be
a size in pt, but a standard LATEX size, like \small.

6.4.2   For the lemma

\Xlemmadisablefontselection   By default, font of the lemma in footnote is the same as font of the lemma in the main
text. For example, if the lemma is in italic in the main text, it is also in italic in note.
The \Xlemmadisablefontselection[⟨*s*⟩] command allows to disable it for a specific
series.

\Xendlemmadisablefontselection   By default, font of the lemma in endnote is the same as font of the lemma in the main
text. For example, if the lemma is in italic in the main text, it is also in italic in note. The
command allows \Xendlemmadisablefontselection[⟨*s*⟩] to disable it for a specific
series.

6.5   Styles of notes content

\Xparindent   By default, reledmac does not add indentation before the paragraphs inside critical
footnotes. Use \Xparindent[⟨*series*⟩] to enable indentation.

\parindentX   By default, reledmac does not add indentation before the paragraphs inside familiar
footnotes. Use \parindentX[⟨*series*⟩] to enable indentation.

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

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

6.6   Arbitrary code at the beginning of notes

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

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

\Xbhooknote   \Xbhooknote[⟨*series*⟩]{⟨*code*⟩} is to be used at the beginning of the critical foot-
notes.

\bhooknoteX   \bhooknoteX[⟨*series*⟩]{⟨*code*⟩} is to be used at the beginning of the familiar foot-
notes.

\Xendbhooknote   \Xendbhooknote[⟨*series*⟩]{⟨*code*⟩} is to be used at the beginning of the endnotes.

6.7 Options for footnotes in columns

6.7.1 Alignment

By default, texts in footnotes in two or three columns are flushed left without hyphenation. However, you can change this with \Xcolalign[⟨*s*⟩]{⟨*code*⟩}, for critical footnotes, and \colalignX[⟨*s*⟩]{⟨*code*⟩}, for familiar footnotes.

<code> must be one of the following command:

\justifying to have text justified, as usual with LaTeX. You can also let <code> empty.

\raggedright to have text left aligned, but *without hyphenation*. That is the default eledmac setting.

\RaggedRight to have text left aligned *with hyphenation*.

\raggedleft to have text right aligned, but *without hyphenation*.

\RaggedLeft to have text right aligned *with hyphenation*.

\centering to have text centered, but *without hyphenation*.

\Centering to have text centered *with hyphenation*.

6.7.2 Size of the columns

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

\Xhsizetwocol   \Xhsizetwocol[⟨*s*⟩]{⟨*l*⟩} is used to change width of a column when critical notes are displaying in two columns. Defaut value is .45 \hsize.

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

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

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

6.8 Options for paragraphed footnotes

6.8.1 Mark separation of notes

\Xafternote   You can add some space after a note by using \Xafternote[⟨*s*⟩]{⟨*l*⟩} (for critical
\afternoteX   footnotes) or \afternoteX[⟨*s*⟩]{⟨*l*⟩} (for familiar footnotes). The default value is 1em plus.4em minus.4em.

\Xparafootsep   For paragraphed footnotes (see below), you can choose the separator between each
\parafootsepX   note by using \Xparafootsep[⟨*s*⟩]{⟨*text*⟩} for critical notes and \parafootsepX for familiar notes. A common separator is the double pipe (‖), which you can set by using \parafootsep{$\parallel$}.

Note that if the symbol defined by \Xsymlinenum must be used at the beginning of a note, the \Xparafootsep / \parafootsepX is not used before this note.

6.8.2   Ragging

\Xragged   Text in paragraphed critical notes is justified, but you can use \Xragged[⟨s⟩]+L+ if you
            want it to be ragged left, or \Xragged[⟨s⟩]+R if you want it to be ragged right.
\raggedX   Text in paragraphed footnotes is justified, but you can use \raggedX[⟨s⟩]+L+ if you
            want it to be ragged left, or \raggedX[⟨s⟩]+R if you want it to be ragged right.

6.9   Options for block of notes

6.9.1   Text before notes

\Xtxtbeforenotes   You can add some text before critical notes with \Xtxtbeforenotes[⟨s⟩]{⟨text⟩}.

6.9.2   Spacing

\Xbeforenotes   You can change the vertical space printed before the rule of the critical notes with
            \Xbeforenotes[⟨s⟩]{⟨l⟩}. The default value is 1.2em plus .6em minus .6em.
                 Be careful, the standard LATEX footnote rule, which is used by eledmac, de-
\beforenotesX   **creases by 3pt. This 3pt decrease is not changed by this command..**
                 You can change the vertical space printed before the rule of the familiar notes with
            \beforenotesX[⟨s⟩]{⟨l⟩}. The default value is 1.2em plus .6em minus .6em.
                 Be careful, the standard LATEX footnote rule, which is used by eledmac, de-
            creases 3pt. These 3pt are not changed by this command.
\preXnotes   You can set the space before the first series of critical notes printed on each page
            and set a different amount of space for subsequent the series on the page. You can do it
            with \preXnotes{⟨l⟩}. Default value is 0pt. You can disable this feature by setting the
            length to 0pt.
\prenotesX   You can want the space before the first printed (in a page) series of familiar notes
            not to be the same as before other series. Default value is 0pt. You can do it with
            \prenotesX{⟨l⟩}. You can disable this feature by setting the length to 0 pt.
\preXnotes   You can set the space before the first series of critical notes printed on each page
            and set a different amount of space for subsequent the series on the page. You can do it
            with \preXnotes{⟨l⟩}. Default value is 0pt. You can disable this feature by setting the
            length to 0pt.
\prenotesX   You can want the space before the first printed (in a page) series of familiar notes
            not to be the same as before other series. Default value is 0pt. You can do it with
            \prenotesX{⟨l⟩}. You can disable this feature by setting the length to 0 pt.

6.9.3   Rule

\Xafterrule   You can change the vertical space printed after the rule of the critical notes with
            \Xafterrule[⟨s⟩]{⟨l⟩}. The default value is 0pt.
                 Be careful, the standard LATEX footnote rule, which is used by eledmac, adds
            2.6pt. These 2.6pt are not changed by this command.
\afterruleX   You can change the vertical space printed after the rule of the familiar notes with
            \beforenotesX[⟨s⟩]{⟨l⟩}. The default value is 0pt.

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

6.9.4   Maximum height

\Xmaxhnotes   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 `\Xmaxhnotes[⟨s⟩]{⟨l⟩}`. Be care-
ful : 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 `\Xmaxhnotes{.33\textheight}`.

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

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

6.10   Footnotes and the `reledpar` columns

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

- `\Xnoteswidthliketwocolumns[⟨s⟩]` to create critical notes with a two-column
  size width. Use `\Xnoteswidthliketwocolumns[⟨s⟩][false]` to disable it.

- `\noteswidthliketwocolumnsX[⟨s⟩]` to create familiar notes with a two-column
  size width. Use `\noteswidthliketwocolumnsX[⟨s⟩][false]` to disable it.

6.11   Endnotes in one paragraph

\Xendparagraph   By default, any new endnote starts a new paragraph. Use `\Xendparagraph[⟨series⟩]`
to have all end notes of one given series set in one paragraph.

\Xendafternote   You can add some space after a endnote series by using `\Xendafternote[⟨s⟩]{⟨l⟩}`.
The default value is `1em plus.4em minus.4em`.

\Xendsep   You can choose the separator between each note by `\Xendsep[⟨s⟩]{⟨text⟩}`. A com-
mon separator is the double pipe ($||$), which you can set by using `\Xendsep{$\parallel$}`.

7   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, es-
pecially within the notes, and then in subsequent sections specify how these commands
are used.

For those who are setting up for a large job, here is a list of the complete set of
`reledmac` 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 some-
\fullstop times be spans of line numbers like this: 12–34, or lines with sub-line numbers like
\rbracket this: 55.6. The en-dash and the full stop are taken from the same font as the numbers, and it all works nicely. But what if you wanted to use old-style numbers, like 12 and 34? These look nice in an edition, but when you use the fonts provided by Plain TeX they are taken from a math font which does not have the en-dash or full stop in the same places as a text font. If you (or your macros) just typed `$\oldstyle 12--34$` or `$\oldstyle 55.6$` you would get '12″34'and '55▷6'. So we define `\endashchar` and `\fullstop`, which produce an en-dash and a full stop respectively from the normal document font, whatever font you are using for the numbers. These two macros are used in the macros which format the line numbers in the margins and footnotes, instead of explicit punctuation. We also define an `\rbracket` macro for the right square bracket printed at the end of the lemma in many styles of textual notes (including `reledmac`'s standard style). For `polyglossia`, when the lemma is RTL, the bracket automatically switches to a left bracket.

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

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

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

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

8   Verse

8.1   Basic

\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.

8.2   Define stanza indents

\stanzaindentbase   Lines within a stanza may be indented. The indents are integer multiples of the length

`\setstanzaindents`

\stanzaindentbase, whose default value is 20pt.

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 \sethanginsymbol: see p. 8.6 p. 40.

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.

8.3 Repeating stanza indents

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

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

is like

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

Be careful: the feature is changed in eledmac 1.5.1. See Appendix A.3 p. 284.

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

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

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

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

8.4   Manual stanza indent

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

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

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

8.5   Stanza breaking

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

The command
\setstanzapenalties{1,5000,10100,5000,0}
results in a penalty of 5000 being placed after the first and third lines of the stanza, and
a penalty of −100 after the second.

The first entry "1" is a control value. If it is zero, then no penalties are passed on
to TEX, which is the default. Values between 0 and 10000 are penalty values; values
between 10001 and 20000 have 10000 subtracted and the result is given as a negative
penalty. The mechanism used for indentations and penalties requires unsigned values
less than 32768. No penalty is placed after the last line, so the final ,0 in then example
above could be omitted. 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.

8.6   Hanging symbol

It is possible to insert a symbol in each line of hanging verse, as in French typography
\sethangingsymbol   for '['. To insert it in reledmac, use macro \hangingsymbol⟨*h*⟩ with this code. In the
example of French typography, do

    \sethangingsymbol{[\,}

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

    \sethangingsymbol{\protect\hfill}

8.7 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 16.2 p. 53 for further details.

8.8 Hanging symbol

It is possible to insert a symbol on each line of hanging verse, as in French typography for

\sethangingsymbol   '['. To insert in reledmac, use macro \sethangingsymbol{⟨*symbol*⟩}. Fro exemple:

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

8.9 Content before/after verses

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

- \stanza command can take a optional argument (in brackets). Its content will be printed before the stanza.

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

- \& can take a optional argument (in brackets). Its content will be printed after the stanza.

8.10 Various tools

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

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

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

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

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

9   Grouping

In a `minipage` environment LaTeX changes \footnote numbering from arabic to alpha-
betic 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 5.4) in a minipage but unlike with
\footnote the numbering scheme is unaltered.

`ledgroup`   Minipages, of course, are not broken across pages. Footnotes in a `ledgroup` envi-
ronment are typeset at the end of the environment, as with minipages, but the environ-
ment 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.

`ledgroupsized`   The `ledgroupsized` environment is similar to `ledroup` except that you must spec-
ify a width for the environment, as with a minipage.
\begin{ledgroupsized}[⟨*pos*⟩]{⟨*width*⟩}.

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

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

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

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

Note that normal text, footnotes, and so forth are all flush left.
\begin{ledgroupsized}{\textwidth} is effectively the same as \begin{ledgroup}

10   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.

10.1   Basic use

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

`\edpageref`   Elsewhere in the text, either before or after the \edlabel, you can refer to its lo-
`\edlineref`   cation via \edpageref{⟨*lab*⟩}, or \edlineref{⟨*lab*⟩} will produce, respectively, the
`\sublineref`   page, line, sub-line and pstart on which the \edlabel{⟨*lab*⟩} command occurred.
`\pstartref`   An \edlabel command may appear in the main text, or in the first argument of
\edtext, but not in the apparatus itself. But \edpageref, \edlineref, \sublineref,

---

[18]More precisely, you should stick to characters in the TeX categories of "letter" and "other".

\pstartref commands can also be used in the apparatus to refer to \edlabels in the
text.

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

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

10.2   Refer to a critical notes

If you want to refer to a word inside an \edtext{⟨*lemma*⟩}{⟨*app*⟩} 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.}}
```

If you add the \edlabel inside some \Xfootnote command, it will refer to that
note, and a suffix *n* will be added to the reference. You can redefine this suffix by re-
defining the command \ledinnotemark. Its actual definition is:

```
\newcommand{\ledinnotemark}[1]{#1\emph{n}}
```

10.3   Cross-referencing which return a number in any case

\xpageref   Where #1 stands for the reference.
\xlineref   However, there are situations in which you will want reledmac to return a number
\xsublineref   without displaying any warning messages about undefined labels or the like: if you want
\xpstartref   to use the reference in a context where LATEX is looking for a number, such a warning
will lead to a complaint that the number is missing. This is the case for references used
within the argument to \linenum, for example (see 5.2.5 p. 23).

For this situation, four variants of the reference commands, with the x prefix, are
supplied: \xpageref, \xlineref, \xsublineref and \xpstartref. They have these
limitations:

  • They will not tell you if the label is undefined.

  • They must be preceded in the file by at least one of the four other cross-reference
    commands—e.g., a \edlabel{foo} command, even if you never refer to that
    label—since those commands can all do the necessary processing of the .aux file,
    and the \x... ones cannot.

  • When hyperref is loaded, the hyperref link will not be added. (Indeed, it is not
    a limitation, but a feature.)

10.3.1   Cross-referencing in order to define line number of a critical note

\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., 5.2.5 p. 23 above) and sets the beginning page, line and subline numbers to those of the place where \edlabel{mouse} was placed, and the ending numbers to those where \edlabel{elephant} occurs.

10.4   Not automatic cross-referencing

\edmakelabel   Sometimes the \edlabel command cannot be used to specify exactly the page and line desired—for example, if you want to refer to a page and line number in another volume of your edition. In such cases, you can use the \edmakelabel{⟨*lab*⟩}{⟨*numbers*⟩} macro so that you can 'roll your own' label.

For example, if you say '\edmakelabel{elephant}{10|25|0}' you will create a new label, and a later call to \edpageref{elephant} would print '10' and \lineref{elephant} would print '25'. The sub-line number here is zero. It is usually best to collect your \edmakelabel statements near the top of your document, so that you can see them at a glance.

10.5   Normal LaTeX cross-referencing

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

10.6   References to lines commented in the apparatus

You may want to make a cross-reference to a passage that is referred to by \edtext. reledmac provides specific tools for this scenario.

\applabel   If you use \applabel{⟨*label*⟩} inside the second argument of a \edtext, reledmac will add a \edlabel at the beginning and end of the marked passage. The label at the beginning of the passage will have the title ⟨*label*⟩:start, while the label at the end will have the title ⟨*label*⟩:end.

If you use \linenum (5.2.5 p. 23) to refer to these labels, reledmac will use your line settings to refer to the passage.

\appref   You can also use \appref{⟨*label*⟩} and \apprefwithpage{⟨*label*⟩} to refer to these
\apprefwithpage   lines. The first one will print the lines as they are printed in the critical footnotes, while the second will print the lines as they are printed in endnotes.

\setapprefprefixsingle   If you use \setapprefprefixsingle{⟨*prefix*⟩}, ⟨*prefix*⟩ will be printed before the
\setapprefprefixmore   line numbers of a \appref-reference. If you use \setapprefprefixmore{⟨*prefix*⟩}, ⟨*prefix*⟩ will be printed before the line numbers, if you refer to more than one line.

For example, you may use:

    \setapprefprefixsingle{line~}

```
\setapprefprefixmore{lines~}
```

Note that if you have not used \setapprefprefixmore is empty, argument of \setapprefprefixsingle will used in any case.

\Xtwolinesappref   If you use \Xtwolines, \Xmorethantwolines, \Xtwolinesbutnotmore and/or
\Xmorethantwolinesappref \Xtwolinesonlyinsamepage (6.2.2 p. 30) *without the optional series argument*, the set-
\Xtwolinesbutnotmoreappref ting will also be available for \appref.
\Xtwolinesonlyinsamepage

    The commands \Xtwolinesappref{⟨*text*⟩}, \Xmorethantwolinesappref{⟨*text*⟩}, \Xtwolinesbutnotmoreappref \Xtwolinesonlyinsamepageappref can also be used, if you only want to change the reference style of \appref.

    It is possible to disable this setting for a specific \appref command by using \appref[fulllines]{⟨*label*⟩}.

\Xendtwolinesapprefwithpage   If you use one of \Xendtwolines, \Xendmorethantwolines, \Xendtwolinesbutnotmore,
\Xmorethantwolinesapprefwithpage \Xendtwolinesonlyinsamepage) (6.2.2 p. 30) *without the optional series argument*, the
\Xendtwolinesbutnotmoreapprefwithpage setting will also be available for \apprefwithpage.
\Xendtwolinesonlyinsamepageapprefwithpage

    The commands \Xendtwolinesappref{⟨*text*⟩}, \Xendmorethantwolinesappref{⟨*text*⟩}, \Xendtwolinesbutnotmoreappref, \Xendtwolinesonlyinsamepageappref can also be used, if you only want to change the reference style of \apprefwithpage.

    It is possible to disable this setting for a specific \apprefwithpage command by using \apprefwithpage[fulllines]{⟨*label*⟩}.

11 Side notes

11.1 Basis

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

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

\ledleftnote   \ledsidenote{⟨*text*⟩} will put ⟨*text*⟩ into the margin specified by the current
\ledrightnote setting of \sidenotemargin{⟨*location*⟩}. The permissible value for ⟨*location*⟩ is one
\ledsidenote out of the list left, right, inner, or outer, for example \sidenotemargin{outer}.
\sidenotemargin 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.

11.2   Setting

11.2.1   Width

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

11.2.2   Vertical position

\rightnoteupfalse   By default, sidenotes are placed to align with the last line of the note to which it refers.
\leftnoteupfalse   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).

11.2.3   Distance to the main text

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

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

These can of course be changed to suit.

11.2.4   Separator between notes

\setsidenotesep   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 use \setsidenotesep{⟨*sep*⟩}.

12   Indexing

12.1   Basis

\edindex   LATEX provides the \index{⟨*item*⟩} command for specifying that ⟨*item*⟩ and the current
page number should be added to the raw index (idx) file. The \edindex{⟨*item*⟩} macro
can be used in numbered text to specify that ⟨*item*⟩ and the current page & linenumber
should be added to the raw index file.
     Note that the file .idx will contain the right reference only after the third run, be-
cause of the internal indexing mechanism of eledmac. That means you must first run
three times (Xe/Lua)LATEX, then run makeindex and finally run again (Xe/Lua)LATEXto
get an index with the right page numbers.
     If the imakeidx or indextools package is used then the macro takes an optional
argument, which is the name of a raw index file. For example \edindex[line]{item}
will use line.idx as the raw file instead of \jobname.idx.

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

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

1. Load `imakeidx` or `indextools` .

2. Load `eledmac`.

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

12.2   Separator between page and line numbers

`\pagelinesep` The page & linenumber combination is written as `page\pagelinesep line`, where the default definition is `\newcommand{\pagelinesep}{-}` so that an item on page 3, line 5 will be noted as being at 3-5. You can renew `\pagelinesep` to get a different separator.

– is the default separator used by the MakeIndex program.

Consequently, if you want to use an other `\pagelinesep`, you have to configure your `.ist` index style file. For example if you use : as separator[19].

```
page_compositor ":"
delim_r ":
```

Read the MakeIndex program's handbook about the `.ist` file.

12.3   Using xindy

Should you decide to use `xindy` instead of `makeindex` to transform your `.idx` files into `.ind` files, you must use some specific configuration file (`.xdy`) so that `xindy` can understand `eledmac` reference syntax of which the scheme is:
`pagenumber-linenumber`

An example of such a file is provided in the "examples" folder. Read the xindy handbook to learn how to use it.[20]

This file also provides, with an explanation, the settings that are needed to put `reledmac` lines numbers in parenthesis, in order to make a better distinction between line numbers and page ranges.

In any case, you must load `reledmac` with the xindy option, in order to generate a `.xdy` file which is specific to your document. This file is needed by the `.xdy` example file which is in the "examples" folder. Its default name is `reledmac-markup-attr.xdy`, but you can change it by using your own as an argument of the `xindy+hyperref` option.

If you chose to use both `xindy` and the `hyperref` package, you must do three more things:

---

[19]For further detail, you can read `http://tex.stackexchange.com/a/32783/7712`.

[20]Or, for people who read French, read `http://geekographie.maieul.net/174`.

1. Use `xindy+hyperref` option when loading the `reledmac` package. When you
   run (Xe/Lua)LATEXwith this option, a `.xdy` configuration file will be generated with
   all the settings needed to allow internal hyperlinking in each index entry which
   is created by `\edindex`.

2. Use `hyperindex=false` option when loading hyperref.

3. Uncomment — by removing the semicolons at the beginning of the relevant lines
   — some lines in the <code>.xdy</code> file provided in the "examples" folder
   in order to restore internal links in the index to be used by the standard `index`
   command.[21].

12.4   Advanced setting

\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 number-
ing is being done; more precisely, they can be used but with odd results, so don't use
them. However, `reledmac` 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 edtabular*
edarrayc   for text entries. The final l, c, or r in the environment names indicate that the entries
edarrayr   will be flushleft (l), centered (c) or flushright (r). There is no means of specifying dif-
edtabularl ferent formats for each column, nor for specifying a fixed width for a column. The
edtabularc environments are centered with respect to the surrounding text.
edtabularr

```
\begin{edtabularc}
1 & 2 & 3 \\
a & bb & ccc \\
AAA & BB & C
\end{edtabularc}
```

1	2	3
a	bb	ccc
AAA	BB	C

Entries in the environments are the same as for the normal `array` and `tabular`
environments but there must be no ending \\ at the end of the last row. *There must be
the same number of column designators (the &) 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.

---

[21]These are the recommended lines to provide the best possible compatibility between hyperref and xindy,
even without using `reledmac`.

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 the

\spreadtext   calculation of column widths. \spreadtext{⟨*text*⟩} is the analagous command for use in edtabular environments.

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

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

\edrowfill   The macro \edrowfill{⟨*start*⟩}{⟨*end*⟩}{⟨*fill*⟩} fills columns number ⟨*start*⟩ to ⟨*end*⟩ inclusive with ⟨*fill*⟩. The ⟨*fill*⟩ argument can be any horizontal 'fill'. For example \hrulefill or \upbracefill.

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

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

```
\begin{edtabularr}
1                           & 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                           &      & l  & 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        ⎵‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾⎵
                     pq                         dgh
    k            l    co    ghweropjklmnbvcxys
    1       2    3           ‾‾‾‾‾‾‾‾‾‾‾‾
```

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}
```

$$
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
a & \llcorner\!\lrcorner & & d \\
A & B & C & D
\end{array}
$$

\edatleft   \edatleft[⟨*math*⟩]{⟨*symbol*⟩}{⟨*halfheight*⟩} typesets the math ⟨*symbol*⟩ as \left{⟨*symbol*⟩}
\edatright   with the optional ⟨*math*⟩ centered before it. The ⟨*symbol*⟩ is twice ⟨*halfheight*⟩ tall. The
                \edatright macro is similar and it typesets \right{⟨*symbol*⟩} with ⟨*math*⟩ 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`
`\edaftertab` \edbeforetab{⟨*text*⟩}{⟨*entry*⟩}, where ⟨*entry*⟩ is an entry in the leftmost column, typesets ⟨*text*⟩ left justified before the ⟨*entry*⟩. Similarly \edaftertab{⟨*entry*⟩}{⟨*text*⟩}, where ⟨*entry*⟩ is an entry in the rightmost column, typesets ⟨*text*⟩ right justified after the ⟨*entry*⟩.

For example:

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

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

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

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

$$\left.\begin{array}{cccc}
a & b & C & d \\
v & w & x & y \\
m & n & o & p \\
k &   & L & cvb
\end{array}\right|$$

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

14 Sectioning commands

14.1 Sectioning commands without line numbers or critical notes

The standard sectioning commands (\chapter, \section etc.) can be used inside numbered text. In this case, you must call them as an optional argument of \pstart (4.2.3 p. 16):

```
\pstart[\section{section}]
Pstart content.
\pend
```

The line which contains them will not been numbered, and you can not add critical notes inside.

14.2   Sectioning commands with line numbering and critical notes

You have to use the following commands:

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

- \eledchapter*

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

- \eledsection*

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

- \eledsubsection*

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

- \eledsubsubsection*

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

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

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

For technical reasons, the page break before \elechapter ca not be added automatically. You have to insert it manually via \beforeeledchapter, which must be called outside of a numbered section.

14.3   Optimization

\noeledsec   If you are not going to have any \eledxxx commands, then load reledmac with \noeledsec option. That will suppress the generation of unneeded .eledsec file, keep memory and make reledmac faster.

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 numbered section. You must open any quotation environments inside a `\start`-`\pend` block, not outside. A quotation environment MUST not be opened immediately after a `\pstart` and MUST not be closed immediately before a `\pend`.

In some cases, you do not want these environments to be redefined in numbered sections. You can load the package with the option `noquotation` to prevent this redefinition.

16 Page breaks

16.1 Control page breaking

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

`\ledpb`
`\lednopb`

- `\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

`\ledpbsetting`   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 reledpar 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.

16.2 Prevent page break in a long verses

`\lednopbinversetrue`   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

17.2   'No room for a new'

Sometime, especially when using reledmac with other packages, you could obtain
warning message such 'no room for a new count' or 'no room for a new write'.

The first thing in order to prevent such problem is to use the options to optimize
reledmac. For example, if you need only two series of notes, use series={A,B} option.
Read ?? p. ?? in order to know which are there options.

However, if with these options you still have such message, here are some tricks.

'no room for a new count' is often caused by a conjunction with biblatex. Load
    reledmac (and reledpar) *before* biblatex.

'no room for a new write' can be caused by with multiple indexes. In this case, use
    indextools of imakeidx with the splitindex option, in order to obtain only
    one .idx file. If that does not solve your problem, you can use morewrites pack-
    age. That should solve the problem, but LaTeX will be slower.

If after reading and applying these advices you have still problem, contact us with a
minimal working example.

17.2.1 Marginal notes

In general, `reledmac`'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.

17.2.2 Paragraph shape

`\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, `reledmac` 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 does not crop up at all often.

17.2.3 Paragraphed footnotes

The restriction on explicit line-breaking in paragraphed footnotes, mentioned in a footnote 17 p. 29, and described in more detail on XII.6.3 p. 146, really is a nuisance if that is something you need to do. There are some possible solutions, described by Michael Downes, but this area remains unsatisfactory.

`\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}`

17.2.4 Use with other packages

Because of `reledmac`'s complexity it may not play well with other packages. In particular `reledmac` is sensitive to commands in the arguments to the `\edtext` and `\*footnote` macros (this is discussed in more detail in section VI, 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.

`\morenoexpands`　　You can define the macro `\morenoexpands` to modify macros that you call within `\edtext`. Because of the way `reledmac` 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 this[22] 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
```

(\@secondoftwo is an internal LaTeX macro that takes two arguments and thows 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 Peter Wilson 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.

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

17.3   Parallel typesetting

Peter Wilson has developed the ledpar package as an extension to ledmac specifically for parallel typesetting of critical texts. This also cooperates with the babel / polyglossia packages for typesetting in multiple languages. reledpar is the successor of the primitive ledpar package.

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

---

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

I Implementation overview

We present the `reledmac` code in roughly the order in which it is used during a run of TeX. The order is *exactly* that in which it is 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.

After package options, we begin with the commands you use to start and stop line numbering in a section of text (Section II). 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 V); 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 VI), followed by the macros that take each paragraph apart, attach the line numbers and insertions, and send the result to the vertical list (Section VII). The footnote commands (Section XII) and output routine (Section **??**) finish the main part of the processing; cross-referencing (Section XXIII) and endnotes (Section XIX) complete the story.

In what follows, macros with an @ in their name are more internal to the workings of `reledmac` 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.

II Preliminaries

II.1 Links with original `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.

II.2 Package declaration

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

```
1  %<*code>
2  \NeedsTeXFormat{LaTeX2e}
3  \ProvidesPackage{reledmac}[2015/07/27 v2.0.1 typeset critical edition]%
4  %
```

II.3   Package options

\ifledfinal
\ifnocritical@
\if@noeled@sec
\ifnoend@
\ifnofamiliar@
\ifnoledgroup@
\ifparapparatus@
\ifnoquotation@
\iflednopbinverse
\ifparledgroup
\ifwidthliketwocolumns
\ifxindy@
\ifxindyhyperref@
\ifeledmaccompat@

Use this to remember which option is used, set and execute the options with final as the default. We use xkeyval in order to manage options with argument.

```
5  \RequirePackage{xkeyval}
6  %
```

The parledgroup option is for reledpar. However, it has consequence on reledmac internal command. So we need to define the boolean now.

```
7  \newif\ifparledgroup
8  %
```

And now, the options of reledmac.

```
9  \DeclareOptionX{series}[A,B,C,D,E]{\xdef\default@series{#1}}
10 \ExecuteOptionsX{series}%
11
12 \newif\if@noeled@sec%
13 \DeclareOptionX{noeledsec}{\@noeled@sectrue}
14
15 \newif\ifnocritical@%
16 \DeclareOptionX{nocritical}{\nocritical@true}%
17
18 \newif\ifnofamiliar@%
19 \DeclareOptionX{nofamiliar}{\nofamiliar@true}%
20
21 \newif\ifnoledgroup@%
22 \DeclareOptionX{noledgroup}{\noledgroup@true}%
23
24 \newif\ifnoend@%
25 \DeclareOptionX{noend}{%
26  \let\l@dend@open\@gobble%
27  \let\l@dend@close\relax%
28   \global\let\l@dend@stuff=\relax%
29   \noend@true%
30 }%
31
32 \newif\ifnoquotation@
33 \DeclareOptionX{noquotation}{\noquotation@true}
34
35
36 \newif\ifledfinal
37 \DeclareOptionX{final}{\ledfinaltrue}
38 \DeclareOptionX{draft}{\ledfinalfalse}
```

```
39 \ExecuteOptionsX{final}
40
41 \newif\ifparapparatus@
42 \DeclareOptionX{parapparatus}{\parapparatus@true}
43
44 \newif\iflednopbinverse
45 \DeclareOptionX{nopbinverse}{\lednopbinversetrue}
46
47 \newif\ifwidthliketwocolumns%
48 \DeclareOptionX{widthliketwocolumns}{\widthliketwocolumnstrue}%
49
50 \newif\ifxindy@
51 \DeclareOptionX{xindy}[eledmac-markup-attr.xdy]{%
52   \AtBeginDocument{\immediate\openout\eledmac@xindy@out=#1}%
53   \newwrite\eledmac@xindy@out%
54   \xindy@true%
55   \gdef\eledmacmarkuplocrefdepth{:depth 1}%
56   \AtEndDocument{\immediate\closeout\eledmac@xindy@out}%
57 }%
58
59 \newif\ifxindyhyperref@
60 \DeclareOptionX{xindy+hyperref}{%
61   \xindyhyperref@true%
62 }%
63
64 \newif\ifeledmaccompat@%
65 \DeclareOptionX{eledmac-compat}{%
66   \eledmaccompat@true%
67 }%
68 %
```

We use the starred form of `\ProcessOptionsX` 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.

```
69 \ProcessOptionsX*\relax
70
71 %
```

II.4 Loading packages

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

```
72 \RequirePackage{xargs}
73 \RequirePackage{etoolbox}
```

```
74  \RequirePackage{etex}
75  \reserveinserts{32}
76  \RequirePackage{suffix}
77  \RequirePackage{xstring}
78  \RequirePackage{ifluatex}
79  \RequirePackage{ragged2e}
80  \RequirePackage{ragged2e}
81  \RequirePackage{ifxetex}%
82  %
```

II.5   Boolean flags

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

```
83  \newif\ifl@dmemoir
84  \@ifclassloaded{memoir}{\l@dmemoirtrue}{\l@dmemoirfalse}
85
86  %
```

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

```
87  \newif\ifl@imakeidx
88  \@ifpackageloaded{imakeidx}{\l@imakeidxtrue}{}%False is the default value
89  %
```

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

```
90  \newif\ifl@indextools%
91  \@ifpackageloaded{indextools}{%
92    \l@indextoolstrue%
93    \l@imakeidxtrue%
94    \let\imki@wrindexentry\indtl@wrindexentry%
95    }{}%
96  %
```

False is the default value. We consider indextools as a variant of imakeidx. That is why we set \ifl@imakeidx to true. We also let \imki@wrindexentry to \indtl@wrindexentry.

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

```
97  \ifdef{\if@RTL}{}{\newif\if@RTL}
98  %
```

II.6   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.

`\reledmac@warning`   Write a warning message.

```
99 \newcommand{\reledmac@warning}[1]{\PackageWarning{reledmac}{#1}}
100 %
```

`\reledmac@error`   Write an error message.

```
101 \newcommand{\reledmac@error}[2]{\PackageError{reledmac}{#1}{#2}}
102 %
```

`\led@err@NumberingStarted`
`\led@err@NumberingNotStarted`
`\led@err@NumberingShouldHaveStarted`
```
103 \newcommand*{\led@err@NumberingStarted}{%
104    \reledmac@error{Numbering has already been started}{\@ehc}}
105 \newcommand*{\led@err@NumberingNotStarted}{%
106    \reledmac@error{Numbering was not started}{\@ehc}}
107 \newcommand*{\led@err@NumberingShouldHaveStarted}{%
108    \reledmac@error{Numbering should already have been started}{\@ehc}}
109 %
```

`\led@err@edtextoutsidepstart`
```
110 \newcommand*{\led@err@edtextoutsidepstart}{%
111    \reledmac@error{\string\edtext\space outside numbered paragraph (\pstart\
    ldots\pend)}{\@ehc}}%
112 %
```

`\led@mess@NotesChanged`
```
113 \newcommand*{\led@mess@NotesChanged}{%
114    \typeout{reledmac reminder: }%
115    \typeout{ The number of the footnotes in this section
116             has changed since the last run.}%
117    \typeout{ You will need to run LaTeX two more times
118             before the footnote placement}%
119    \typeout{ and line numbering in this section are
120             correct.}}
121 %
```

`\led@mess@SectionContinued`
```
122 \newcommand*{\led@mess@SectionContinued}[1]{%
123    \message{Section #1 (continuing the previous section)}}
124 %
```

`\led@err@LineationInNumbered`
```
125 \newcommand*{\led@err@LineationInNumbered}{%
126    \reledmac@error{You can't use \string\lineation\space within
127                   a numbered section}{\@ehc}}
128 %
```

`\led@warn@BadLineation`₂₉
`\led@warn@BadLinenummargin`₃₀
`\led@warn@BadLockdisp`₃₁
`\led@warn@BadSublockdisp`₃₂

```
\newcommand*{\led@warn@BadLineation}{%
  \reledmac@warning{Bad \string\lineation\space argument}}
\newcommand*{\led@warn@BadLinenummargin}{%
  \reledmac@warning{Bad \string\linenummargin\space argument}}
\newcommand*{\led@warn@BadLockdisp}{%
  \reledmac@warning{Bad \string\lockdisp\space argument}}
\newcommand*{\led@warn@BadSublockdisp}{%
  \reledmac@warning{Bad \string\sublockdisp\space argument}}
%
```

`\led@warn@NoLineFile`₃₈

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

`\led@warn@LineFileObsolete`₄₁

```
\newcommand*{\led@warn@Obsolete}[1]{%
  \reledmac@warning{Line-list file #1 was obsolete. We have not read it.
  Please run LaTeX again.}}
%
```

`\led@warn@BadAdvancelineSubline`₄₄
`\led@warn@BadAdvancelineLine`₄₅

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

`\led@warn@BadSetline`₅₁
`\led@warn@BadSetlinenum`₅₂

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

`\led@err@PstartNotNumbered`₅₆
`\led@err@PstartInPstart`₅₇
`\led@err@PendNotNumbered`₅₈
`\led@err@PendNoPstart`₅₉
`\led@err@AutoparNotNumbered`₆₀
`\led@err@NumberingWithoutPstart`₆₁

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

```
167 \newcommand*{\led@err@AutoparNotNumbered}{%
168   \reledmac@error{\string\autopar\space must be used within a
169                   numbered section}{\@ehc}}
170 \newcommand*{\led@err@NumberingWithoutPstart}{%
171   \reledmac@error{\string\beginnumbering...\string\endnumbering\space
    without \string\pstart}{\@ehc}}%
172 %
```

```
\led@warn@BadAction 173 \newcommand*{\led@warn@BadAction}{%
174   \reledmac@warning{Bad action code, value \next@action.}}
175 %
```

```
 \led@warn@DuplicateLabel 176 \newcommand*{\led@warn@DuplicateLabel}[1]{%
ed@warn@AppLabelOutEdtext 177   \reledmac@warning{Duplicate definition of label `#1' on page \the\pageno
   \led@warn@RefUndefined    .}}
                         178 \newcommand*{\led@warn@AppLabelOutEdtext}[1]{%
                         179   \reledmac@warning{\string\applabel\space outside of \string\edtext\space
                             `#1' on page \the\pageno.}}%
                         180 \newcommand*{\led@warn@RefUndefined}[1]{%
                         181   \reledmac@warning{Reference `#1' on page \the\pageno\space undefined.
                         182                   Using `000'.}}
                         183 %
```

```
\led@warn@NoMarginpars 184 \newcommand*{\led@warn@NoMarginpars}{%
185   \reledmac@warning{You can't use \string\marginpar\space in numbered text
    }}
186 %
```

```
ed@warn@BadSidenotemargin 187 \newcommand*{\led@warn@BadSidenotemargin}{%
188   \reledmac@warning{Bad \string\sidenotemmargin\space argument}}
189 %
```

```
\led@warn@NoIndexFile 190 \newcommand*{\led@warn@NoIndexFile}[1]{%
191   \reledmac@warning{Undefined index file #1}}
192 %
```

```
led@warn@SeriesStillExist 193 \newcommand{\led@warn@SeriesStillExist}[1]{%
194   \reledmac@warning{Series #1 is still existing !}%
195 }%
196 %
```

`\led@err@ManySidenotes`97 `\newcommand{\led@err@ManySidenotes}{%`
`\led@err@ManyLeftnotes`98   `\ifledRcol@%`
`\led@err@ManyRightnotes`99   `\reledmac@warning{\itemcount@\space sidenotes on line \the\line@numR\`
`space p. \the\page@numR}%`
200   `\else%`
201   `\reledmac@warning{\itemcount@\space sidenotes on line \the\line@num\`
`space p. \the\page@num}%`
202   `\fi%`
203 `}%`
204 `\newcommand{\led@err@ManyLeftnotes}{%`
205   `\ifledRcol@%`
206   `\reledmac@warning{\itemcount@\space leftnotes on line \the\line@numR\`
`space p. \the\page@numR}%`
207   `\else%`
208   `\reledmac@warning{\itemcount@\space leftnotes on line \the\line@num\`
`space p. \the\page@num}%`
209   `\fi%`
210 `}%`
211 `\newcommand{\led@err@ManyRightnotes}{%`
212   `\ifledRcol@%`
213   `\reledmac@warning{\itemcount@\space rightnotes on line \the\line@numR\`
`space p. \the\page@numR}%`
214   `\else%`
215   `\reledmac@warning{\itemcount@\space rightnotes on line \the\line@num\`
`space p. \the\page@num}%`
216   `\fi%`
217 `}%`
218 `%`

`\led@err@TooManyColumns`219 `\newcommand*{\led@err@TooManyColumns}{%`
`\led@err@UnequalColumns`220   `\reledmac@error{Too many columns}{\@ehc}}`
`\led@err@LowStartColumn`221 `\newcommand*{\led@err@UnequalColumns}{%`
`\led@err@HighEndColumn`222   `\reledmac@error{Number of columns is not equal to the number`
`in the previous row (or \protect\\ \space forgotten?)}{\`
`@ehc}}`
`\led@err@ReverseColumns`223
224 `\newcommand*{\led@err@LowStartColumn}{%`
225   `\reledmac@error{Start column is too low}{\@ehc}}`
226 `\newcommand*{\led@err@HighEndColumn}{%`
227   `\reledmac@error{End column is too high}{\@ehc}}`
228 `\newcommand*{\led@err@ReverseColumns}{%`
229   `\reledmac@error{Start column is greater than end column}{\@ehc}}`
230 `%`

`\led@err@EdtextWithoutFootnote`231 `\newcommand{\led@err@EdtextWithoutFootnote}{%`
232   `\reledmac@error{edtext without Xfootnote. Check syntaxis.}{\@ehc}%`
233 `}%`
234 `%`

```
err@FootnoteWithoutEdtext 235 \newcommand{\led@err@FootnoteWithoutEdtext}{%
                          236   \reledmac@error{Xfootnote without edtext. Check syntax.}{\@ehc}%
                          237 }%
                          238 %
```

```
rror@ImakeidxAfterEledmac 239 \newcommand{\led@error@ImakeidxAfterEledmac}{%
                          240   \reledmac@error{Imakeidx must be loaded before eledmac.}{\@ehc}%
                          241 }%
                          242 %
```

```
or@IndextoolsAfterEledmac 243 \newcommand{\led@error@IndextoolsAfterEledmac}{%
                          244   \reledmac@error{Indextools must be loaded before eledmac.}{\@ehc}%
                          245 }%
                          246 %
```

```
error@fail@patch@@makecol 247 \newcommand{\led@error@fail@patch@@makecol}{%
                          248   \reledmac@error{Fail to patch \string\@makecol\space command.}{\@ehc}%
                          249 }%
                          250 %
```

```
ror@fail@patch@@reinserts 251 \newcommand{\led@error@fail@patch@@reinserts}{%
                          252   \reledmac@error{Fail to patch \string\@reinserts\space command.}{\@ehc}%
                          253 }%
                          254 %
```

```
r@fail@patch@@doclearpage 255 \newcommand{\led@error@fail@patch@@doclearpage}{%
                          256   \reledmac@error{Fail to patch \string\@doclearpage\space command.}{\@ehc}
                              %
                          257 }%
                          258 %
```

```
r@fail@patch@@iiiminipage 259 \newcommand{\led@error@fail@patch@@iiiminipage}{%
                          260   \reledmac@error{Fail to patch \string\@iiiminipage\space command.}{\@ehc}
                              %
                          261 }%
                          262 %
```

```
or@fail@patch@endminipage 263 \newcommand{\led@error@fail@patch@endminipage}{%
                          264   \reledmac@error{Fail to patch \string\endminipage\space command.}{\@ehc}%
                          265 }%
                          266 %
```

II.7 Gobbling

Here, we define some commands which gobble their arguments.

```
\@gobblethree₆₇  \providecommand*{\@gobblethree}[3]{}
 \@gobblefour₆₈  \providecommand*{\@gobblefour}[4]{}
 \@gobblefive₆₉  \providecommand*{\@gobblefive}[5]{}
         270  %
```

II.8 Miscellaneous commands

\showlemma \showlemma{⟨*lemma*⟩} typesets the lemma text in the body. It depends on the option.

```
271 \ifledfinal
272   \newcommand*{\showlemma}[1]{#1}
273 \else
274   \newcommand*{\showlemma}[1]{\underline{#1}}
275 \fi
276
277 %
```

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

Initialize it as \empty.

```
278 \let\linenumberlist=\empty
279
280 %
```

\@l@dtempcnta In imitation of LaTeX, we create a couple of scratch counters.
\@l@dtempcntb   LaTeX already defines \@tempcnta and \@tempcntb but Peter Wilson found in the past that it can be dangerous to use these (for example one of the AMS packages did something nasty to the ccaption package's use of one of these).

```
281 \newcount\@l@dtempcnta \newcount\@l@dtempcntb
282 %
```

II.9 Prepare `reledpar`

\ifnumberingR In preparation for the reledpar package, these are related to the 'right' text of paral-
\ifl@dpairing lel texts (when \ifl@dpairing is TRUE). They are explained in the eledpar manual.
\ifl@dpaging
\l@dpagingtrue
\l@dpagingfalse ₂₈₃ \newif\ifl@dpairing
\ifl@dprintingpages ₂₈₄ \newif\ifl@dpaging%
\l@dprintingpagestrue ₂₈₅ \newif\ifl@dprintingpages%
\l@dprintingpagesfalse ₂₈₆ \newif\ifl@dprintingcolumns%
\ifl@dprintingcolumns ₂₈₇ \newif\ifpst@rtedL
\l@dprintingcolumnstrue
\l@dprintingcolumnsfalse
\l@dpairingtrue
\l@dpairingfalse
\ifpst@rtedL
\pst@rtedLtrue
\pst@rtedLfalse
\l@dnumpstartsL
\ifledRcol
```

```
288 \newcount\l@dnumpstartsL
289 %
```

`\ifledRcol` is set to true in the `Rightside` environnement. It must be not confused with `\ifledRcol@` which is set to true when a right line is processed, in `\Pages` or `\Columns`.

```
290 \newif\ifledRcol
291 \newif\ifledRcol@
292 %
```

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

```
293 \newif\ifnumberingR
294 %
```

III   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 `\resumenumbering` commands have appeared; it need not 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.

```
295 \newcount\section@num
296 \section@num=0
297 \let\extensionchars=\empty
298 %
```

`\ifnumbering` The `\ifnumbering` flag is set to `true` if we are within a numbered section (that is,
`\numberingtrue` between `\beginnumbering` and `\endnumbering`). You can use `\ifnumbering` in your
`\numberingfalse` own code to check whether you are in a numbered section, but do not change the flag's value.

```
299 \newif\ifnumbering
300 %
```

`\beginnumbering`
`\initnumbering@reg`

`\beginnumbering` begins a section of numbered text. When it is 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 is done the lists will remain but the counters will return to zero. Those same counters will then be used as we process the text of this section, but the assignments will be made globally. These initializations actually apply to both uses, though in all other respects there should be no direct interaction between the use of these counters and variables in the two processing steps.   For parallel processing :

- zero `\l@dnumpstartsL` — the number of chunks to be processed.

- set `\ifpst@rtedL` to FALSE.

```
301 \newcommand*{\beginnumbering}{%
302   \ifnumbering
303     \led@err@NumberingStarted
304     \endnumbering
305   \fi
306   \global\numberingtrue
307   \global\advance\section@num \@ne
308   \initnumbering@reg
309   \message{Section \the\section@num }%
310   \line@list@stuff{\jobname.\extensionchars\the\section@num}%
311   \l@dend@stuff
312   \setcounter{pstart}{1}
313   \ifl@dpairing
314     \global\l@dnumpstartsL \z@
315     \global\pst@rtedLfalse
316 %
```

The tools for section's title commands are called:

- Define an empty list of pstart number where sectioning commands are called.

- Input auxiliary file with the description of section titles.

- Open the same auxiliary file to write in.

```
317   \else
318     \begingroup
319     \global\@afterindenttrue%In order to retablish normal feature if the \
    begingroup was not here
320     \initnumbering@quote
321     \ifwidthliketwocolumns%
322       \csuse{setwidthliketwocolumns@\columns@position}%
323       \csuse{setpositionliketwocolumns@\columns@position}%
324     \fi%
```

```
325    \fi
326    \gdef\eled@sections@@{}%
327    \if@noeled@sec\else%
328      \makeatletter\InputIfFileExists{\jobname.eledsec\the\section@num}{}{}\
  makeatother%
329      \immediate\openout\eled@sectioning@out=\jobname.eledsec\the\section@num
  \relax%
330    \fi%
331  }
332  \newcommand*{\initnumbering@reg}{%
333    \global\pst@rtedLfalse
334    \global\l@dnumpstartsL \z@
335    \global\absline@num \z@
336    \gdef\normal@page@break{}
337    \gdef\l@prev@pb{}
338    \gdef\l@prev@nopb{}
339    \global\line@num \z@
340    \global\subline@num \z@
341    \global\@lock \z@
342    \global\sub@lock \z@
343    \global\sublines@false
344    \global\let\next@page@num=\relax
345    \global\let\sub@change=\relax
346    \resetprevline@
347    \resetprevpage@num
348    }
349
350  %
```

\endnumbering must follow the last text for a numbered section. It takes care of no-
tifying you when changes have been noted in the input that require running the file
through again to move everything to the right place.

```
351  \def\endnumbering{%
352    \ifnumbering
353      \global\numberingfalse
354      \normal@pars
355      \ifnum\l@dnumpstartsL=0%
356        \led@err@NumberingWithoutPstart%
357      \fi%
358      \ifl@dpairing
359        \global\pst@rtedLfalse
360      \else
361        \ifx\insertlines@list\empty\else
362          \global\noteschanged@true
363        \fi
364        \ifx\line@list\empty\else
365          \global\noteschanged@true
366        \fi
367      \fi
```

```
368        \ifnoteschanged@
369          \led@mess@NotesChanged
370        \fi
371      \else
372        \led@err@NumberingNotStarted
373      \fi
374      \autoparfalse
375      \if@noeled@sec\else%
376        \immediate\closeout\eled@sectioning@out%
377      \fi%
378      \ifl@dpairing\else
379        \global\l@dnumpstartsL=\z@%
380        \endgroup
381      \fi
382  }
383  %
```

\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.[23]

```
384  \newcommand{\pausenumbering}{%
385    \ifautopar\global\autopar@pausetrue\fi%
386    \endnumbering\global\numberingtrue}
387  %
```

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.

```
388  \newcommand*{\resumenumbering}{%
389    \ifnumbering
390        \ifautopar@pause\autopar\fi
391        \global\pst@rtedLtrue
392        \global\advance\section@num \@ne
393        \led@mess@SectionContinued{\the\section@num}%
394        \line@list@stuff{\jobname.\extensionchars\the\section@num}%
395        \l@dend@stuff
396        \ifl@dpairing\else%
397            \begingroup%
398            \initnumbering@quote%
399            \ifwidthliketwocolumns%
400               \csuse{setwidthliketwocolumns@\columns@position}%
401               \csuse{setpositionliketwocolumns@\columns@position}%
402            \fi%
403        \fi%
404    \else
405      \led@err@NumberingShouldHaveStarted
406      \endnumbering
```

---

[23]Peter Wilson's thanks to Wayne Sullivan, who suggested the idea behind these macros.

```
407      \beginnumbering
408    \fi}
409
410
411 %
```

IV   List macros

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.

The historical list tools of ledmac are keept, because in many cause there are more useful than etoolbox's lists. They allows to get and delete the first element of a list in one operation. They also expands the items add to the list.

However, etoolbox's lists are more useful to loop on them. Consequently, depending of what we need, we use one or either.

It could be nice to unify them to the LaTeX3 list, however such migration would take quite time with some risk of error, for a gain which will be minor.

\list@create   The \list@create macro creates a new list. This macro does not do anything beyond initializing an empty list macro.

```
412 \newcommand*{\list@create}[1]{%
413    \global\let#1=\empty%
414 }%
415 %
```

\list@clear   The \list@clear macro just initializes a list to the empty list; it is no different from \list@create in its effect, but it is in its semantic .

```
416 \newcommand*{\list@clear}[1]{%
417    \global\let#1=\empty%
418 }
419 %
```

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

```
420 \newtoks\led@toksa \newtoks\led@toksb
421 \global\led@toksa={\\}
422 \long\def\xright@appenditem#1\to#2{%
423    \global\led@toksb=\expandafter{#2}%
```

```
424    \xdef#2{\the\led@toksb\the\led@toksa\expandafter{#1}}%
425    \global\led@toksb={}}
426 %
```

\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.

```
427 \long\def\xleft@appenditem#1\to#2{%
428   \global\led@toksb=\expandafter{#2}%
429   \xdef#2{\the\led@toksa\expandafter{#1}\the\led@toksb}%
430   \global\led@toksb={}}
431 %
```

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

```
432 \def\gl@p#1\to#2{\expandafter\gl@poff#1\gl@poff#1#2}
433 \long\def\gl@poff\\#1#2\gl@poff#3#4{\gdef#4{#1}\gdef#3{#2}}
434
435 %
```

V   Line counting

V.1   Choosing the system of lineation

Line number can be reset at each section (default) ; at each page ; at each pstart. Here
we define internal codes for these systems and the macros.

\ifbypstart@   The \ifbypage@ and \ifbypstart@ flag specifie the current lineation system:
\bypstart@true
\bypstart@false
\ifbypage@   • line-of-page: bypstart@ = false and bypage@ = true.
\bypage@true
\bypage@false   • line-of-pstart: bypstart@ = true and bypage@ = false.

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

```
436 \newif\ifbypage@
437 \newif\ifbypstart@
438 %
```

The \ifbypage@R and \ifbypstart@R flag specifie the current lineation for right
side in case of using reledpar. They are now defined because they are used in some
specific code. reledpar will use the line-of-section system unless instructed otherwise.

\ifbypage@R `439` \newif\ifbypage@R
\ifbypstart@R `440` \newif\ifbypstart@R
`441` %

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

```
442 \newcommand*{\lineation}[1]{{
443 %
```

We can't change the lineation system inside numbering section.

```
444    \ifnumbering
445       \led@err@LineationInNumbered
446    \else
447 %
```

If the argument is page.

```
448       \def\@tempa{#1}\def\@tempb{page}%
449       \ifx\@tempa\@tempb
450          \global\bypage@true
451          \global\bypstart@false
452          \unless\ifnocritical@%
453             \Xpstart[][false]%
454          \fi%
455 %
```

If the argument is pstart.

```
456       \else
457          \def\@tempb{pstart}%
458          \ifx\@tempa\@tempb
459             \global\bypage@false
460             \global\bypstart@true
461             \unless\ifnocritical@%
462                \Xpstart%
463             \fi%
464 %
```

And finally, if the argument is section (default).

```
465             \else
466                \def\@tempb{section}
467                \ifx\@tempa\@tempb
468                \global\bypage@false
469                \global\bypstart@false
470                \unless\ifnocritical@%
471                   \Xpstart[][false]%
472                \fi%
473 %
```

In other case, it is an error.

```
474             \else
475                   \led@warn@BadLineation
476                \fi
477             \fi
478       \fi
```

```
479    \fi}}
480  %
```

V.2   Line number margin

\linenummargin   \linenummargin{⟨*word*⟩} specify which margin line numbers are in; it takes one ar-
\line@margin   gument, a string, which value can be left ; right; inner or outer.
\l@dgetline@margin   The selection is recorded in the count \line@margin: 0 for left, 1 for right, 2 for
                     outer, and 3 for inner.

```
481  \newcount\line@margin
482
483  \newcommand*{\linenummargin}[1]{{%
484    \l@dgetline@margin{#1}%
485    \ifnum\@l@dtempcntb>\m@ne
486      \ifledRcol
487        \global\line@marginR=\@l@dtempcntb
488      \else
489        \global\line@margin=\@l@dtempcntb
490      \fi
491  \fi}}
492
493  \newcommand*{\l@dgetline@margin}[1]{%
494    \def\@tempa{#1}\def\@tempb{left}%
495    \ifx\@tempa\@tempb
496      \@l@dtempcntb \z@
497    \else
498      \def\@tempb{right}%
499      \ifx\@tempa\@tempb
500        \@l@dtempcntb \@ne
501      \else
502        \def\@tempb{outer}%
503        \ifx\@tempa\@tempb
504          \@l@dtempcntb \tw@
505        \else
506          \def\@tempb{inner}%
507          \ifx\@tempa\@tempb
508            \@l@dtempcntb \thr@@
509          \else
510            \led@warn@BadLinenummargin
511            \@l@dtempcntb \m@ne
512          \fi
513        \fi
514      \fi
515  \fi}
516
517  %
```

V.3   Line number initialization and increment

\c@firstlinenum
\c@linenumincrement
The following counters tell reledmac 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.

```
518 \newcounter{firstlinenum}
519   \setcounter{firstlinenum}{5}
520 \newcounter{linenumincrement}
521   \setcounter{linenumincrement}{5}
522 %
```

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

```
523 \newcounter{firstsublinenum}
524   \setcounter{firstsublinenum}{5}
525 \newcounter{sublinenumincrement}
526   \setcounter{sublinenumincrement}{5}
527
528 %
```

\firstlinenum
\linenumincrement
\firstsublinenum
\sublinenumincrement
These macros can be used to set the corresponding counters.

```
529
530 \newcommand*{\firstlinenum}[1]{%
531   \ifledRcol%
532     \setcounter{firstlinenumR}{#1}%
533   \else%
534     \setcounter{firstlinenum}{#1}%
535   \fi%
536 }
537 \newcommand*{\linenumincrement}[1]{%
538   \ifledRcol%
539     \setcounter{linenumincrementR}{#1}%
540   \else%
541     \setcounter{linenumincrement}{#1}%
542   \fi%
543 }
544 \newcommand*{\firstsublinenum}[1]{%
545   \ifledRcol%
546     \setcounter{firstsublinenumR}{#1}%
547   \else%
548     \setcounter{firstsublinenum}{#1}%
549   \fi%
550 }
551 \newcommand*{\sublinenumincrement}[1]{%
552   \ifledRcol%
```

```
553        \setcounter{sublinenumincrementR}{#1}%
554    \else%
555        \setcounter{sublinenumincrement}{#1}%
556    \fi%
557 }
558
559 %
```

V.4   Line number locking

\lockdisp   When line locking is being used, the \lockdisp{⟨*word*⟩} macro specifies whether a
\lock@disp   line number—if one is due to appear—should be printed on the first printed line or on
\l@dgetlock@disp 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.

```
560 \newcount\lock@disp
561 \newcommand{\lockdisp}[1]{{%
562    \l@dgetlock@disp{#1}%
563    \ifnum\@l@dtempcntb>\m@ne
564      \global\lock@disp=\@l@dtempcntb
565    \else
566      \led@warn@BadLockdisp
567    \fi}}
568 \newcommand*{\l@dgetlock@disp}[1]{
569    \def\@tempa{#1}\def\@tempb{first}%
570    \ifx\@tempa\@tempb
571      \@l@dtempcntb \z@
572    \else
573      \def\@tempb{last}%
574      \ifx\@tempa\@tempb
575        \@l@dtempcntb \@ne
576      \else
577        \def\@tempb{all}%
578        \ifx\@tempa\@tempb
579          \@l@dtempcntb \tw@
580        \else
581          \@l@dtempcntb \m@ne
582        \fi
583      \fi
584    \fi}
585
586 %
```

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

```
587 \newcount\sublock@disp
588 \newcommand{\sublockdisp}[1]{{%
```

```
589    \l@dgetlock@disp{#1}%
590    \ifnum\@l@dtempcntb>\m@ne
591      \global\sublock@disp=\@l@dtempcntb
592    \else
593      \led@warn@BadSublockdisp
594    \fi}}
595
596 %
```

V.5   Line number style

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

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

NOTE: In v0.7 \linenumrep and \sublinenumrep replaced the internal \linenumr@p and \sublinenumr@p.

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

```
597 \newcommand*{\linenumberstyle}[1]{%
598    \def\linenumrep##1{\@nameuse{@#1}{##1}}}
599 \newcommand*{\sublinenumberstyle}[1]{%
600    \def\sublinenumrep##1{\@nameuse{@#1}{##1}}}
601 %
```

Initialise the number styles to arabic.

```
602 \linenumberstyle{arabic}
603    \let\linenumr@p\linenumrep
604 \sublinenumberstyle{arabic}
605    \let\sublinenumr@p\sublinenumrep
606
607 %
```

V.6   Line number printing

\leftlinenum
\rightlinenum
\linenumsep
\numlabfont
\ledlinenum

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

Whatever these macros output gets printed in a box that will be put into the appropriate margin without any space between it and the line of text. You will 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.

```
608  \newlength{\linenumsep}
609    \setlength{\linenumsep}{1pc}
610  \newcommand*{\numlabfont}{\normalfont\scriptsize}
611  \newcommand*{\ledlinenum}{%
612    \bgroup%
613    \ifluatex%
614      \luatextextdir TLT%
615    \fi%
616    \numlabfont\linenumrep{\line@num}%
617    \ifsublines@
618      \ifnum\subline@num>0\relax
619        \unskip\fullstop\sublinenumrep{\subline@num}%
620      \fi
621    \fi%
622    \egroup%
623  }%
624
625  \newcommand*{\leftlinenum}{%
626    \ledlinenum
627    \kern\linenumsep}
628  \newcommand*{\rightlinenum}{%
629    \kern\linenumsep
630    \ledlinenum}
631
632  %
```

V.7   Line number counters and lists

Footnote references using line numbers rather than symbols can't be generated in one pass, because we do not 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 is used in marginal line numbering and in notes: counting either by section, page or pstart, depending on your choice for this section. This may be qualified by `\subline@num`.

```
633  \newcount\line@num
```

634 `%`

\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.

635 `\newcount\subline@num`
636 `%`

\ifsublines@   We maintain an associated flag, `\ifsublines@`, to tell us whether we're within a sub-
\sublines@true   line range or not.
\sublines@false   You may wonder why we do not 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.

637 `\newif\ifsublines@`
638 `%`

\absline@num   The count `\absline@num` stores the absolute number of lines since the start of the section: that is, the number we have 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 does not depend on the lineation system in use.

639 `\newcount\absline@num`
640 `%`

We will call `\absline@num` numbers "absolute" numbers, and `\line@num` and `\subline@num` numbers "visible" numbers.

V.8   Line number locking counter

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

641 `\newcount\@lock`
642 `\newcount\sub@lock`
643 `%`

V.9   Line number associated to lemma

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

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

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

    These line numbers are all visible numbers. The font specifier is a set of four codes for font encoding, family, series, and shape, separated by / characters. Thus a lemma that started on page 23, line 35 and went on until page 24, line 3 (with no sub-line numbering), and was typeset in a normal roman font would have a line list entry like this:
    `23|35|0|24|3|0|OT1/cmr/m/n`.

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

- `\insertlines@list`: the line numbers of lines that have footnotes or other insertions. These are the absolute numbers where the corresponding lemmas begin. This list contains one entry for every footnote in the section; one lemma may contribute no footnotes or many footnotes. This list is used by `\add@inserts` within `\do@line`, to tell it where to insert notes.

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

- `\actions@list`: action codes corresponding to the line numbers in `\actionlines@list`. These codes tell `reledmac` what action it is supposed to take at each of these lines. One action, the page-start action, is generated behind the scenes by `reledmac` 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 `reledmac` computes the visible line numbers from the absolute line numbers with reference to the other action codes and the settings they invoke; it does not require an entry in the action-code list for every line.

Here are the commands to create these lists:

```
644  \list@create{\line@list}
645  \list@create{\insertlines@list}
646  \list@create{\actionlines@list}
```

```
647  \list@create{\actions@list}
648
649  %
```

\page@num   We will need some counts while we read the line-list, for the page number and the ending
\endpage@num   page, line, and sub-line numbers. Some of these will be used again later on, when we
\endline@num   are acting on the data in our list macros.
\endsubline@num
```
650  \newcount\page@num
651  \newcount\endpage@num
652  \newcount\endline@num
653  \newcount\endsubline@num
654  %
```

\ifnoteschanged@   If the number of the footnotes in a section is different from what it was during the last
\noteschanged@true   run, or if this is the very first time you've run LaTeX, on this file, the information from
\noteschanged@false   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 do
not 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 is 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.

```
655  \newif\ifnoteschanged@
656  %
```

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

```
\resetprevline@657  \newcommand*{\resetprevline@}{%
658       \def\do##1{\global\csundef{prevline##1}}%
659       \dolistloop{\@series}%
660  }
661  %
```

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

```
\resetprevpage@662  \newcommand*{\resetprevpage@num}{%
663       \def\do##1{\ifcsdef{prevpage##1@num}{\global\csname prevpage##1@num\
endcsname=0}{}}%
```

```
664        \dolistloop{\@series}%
665 }
666 %
```

V.10   Reading the line-list file

`\read@linelist` `\read@linelist{`⟨*file*⟩`}` is the control sequence that is called by `\beginnumbering`
(via `\line@list@stuff`) to open and process a line-list file; its argument is the name
of the file. . First, it clear all previous line's list.

```
667 \newread\@inputcheck
668 \newcommand*{\read@linelist}[1]{%
669   \ifledRcol%
670     \list@clearing@regR%
671   \else%
672     \list@clearing@reg%
673   \fi%
674 %
```

When using `reledpar`, make sure that the `\maxlinesinpar@list` is empty (oth-
erwise things will be thrown out of kilter if there is any old stuff still hanging in there).

```
675   \list@clear{\maxlinesinpar@list}
676 %
```

Now get the file and interpret it. When the file is there we start a new group and make
some special definitions we will need to process it. It is a sequence of TEX commands,
but they require a few special settings. We make [and] become grouping characters:
they are 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 is 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 are in horizontal mode they can get interpreted as
spaces to be printed.

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

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

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

```
677   \get@linelistfile{#1}%
678   \endgroup
679 %
```

When the reading is done, we are all through with the line-list file. All the informa-
tion 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.

```
680    \ifledRcol
681      \global\page@numR=\m@ne
682      \ifx\actionlines@listR\empty
683        \gdef\next@actionlineR{1000000}%
684      \else
685        \gl@p\actionlines@listR\to\next@actionlineR
686        \gl@p\actions@listR\to\next@actionR
687      \fi
688    \else
689      \global\page@num=\m@ne
690      \ifx\actionlines@list\empty
691        \gdef\next@actionline{1000000}%
692      \else
693        \gl@p\actionlines@list\to\next@actionline
694        \gl@p\actions@list\to\next@action
695      \fi
696    \fi
697 }
698 %
```

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

```
699 \newcommand*{\list@clearing@reg}{%
700    \list@clear{\line@list}%
701    \list@clear{\insertlines@list}%
702    \list@clear{\actionlines@list}%
703    \list@clear{\actions@list}%
704    \list@clear{\linesinpar@listL}%
705    \list@clear{\linesonpage@listL}%
706    }%
707 %
```

\get@linelistfile   reledmac can take advantage of the LATEX 'safe file input' macros to get the line-list file.

```
708 \newcommand*{\get@linelistfile}[1]{%
709    \InputIfFileExists{#1}{%
710      \global\noteschanged@false
711      \begingroup
712        \catcode`\[=1 \catcode`\]=2
713        \makeatletter \catcode`\^^M=9}{%
714      \led@warn@NoLineFile{#1}%
715      \global\noteschanged@true
716      \begingroup}%
717 }
718
719 %
```

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 would have to do some file renaming outside of LaTeX for that to work. We have 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 4.2.7 p. 17 above).

V.11   Commands within the line-list file

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

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

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

\line@list@version   The \line@list@version check if the line-list file does not refers to the older commands of reledmac. In this case, we stop reading the line-list file. Consequently, \line@list@version must be the first line of a line-number file.

```
720  \newcommand{\line@list@version}[1]{%
721    \IfStrEq{#1}{\this@line@list@version}%
722      {}%
723      {\ifledRcol%
724         \led@warn@Obsolete{\jobname.\extensionchars\the\section@num}%
725       \else%
726         \led@warn@Obsolete{\jobname.\extensionchars\the\section@num}%
727       \fi%
728       \endinput%
729      }%
730  }%
731  %
```

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

In November 2004 in order to accurately determine page numbers Peter Wilson added these to the macro. It is now:

\@nl{⟨*page counter number*⟩}{⟨*printed page number*⟩}

We do not (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.

Exactly what \@nl does depends on whether right text is being processed. That's why many code is defined in \@nl@reg or \nl@regR.

```
732
733  \newcommand*{\@nl}[2]{%
734    \fix@page{#1}%
735    \ifledRcol%
736      \@nl@regR%
737    \else%
738      \@nl@reg%
739    \fi%
740  }
741  \newcommand*{\@nl@reg}{%
742    \ifx\l@dchset@num\relax \else
743      \advance\absline@num \@ne
744      \set@line@action
745      \let\l@dchset@num=\relax
746      \advance\absline@num \m@ne
747      \advance\line@num \m@ne
748    \fi
749  %
```

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

```
750    \advance\absline@num \@ne
751        \ifx\next@page@num\relax \else
752            \page@action
753            \let\next@page@num=\relax
754        \fi
755        \ifx\sub@change\relax \else
756            \ifnum\sub@change>\z@
757                \sublines@true
758            \else
759                \sublines@false
760            \fi
761            \sub@action
762            \let\sub@change=\relax
763        \fi
764  %
```

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

```
765            \ifcase\@lock
766                \or
```

```
767              \@lock \tw@
768          \or \or
769              \@lock \z@
770        \fi
771        \ifcase\sub@lock
772          \or
773              \sub@lock \tw@
774          \or \or
775              \sub@lock \z@
776        \fi
777 %
```

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

```
778        \ifsublines@
779              \ifnum\sub@lock<\tw@
780                \advance\subline@num \@ne
781              \fi
782        \else
783              \ifnum\@lock<\tw@
784                \advance\line@num \@ne \subline@num \z@
785              \fi
786        \fi}
787
788 %
```

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

```
789 \newcount\last@page@num
790   \last@page@num=-10000
791
792 \newcommand*{\fix@page}[1]{%
793   \ifledRcol
794     \ifnum #1=\last@page@numR
795     \else
796       \ifbypage@R
797         \line@numR \z@ \subline@numR \z@
798       \fi
799       \page@numR=#1\relax
800       \last@page@numR=#1\relax
801       \def\next@page@numR{#1}%
802     \fi
803   \else
804     \ifnum #1=\last@page@num
805     \else
806       \ifbypage@
807         \line@num \z@ \subline@num \z@
808       \fi
809       \page@num=#1\relax
810       \last@page@num=#1\relax
```

```
811        \def\next@page@num{#1}%
812        \listxadd{\normal@page@break}{\the\absline@num}
813      \fi
814    \fi}
815 %
```

\@pend These do not do anything at this point, but will have been added to the auxiliary file(s)
\@pendR if the reledpar package has been used. They are just here to stop reledmac from
\@lopL moaning if the reledpar is used for one run and then not for the following one.
\@lopR
```
816 \newcommand*{\@pend}[1]{}
817 \newcommand*{\@pendR}[1]{}
818 \newcommand*{\@lopL}[1]{}
819 \newcommand*{\@lopR}[1]{}
820
821 %
```

\sub@on The \sub@on and \sub@off macros turn sub-lineation on and off: but not directly, since
\sub@off such changes do not really take effect until the next line of text. Instead they set a flag
that notifies \@nl of the necessary action.
```
822 \newcommand*{\sub@on}{\ifsublines@
823        \let\sub@change=\relax
824    \else
825        \def\sub@change{1}%
826    \fi}
827 \newcommand*{\sub@off}{\ifsublines@
828        \def\sub@change{-1}%
829    \else
830        \let\sub@change=\relax
831    \fi}
832
833 %
```

\@adv The \@adv{⟨*num*⟩} macro advances the current visible line number by the amount spec-
ified as its argument. This is used to implement \advanceline.
```
834
835 \newcommand*{\@adv}[1]{%
836    \ifsublines@
837      \ifledRcol
838        \advance\subline@numR by #1\relax
839        \ifnum\subline@numR<\z@
840          \led@warn@BadAdvancelineSubline
841          \subline@numR \z@
842        \fi
843      \else
844        \advance\subline@num by #1\relax
845        \ifnum\subline@num<\z@
846          \led@warn@BadAdvancelineSubline
```

```
847        \subline@num \z@
848      \fi
849    \fi
850  \else
851    \ifledRcol
852      \advance\line@numR by #1\relax
853      \ifnum\line@numR<\z@
854        \led@warn@BadAdvancelineLine
855        \line@numR \z@
856      \fi
857    \else
858      \advance\line@num by #1\relax
859      \ifnum\line@num<\z@
860        \led@warn@BadAdvancelineLine
861        \line@num \z@
862      \fi
863    \fi
864  \fi
865  \set@line@action}
866
867 %
```

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

```
868
869 \newcommand*{\@set}[1]{%
870   \ifledRcol
871     \ifsublines@
872       \subline@numR=#1\relax
873     \else
874       \line@numR=#1\relax
875     \fi
876     \set@line@action
877   \else
878     \ifsublines@
879       \subline@num=#1\relax
880     \else
881       \line@num=#1\relax
882     \fi
883     \set@line@action
884   \fi}
885
886 %
```

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

```
887
888 \newcommand*{\l@d@set}[1]{%
889   \ifledRcol
890     \line@numR=#1\relax
891     \advance\line@numR \@ne
892     \def\l@dchset@num{#1}
893   \else
894     \line@num=#1\relax
895     \advance\line@num \@ne
896     \def\l@dchset@num{#1}
897   \fi}
898 \let\l@dchset@num\relax
899
900 %
```

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

```
901
902 \newcommand*{\page@action}{%
903   \ifledRcol
904     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
905     \xright@appenditem{\next@page@numR}\to\actions@listR
906   \else
907     \xright@appenditem{\the\absline@num}\to\actionlines@list
908     \xright@appenditem{\next@page@num}\to\actions@list
909   \fi}
910 %
```

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

```
911
912 \newcommand*{\set@line@action}{%
913   \ifledRcol
914     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
915     \ifsublines@
916       \@l@dtempcnta=-\subline@numR
917     \else
918       \@l@dtempcnta=-\line@numR
919     \fi
920     \advance\@l@dtempcnta by -5000\relax
921     \xright@appenditem{\the\@l@dtempcnta}\to\actions@listR
922   \else
923     \xright@appenditem{\the\absline@num}\to\actionlines@list
924     \ifsublines@
925       \@l@dtempcnta=-\subline@num
926     \else
927       \@l@dtempcnta=-\line@num
928     \fi
929     \advance\@l@dtempcnta by -5000\relax
```

```
930      \xright@appenditem{\the\@l@dtempcnta}\to\actions@list
931    \fi}
932 %
```

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

```
933
934 \newcommand*{\sub@action}{%
935   \ifledRcol
936     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
937     \ifsublines@
938       \xright@appenditem{-1001}\to\actions@listR
939     \else
940       \xright@appenditem{-1002}\to\actions@listR
941     \fi
942   \else
943     \xright@appenditem{\the\absline@num}\to\actionlines@list
944     \ifsublines@
945       \xright@appenditem{-1001}\to\actions@list
946     \else
947       \xright@appenditem{-1002}\to\actions@list
948     \fi
949   \fi}
950 %
```

\lock@on   \lock@on adds an entry to the action-code list to turn line number locking on. The
\do@lockon   current setting of the sub-lineation flag tells us whether this applies to line numbers or
\do@lockonL   sub-line numbers.
              Adding commands to the action list is slow, and it is 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.

```
951 \newcommand*{\lock@on}{\futurelet\next\do@lockon}
952
953 \newcommand*{\do@lockon}{%
954   \ifx\next\lock@off
955     \global\let\lock@off=\skip@lockoff
956   \else
957     \ifledRcol
958       \do@lockonR
959     \else
960       \do@lockonL
961     \fi
962   \fi}
963
964
965 \newcommand*{\do@lockonL}{%
```

```
966   \xright@appenditem{\the\absline@num}\to\actionlines@list
967   \ifsublines@
968     \xright@appenditem{-1005}\to\actions@list
969     \ifnum\sub@lock=\z@
970       \sub@lock \@ne
971     \else
972       \ifnum\sub@lock=\thr@@
973         \sub@lock \@ne
974       \fi
975     \fi
976   \else
977     \xright@appenditem{-1003}\to\actions@list
978     \ifnum\@lock=\z@
979       \@lock \@ne
980     \else
981       \ifnum\@lock=\thr@@
982         \@lock \@ne
983       \fi
984     \fi
985   \fi}
986
987 %
```

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

```
988  \newcommand*{\do@lockoffL}{%
989    \xright@appenditem{\the\absline@num}\to\actionlines@list
990    \ifsublines@
991      \xright@appenditem{-1006}\to\actions@list
992      \ifnum\sub@lock=\tw@
993        \sub@lock \thr@@
994      \else
995        \sub@lock \z@
996      \fi
997    \else
998      \xright@appenditem{-1004}\to\actions@list
999      \ifnum\@lock=\tw@
1000       \@lock \thr@@
1001     \else
1002       \@lock \z@
1003     \fi
1004   \fi}
1005
1006 \newcommand*{\do@lockoff}{%
1007   \ifledRcol
1008     \do@lockoffR
1009   \else
1010     \do@lockoffL
1011   \fi}
1012 \newcommand*{\skip@lockoff}{\global\let\lock@off=\do@lockoff}
```

```
1013 \global\let\lock@off=\do@lockoff
1014
1015 %
```

\n@num These macros implement the \skipnumbering command. They use action code 1007.

```
1016 \newcommand*{\n@num}{%
1017   \ifledRcol%
1018     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
1019     \xright@appenditem{-1007}\to\actions@listR
1020   \else%
1021     \xright@appenditem{\the\absline@num}\to\actionlines@list%
1022     \xright@appenditem{-1007}\to\actions@list%
1023   \fi%
1024 }%
1025
1026 %
```

\n@num@stanza This macro implements the \skipnumbering for stanza command. It uses action code 1008.

```
1027 \newcommand*{\n@num@stanza}{%
1028   \ifledRcol%
1029     \xright@appenditem{\the\absline@numR}\to\actionlines@listR%
1030     \xright@appenditem{-1008}\to\actions@listR%
1031   \else%
1032     \xright@appenditem{\the\absline@num}\to\actionlines@list%%
1033     \xright@appenditem{-1008}\to\actions@list%
1034   \fi%
1035 }
1036 %
```

\ifl@dhidenumber \hidenumbering hides number in margin. It uses action code 1009.
\hidenumbering
\h@num
```
1037 \newif\ifl@dhidenumber
1038 \newcommand*{\hidenumbering}{
1039   \ifledRcol%
1040     \write\linenum@outR{\string\hide@num}%
1041   \else%
1042     \write\linenum@out{\string\hide@num}%
1043   \fi%
1044 }%
1045 \newcommand*{\hide@num}{%
1046   \ifledRcol%
1047     \xright@appenditem{\the\absline@numR}\to\actionlines@listR%
1048     \xright@appenditem{-1009}\to\actions@listR%
1049   \else%
1050     \xright@appenditem{\the\absline@num}\to\actionlines@list%%
```

```
1051        \xright@appenditem{-1009}\to\actions@list%
1052    \fi%
1053 }
1054 %
```

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

```
1055 \newcount\insert@count
1056 %
```

- #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 is necessary to temporarily redefine
                 \@ref within \@ref, so that we are only doing one of these at a time.

```
1057 \newcommand*{\dummy@ref}[2]{#2}
1058 %
```

\@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.

```
1059 \newcommand*{\@ref}[2]{%
1060    \ifledRcol%
1061       \@ref@regR{#1}{#2}%
1062    \else%
1063       \@ref@reg{#1}{#2}%
1064    \fi%
1065 }%
1066 \newcommand*{\@ref@reg}[2]{%
1067    \global\insert@count=#1\relax
1068    \global\advance\@edtext@level by 1%
1069    \loop\ifnum\insert@count>\z@
1070       \xright@appenditem{\the\absline@num}\to\insertlines@list
1071       \global\advance\insert@count \m@ne
1072    \repeat
1073 %
```

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.

```
1074    \begingroup
1075      \let\@ref=\dummy@ref
1076      \let\@lopL\@gobble
1077      \let\page@action=\relax
1078      \let\sub@action=\relax
1079      \let\set@line@action=\relax
1080      \let\@lab=\relax
1081      \let\@lemma=\relax%
1082      \let\@sw\@gobblethree%
1083      #2
1084      \global\endpage@num=\page@num
1085      \global\endline@num=\line@num
1086      \global\endsubline@num=\subline@num
1087    \endgroup
1088 %
```

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

```
1089      \xright@appenditem%
1090        {\the\page@num|\the\line@num|%
1091         \ifsublines@ \the\subline@num \else 0\fi|%
1092         \the\endpage@num|\the\endline@num|%
1093         \ifsublines@ \the\endsubline@num \else 0\fi}\to\line@list
1094 %
```

Create a list which stores every second argument of each `\@sw` in this lemma, at this
level. Also set the boolean about the use of lemma in this edtext level to false.

```
1095        \expandafter\list@create\expandafter{\csname sw@list@edtext@tmp@\the\
     @edtext@level\endcsname}%
1096        \providebool{lemmacommand@\the\@edtext@level}%
1097        \boolfalse{lemmacommand@\the\@edtext@level}%
1098 %
```

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

```
1099      #2%
1100 %
```

Now, we store the list of `\@sw` of this current `\edtext` as an element of the global list
of list of `\@sw` for a `\edtext` depth.

```
1101    \ifnum\@edtext@level>0%
1102      \def\create@this@edtext@level{\expandafter\list@create\expandafter{\
     csname sw@list@edtext@\the\@edtext@level\endcsname}}%
1103      \ifcsundef{sw@list@edtext@\the\@edtext@level}{\create@this@edtext@level
     }{}%
1104      \letcs{\@tmp}{sw@list@edtext@\the\@edtext@level}%
1105      \letcs{\@tmpp}{sw@list@edtext@tmp@\the\@edtext@level}
1106      \xright@appenditem{\expandonce\@tmpp}\to\@tmp%
1107      \global\cslet{sw@list@edtext@\the\@edtext@level}{\@tmp}%
```

```
1108    \fi%
1109 %
```

Decrease edtext level counter.

```
1110    \global\advance\@edtext@level by -1%
1111 %

1112 }
1113
1114 %
```

V.12   Writing to the line-list file

We have now defined all the counters, lists, and commands involved in reading the line-list file at the start of a section. Now we will cover the commands that reledmac 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.

```
1115 \newwrite\linenum@out
1116 %
```

\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 would have to write it at the start of every line. But it is not very easy for the output routine to tell whether an output stream is open or not. There is 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 in-elegant name suggests the nature of the problem that made its creation necessary. It is set to be true before any \linenum@out file is opened. When such a file is opened for the first time, it is 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.

```
1117 \newif\iffirst@linenum@out@
1118    \first@linenum@out@true
1119 %
```

\this@line@list@version   The commands allowed in the line-list file and their arguments can change between two version of reledmac. The \this@line@list@version command is upgraded when it happens. It is written in the file list. If we process a line-list file which used a older version, that means the commands used inside are deprecated, and we can't use them.

```
1120 \newcommand{\this@line@list@version}{2}%
1121 %
```

\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.

```
1122 \newcommand*{\line@list@stuff}[1]{%
1123 %
```

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

```
1124    \read@linelist{#1}%
1125 %
```

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.

```
1126    \iffirst@linenum@out@
1127        \immediate\closeout\linenum@out%
1128        \global\first@linenum@out@false%
1129        \immediate\openout\linenum@out=#1\relax%
1130        \immediate\write\linenum@out{\string\line@list@version{\
    this@line@list@version}}%
1131    \else
1132 %
```

If we get here, then this is not the first line-list we have seen, so we do not open or close the files immediately.

```
1133        \if@minipage%
1134          \leavevmode%
1135        \fi%
1136        \closeout\linenum@out%
1137        \openout\linenum@out=#1\relax%
1138    \fi}
1139
1140 %
```

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

```
1141 \newcommand*{\new@line}{%
1142    \IfStrEq{\led@pb@setting}{after}%
1143      {\xifinlist{\the\absline@num}{\l@prev@nopb}%
1144        {\xifinlist{\the\absline@num}{\normal@page@break}%
1145          {\numgdef{\@next@page}{\c@page+1}%
1146          \write\linenum@out{\string\@nl[\@next@page][\@next@page]}%
1147        }%
1148        {\write\linenum@out{\string\@nl[\the\c@page][\thepage]}}%
1149      }%
1150      {\write\linenum@out{\string\@nl[\the\c@page][\thepage]}}}%
1151    {}%
```

```
1152    \IfStrEq{\led@pb@setting}{before}%
1153      {\numdef{\next@absline}{\the\absline@num+1}%
1154      \xifinlist{\next@absline}{\l@prev@nopb}%
1155        {\xifinlist{\the\absline@num}{\normal@page@break}%
1156          {\numgdef{\nc@page}{\c@page+1}%
1157          \write\linenum@out{\string\@nl[\nc@page][\nc@page]}%
1158          }%
1159          {\write\linenum@out{\string\n@l[\the\c@page][\thepage]}}%
1160        }%
1161        {\write\linenum@out{\string\@nl[\the\c@page][\thepage]}}%
1162      }%
1163      {}%
1164    \IfStrEqCase{\led@pb@setting}{{before}{\relax}{after}{\relax}}[\write\
       linenum@out{\string\@nl[\the\c@page][\thepage]}]%
1165  }
1166
1167  %
```

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

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

```
1168  \newif\if@noneed@Footnote%
1169
1170  \newcommand*{\flag@start}{%
1171    \ifledRcol%
1172      \edef\next{\write\linenum@outR{%
1173                    \string\@ref[\the\insert@countR][]}}%
1174      \next%
1175      \ifnum\insert@countR<1%
1176        \if@noneed@Footnote\else%
1177          \led@err@EdtextWithoutFootnote%
1178        \fi%
1179      \fi%
1180    \else%
1181      \edef\next{\write\linenum@out{%
1182                    \string\@ref[\the\insert@count][]}}%
1183      \next%
1184      \ifnum\insert@count<1%
1185        \if@noneed@Footnote\else%
1186          \led@err@EdtextWithoutFootnote%
1187        \fi%
1188      \fi%
1189    \fi}%
1190
1191  %
```

\startsub   `\startsub` and `\endsub` turn sub-lineation on and off, by writing appropriate instruc-
\endsub   tions 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 does not 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 at-
tached 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.

```
1192
1193
1194 \newcommand*{\startsub}{\dimen0\lastskip
1195   \ifdim\dimen0>0pt \unskip \fi
1196   \ifledRcol \write\linenum@outR{\string\sub@on}%
1197   \else      \write\linenum@out{\string\sub@on}%
1198   \fi
1199   \ifdim\dimen0>0pt \hskip\dimen0 \fi}
1200 \def\endsub{\dimen0\lastskip
1201   \ifdim\dimen0>0pt \unskip \fi
1202   \ifledRcol \write\linenum@outR{\string\sub@off}%
1203   \else      \write\linenum@out{\string\sub@off}%
1204   \fi
1205   \ifdim\dimen0>0pt \hskip\dimen0 \fi}
1206
1207 %
```

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

```
1208 \newcommand*{\advanceline}[1]{%
1209   \ifledRcol \write\linenum@outR{\string\@adv[#1]}%
1210   \else      \write\linenum@out{\string\@adv[#1]}%
1211   \fi%
1212 }
1213 %
```

\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.

```
1214
1215 \newcommand*{\setline}[1]{%
1216   \ifnum#1<\z@
1217     \led@warn@BadSetline
1218   \else
1219     \ifledRcol \write\linenum@outR{\string\@set[#1]}%
```

```
1220        \else       \write\linenum@out{\string\@set[#1]}%
1221      \fi
1222    \fi}
1223
1224 %
```

\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.

```
1225
1226 \newcommand*{\setlinenum}[1]{%
1227    \ifnum#1<\z@
1228       \led@warn@BadSetlinenum
1229    \else
1230       \ifledRcol \write\linenum@outR{\string\l@d@set[#1]}
1231       \else       \write\linenum@out{\string\l@d@set[#1]} \fi
1232    \fi}
1233
1234 %
```

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

```
1235
1236 \newcommand*{\startlock}{%
1237    \ifledRcol \write\linenum@outR{\string\lock@on}%
1238    \else       \write\linenum@out{\string\lock@on}%
1239    \fi}
1240 \def\endlock{%
1241    \ifledRcol \write\linenum@outR{\string\lock@off}%
1242    \else       \write\linenum@out{\string\lock@off}%
1243    \fi}
1244 %
```

\ifl@dskipnumber   In numbered text \skipnumbering will suspend the numbering for that particular line.
\ifl@dskipversenumber
\l@dskipnumbertrue
\l@dskipnumberfalse
\skipnumbering
```
1245 \newif\ifl@dskipnumber
1246 \newif\ifl@dskipversenumber%
1247 \newcommand*{\skipnumbering}{%
1248    \leavevmode%
1249    \ifledRcol%
1250       \ifinstanza%
1251          \write\linenum@outR{\string\n@num@stanza}%
1252       \else%
1253          \write\linenum@outR{\string\n@num}%
1254       \fi%
1255       \advanceline{-1}%
1256    \else%
```

```
1257      \ifinstanza%
1258         \write\linenum@out{\string\n@num@stanza}%
1259      \else%
1260         \write\linenum@out{\string\n@num}%
1261      \fi%
1262      \advanceline{-1}%
1263    \fi%
1264 }%
1265
1266 %
```

VI   Marking text for notes

The `\edtext` 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.

The `\edtext` macro takes two arguments.

`\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.

The `\edtext` macro may be used (somewhat) recursively; that is, `\edtext` 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 is quite likely that we will 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 are not 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 `\edtext` will fail if you try to use a copy that is called something other than `\edtext`. In order to handle recursion, `\edtext` needs to redefine its own definition temporarily at one point, and that does not work if the macro you are calling is not actually named `\edtext`. There is no problem as long as `\edtext` is not invoked in the first argument. If you want to call `\edtext` something else, it is best to create instead a macro that expands to an invocation of `\edtext`, rather than copying `\edtext` 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 foot-note macros directly within a paragraph whose lines are numbered (see comments to

`\do@line`, VII.2.1 p. 121). 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 do not provide previous-note information, although it is often wanted; your own macros must handle that. We ca not do it correctly without keeping track of what kind of notes have gone past: it is not just a matter of remembering the line numbers associated with the previous invocation of `\edtext`, 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.

VI.1   `\edtext` itself

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

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

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

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

```
1267  \list@create{\end@lemmas}
1268  %
```

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

```
1269  \newcommand{\dummy@edtext}[2]{#1}
1270  %
```

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

```
1271 \newcommand{\dummy@edtext@showlemma}[2]{\showlemma{#1}}%
1272 %
```

We are 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 are likely to see
\morenoexpands 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.[24] 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 is 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 is not used enough to bother with. That macro, and any other that causes trouble, will get by all right if you put a \protect in front of it in your file.)

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

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

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

```
1273 \newcommand*{\no@expands}{%
```

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

```
1274    \let\select@@lemmafont=0%
1275    \let\startsub=\relax  \let\endsub=\relax
1276    \let\startlock=\relax \let\endlock=\relax
1277    \let\edlabel=\@gobble
1278    \let\setline=\@gobble \let\advanceline=\@gobble
1279    \let\sameword\sameword@inedtext%
1280    \let\edtext=\dummy@edtext
1281    \l@dtabnoexpands
1282    \morenoexpands}
1283 \let\morenoexpands=\relax
1284
1285 %
```

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

```
1286 \newcommand{\@tag}{}
1287 %
```

\@edtext@level   This counter is increased by 1 at each level of \edtext. That is useful for some com-
                  mands which can have a different behavior if called inside or outside of the {⟨*lemma*⟩}
                  argument.

```
1288 \newcount\@edtext@level%
1289 \@edtext@level=0%
1290 %
```

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

```
1291 \newcommand{\edtext}[2]{\leavevmode%
1292 %
```

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

```
1293    \ifnumberedpar@%
1294 %
```

         We increase the \@edtext@ counter to know in which level of \edtext we are.

```
1295        \global\advance\@edtext@level by 1%
1296 %
```

         By default, we do not use \lemma

```
1297        \global\@lemmacommand@false%
1298 %

1299        \begingroup%
1300 %
```

         We get the next series of samewords data in the list of samewords data for the current
         edtext level. We push them inside \sw@inthisedtext.

```
1301        \ifledRcol%
1302          \ifcsundef{sw@list@edtextR@\the\@edtext@level}%
1303                {\global\let\sw@inthisedtext\empty}%
1304                {\ifcsempty{sw@list@edtextR@\the\@edtext@level}%
1305      {\global\let\sw@inthisedtext\empty}%
1306      {\expandafter\gl@p\csname sw@list@edtextR@\the\@edtext@level\endcsname\
     to\sw@inthisedtext}%
1307          }%
1308        \else%
1309          \ifcsundef{sw@list@edtext@\the\@edtext@level}%
1310                {\global\let\sw@inthisedtext\empty}%
1311                {\ifcsempty{sw@list@edtext@\the\@edtext@level}%
1312      {\global\let\sw@inthisedtext\empty}%
1313      {\expandafter\gl@p\csname sw@list@edtext@\the\@edtext@level\endcsname\
     to\sw@inthisedtext}%
1314          }%
1315        \fi%
1316 %
```

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

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

```
1317            \global\renewcommand{\@tag}{%
1318              \no@expands #1%
1319              }%
1320 %
```

\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.

```
1321            \set@line%
1322 %
```

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

```
1323            \ifledRcol \global\insert@countR \z@%
1324            \else      \global\insert@count \z@ \fi%
1325 %
```

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.

```
1326            \ignorespaces #2\relax%
1327 %
```

With `polyglossia`, you must track whether the language reads left to right (English) or right to left (Arabic).

```
1328            \@ifundefined{xpg@main@language}{%if not polyglossia
1329               \flag@start}%
1330               {\if@RTL\flag@end\else\flag@start\fi%
1331               }%
1332 %
```

We write in the numbered file wether the current \edtext has a \lemma in the the second argument.

```
1333            \if@lemmacommand@%
1334              \ifledRcol%
1335                \write\linenum@outR{\string\@lemma}%
1336              \else%
1337                \write\linenum@out{\string\@lemma}%
1338              \fi%
1339            \fi%
1340 %
```

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

It is 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.

```
1341          \endgroup%
1342          \showlemma{#1}%
1343 %
```

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.

```
1344          \ifx\end@lemmas\empty \else%
1345            \gl@p\end@lemmas\to\x@lemma%
1346            \x@lemma%
1347            \global\let\x@lemma=\relax%
1348          \fi%
1349          \@ifundefined{xpg@main@language}{%if not polyglossia
1350              \flag@end}%
1351              {\if@RTL\flag@start\else\flag@end\fi% With polyglossia, you must
    track whether the language reads left to right (English) or right to left
    (Arabic).
1352              }%
1353 %
```

We switch to false some flags.

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

- The one that says we are inside a \edtext. In fact, it is not a flag, but a counter which is increased to 1 in each leavel of \edtext.

- The one that says we are inside à \@lemma.

```
1354        \global\@noneed@Footnotefalse%
1355        \global\advance\@edtext@level by -1%
1356        \global\@lemmacommand@false%
1357 %
```

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

```
1358    \else%
1359    \showlemma{#1} (\textbf{\textsc{Edtext outside numbered paragraph}})\
       led@err@edtextoutsidepstart%
1360    \fi%
1361 }%
1362
1363 \newcommand*{\flag@end}{%
1364    \ifledRcol%
1365        \write\linenum@outR{]}%
1366    \else%
1367        \write\linenum@out{]}%
1368    \fi}%
1369
1370 %
```

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

```
1371 \newif\ifnumberline
1372 \numberlinetrue
1373 %
```

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

One instance of \edtext may generate several notes, or it may generate none — it is legitimate for argument #2 to \edtext 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 is vital to also remove one and only one \line@list entry here.

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

```
1374 \newcommand*{\set@line}{%
1375    \ifledRcol
```

```
1376        \ifx\line@listR\empty
1377          \global\noteschanged@true
1378          \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
1379        \else
1380          \gl@p\line@listR\to\@tempb
1381          \xdef\l@d@nums{\@tempb|\edfont@info}%
1382          \global\let\@tempb=\undefined
1383        \fi
1384      \else
1385        \ifx\line@list\empty
1386          \global\noteschanged@true
1387          \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
1388        \else
1389          \gl@p\line@list\to\@tempb
1390          \xdef\l@d@nums{\@tempb|\edfont@info}%
1391          \global\let\@tempb=\undefined
1392        \fi
1393      \fi}
1394
1395  %
```

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

```
1396  \newcommand*{\edfont@info}{\f@encoding/\f@family/\f@series/\f@shape}
1397
1398  %
```

VI.2   Substitute lemma

\lemma   The \lemma{⟨*text*⟩} macro allows you to change the lemma that is passed on to the notes.   Read about \@tag in normal \edtext macro for more details about \sw@list@inedtext and \no@expands (VI.1 p. 105).

```
1399  \unless\ifnocritical@
1400  \newcommand*{\lemma}[1]{%
1401    \global\@lemmacommand@true%
1402    \global\renewcommand{\@tag}{%
1403      \no@expands #1%
1404    }%
1405    \ignorespaces%
1406  }%
1407  %
```

\@lemma   The \@lemma is written in the numbered file to set which \edtext has an \lemma as second argument.

```
1408  \newcommand{\@lemma}{%
1409    \booltrue{lemmacommand@\the\@edtext@level}%
1410  }%
```

```
1411  \fi
1412  %
```

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

```
1413  \newif\if@lemmacommand@%
1414  %
```

VI.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 V.9 p. 80): 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 do not 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.

```
1415  \newcommand*{\linenum}[1]{%
1416    \xdef\@tempa{#1|||||||\noexpand\\l@d@nums}%
1417    \global\let\l@d@nums=\empty
1418    \expandafter\line@set\@tempa|\\\ignorespaces}
1419  %
```

\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.

```
1420  \def\line@set#1|#2\\#3|#4\\{%
1421    \gdef\@tempb{#1}%
1422    \ifx\@tempb\empty
1423        \l@d@add{#3}%
1424    \else
1425        \l@d@add{#1}%
1426    \fi
1427    \gdef\@tempb{#4}%
1428    \ifx\@tempb\empty\else
1429        \l@d@add{|}\line@set#2\\#4\\%
1430    \fi}
1431  %
```

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

```
1432  \newcommand{\l@d@add}[1]{\xdef\l@d@nums{\l@d@nums#1}}
1433
1434  %
```

VI.4   Lemma disambiguation

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

The general mechanism is the following:

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

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

- **When this auxiliary file is read at the second run**, different operations are achieved:

  1. Get the rank of each \sameword in a line (relative rank) from the rank of each \sameword in all the numbered section (absolute rank):

     - For each paired \sameword argument and absolute line number, a counter is defined. Its value corresponds to the number of times \sameword{⟨*argument*⟩}+ is called from the beginning of the lineation to the end of the current line. We also store the same data for the preceding absolute line number, if it does not have \sameword{⟨*argument*⟩}.

     - For each \sameword having the same argument, we subtract from its absolute rank the number stored for the paired \sameword argument and previous absolute line number. Consequently, we obtain the relative rank.

     - See the following example which explain how for same \sameword absolute ranks are transformed to relative rank.
       ```
       At line 1:
       absolute rank 1 becomes relative rank 1-0 = 1
       1 is stored for this \sameword and the line 1
       At line 2:
       absolute rank 2 becomes relative rank 2-1 = 1
       absolute rank 3 becomes relative rank 3-2 = 2
       3 is stored for this \sameword and the line 2
       At line 3:
       no \sameword for this line.
       3 is stored for this \sameword and the line 3
       At line 4:
       absolute rank 4 becomes relative rank 4-3 = 1
       3 is stored for this \sameword and the line 4
       ```

2. Create lists of lists of \sameword by depth of \edtext. That is: create a list for \edtext of level 1, a list for \edtext of level 2, a list for \edtext of level 3 etc. For each \edtext in these list, we store all the relative rank of \saweword which are called as lemma information, that is 1) or called in the first argument of \sameword 2) or called in the \lemma macro of the second argument of \sameword AND marked by the optional argument of \saweword in first argument of \edtext.

   For example, suppose a line with nested \edtexts which contains some word marked by \sameword and having the following relative rank:

   bar^1 | | $\text{foo}^1\ \text{foo}^2\ \text{bar}^2\ \text{foo}^3$ | (A)(B) $\text{foo}^4\ \text{bar}^3$ | (C) | | foo^5 | (D) bar^4 | (E)

   In this example, all lemma information for \edtext is framed. The text in parenthesis is the content of critical notes associated to the preceding frame. As you can see, we have two level of \edtext.

   The list for \edtexts of level 1 is $\{\{1, 2, 2, 3, 4, 3\}, \{5, 4\}\}$.

   The list for \edtexts of level 2 is $\{\{1, 2, 2, 3\}, \{5\}\}$.

   As you can see, the mandatory argument of \sameword does not matter: we store the rank informations for every word potentially ambiguous.

- At the second run, when a critical notes is called, we associate it to the next item of the list associated to is \edtext level. So, in the previous example:

  - Critical notes (A) and (B) are associated with $\{1, 2, 2, 3\}$.
  - Critical note (C) is associated with $\{1, 2, 2, 3, 4, 3\}$.
  - Critical note (D) is associated with $\{5\}$.
  - Critical note (E) is associated with $\{5, 4\}$.

- At the second run, when a critical note is printed:

  - The \sameword command is let \sameword@inedtext.
  - At each call of this \sameword@inedtext, we step to the next element of the list associated to the note. Let it be r.
  - For the word marked by \sameword, we calculate how many time it is called in its line. To do it:
    * We get the absolute line number of the current \sameword. This absolute line number was stored with list of relative rank for the current \edtext. That means, in the previous example, that, if the absolute line number of \edtext was 1, that critical notes (A) and (B) were not associated with $\{1, 2, 2, 3\}$ but with $\{(1, 1), (2, 1), (2, 1), (3, 1)\}$. Such method to know the absolute line number associated to a \sameword is required because a \edtext can be overlap many lines, but \sameword can't get it.
    * We get the value associated, when reading the auxiliary file, to the pair compose by the current marked word and the current absolute line number. Let this value be n.

– If $n > 1$, that mean the current word appears more than once time in its line. In this case, we call \showwordrank with the word as first argument and r as second argument. If the word is called only once, we just print it.

After theory, implementation.

\get@sw@txt As the argument of \sameword can contain active character if we use inputenc with utf8 option instead of native UTF-8 engine, we store its detokenized content in a macro in order to allow dynamic name of macro with \csname.[25]

Because there is a bug with \detokenize and XƎTEX when using non BMP characters[26], we detokenize only for not XƎTEXengines. In any case, in XƎTEX, a \csname construction can contain UTF-8 characters without a problem, as UTF-8 characters are not managed with category code, but instead read directly as UTF-8 characters.

```
1435 \newcommand{\get@sw@txt}[1]{%
1436   \ifxetex%
1437     \xdef\sw@txt{#1}%
1438   \else%
1439     \expandafter\xdef\expandafter\sw@txt\expandafter{\detokenize{#1}}%
1440   \fi%
1441 }%
1442 %
```

\sameword The hight level macro \sameword, used by the editor.

```
1443 \newcommandx{\sameword}[2][1,usedefault]{%
1444     \leavevmode%
1445     \get@sw@txt{#2}%
1446 %
```

Now, the real code. First, increment the counter corresponding to the argument.

```
1447   \unless\ifledRcol%
1448     \csnumgdef{sw@\sw@txt}{\csuse{sw@\sw@txt}+1}%
1449 %
```

Then, write its value to the numbered file.

```
1450     \protected@write\linenum@out{}{\string\@sw{\sw@txt}{\csuse{sw@\sw@txt
}}{#1}}%
1451 %
```

Do the same thing if we are in the right columns.

```
1452   \else%
1453     \csnumgdef{sw@\sw@txt@R}{\csuse{sw@\sw@txt@R}+1}%
1454     \protected@write\linenum@outR{}{\string\@sw{\sw@txt}{\csuse{sw@\
sw@txt@R}}{#1}}%
1455   \fi%
1456 %
```

---

[25]See http://tex.stackexchange.com/q/244538/7712.
[26]http://sourceforge.net/p/xetex/bugs/108/

And print the word.

```
1457    #2%
1458 }%
1459 %
```

A flag set to true if a `\@sw` relative rank must be added to the list of ranks for a specific `\edtext`.

`\if@addsw` `\newif\if@addsw%`
```
1460
1461 %
```

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

```
1462 \newcommand{\@sw}[3]{%
1463    \get@sw@txt{#1}%
1464    \unless\ifledRcol%
1465 %
```

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

```
1466        \csxdef{sw@\sw@txt @\the\absline@num @\the\section@num}{#2}%
1467 %
```

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

```
1468        \numdef{\prev@line}{\the\absline@num-1}%
1469        \ifcsundef{sw@\sw@txt @\prev@line @\the\section@num}{%
1470          \csnumgdef{sw@\sw@txt @\prev@line @\the\section@num}{#2-1}%
1471        }{}%
1472 %
```

Then, calculate the position of the word in the line.

```
1473        \numdef{\the@sw}{#2-\csuse{sw@\sw@txt @\prev@line @\the\section@num}}%
1474 %
```

And do the same thing for the right side.

```
1475    \else%
1476        \csxdef{sw@\sw@txt @\the\absline@numR @\the\section@numR @R}{#2}%
1477        \numdef{\prev@line}{\the\absline@numR-1}%
1478        \ifcsundef{sw@\sw@txt @\prev@line @\the\section@numR @R}{%
1479          \csnumgdef{sw@\sw@txt @\prev@line @\the\section@numR @R}{#2-1}%
1480        }{}%
1481        \numdef{\the@sw}{#2-\csuse{sw@\sw@txt @\prev@line @\the\section@numR @R
      }}%
1482    \fi%
1483 %
```

And now, add it to the list of `\@sw` for the current edtext, in all depth.

```
1484    \@tempcnta=\@edtext@level
1485    \@whilenum{\@tempcnta>0}\do{%
1486      \ifcsdef{sw@list@edtext@tmp@\the\@tempcnta}%
1487        {%
1488          \@addswfalse%
1489          \notbool{lemmacommand@\the\@tempcnta}%
1490            {\@addswtrue}%
1491            {\IfStrEq{#3}{inlemma}%
1492              {\@addswtrue}%
1493              {%
1494               \def\do##1{%
1495                  \ifnumequal{##1}{\the\@tempcnta}%
1496                    {\@addswtrue\listbreak}%
1497                    {}%
1498               }%
1499               \docsvlist{#3}%
1500              }%
1501            }%
1502          \if@addsw%
1503            \letcs{\@tmp}{sw@list@edtext@tmp@\the\@tempcnta}%
1504            \ifledRcol%
1505              \xright@appenditem{{\the@sw}{\the\absline@numR}}\to\@tmp%
1506            \else%
1507              \xright@appenditem{{\the@sw}{\the\absline@num}}\to\@tmp%
1508            \fi%
1509            \cslet{sw@list@edtext@tmp@\the\@tempcnta}{\@tmp}%
1510          \fi%
1511        }%
1512        {}%
1513      \advance\@tempcnta by -1%
1514    }%
1515 }%
1516 %
```

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

```
1517 \newcommandx{\sameword@inedtext}[2][1,usedefault]{%
1518    \get@sw@txt{#2}%
1519    \unless\ifledRcol@%
1520 %
```

Just a precaution.

```
1521        \ifx\sw@list@inedtext\empty%
1522          \def\the@sw{999}%
1523          \def\this@absline{-99}%
1524        \else%
1525 %
```

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

```
1526          \gl@p\sw@list@inedtext\to\@tmp%
1527          \edef\the@sw{\expandafter\@firstoftwo\@tmp}%
1528          \edef\this@absline{\expandafter\@secondoftwo\@tmp}%
1529        \fi%
1530 %
```

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

```
1531        \ifcsdef{sw@\sw@txt @\this@absline @\the\section@num}{%
1532          \numdef{\prev@line}{\this@absline-1}%
1533          \numdef{\sw@atthisline}{\csuse{sw@\sw@txt @\this@absline @\the\
      section@num}-\csuse{sw@\sw@txt @\prev@line @\the\section@num}}%
1534          }%
1535          {\numdef{\sw@atthisline}{0}}%
1536 %
```

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

```
1537        \ifnumgreater{\sw@atthisline}{1}%
1538            {\showwordrank{#2}{\the@sw}}%
1539            {#2}%
1540 %
```

And the same for right side.

```
1541    \else%
1542        \ifx\sw@list@inedtext\empty%
1543          \def\the@sw{999}%
1544          \def\this@absline{-99}%
1545        \else%
1546          \gl@p\sw@list@inedtext\to\@tmp%
1547          \edef\the@sw{\expandafter\@firstoftwo\@tmp}%
1548          \edef\this@absline{\expandafter\@secondoftwo\@tmp}%
1549        \fi%
1550        \ifcsdef{sw@\sw@txt @\this@absline @\the\section@numR @R}{%
1551          \numdef{\prev@line}{\this@absline-1}%
1552          \numdef{\sw@atthisline}{\csuse{sw@\sw@txt @\this@absline @\the\
      section@numR @R}-\csuse{sw@\sw@txt @\prev@line @\the\section@numR @R}}%
1553          }%
1554          {\numdef{\sw@atthisline}{0}}%
1555        \ifnumgreater{\sw@atthisline}{1}%
1556            {\showwordrank{#2}{\the@sw}}%
1557            {#2}%
1558    \fi%
1559 }%
1560 %
```

```
\showwordrank 1561 % Finally, the way the rank will be printed.
           1562 \newcommand{\showwordrank}[2]{%
           1563   #1\textsuperscript{#2}%
```

```
1564 }%
1565 %
```

VII   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.

VII.1   Boxes, counters, \pstart and \pend

\raw@text
\ifnumberedpar@
\numberedpar@true
\numberedpar@false
\num@lines
\one@line
\par@line

Here are numbers and flags that are used internally in the course of the paragraph decomposition.

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

```
1566 \newbox\raw@text
1567 \newif\ifnumberedpar@
1568 \newcount\num@lines
1569 \newbox\one@line
1570 \newcount\par@line
1571 %
```

\pstart
\AtEveryPstart
\numberpstarttrue
\numberpstartfalse
\labelpstarttrue
\labelpstartfalse
\thepstart

\pstart starts the paragraph by clearing the \inserts@list list and other relevant variables, and then arranges for the subsequent text to go into the \raw@text box. \pstart needs to appear at the start of every paragraph that is to be numbered; the \autopar command below may be used to insert these commands automatically.

Beware: everything that occurs between \pstart and \pend is happening within a group; definitions must be global if you want them to survive past the end of the paragraph.

```
1572
1573 \newcommand{\AtEveryPstart}[1]{%
1574   \ifstrempty{#1}%
1575     {\xdef\at@every@pstart{}}%
1576     {\gdef\at@every@pstart{\noindent#1}}%
1577 }%
1578 \xdef\at@every@pstart{}%
1579
1580 \newcounter{pstart}
```

```
1581  \renewcommand{\thepstart}{{\bfseries\@arabic\c@pstart}. }
1582  \newif\ifnumberpstart
1583  \numberpstartfalse
1584  \newif\iflabelpstart
1585  \labelpstartfalse
1586  \newcommandx*{\pstart}[1][1]{%
1587    \normal@pars%
1588    \ifstrempty{#1}{\at@every@pstart}{\noindent#1}%
1589    \ifautopar%
1590      \autopar%
1591    \fi%
1592    \ifluatex%
1593      \edef\l@luatextextdir@L{\the\luatextextdir}%
1594    \fi%
1595    \if@nobreak%
1596      \let\@oldnobreak\@nobreaktrue%
1597    \else%
1598      \let\@oldnobreak\@nobreakfalse%
1599    \fi%
1600    \@nobreaktrue%
1601    \ifnumbering \else%
1602      \led@err@PstartNotNumbered%
1603      \beginnumbering%
1604    \fi%
1605    \ifnumberedpar@%
1606      \led@err@PstartInPstart%
1607      \pend%
1608    \fi%
1609    \list@clear{\inserts@list}%
1610    \global\let\next@insert=\empty%
1611    \begingroup\normal@pars%
1612    \global\advance \l@dnumpstartsL\@ne
1613    \global\setbox\raw@text=\vbox\bgroup%
1614      \ifautopar\else%
1615      \ifnumberpstart%
1616        \ifinstanza\else%
1617         \ifsidepstartnum\else%
1618           \thepstart%
1619          \fi%
1620         \fi%
1621        \fi%
1622       \fi%
1623    \numberedpar@true%
1624    \iflabelpstart\protected@edef\@currentlabel%
1625        {\p@pstart\thepstart}
1626    \fi%
1627    \l@dzeropenalties%
1628    \ignorespaces%because not automatically ignored if an optional argument
  is used (classical TeX behavior for space after commands)
1629    }
```

```
1630 %
```

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

```
1631 \newcommandx*{\pend}[1][1]{\ifnumbering \else%
1632     \led@err@PendNotNumbered%
1633   \fi%
1634   \global\l@dskipversenumberfalse%
1635   \ifnumberedpar@ \else%
1636     \led@err@PendNoPstart%
1637   \fi%
1638 %
```

We set all the usual interline penalties to zero and then immediately call \endgraf to end the paragraph; this ensures that there will 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 are not any more lines left.

```
1639   \l@dzeropenalties%
1640   \endgraf\global\num@lines=\prevgraf\egroup%
1641   \global\par@line=0%
1642 %
```

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

```
1643   \csnumdef{pstartline}{0}%
1644   \loop\ifvbox\raw@text%
1645     \csnumdef{pstartline}{\pstartline+1}%
1646     \do@line%
1647     \ifbypstart@%
1648       \ifnumequal{\pstartline}{1}{\setline{1}\resetprevline@}{}%
1649     \fi%
1650   \repeat%
1651 %
```

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

```
1652   \flush@notes%
1653   \endgroup%
1654   \ignorespaces%
1655 %
```

Increase pstart counter.

```
1656   \ifnumberpstart%
1657       \pstartnumtrue%
1658   \fi%
1659   \addtocounter{pstart}{1}%
1660 %
```

Restore paragraph, nobreak setting and autopar setting.

```
1661    \normal@pars%
1662    \@oldnobreak%
1663    \ifautopar%
1664      \autopar%
1665    \fi%
1666 %
```

Print the optional argument of \pend or the content printed after every \pend

```
1667    \ifstrempty{#1}{\at@every@pend}{\noindent#1}%
1668 }
1669
1670 %
```

Here, two macros to insert content after every \pend, between numbered line. \AtEveryPend is the user macro, \at@every@pend is macro set by it.

\AtEveryPend
\at@every@pend
```
1671
1672 \newcommand{\AtEveryPend}[1]{%
1673    \ifstrempty{#1}%
1674      {\xdef\at@every@pend{}}%
1675      {\gdef\at@every@pend{\noindent#1}}%
1676 }%
1677 \xdef\at@every@pend{}%
1678
1679 %
```

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

```
1680 \newcommand*{\l@dzeropenalties}{%
1681    \brokenpenalty \z@ \clubpenalty \z@
1682    \displaywidowpenalty \z@ \interlinepenalty \z@ \predisplaypenalty \z@
1683    \postdisplaypenalty \z@ \widowpenalty \z@}
1684
1685 %
```

\autopar   In most cases it is 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 do not 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 has been added for this paragraph. Then we start again with \pstart, restoring the indentation that we saved, and locally change \par so that it will do our \pend for us.

```
1686 \newif\ifautopar
1687 \autoparfalse
1688 \newcommand*{\autopar}{
1689   \ifledRcol
1690     \ifnumberingR \else
1691     \led@err@AutoparNotNumbered
1692     \beginnumberingR
1693     \fi
1694   \else
1695     \ifnumbering \else
1696     \led@err@AutoparNotNumbered
1697     \beginnumbering
1698     \fi
1699   \fi
1700   \autopartrue
1701   \everypar{\setbox0=\lastbox
1702     \endgraf \vskip-\parskip
1703     \pstart \noindent \kern\wd0 \ifnumberpstart\ifinstanza\else\thepstart\
fi\fi
1704     \let\par=\pend}%
1705   \ignorespaces}
1706 %
```

\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 will want to do this within a footnotes, for example.

```
1707 \newcommand*{\normal@pars}{\everypar{}\let\par\endgraf}
1708
1709 %
```

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

```
1710 \newif\ifautopar@pause
1711 %
```

VII.2   Processing one line

VII.2.1   General process

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

```
1712  \newcommand*{\l@dunhbox@line}[1]{\unhbox #1}
1713  \newcommand*{\do@line}{%
1714   {\vbadness=10000
1715    \splittopskip=\z@
1716    \do@linehook
1717  \l@demptyd@ta
1718    \global\setbox\one@line=\vsplit\raw@text to\baselineskip}%
1719   \unvbox\one@line \global\setbox\one@line=\lastbox
1720   \getline@num
1721   \IfStrEq{\led@pb@setting}{before}{\led@check@pb\led@check@nopb}{}
1722   \ifnum\@lock>\@ne
1723     \inserthangingsymboltrue
1724   \else
1725     \inserthangingsymbolfalse
1726   \fi
1727   \check@pb@in@verse
1728   \ifl@dhidenumber%
1729     \global\l@dhidenumberfalse%
1730     \f@x@l@cks%
1731   \else%
1732     \affixline@num%
1733   \fi%
1734  %
```

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

```
1735    \xifinlist{\the\l@dnumpstartsL}{\eled@sections@@}%
1736      {\print@eledsection}%
1737      {\print@line}%
1738   \IfStrEq{\led@pb@setting}{after}{\led@check@pb\led@check@nopb}{}
1739   }%
1740  %
```

VII.2.2   Process for "normal" line

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

```
1741  \def\print@line{
1742  %
```

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

```
1743      \affixpstart@num%
1744  %
```

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

```
1745      \hb@xt@ \linewidth{%
1746 %
```

User hook.

```
1747      \do@insidelinehook%
1748 %
```

Left line number

```
1749      \l@dld@ta%
1750 %
```

Restore marginal and footnotes.

```
1751          \add@inserts\affixside@note%
1752 %
```

Print left notes.

```
1753          \l@dlsn@te
1754 %
```

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

```
1755          {\ledllfill\hb@xt@ \wd\one@line{\new@line%
1756 %
```

If we use LuaLATEX then restore the direction.

```
1757          \ifluatex%
1758            \luatextextdir\l@luatextextdir@L%
1759          \fi%
1760 %
```

Insert, if needed, the hanging symbol.

```
1761          \inserthangingsymbol %Space keept for backward compatibility
1762 %
```

And so, print the line.

```
1763          \l@dunhbox@line{\one@line}}%
1764 %
```

Right line number

```
1765          \ledrlfill\l@drd@ta%
1766 %
```

Print right notes.

```
1767          \l@drsn@te
1768        }}%
1769 %
```

And reinsert penalties (for page breaking)...

```
1770      \add@penalties%
1771 }
1772 %
```

VII.2.3   Process for line containing \eledsection command

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

```
1773 \def\print@eledsection{%
1774     \add@inserts\affixside@note%
1775     \numdef{\temp@}{\l@dnumpstartsL-1}%
1776     \xifinlist{\temp@}{\eled@sections@@}{\@nobreaktrue}{\@nobreakfalse}%
1777     \@eled@sectioningtrue%
1778     \csuse{eled@sectioning@\the\l@dnumpstartsL}%
1779     \@eled@sectioningfalse%
1780     \global\csundef{eled@sectioning@\the\l@dnumpstartsL}%
1781     \if@RTL%
1782       \hspace{-3\paperwidth}%
1783     {\hbox{\l@dunhbox@line{\one@line}} \new@line}%
1784     \else%
1785       \hspace{3\paperwidth}%
1786     {\new@line \hbox{\l@dunhbox@line{\one@line}}}%
1787     \fi%
1788     \vskip-\baselineskip%
1789 }
1790 %
```

VII.2.4   Hooks

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

```
1791 \newcommand*{\do@linehook}{}
1792 \newcommand*{\do@insidelinehook}{}
1793 %
```

\dolinehook   These hight level commands just redefine the low level commands. They have to be used
\doinsidelinehook   be user, without \makeatletter.

```
1794 \newcommand*{\dolinehook}[1]{\gdef\do@linehook{#1}}%
1795 \newcommand*{\doinsidelinehook}[1]{\gdef\do@insidelinehook{#1}}%
1796
1797 %
```

VII.2.5   Sidenotes and marginal line number initialization

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

```
1798 \newcommand*{\l@demptyd@ta}{%
1799   \gdef\l@dld@ta{}%
1800   \gdef\l@drd@ta{}%
1801   \gdef\l@dcsnotetext@l{}%
1802   \gdef\l@dcsnotetext@r{}%
1803   \gdef\l@dcsnotetext{}}
1804
1805 %
```

\l@dlsn@te Zero width boxes of the left and right side notes, together with their kerns.
\l@drsn@te
```
1806 \newcommand{\l@dlsn@te}{%
1807   \hb@xt@ \z@{\hss\box\l@dlp@rbox\kern\ledlsnotesep}}
1808 \newcommand{\l@drsn@te}{%
1809   \hb@xt@ \z@{\kern\ledrsnotesep\box\l@drp@rbox\hss}}
1810
1811 %
```

\ledllfill These macros are called at the left (\ledllfill) and the right (\ledllfill) of each
\ledrlfill numbered line. The initial definitions correspond to the original code for \do@line.
```
1812 \newcommand*{\ledllfill}{\hfil}
1813 \newcommand*{\ledrlfill}{}
1814
1815 %
```

VIII   Line and page number computation

\getline@num The \getline@num macro determines the page and line numbers for the line we are
about to send to the vertical list.
```
1816 \newcommand*{\getline@num}{%
1817   \global\advance\absline@num \@ne%
1818   \do@actions
1819   \do@ballast
1820   \ifnumberline
1821       \ifsublines@
1822           \ifnum\sub@lock<\tw@
1823               \global\advance\subline@num \@ne
1824           \fi
1825       \else
1826           \ifnum\@lock<\tw@
1827               \global\advance\line@num \@ne
1828               \global\subline@num \z@
1829           \fi
1830       \fi
1831   \fi
1832 }
1833 %
```

\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 XI.2 p. 134).

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

```
1834 \newcount\ballast@count
1835 \newcounter{ballast}
1836   \setcounter{ballast}{0}
1837 %
```

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.

```
1838 \newcommand*{\do@ballast}{\global\ballast@count \z@
1839   \begingroup
1840     \advance\absline@num \@ne
1841     \ifnum\next@actionline=\absline@num
1842       \ifnum\next@action>-1001\relax
1843         \global\advance\ballast@count by -\c@ballast
1844       \fi
1845     \fi
1846   \endgroup}
1847 %
```

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

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

```
1848 \newcommand*{\do@actions}{%
1849   \global\let\do@actions@next=\relax
1850   \ifnum\absline@num<\next@actionline\else
1851 %
```

First, page number changes, which will generally be the most common actions. If we are restarting lineation on each page, this is where it happens.

```
1852     \ifnum\next@action>-1001
1853       \global\page@num=\next@action
1854       \ifbypage@
1855         \global\line@num=\z@ \global\subline@num=\z@
1856         \resetprevline@
1857       \fi
1858 %
```

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.)

```
1859       \else
1860          \ifnum\next@action<-4999
1861             \@l@dtempcnta=-\next@action
1862             \advance\@l@dtempcnta by -5001
1863             \ifsublines@
1864                \global\subline@num=\@l@dtempcnta
1865             \else
1866                \global\line@num=\@l@dtempcnta
1867             \fi
1868 %
```

We rescale the value in \@l@dtempcnta so that we can use a case statement.

```
1869          \else
1870             \@l@dtempcnta=-\next@action
1871             \advance\@l@dtempcnta by -1000
1872             \do@actions@fixedcode
1873          \fi
1874       \fi
1875 %
```

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

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

```
1876       \ifx\actionlines@list\empty
1877          \gdef\next@actionline{1000000}%
1878       \else
1879          \gl@p\actionlines@list\to\next@actionline
1880          \gl@p\actions@list\to\next@action
1881          \global\let\do@actions@next=\do@actions
1882       \fi
1883    \fi
1884 %
```

Make the recursive call, if necessary.

```
1885 \do@actions@next}
1886
1887 %
```

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

```
1888 \newcommand*{\do@actions@fixedcode}{%
1889    \ifcase\@l@dtempcnta
1890    \or%                    % 1001
1891       \global\sublines@true
```

```
1892    \or%                          % 1002
1893       \global\sublines@false
1894    \or%                          % 1003
1895          \global\@lock=\@ne
1896    \or%                          % 1004
1897       \ifnum\@lock=\tw@
1898          \global\@lock=\thr@@
1899       \else
1900          \global\@lock=\z@
1901       \fi
1902    \or%                          % 1005
1903          \global\sub@lock=\@ne
1904    \or%                          % 1006
1905       \ifnum\sub@lock=\tw@
1906          \global\sub@lock=\thr@@
1907       \else
1908          \global\sub@lock=\z@
1909       \fi
1910    \or%                          % 1007
1911       \l@dskipnumbertrue
1912    \or%                          % 1008
1913       \l@dskipversenumbertrue%
1914  \or%     % 1009
1915       \l@dhidenumbertrue
1916    \else
1917       \led@warn@BadAction
1918    \fi}
1919
1920
1921 %
```

IX   Line number printing

\affixline@num   \affixline@num just puts a left line number into \l@dld@ta or a right line number into \l@drd@ta if required.

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

$$n \quad = int((\textit{linenum} - \textit{firstlinenum})/\textit{linenumincrement})$$
$$m \quad = \textit{firstlinenum} + (n \times \textit{linenumincrement})$$

(where *int* truncates a real number to an integer). m will be equal to *linenum* only if we are to paste a number on here. However, the formula breaks down for the first line to number (and any before that), so we check that case separately: if \line@num $\leq$ \firstlinenum, we compare the two directly instead of making these calculations.

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

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

```
1922  \newcommand*{\affixline@num}{%
1923  %
```

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).

```
1924    \ifledgroupnotesL@\else
1925      \ifnumberline
1926        \ifl@dskipnumber
1927          \global\l@dskipnumberfalse
1928        \else
1929          \ifsublines@
1930            \@l@dtempcntb=\subline@num
1931            \ifnum\subline@num>\c@firstsublinenum
1932              \@l@dtempcnta=\subline@num
1933              \advance\@l@dtempcnta by-\c@firstsublinenum
1934              \divide\@l@dtempcnta by\c@sublinenumincrement
1935              \multiply\@l@dtempcnta by\c@sublinenumincrement
1936              \advance\@l@dtempcnta by\c@firstsublinenum
1937            \else
1938              \@l@dtempcnta=\c@firstsublinenum
1939            \fi
1940  %
```

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.

```
1941              \ch@cksub@l@ck
1942  %
```

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

```
1943          \else
1944            \@l@dtempcntb=\line@num
1945  %
```

Check on the \linenumberlist If it is \empty use the standard algorithm.

```
1946            \ifx\linenumberlist\empty
1947              \ifnum\line@num>\c@firstlinenum
1948                \@l@dtempcnta=\line@num
1949                \advance\@l@dtempcnta by-\c@firstlinenum
1950                \divide\@l@dtempcnta by\c@linenumincrement
1951                \multiply\@l@dtempcnta by\c@linenumincrement
1952                \advance\@l@dtempcnta by\c@firstlinenum
1953              \else
1954                \@l@dtempcnta=\c@firstlinenum
1955              \fi
1956            \else
1957  %
```

The `\linenumberlist` was not `\empty`, so here is Wayne's numbering mechanism. This takes place in TEX's mouth.

```
1958                \@l@dtempcnta=\line@num
1959                \edef\rem@inder{,\linenumberlist,\number\line@num,}%
1960                \edef\sc@n@list{\def\noexpand\sc@n@list
1961                  ####1,\number\@l@dtempcnta,####2|{\def\noexpand\rem@inder
{####2}}}%
1962                \sc@n@list\expandafter\sc@n@list\rem@inder|%
1963                \ifx\rem@inder\empty%
1964                  \advance\@l@dtempcnta\@ne
1965                \fi
1966              \fi
1967 %
```

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

```
1968                \ch@ck@l@ck
1969              \fi
1970 %
```

The following tests are true if we need to print a line number.

```
1971              \ifnum\@l@dtempcnta=\@l@dtempcntb
1972                \ifl@dskipversenumber\else
1973 %
```

If we got here, we are 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 is 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 is 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 Peter Wilson thought 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`

```
1974                \if@twocolumn
1975                  \if@firstcolumn
1976                    \gdef\l@dld@ta{\llap{{\leftlinenum}}}%
1977                  \else
1978                    \gdef\l@drd@ta{\rlap{{\rightlinenum}}}%
1979                  \fi
1980                \else
1981                  \@l@dtempcntb=\line@margin
1982                  \ifnum\@l@dtempcntb>\@ne
1983                    \advance\@l@dtempcntb \page@num
1984                  \fi
1985                  \ifodd\@l@dtempcntb
```

```
1986                    \gdef\l@drd@ta{\rlap{{\rightlinenum}}}%
1987                  \else
1988                    \gdef\l@dld@ta{\llap{{\leftlinenum}}}%
1989                \fi
1990              \fi
1991            \fi
1992          \fi
1993 %
```

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

```
1994          \f@x@l@cks
1995        \fi
1996      \fi
1997    \fi
1998 }
1999
2000 %
```

\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 the
   \f@x@l@cks   counters to arbitrary but unequal values.

```
2001 \newcommand*{\ch@cksub@l@ck}{%
2002    \ifcase\sub@lock
2003      \or
2004        \ifnum\sublock@disp=\@ne
2005          \@l@dtempcntb=\z@ \@l@dtempcnta=\@ne
2006        \fi
2007      \or
2008        \ifnum\sublock@disp=\tw@ \else
2009          \@l@dtempcntb=\z@ \@l@dtempcnta=\@ne
2010        \fi
2011      \or
2012        \ifnum\sublock@disp=\z@
2013          \@l@dtempcntb=\z@ \@l@dtempcnta=\@ne
2014        \fi
2015    \fi}
2016 %
```

Similarly for line numbers.

```
2017 \newcommand*{\ch@ck@l@ck}{%
2018    \ifcase\@lock
2019      \or
2020        \ifnum\lock@disp=\@ne
2021          \@l@dtempcntb=\z@ \@l@dtempcnta=\@ne
2022        \fi
2023      \or
2024        \ifnum\lock@disp=\tw@ \else
```

```
2025        \@l@dtempcntb=\z@ \@l@dtempcnta=\@ne
2026      \fi
2027    \or
2028      \ifnum\lock@disp=\z@
2029        \@l@dtempcntb=\z@ \@l@dtempcnta=\@ne
2030      \fi
2031    \fi}
2032 %
```

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

```
2033 \newcommand*{\f@x@l@cks}{%
2034   \ifcase\@lock
2035   \or
2036     \global\@lock=\tw@
2037   \or \or
2038     \global\@lock=\z@
2039   \fi
2040   \ifcase\sub@lock
2041   \or
2042     \global\sub@lock=\tw@
2043   \or \or
2044     \global\sub@lock=\z@
2045   \fi}
2046
2047 %
```

X Pstart number printing in side

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

\affixpstart@num
\pstartnum

- The pstarts counter is upgrade in the \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 is tried in the \leftpstartnum and \rightpstartnum commands. After the try, it is set to FALSE.

\leftpstartnum
\rightpstartnum
\ifsidepstartnum

```
2048
2049 \newif\ifsidepstartnum
2050 \newcommand*{\affixpstart@num}{%
2051   \ifsidepstartnum
2052     \if@twocolumn
2053       \if@firstcolumn
2054         \gdef\l@dld@ta{\llap{{\leftpstartnum}}}%
```

```
2055              \else
2056                   \gdef\l@drd@ta{\rlap{{\rightpstartnum}}}%
2057              \fi
2058          \else
2059               \@l@dtempcntb=\line@margin
2060              \ifnum\@l@dtempcntb>\@ne
2061                   \advance\@l@dtempcntb \page@num
2062              \fi
2063              \ifodd\@l@dtempcntb
2064                   \gdef\l@drd@ta{\rlap{{\rightpstartnum}}}%
2065              \else
2066                   \gdef\l@dld@ta{\llap{{\leftpstartnum}}}%
2067              \fi
2068          \fi
2069      \fi
2070
2071 }
2072 %
2073
2074 \newif\ifpstartnum
2075 \pstartnumtrue
2076 \newcommand*{\leftpstartnum}{
2077      \ifpstartnum\thepstart
2078      \kern\linenumsep\fi
2079      \global\pstartnumfalse
2080 }
2081 \newcommand*{\rightpstartnum}{
2082      \ifpstartnum
2083      \kern\linenumsep
2084      \thepstart
2085      \fi
2086      \global\pstartnumfalse
2087 }
2088 %
```

XI   Restoring footnotes and penalties

Because of the paragraph decomposition process in order to number line, reledmac
must hack the standard way TEX works in order to manage insertion of footnotes, both
critical and familiar.

We need to call the \insert commands not when the content of \pstart…\pend
is read by TEX by when each individual line is typeset.

Consequently, when reading the content of \pstart…\pend, we store the insertion
(footnotes) in an specific reledmac's list, and we restore them to the vertical list when
printing each individual line.

XI.1   Add insertions to the vertical list

\inserts@list   \inserts@list is the list macro that contains the inserts that we save up for one para-
graph.

```
2089  \list@create{\inserts@list}
2090  %
```

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

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

```
2091  \newcommand*{\add@inserts}{%
2092    \global\let\add@inserts@next=\relax
2093  %
```

If \inserts@list is empty, there are not any more notes or insertions for this
paragraph, and we need not waste our time.

```
2094    \ifx\inserts@list\empty \else
2095  %
```

The \next@insert macro records the number of the line that receives the next foot-
note or other insert; it is empty when we start out, and just after we have affixed a note
or insert.

```
2096    \ifx\next@insert\empty
2097      \ifx\insertlines@list\empty
2098        \global\noteschanged@true
2099        \gdef\next@insert{100000}%
2100      \else
2101        \gl@p\insertlines@list\to\next@insert
2102      \fi
2103    \fi
2104  %
```

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 will call ourself recursively: there might be another insert for this same line.

```
2105    \ifnum\next@insert=\absline@num
2106      \gl@p\inserts@list\to\@insert
2107      \@insert
2108      \global\let\@insert=\undefined
2109      \global\let\next@insert=\empty
2110      \global\let\add@inserts@next=\add@inserts
2111    \fi
2112  \fi
2113  %
```

Make the recursive call, if necessary.

```
2114  \add@inserts@next}
2115
2116  %
```

XI.2   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 para-
graph; 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 are working on at the moment. The count \@l@dtempcnta
is used to calculate and accumulate the penalty; it is initially set to the value of
\ballast@count, which has been worked out in \do@ballast above (VIII p. 125). Fi-
nally, the penalty is checked to see that it does not go below -10000.

```
2117  \newcommand*{\add@penalties}{\@l@dtempcnta=\ballast@count
2118    \ifnum\num@lines>\@ne
2119      \global\advance\par@line \@ne
2120      \ifnum\par@line=\@ne
2121        \advance\@l@dtempcnta \clubpenalty
2122      \fi
2123      \@l@dtempcntb=\par@line \advance\@l@dtempcntb \@ne
2124      \ifnum\@l@dtempcntb=\num@lines
2125        \advance\@l@dtempcnta \widowpenalty
2126      \fi
2127      \ifnum\par@line<\num@lines
2128        \advance\@l@dtempcnta \interlinepenalty
2129      \fi
2130    \fi
2131    \ifnum\@l@dtempcnta=\z@
2132      \relax
2133    \else
2134      \ifnum\@l@dtempcnta>-10000
2135        \penalty\@l@dtempcnta
2136      \else
2137        \penalty -10000
2138      \fi
2139    \fi}
2140
2141  %
```

XI.3   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 previous run of TₑX, then there can be leftover notes that have not yet been printed. An appropriate error message will be printed elsewhere; but it is best to go ahead and print these notes somewhere, even if it is 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 is not too far from the proper location, to which they will move on the next run.

```
2142  \newcommand*{\flush@notes}{%
2143    \@xloop
2144      \ifx\inserts@list\empty \else
2145        \gl@p\inserts@list\to\@insert
2146        \@insert
2147        \global\let\@insert=\undefined
2148    \repeat}
2149
2150  %
```

`\@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.

```
2151  \def\@xloop#1\repeat{%
2152    \def\body{#1\expandafter\body\fi}%
2153    \body}
2154
2155  %
```

XII   Critical footnotes

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

XII.1   Fonts

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

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

```
2156    \def\select@lemmafont#1|#2|#3|#4|#5|#6|#7|{\select@@lemmafont#7|}
2157    \def\select@@lemmafont#1/#2/#3/#4|%
2158      {\fontencoding{#1}\fontfamily{#2}\fontseries{#3}\fontshape{#4}%
2159      \selectfont}
2160
2161 %
```

XII.2   Individual note options

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

```
2162 \newcommandx*{\footnoteoptions@}[3][1=L,usedefault]{%
2163     \def\do##1{%
2164        \ifstrequal{#1}{L}{% In Leftside
2165           \xright@appenditem{\global\noexpand\settoggle{##1@}{#3}}\to\
inserts@list% Switch toogle, in all case
2166           \global\advance\insert@count \@ne% Increment the left insert
counter.
2167        }%
2168        {%
2169           \xright@appenditem{\global\noexpand\settoggle{##1@}{#3}}\to\
inserts@listR% Switch toogle, in all case
2170           \global\advance\insert@countR \@ne% Increment the right insert
counter insert.
2171        }%
2172     }%
2173     \notblank{#2}{\docsvlist{#2}}{}% Parsing all options
2174 }
2175 %
```

XII.3   Notes language

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

```
2176 \newcommandx*{\footnotelang@lua}[1][1=L,usedefault]{%
2177     \ifstrequal{#1}{L}{%
2178     \xright@appenditem{{\csxdef{footnote@luatextextdir}{\the\luatextextdir
}}}\to\inserts@list%Know the dir of lemma
2179     \global\advance\insert@count \@ne%
2180     \xright@appenditem{{\csxdef{footnote@luatexpardir}{\the\luatexpardir
}}}\to\inserts@list%Know the dir of lemma
2181     \global\advance\insert@count \@ne%
2182     }%
```

```
2183     {%
2184     \xright@appenditem{{\csxdef{footnote@luatextextdir}{\the\luatextextdir
    }}}\to\inserts@listR%Know the dir of lemma
2185       \global\advance\insert@countR \@ne%
2186       \xright@appenditem{{\csxdef{footnote@luatexpardir}{\the\luatexpardir
    }}}\to\inserts@listR%Know the dir of lemma
2187       \global\advance\insert@countR \@ne%
2188     }%
2189 }
2190 %
```

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

```
2191 \newcommandx*{\footnotelang@poly}[1][1=L,usedefault]{%
2192     \ifstrequal{#1}{L}{%
2193     \if@RTL%
2194         \xright@appenditem{{\csxdef{footnote@dir}{@RTLtrue}}}\to\
    inserts@list%Know the language used in the lemma
2195         \global\advance\insert@count \@ne%
2196     \else
2197         \xright@appenditem{{\csxdef{footnote@dir}{@RTLfalse}}}\to\
    inserts@list%Know the language of lemma
2198         \global\advance\insert@count \@ne%
2199     \fi%
2200     \xright@appenditem{{\csxdef{footnote@lang}{\expandonce\languagename}}}\
    to\inserts@list%Know the language of lemma
2201     \global\advance\insert@count \@ne%
2202     }%
2203     {%
2204     \if@RTL
2205         \xright@appenditem{{\csxdef{footnote@dir}{@RTLtrue}}}\to\
    inserts@listR%Know the language of lemma
2206         \global\advance\insert@countR \@ne%
2207     \else
2208         \xright@appenditem{{\csxdef{footnote@dir}{@RTLfalse}}}\to\
    inserts@listR%Know the language of lemma
2209         \global\advance\insert@countR \@ne%
2210     \fi
2211     \xright@appenditem{{\csxdef{footnote@lang}{\expandonce\languagename}}}\
    to\inserts@listR%Know the language of lemma
2212     \global\advance\insert@countR \@ne%
2213     }%
2214 }
2215 %
```

XII.4   General survey of the way we manage notes

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 format-
ting 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 foot-
note 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 distin-
guished by the 'series letter' that indicates which set of the footnotes we are 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.

These macros are changed depending of the footnotes arrangement: "normal", "para-
graphed", "two columns" or "three columns".

XII.5   General setup

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

- \interlinepenalty.

- \splittopskip (skip before last part of notes that flow from one page to another).

- \splitmaxdepth.

- \floatingpenalty, that is penalty values being added when a long note flows
  from one page to another. Here, we let it to 0 when we are processing paral-
  lel pages in eledpar, in order to allow notes to flow from left to right pages
  and *vice-versa*. Otherwise, we let it to \@MM, which is the standard LaTeX
  \floatingpenalty.

```
2216 \newcommand*{\footsplitskips}{%
2217   \interlinepenalty=\interfootnotelinepenalty
2218   \unless\ifl@dprintingpages%
2219     \floatingpenalty=\@MM%
2220   \fi%
2221   \splittopskip=\ht\strutbox \splitmaxdepth=\dp\strutbox
2222   \leftskip=\z@skip \rightskip=\z@skip}
2223
2224 %
```

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

```
2225 \let\normalfootnoterule=\footnoterule
2226 %
```

XII.6   Footnotes arrangement

XII.6.1   User level macro

\Xarrangement   \Xarrangement[⟨*s*⟩]{⟨*arrangement*⟩} The command calls, for each series, a specific command which set many counters and commands in order to define specific arrangement.

```
2227 \newcommandx{\Xarrangement}[2][1,usedefault]{%
2228   \def\do##1{%
2229     \csname Xarrangement@#2\endcsname{##1}%
2230   }%
2231   \ifstrempty{#1}%
2232     {%
2233     \dolistloop{\@series}%
2234     }%
2235     {
2236     \docsvlist{#1}%
2237     }%
2238 }%
2239 %
```

XII.6.2   Normal footnote

\Xarrangement@normal   We can now define all the parameters for the series of footnotes; initially they use the "normal" footnote formatting.

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 `reledmac` code.)

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

(Read *The TeXbook* in order to understand what are the counter, skip and dimen associated to an insertion.)

Instead of repeating ourselves, we define a \Xarrangement@normal macro that makes all these assignments for us, for any given series letter. This command is called when people use \Xarrangement[⟨*series*⟩]{normal}

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

```
2240 \newcommand*{\Xarrangement@normal}[1]{%
2241   \csgdef{series@display#1}{normal}
2242   \expandafter\let\csname #1footstart\endcsname=\normalfootstart
2243   \expandafter\let\csname v#1footnote\endcsname=\normalvfootnote
2244   \expandafter\let\csname #1footfmt\endcsname=\normalfootfmt
```

```
2245    \expandafter\let\csname #1footgroup\endcsname=\normalfootgroup
2246    \expandafter\let\csname #1footnoterule\endcsname=%
2247                                                    \normalfootnoterule
2248    \count\csname #1footins\endcsname=1000
2249    \dimen\csname #1footins\endcsname=\csuse{Xmaxhnotes@#1}
2250    \skip\csname #1footins\endcsname=\csuse{Xbeforenotes@#1}%
2251    \advance\skip\csname #1footins\endcsname by\csuse{Xafterrule@#1}%
2252 %
```

The `reledpar` provides tools in order to confine notes to one side. The mechanism is explained in the `reledpar`'s handbook. For now, just retain we need to store default value of the counter associated to the notes TEX's inserts.

```
2253    \csxdef{default@#1footins}{1000}%Use this to confine the  notes to one
      side only
2254 %
```

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).

```
2255    \ifnoledgroup@\else%
2256      \expandafter\let\csname mpv#1footnote\endcsname=\mpnormalvfootnote
2257      \expandafter\let\csname mp#1footgroup\endcsname=\mpnormalfootgroup
2258      \count\csname mp#1footins\endcsname=1000
2259      \dimen\csname mp#1footins\endcsname=\csuse{Xmaxhnotes@#1}
2260      \skip\csname mp#1footins\endcsname=\csuse{Xbeforenotes@#1}%
2261      \advance\skip\csname mp#1footins\endcsname by\csuse{Xafterrule@#1}%
2262    \fi
2263 }
2264
2265 %
```

\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.

```
2266 \notbool{parapparatus@}{\newcommand*}{\newcommand}{\normalvfootnote}[2]{%
2267    \insert\csname #1footins\endcsname\bgroup
2268    \csuse{Xbhooknote@#1}
2269    \csuse{Xnotefontsize@#1}
2270    \footsplitskips
2271    \ifl@dpairing\ifl@dpaging\else%
2272      \setXnoteswidthliketwocolumns@{#1}%
2273    \fi\fi%
2274    \setXnotespositionliketwocolumns@{#1}%
2275    \spaceskip=\z@skip \xspaceskip=\z@skip
2276    \csname #1footfmt\endcsname #2{#1}\egroup}
2277 %
```

\mpnormalvfootnote   And a somewhat different version for minipages.

```
2278 \notbool{parapparatus@}{\newcommand*}{\newcommand}{\mpnormalvfootnote}[2]{%
2279   \global\setbox\@nameuse{mp#1footins}\vbox{%
2280     \unvbox\@nameuse{mp#1footins}
2281     \csuse{Xbhooknote@#1}
2282     \csuse{Xnotefontsize@#1}
2283     \hsize\columnwidth
2284     \@parboxrestore
2285     \color@begingroup
2286     \csname #1footfmt\endcsname #2{#1}\color@endgroup}}
2287
2288 %
```

\normalfootfmt   \normalfootfmt is a 'normal' macro to take the footnote line and page number infor-
mation (see V.9 p. 80), 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.

```
2289
2290
2291 \notbool{parapparatus@}{\newcommand*}{\newcommand}{\normalfootfmt}[4]{%
2292   \Xledsetnormalparstuff{#4}%
2293   \hangindent=\csuse{Xhangindent@#4}
2294   \strut{\printlinefootnote{#1}{#4}}%
2295   {\nottoggle{Xlemmadisablefontselection@#4}{\select@lemmafont#1|#2}{#2}}%
2296   \iftoggle{nosep@}{\hskip\csuse{Xinplaceoflemmaseparator@#4}}{\ifcsempty{
Xlemmaseparator@#4}%
2297     {\hskip\csuse{Xinplaceoflemmaseparator@#4}}%
2298     {\nobreak\hskip\csuse{Xbeforelemmaseparator@#4}\csuse{Xlemmaseparator@
#4}\hskip\csuse{Xafterlemmaseparator@#4}\relax%
2299   }}%
2300   #3\strut\par}
2301 %
```

\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 ex-
actly equal to the \skip\Xfootins value for the associated series of notes. TeX makes
page computations based on that \skip value, and the output pages will suffer from
spacing problems if what you add takes up a different amount of space.

   But if the skip \preXnotes@ is greater than 0 pt, it is used instead of \skip\footins
for the first printed series in one page.

   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 speak-
ing, 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 reledmac in this respect is general across all footnote types. 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.

```
2302 \newcommand*{\normalfootstart}[1]{%
2303 %
```

The first series of notes printed in a page can have a specific skip before it. In order to insert this specific skip without overlap the bottom margin of the page, Maïeul Rouquette have defined an algorithm explained in XVIII p. 182. Here is part of this algorithm, when the block of notes are ready to be printed.

```
2304    \ifdimequal{0pt}{\preXnotes@}{}%
2305        {%
2306        \iftoggle{preXnotes@}{%
2307            \togglefalse{preXnotes@}%
2308            \skip\csname #1footins\endcsname=%
2309              \dimexpr\csuse{preXnotes@}+\csuse{Xafterrule@#1}\relax%
2310            }%
2311          {}%
2312        }%
2313    \vskip\skip\csname #1footins\endcsname%
2314 %
```

And now, the problem of left and right skip for notes. Especially when using one feature of reledpar which allows to have the footnotes horizontal size as the size of columns printed by \Columns. Read XV p. 180 for the general description of the problem.

```
2315    \leftskip0pt \rightskip0pt
2316    \ifl@dpairing\else%
2317        \hsize=\old@hsize%
2318    \fi%
2319    \setXnoteswidthliketwocolumns@{#1}%
2320    \setXnotespositionliketwocolumns@{#1}%
2321 %
```

And now, print the footnote's rule to finish the footnote's introduction.

```
2322    \print@Xfootnoterule{#1}%
2323    \noindent\leavevmode}
2324 %
```

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

```
2325 \newcommand*{\normalfootgroup}[1]{%
2326   {\csuse{Xnotefontsize@#1}\noindent\csuse{Xtxtbeforenotes@#1}}%
2327   \unvbox\csname #1footins\endcsname%
2328   \hsize=\old@hsize%
2329   }%
2330
2331 %
```

`\mpnormalfootgroup`   A somewhat different version for minipages. Notes that, in this case, we don not make distinction between `\Xfootgroup` and `\Xfootstarts` macro.

```
2332  \unless\ifnoledgroup@
2333  \newcommand*{\mpnormalfootgroup}[1]{{
2334    \vskip\skip\@nameuse{mp#1footins}
2335    \ifl@dpairing\ifparledgroup%
2336      \leavevmode\marks\parledgroup@{begin}%
2337      \marks\parledgroup@series{#1}%
2338      \marks\parledgroup@type{Xfootnote}%
2339    \fi\fi\normalcolor%
2340    \ifparledgroup%
2341      \ifl@dpairing%
2342      \else%
2343        \setXnoteswidthliketwocolumns@{#1}%
2344        \setXnotespositionliketwocolumns@{#1}%
2345        \print@Xfootnoterule{#1}%%
2346      \fi%
2347    \else%
2348      \setXnoteswidthliketwocolumns@{#1}%
2349      \setXnotespositionliketwocolumns@{#1}%
2350      \print@Xfootnoterule{#1}%%
2351    \fi%
2352    \setlength{\parindent}{0pt}
2353   {\csuse{Xnotefontsize@#1}\csuse{Xtxtbeforenotes@#1}}
2354   \unvbox\csname mp#1footins\endcsname}}
2355  \fi
2356  %
```

XII.6.3   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 TeXbook*, pp. 398–400, with alterations for our environment. This algorithm uses a considerable amount of save-stack space: a TEX of ordinary size may not be able to handle more than about 100 notes of this kind on a page.

`\Xarrangement@paragraph`   The `\Xarrangement@paragraph` macro sets up everything for one series of the footnotes so that they will 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 user is switching to paragraphed footnotes after having columnar ones, since they change this value (see below).

   The argument of `\Xarrangement@footparagraph` is the letter denoting the series of notes to be paragraphed.

```
2357  \newcommand*{\Xarrangement@paragraph}[1]{%
2358    \csgdef{series@display#1}{paragraph}
2359    \expandafter\newcount\csname #1prevpage@num\endcsname
2360    \expandafter\let\csname #1footstart\endcsname=\parafootstart
```

```
2361    \expandafter\let\csname v#1footnote\endcsname=\paravfootnote
2362    \expandafter\let\csname #1footfmt\endcsname=\parafootfmt
2363    \expandafter\let\csname #1footgroup\endcsname=\parafootgroup
2364    \count\csname #1footins\endcsname=1000
2365    \csxdef{default@#1footins}{1000}%Use this to confine the  notes to one
     side only
2366    \dimen\csname #1footins\endcsname=\csuse{Xmaxhnotes@#1}
2367    \skip\csname #1footins\endcsname=\csuse{Xbeforenotes@#1}%
2368    \advance\skip\csname #1footins\endcsname by\csuse{Xafterrule@#1}%
2369    \para@footsetup{#1}
2370 %
```

And the extra setup for minipages.

```
2371    \ifnoledgroup@\else
2372      \expandafter\let\csname mpv#1footnote\endcsname=\mpparavfootnote
2373      \expandafter\let\csname mp#1footgroup\endcsname=\mpparafootgroup
2374      \count\csname mp#1footins\endcsname=1000
2375      \dimen\csname mp#1footins\endcsname=\csuse{Xmaxhnotes@#1}
2376      \skip\csname mp#1footins\endcsname=\csuse{Xbeforenotes@#1}%
2377      \advance\skip\csname mp#1footins\endcsname by\csuse{Xafterrule@#1}%
2378    \fi
2379 }
2380 %
```

\footfudgefiddle   For paragraphed footnotes TEX has to estimate the amount of space required. If it un-
derestimates 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.

```
2381 \providecommand{\footfudgefiddle}{64}
2382 %
```

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

Peter Wilson thinks that \columnwidth should be used here for LATEX not \hsize.
Peter Wilson have also included \footfudgefiddle.

```
2383 \newcommand*{\para@footsetup}[1]{{\csuse{Xnotefontsize@#1}
2384    \setXnoteswidthliketwocolumns@{#1}%
2385    \dimen0=\baselineskip
2386    \multiply\dimen0 by 1024
2387    \divide \dimen0 by \columnwidth \multiply\dimen0 by \footfudgefiddle\
     relax
2388    \csxdef{#1footfudgefactor}{%
2389      \expandafter\strip@pt\dimen0 }}}
2390
2391 %
```

\strip@pt strip the characters pt from a dimen value.

\parafootstart \parafootstart is the same as \normalfootstart, but we give it again to en-
sure that \rightskip and \leftskip are zeroed (this needs to be done before
\para@footgroup in the output routine). 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.

```
2392  \newcommand*{\parafootstart}[1]{%
2393    \rightskip=0pt \leftskip=0pt \parindent=0pt
2394      \ifdimequal{0pt}{\preXnotes@}{}%
2395        {%
2396        \iftoggle{preXnotes@}{%
2397            \togglefalse{preXnotes@}%
2398            \skip\csname #1footins\endcsname=%
2399              \dimexpr\csuse{preXnotes@}+\csuse{Xafterrule@#1}\relax%
2400          }%
2401        {}%
2402      }%
2403    \vskip\skip\csname #1footins\endcsname%
2404    \setXnoteswidthliketwocolumns@{#1}%
2405    \setXnotespositionliketwocolumns@{#1}%
2406    \print@Xfootnoterule{#1}%%
2407    \noindent\leavevmode}
2408  %
```

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

The initial model for this insertion is, of course, the \insert\footins definition
in *The TeXbook*, p. 398. There, the footnotes are first collected up in hboxes, and these
hboxes are later unpacked and stuck together into a paragraph.

However, Michael Downes has pointed out that because text in hboxes gets typeset
in restricted horizontal mode, there are some undesirable side-effects if you later want to
break such text across lines. In restricted horizontal mode, where TeX does not expect
to have to break lines, it does not insert certain items like \discretionarys. 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.[27]

Wayne Sullivan pointed out to us another subtle problem that arises from the same
cause: TeX also leaves the \language whatsit nodes out of the horizontal list.[28] So
changes from one language to another will not invoke the proper hyphenation rules in

---

[27] Michael Downes, 'Line Breaking in \unhboxed Text', *TUGboat* **11** (1990), pp. 605–612.
[28] See *The TeXbook*, p. 455 (editions after January 1990).

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 is not too hard. For details, we refer you to Michael's article, where the issues are clearly explained.[29] Michael's unboxing macro is called \Xunvxh: 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.[30] In other words, be very careful not to say \break, or \penalty-10000, or any equivalent inside your para-footnote. If you do, most of the note will probably disappear. You *are* allowed to make strong suggestions; in fact \penalty-9999 will be quite okay. Just do not make the break mandatory. We have not applied any of Michael's solutions here, since we feel that the problem is exiguous, and reledmac 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. XII.6.2 p. 141 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.

```
2409  \newcommand*{\paravfootnote}[2]{%
2410    \insert\csname #1footins\endcsname
2411    \bgroup
2412      \csuse{Xbhooknote@#1}
2413      \csuse{Xnotefontsize@#1}
2414      \footsplitskips
2415      \setbox0=\vbox{\hsize=\maxdimen
2416        \noindent\csname #1footfmt\endcsname #2{#1}}%
2417      \setbox0=\hbox{\Xunvxh{0}{#1}}%
2418      \dp0=0pt
2419      \ht0=\csname #1footfudgefactor\endcsname\wd0
2420  %
```

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

```
2421      \if@RTL\noindent \leavevmode\fi\box0%
2422      \penalty0
```

---

[29]Wayne supplied his own macros to do this, but since they were almost identical to Michael's, Peter Wilson have used the latter's \Xunvxh macro since it is publicly documented.

[30]'Line Breaking', p. 610.

```
2423    \egroup}
2424
2425 %
```

The final penalty of 0 was added here at Wayne's suggestion to avoid a weird page-breaking problem, which occurs on those occasions when TeX attempts to split foot paragraphs. After trying out such a split (see *The TeXbook*, p. 124), TeX inserts a penalty of −10000 here, which nearly always forces the break at the end of the whole footnote paragraph (since individual notes can't be split) even when this leads to an overfull vbox. The change above results in a penalty of 0 instead which allows, but does not force, such breaks. This penalty of 0 is later removed, after page breaks have been decided, by the \unpenalty macro in \makehboxofhboxes. So it does not affect how the footnote paragraphs are typeset (the notes still have a penalty of −10 between them, which is added by \parafootfmt).

\mpparavfootnote This version is for minipages.

```
2426 \newcommand*{\mpparavfootnote}[2]{%
2427   \global\setbox\@nameuse{mp#1footins}\vbox{%
2428     \unvbox\@nameuse{mp#1footins}%
2429     \csuse{Xbhooknote@#1}
2430     \csuse{Xnotefontsize@#1}
2431     \footsplitskips
2432     \setbox0=\vbox{\hsize=\maxdimen
2433       \noindent\color@begingroup\csname #1footfmt\endcsname #2{#1}\
color@endgroup}%
2434     \setbox0=\hbox{\Xunvxh{0}{#1}}%
2435     \dp0=\z@
2436     \ht0=\csname #1footfudgefactor\endcsname\wd0
2437     \box0
2438     \penalty0
2439 }}
2440
2441 %
```

\Xunvxh 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.

```
2442 \newcommand*{\Xunvxh}[2]{%
2443   \setbox0=\vbox{\unvbox#1%
2444     \global\setbox1=\lastbox}%
2445   \unhbox1
2446   \unskip              % remove \rightskip,
2447   \unskip              % remove \parfillskip,
2448   \unpenalty           % remove \penalty of 10000,
2449   \hskip\csuse{Xafternote@#2}}   % but add the glue to go between the notes
```

```
2450
2451 %
```

\parafootfmt   \parafootfmt is \normalfootfmt adapted to do the special stuff needed for para-
graphed 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).

```
2452 \newcommand*{\parafootfmt}[4]{%
2453   \Xinsertparafootsep{#4}%
2454   \Xledsetnormalparstuff{#4}%
2455   \printlinefootnote{#1}{#4}%
2456   {\nottoggle{Xlemmadisablefontselection@#4}{\select@lemmafont#1|#2}{#2}}%
2457   \iftoggle{nosep@}{\hskip\csuse{Xinplaceoflemmaseparator@#4}}{\ifcsempty{
Xlemmaseparator@#4}%
2458     {\hskip\csuse{Xinplaceoflemmaseparator@#4}}%
2459     {\nobreak\hskip\csuse{Xbeforelemmaseparator@#4}\csuse{Xlemmaseparator@
#4}\hskip\csuse{Xafterlemmaseparator@#4}%
2460   }}%
2461   #3\penalty-10 }
2462 %
```

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

\parafootgroup   This footgroup code is modelled on the macros in *The TeXbook*, p. 399. The only differ-
ence 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 \Xnotefontsize@⟨s⟩ is to ensure that the correct \baselineskip for
the footnotes is used. The argument is the note series letter.

```
2463 \newcommand*{\parafootgroup}[1]{%
2464   \unvbox\csname #1footins\endcsname
2465   \ifcsstring{Xragged@#1}{L}{\RaggedLeft}{}%
2466   \ifcsstring{Xragged@#1}{R}{\RaggedRight}{}%
2467   \makehboxofhboxes
2468   \setbox0=\hbox{{\csuse{Xnotefontsize@#1}\csuse{Xtxtbeforenotes@#1}}\
unhbox0 \removehboxes}%
2469   \csuse{Xnotefontsize@#1}
2470   \noindent\unhbox0\par%
2471   \global\hsize=\old@hsize%
2472   }%
2473
2474 %
```

\mpparafootgroup   The minipage version.

```
2475 \newcommand*{\mpparafootgroup}[1]{{%
2476   \setXnoteswidthliketwocolumns@{#1}%
2477   \vskip\skip\@nameuse{mp#1footins}
2478   \ifl@dpairing\ifparledgroup%
2479     \leavevmode\marks\parledgroup@{begin}%
2480     \marks\parledgroup@series{#1}%
2481     \marks\parledgroup@type{Xfootnote}%
2482   \fi\fi\normalcolor
2483   \ifparledgroup%
2484     \ifl@dpairing%
2485     \else%
2486       \setXnoteswidthliketwocolumns@{#1}%
2487       \setXnotespositionliketwocolumns@{#1}%
2488       \print@Xfootnoterule{#1}%%
2489     \fi%
2490   \else%
2491       \setXnoteswidthliketwocolumns@{#1}%
2492       \setXnotespositionliketwocolumns@{#1}%
2493       \print@Xfootnoterule{#1}%
2494   \fi%
2495   \unvbox\csname mp#1footins\endcsname
2496   \ifcsstring{Xragged@#1}{L}{\RaggedLeft}{}%
2497   \ifcsstring{Xragged@#1}{R}{\RaggedRight}{}%
2498   \makehboxofhboxes
2499   \setbox0=\hbox{{\csuse{Xnotefontsize@#1}\csuse{Xtxtbeforenotes@#1}}\
unhbox0 \removehboxes}%
2500   \csuse{Xnotefontsize@#1}
2501   \noindent\unhbox0\par}}
2502
2503 %
```

And finally, the two macros which are required to transform the long horizontal box
stored in the insert' box to a printable text.

```
\makehboxofhboxes 04 \newcommand*{\makehboxofhboxes}{\setbox0=\hbox{}%
      \removehboxes 05   \loop
2506     \unpenalty
2507     \setbox2=\lastbox
2508   \ifhbox2
2509     \setbox0=\hbox{\box2\unhbox0}%
2510   \repeat}
2511
2512 \newcommand*{\removehboxes}{\setbox0=\lastbox
2513   \ifhbox0{\removehboxes}\unhbox0 \fi}
2514
2515 %
```

Insertion of the footnotes separator   The command \Xinsertparafootsep{⟨*series*⟩}
must be called at the beginning of \parafootftm.

```
\prevpage@num₁₆ \newcommand{\Xinsertparafootsep}[1]{%
\Xinsertparafootsep₁₇    \ifnumequal{\csuse{#1prevpage@num}}{\page@num}%
        2518         {\ifcsdef{prevline#1}% Be sur \prevline#1 exists.
        2519      {\ifnumequal{\csuse{prevline#1}}{\line@num}%
        2520        {\ifcsempty{Xsymlinenum@#1}{\csuse{Xparafootsep@#1}}{}}%
        2521        {\csuse{Xparafootsep@#1}}%
        2522      }%
        2523      {\csuse{Xparafootsep@#1}}%
        2524    }%
        2525    {}%
        2526    \global\csname #1prevpage@num\endcsname=\page@num%
        2527 }
        2528 %
```

XII.6.4   Columnar footnotes

Common tools

\rigidbalance
\dosplits
\splitoff
\@h
\@k

We will now define macros for three-column notes and two-column notes. Both sets of macros will use \rigidbalance, which splits a box (#1) into into a number (#2) of columns, each with a space (#3) between the top baseline and the top of the \vbox. The \rigidbalance macro is taken from *The TeXbook*, p. 397, with a slight change to the syntax of the arguments so that they do not 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.

```
        2529 \newcount\@k \newdimen\@h
        2530 \newcommand*{\rigidbalance}[3]{\setbox0=\box#1 \@k=#2 \@h=#3
        2531   \@@line{\splittopskip=\@h \vbadness=\@M \hfilneg
        2532   \valign{##\vfil\cr\dosplits}}}
        2533
        2534 \newcommand*{\dosplits}{\ifnum\@k>0 \noalign{\hfil}\splitoff
        2535   \global\advance\@k-1\cr\dosplits\fi}
        2536
        2537 \newcommand*{\splitoff}{\dimen0=\ht0
        2538   \divide\dimen0 by\@k \advance\dimen0 by\@h
        2539   \setbox2 \vsplit0 to \dimen0
        2540   \unvbox2 }
        2541
        2542 %
```

Three columns

\Xarrangement@threecol

```
        2543 \newcommand*{\Xarrangement@threecol}[1]{%
        2544   \csgdef{series@display#1}{threecol}
        2545   \expandafter\let\csname v#1footnote\endcsname=\threecolvfootnote
```

```
2546    \expandafter\let\csname #1footfmt\endcsname=\threecolfootfmt
2547    \expandafter\let\csname #1footgroup\endcsname=\threecolfootgroup
2548    \dimen\csname #1footins\endcsname=\csuse{Xmaxhnotes@#1}%
2549    \skip\csname #1footins\endcsname=\csuse{Xbeforenotes@#1}%
2550    \advance\skip\csname #1footins\endcsname by\csuse{Xafterrule@#1}%
2551    \threecolfootsetup{#1}
2552 %
```

The additional setup for minipages.

```
2553    \ifnoledgroup@\else
2554      \expandafter\let\csname mpv#1footnote\endcsname=\mpnormalvfootnote
2555      \expandafter\let\csname mp#1footgroup\endcsname=\mpthreecolfootgroup
2556      \skip\csname mp#1footins\endcsname=\csuse{Xbeforenotes@#1}%
2557      \advance\skip\csname mp#1footins\endcsname by\csuse{Xafterrule@#1}%
2558      \mpthreecolfootsetup{#1}
2559    \fi
2560 }
2561
2562 %
```

The \footstart and \footnoterule macros for these notes assume the normal values
(XII.6.2 p. 141 above).

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

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

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

```
2563 \newcommand*{\threecolfootsetup}[1]{%
2564    \count\csname #1footins\endcsname 333
2565    \csxdef{default@#1footins}{333}%Use this to confine the  notes to one
     side only
2566    \multiply\dimen\csname #1footins\endcsname \thr@@}
2567 %
```

\mpthreecolfootsetup   The setup for minipages.

```
2568 \newcommand*{\mpthreecolfootsetup}[1]{%
2569    \count\csname mp#1footins\endcsname 333
2570    \multiply\dimen\csname mp#1footins\endcsname \thr@@}
```

2571
2572 %

\threecolvfootnote   \threecolvfootnote is the \vfootnote command for three-column notes. The call
to \Xnotefontsize@⟨s⟩ 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 #1 the full text of the note (including
numbers, lemma and text).

```
2573 \notbool{parapparatus@}{\newcommand*}{\newcommand}{\threecolvfootnote}[2]{%
2574    \insert\csname #1footins\endcsname\bgroup
2575    \csuse{Xnotefontsize@#1}
2576    \footsplitskips
2577    \csname #1footfmt\endcsname #2{#1}\egroup}
2578 %
```

\threecolfootfmt   \threecolfootfmt is the command that formats one note. The arguments are #1 the
line numbers, #2 the lemma and #4 the text of the -footnote command #4 optional
(for backward compatibility): the series.

```
2579 \notbool{parapparatus@}{\newcommand*}{\newcommand}{\threecolfootfmt}[4]{%
2580    \normal@pars
2581    \hsize \csuse{Xhsizethreecol@#4}
2582    \nottoggle{Xparindent@#4}{\parindent=\z@}{}
2583    \tolerance=5000
2584    \hangindent=\csuse{Xhangindent@#4}
2585    \leavevmode
2586    \csuse{Xcolalign@#4}%
2587    \strut{\printlinefootnote{#1}{#4}}%
2588    {\nottoggle{Xlemmadisablefontselection@#4}{\select@lemmafont#1|#2}{#2}}%
2589    \iftoggle{nosep@}{\hskip\csuse{Xinplaceoflemmaseparator@#4}}{\ifcsempty{
Xlemmaseparator@#4}%
2590       {\hskip\csuse{Xinplaceoflemmaseparator@#4}}%
2591       {\nobreak\hskip\csuse{Xbeforelemmaseparator@#4}\csuse{Xlemmaseparator@
#4}\hskip\csuse{Xafterlemmaseparator@#4}%
2592    }}%
2593    #3\strut\par\allowbreak}
2594 %
```

\threecolfootgroup   And here is the footgroup macro that is called within the output routine to regroup the
notes into three columns. Once again, the call to \Xnotefontsize@⟨s⟩ 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 ouput 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.

```
2595  \newcommand*{\threecolfootgroup}[1]{{\csuse{Xnotefontsize@#1}%
2596    \noindent\csuse{Xtxtbeforenotes@#1}\par%
2597    \splittopskip=\ht\strutbox
2598    \expandafter
2599    \rigidbalance\csname #1footins\endcsname \thr@@ \splittopskip}}
2600  %
```

\mpthreecolfootgroup   The setup for minipages.

```
2601  \newcommand*{\mpthreecolfootgroup}[1]{{%
2602    \vskip\skip\@nameuse{mp#1footins}
2603    \ifl@dpairing\ifparledgroup%
2604      \leavevmode\marks\parledgroup@{begin}%
2605      \marks\parledgroup@series{#1}%
2606      \marks\parledgroup@type{Xfootnote}%
2607    \fi\fi\normalcolor
2608    \ifparledgroup%
2609      \ifl@dpairing%
2610      \else%
2611        \setXnoteswidthliketwocolumns@{#1}%
2612        \setXnotespositionliketwocolumns@{#1}%
2613        \print@Xfootnoterule{#1}%
2614      \fi%
2615    \else%
2616      \setXnoteswidthliketwocolumns@{#1}%
2617      \setXnotespositionliketwocolumns@{#1}%
2618      \print@Xfootnoterule{#1}%
2619    \fi%
2620    {\csuse{Xnotefontsize@#1}\noindent\csuse{Xtxtbeforenotes@#1}}\par
2621    \splittopskip=\ht\strutbox
2622    \expandafter
2623    \rigidbalance\csname mp#1footins\endcsname \thr@@ \splittopskip}}
2624
2625  %
```

Two columns

\Xarrangement@twocol
```
2626  \newcommand*{\Xarrangement@twocol}[1]{%
2627    \csgdef{series@display#1}{twocol}
2628    \expandafter\let\csname v#1footnote\endcsname=\twocolvfootnote
2629    \expandafter\let\csname #1footfmt\endcsname=\twocolfootfmt
2630    \expandafter\let\csname #1footgroup\endcsname=\twocolfootgroup
2631    \dimen\csname #1footins\endcsname=\csuse{Xmaxhnotes@#1}%
2632    \skip\csname #1footins\endcsname=\csuse{Xbeforenotes@#1}%
```

```
2633    \advance\skip\csname #1footins\endcsname by\csuse{Xafterrule@#1}%
2634    \twocolfootsetup{#1}
2635 %
```

The additional setup for minipages.

```
2636    \ifnoledgroup@\else
2637      \expandafter\let\csname mpv#1footnote\endcsname=\mpnormalvfootnote
2638      \expandafter\let\csname mp#1footgroup\endcsname=\mptwocolfootgroup
2639      \skip\csname mp#1footins\endcsname=\csuse{Xbeforenotes@#1}%
2640      \advance\skip\csname mp#1footins\endcsname by\csuse{Xafterrule@#1}%
2641      \mptwocolfootsetup{#1}
2642    \fi
2643 }
2644
2645 %
```

\twocolfootsetup   Here is a series of macros which are very similar to their three-column counterparts. In
\twocolvfootnote   this case, each note is assumed to contribute only a half a line of text. And the notes are
\twocolfootfmt   set in columns giving a gap between them of one tenth of the \hsize.
\twocolfootgroup
```
2646 \newcommand*{\twocolfootsetup}[1]{%
2647    \count\csname #1footins\endcsname 500
2648    \csxdef{default@#1footins}{500}%Use this to confine the  notes to one
     side only
2649    \multiply\dimen\csname #1footins\endcsname \tw@}
2650 %
```

```
2651 \notbool{parapparatus@}{\newcommand*}{\newcommand}{\twocolvfootnote}[2]{\
     insert\csname #1footins\endcsname\bgroup
2652    \csuse{Xnotefontsize@#1}
2653    \footsplitskips
2654    \csname #1footfmt\endcsname #2{#1}\egroup}
2655 %
```

```
2656 \notbool{parapparatus@}{\newcommand*}{\newcommand}{\twocolfootfmt}[4]{% 4th
      arg is optional, for backward compatibility
2657    \normal@pars
2658    \hsize \csuse{Xhsizetwocol@#4}
2659    \nottoggle{Xparindent@#4}{\parindent=\z@}{}
2660    \tolerance=5000
2661    \hangindent=\csuse{Xhangindent@#4}
2662    \leavevmode
2663    \csuse{Xcolalign@#4}%
2664    \strut{\printlinefootnote{#1}{#4}}%
2665    {\nottoggle{Xlemmadisablefontselection@#4}{\select@lemmafont#1|#2}{#2}}%
2666    \iftoggle{nosep@}{\hskip\csuse{Xinplaceoflemmaseparator@#4}}{\ifcsempty{
     Xlemmaseparator@#4}%
2667      {\hskip\csuse{Xinplaceoflemmaseparator@#4}}%
2668      {\nobreak\hskip\csuse{Xbeforelemmaseparator@#4}\csuse{Xlemmaseparator@
     #4}\hskip\csuse{Xafterlemmaseparator@#4}%
```

```
2669   }}%
2670   #3\strut\par\allowbreak}
2671 %

2672 \newcommand*{\twocolfootgroup}[1]{{\csuse{Xnotefontsize@#1}
2673   \noindent\csuse{Xtxtbeforenotes@#1}\par%
2674   \splittopskip=\ht\strutbox
2675   \expandafter
2676   \rigidbalance\csname #1footins\endcsname \tw@ \splittopskip}}

2677
2678 %
```

\mptwocolfootsetup The versions for minipages.
\mptwocolfootgroup

```
2679 \newcommand*{\mptwocolfootsetup}[1]{%
2680   \count\csname mp#1footins\endcsname 500
2681   \multiply\dimen\csname mp#1footins\endcsname \tw@}
2682 %

2683 \newcommand*{\mptwocolfootgroup}[1]{{%
2684   \vskip\skip\@nameuse{mp#1footins}
2685   \ifl@dpairing\ifparledgroup%
2686     \leavevmode\marks\parledgroup@{begin}%
2687     \marks\parledgroup@series{#1}%
2688     \marks\parledgroup@type{Xfootnote}%
2689   \fi\fi\normalcolor
2690   \ifparledgroup%
2691     \ifl@dpairing%
2692     \else%
2693       \setXnoteswidthliketwocolumns@{#1}%
2694       \setXnotespositionliketwocolumns@{#1}%
2695       \print@Xfootnoterule{#1}%
2696     \fi%
2697   \else%
2698     \setXnoteswidthliketwocolumns@{#1}%
2699     \setXnotespositionliketwocolumns@{#1}%
2700     \print@Xfootnoterule{#1}%
2701   \fi%
2702    {\csuse{Xnotefontsize@#1}\noindent\csuse{Xtxtbeforenotes@#1}}\par
2703   \splittopskip=\ht\strutbox
2704   \expandafter
2705   \rigidbalance\csname mp#1footins\endcsname \tw@ \splittopskip}}

2706
2707 %
```

XII.7   Critical notes presentation

Here, we define some commons macro which are used in order to print a critical notes, that is a note with 1) line number 2) lemma 3) lemma separator 4) text associated to the lemma.

XII.7.1   Font tools

\endashchar
\fullstop
\rbracket

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

The \endashchar macro is simply an en-dash from the normal font and is immune to changes in the surrounding font. The same goes for the full stop. These two are used in \printlines. The right bracket macro is the same again; it crops up in \normalfootfmt and the other footnote macros for controlling the format of the footnotes.

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

```
2708 \def\endashchar{\textnormal{--}}
2709 \newcommand*{\fullstop}{\textnormal{.}}
2710 \newcommand*{\rbracket}{\textnormal{%
2711     \csuse{text\csuse{footnote@lang}}{%
2712             \ifluatex%
2713             \ifdefstring{\footnote@luatextextdir}{TRT}{\thinspace[}{\thinspace
]}%
2714                 \else%
2715             \thinspace]%
2716             \fi}%
2717     }%
2718 }
2719
2720 %
```

XII.7.2   Pstart number in footnote

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

```
2721 \newcommand{\printpstart}[0]{%
2722     \ifboolexpr{bool{l@dpairing} or bool{l@dprintingpages} or bool{
l@dprintingcolumns}}{%
2723             \ifledRcol%
2724                 \thepstartR%
2725             \else%
2726                 \thepstartL%
2727             \fi%
2728     }{%
2729             \thepstart%
2730     }%
2731 }
2732 %
```

XII.7.3   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 series options. Its first argument is the information about lines; its second is the series of the footnote. The printing of the line number is shared in \printlinefootnotenumbers.

```
2733  \newcommand{\printlinefootnote}[2]{%
2734      \def\extractline@##1|##2|##3|##4|##5|##6|##7|{##2}%
2735      \def\extractsubline@##1|##2|##3|##4|##5|##6|##7|{##3}%
2736      \def\extractendline@##1|##2|##3|##4|##5|##6|##7|{##5}%
2737      \def\extractendsubline@##1|##2|##3|##4|##5|##6|##7|{##6}%
2738      \iftoggle{Xnumberonlyfirstintwolines@#2}{%
2739          \edef\lineinfo@{\extractline@ #1| - \extractsubline@ #1| - \
       extractendline@ #1| - \extractendsubline@ #1|}%
2740          }%
2741          {%
2742          \edef\lineinfo@{\extractline@ #1| - \extractsubline@ #1|}%
2743          }%
2744      \iftoggle{nonum@}{%Try if the line number must printed for this specific
       not (by default, yes)
2745      \hspace{\csuse{Xinplaceofnumber@#2}}%
2746      }%
2747      {%
2748          {%
2749              \iftoggle{Xnonumber@#2}%Try if the line number must printed (by
       default, yes)
2750              {%
2751              \hspace{\csuse{Xinplaceofnumber@#2}}%
2752              }%
2753              {%
2754                  {\iftoggle{Xnumberonlyfirstinline@#2}% If for this series the
       line number must be printed only in the first time.
2755                      {%
2756                      \ifcsdef{prevline#2}%
2757                          {%Be sure the \prevline exists.
2758                          \ifcsequal{prevline#2}{lineinfo@}%Try it
2759                              {%
2760                              \ifcsempty{Xsymlinenum@#2}%
2761                              {%
2762                                  \hspace{\csuse{Xinplaceofnumber@#2}}%
2763                              }%
2764                              {\hspace{\csuse{Xbeforesymlinenum@#2}}\csuse{
       Xnotenumfont@#2}%
2765                              \ifdimequal{\csuse{Xboxsymlinenum@#2}}{0pt}%
2766                                  {\csuse{Xsymlinenum@#2}}%
2767                                  {\hbox to \csuse{Xboxsymlinenum@#2}{\csuse{
       Xsymlinenum@#2}\hfill}}%
2768                              \hspace{\csuse{Xaftersymlinenum@#2}}}%
2769                              }%
```

```
2770                        {%
2771                          \printlinefootnotearea{#1}{#2}%
2772                          }%
2773                        }%
2774                        {%
2775                          \printlinefootnotearea{#1}{#2}%
2776                          }%
2777                    }%
2778                    {%
2779                      \printlinefootnotearea{#1}{#2}%
2780                      }%
2781                      \csxdef{prevline#2}{\lineinfo@}%
2782                }%
2783            }%
2784        }%
2785      }%
2786 }
2787 %
```

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

```
2788 \newcommand{\printlinefootnotearea}[2]{%
2789   \printXbeforenumber{#2}%
2790   \csuse{Xnotenumfont@#2}%
2791   \boxfootnotenumbers{#1}{#2}%
2792   \printXafternumber{#2}%
2793 }%
2794 %
```

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

```
2795 \newcommand{\boxfootnotenumbers}[2]{%
2796   \ifdimequal{\csuse{Xboxlinenum@#2}}{0pt}{%
2797        \printlinefootnotenumbers{#1}{#2}%
2798      }%
2799      {%
2800        \hbox to \csuse{Xboxlinenum@#2}%
2801          {%
2802          \IfSubStr{RC}{\csuse{Xboxlinenumalign@#2}}{\hfill}{}%
2803          \printlinefootnotenumbers{#1}{#2}%
2804          \IfSubStr{LC}{\csuse{Xboxlinenumalign@#2}}{\hfill}{}%
2805          }%
2806      }%
2807 }%
2808 %
```

\printlinefootnotenumbers This macro prints, if needed, the pstart number and the line number. The first argument is line information, the second is the notes series (A, B, C, etc.) The previous \boxlinefootnote calls it.

```
2809 \newcommand{\printlinefootnotenumbers}[2]{%
2810   \xdef\@currentseries{#2}%
2811   \ifboolexpr{%
2812     (togl{Xpstart@#2} and bool{numberpstart})%
2813     or togl{Xpstarteverytime@#2}}%
2814   {\printpstart}{}%
2815   \iftoggle{Xonlypstart@#2}{}{\printlines#1|}%
2816 }%
2817 %
```

\printXbeforenumber This macro prints a space (before the line number) in footnote. It is called by \printlinefootnotearea. Its only argument is the note series (A, B, C, etc.)

```
2818 \newcommand{\printXbeforenumber}[1]{%
2819   \hspace{\csuse{Xbeforenumber@#1}}%
2820 }%
2821 %
```

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

```
2822 \newcommand{\printXafternumber}[1]{%
2823   \iftoggle{Xnonbreakableafternumber@#1}{\nobreak}{}%
2824   \hspace{\csuse{Xafternumber@#1}}%
2825 }%
2826 %
```

If we have decided to print the line number in a specific notes, 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 V.9 p. 80: 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.

edmac' creator have defined six boolean in order to know which component of line number description we have to print:

- \ifl@d@pnum for page numbers;

- \ifl@d@ssub for starting sub-line;

- \ifl@d@elin for ending line;

- \ifl@d@esl for ending sub-line; and

- \ifl@d@dash for the dash between the starting and ending groups.

There is no boolean for the line number because it is always printed.

Maïeul Rouquette has added `\ifl@d@Xtwolines` and `\ifl@d@Xmorethantwolines` to print a symbol which stands for "and subsequent" when there are two, three or more lines.

```
\ifl@d@pnum                2827  \newif\ifl@d@pnum
\ifl@d@ssub                2828  \newif\ifl@d@ssub
\ifl@d@elin                2829  \newif\ifl@d@elin
\ifl@d@esl                 2830  \newif\ifl@d@esl
\ifl@d@dash                2831  \newif\ifl@d@dash
\ifl@d@Xtwolines           2832  \newif\ifl@d@Xtwolines%
\ifl@d@Xmorethantwolines   2833  \newif\ifl@d@Xmorethantwolines%
                           2834  %
```

`\l@dparsefootspec` `\l@dparsefootspec{⟨spec⟩}{⟨lemma⟩}{⟨text⟩}` parses a footnote specification. ⟨*lemma*⟩
`\l@dp@rsefootspec` and ⟨*text*⟩ are the lemma and text respectively.   ⟨*spec*⟩ is the line and page num-
`\l@dparsedstartpage` ber and lemma font specifier in `\l@d@nums` style format.   The real work is done by
`\l@dparsedstartline` `\l@dp@rsefootspec` which defines macros holding the numeric values. Just a reminder
`\l@dparsedstartsub` of the arguments:
`\l@dparsedendpage`
`\l@dparsedendline`
`\l@dparsedendsub`

```
\printlines    #1    | #2 |   #3    |   #4    | #5 |   #6    |  #7
\printlines start-page | line | subline | end-page | line | subline | font
```

```
2835  \newcommand*{\l@dparsefootspec}[3]{\l@dp@rsefootspec#1|}
2836  \def\l@dp@rsefootspec#1|#2|#3|#4|#5|#6|#7|{%
2837    \gdef\l@dparsedstartpage{#1}%
2838    \gdef\l@dparsedstartline{#2}%
2839    \gdef\l@dparsedstartsub{#3}%
2840    \gdef\l@dparsedendpage{#4}%
2841    \gdef\l@dparsedendline{#5}%
2842    \gdef\l@dparsedendsub{#6}%
2843  }
2844  %
```

Initialise the several number value macros.

```
2845  \def\l@dparsedstartpage{0}%
2846  \def\l@dparsedstartline{0}%
2847  \def\l@dparsedstartsub{0}%
2848  \def\l@dparsedendpage{0}%
2849  \def\l@dparsedendline{0}%
2850  \def\l@dparsedendsub{0}%
2851
2852  %
```

`\setprintlines` The macro `\setprintlines` does the work of deciding what numbers should be printed.
Its arguments are the same as the first 6 of `\printlines`.

```
2853  \newcommand*{\setprintlines}[6]{%
2854    \l@d@pnumfalse \l@d@dashfalse
2855  %
```

We print the page numbers only if: 1) we are doing the lineation by page, and 2) the ending page number is different from the starting page number.a

```
2856    \ifbypage@
2857        \ifnum#4=#1 \else
2858            \l@d@pnumtrue
2859            \l@d@dashtrue
2860        \fi
2861    \fi
2862 %
```

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

```
2863    \ifl@d@pnum \l@d@elintrue \else \l@d@elinfalse \fi
2864    \ifnum#2=#5 \else
2865        \l@d@elintrue
2866        \l@d@dashtrue
2867    \fi
2868 %
```

We print the starting sub-line if it is nonzero.

```
2869    \l@d@ssubfalse
2870    \ifnum#3=0 \else
2871        \l@d@ssubtrue
2872    \fi
2873 %
```

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

```
2874    \l@d@eslfalse
2875    \ifnum#6=0 \else
2876        \ifnum#6=#3
2877            \ifl@d@elin \l@d@esltrue \else \l@d@eslfalse \fi
2878        \else
2879            \l@d@esltrue
2880            \l@d@dashtrue
2881        \fi
2882    \fi%
2883 %
```

However, if the \Xtwolines is set for the current series, we do not print the last line number.

```
2884    \ifl@d@dash%
2885      \ifboolexpr{togl{fulllines@} or test{\ifcsempty{Xtwolines@\
       @currentseries}}}%
2886          {}%
2887          {%
2888          \setistwofollowinglines{#1}{#2}{#4}{#5}%
2889            \ifboolexpr{%
```

```
2890            (%
2891                togl {Xtwolinesbutnotmore@\@currentseries}%
2892                and not%
2893                  (%
2894                   bool {istwofollowinglines@}%
2895                  )%
2896            )%
2897            or%
2898            (%
2899               (not test{\ifnumequal{#1}{#4}})%
2900                and togl{Xtwolinesonlyinsamepage@\@currentseries}%
2901              )%
2902          }%
2903          {}%
2904          {%
2905          \l@d@dashfalse%
2906          \l@d@Xtwolinestrue%
2907          \l@d@elinfalse%
2908          \l@d@eslfalse%
2909          \ifcsempty{Xmorethantwolines@\@currentseries}%
2910            {}%
2911            {\ifistwofollowinglines@\else%
2912              \l@d@Xmorethantwolinestrue%
2913             \fi%
2914            }%
2915          }%
2916      }%
2917   \fi%
2918 %
```

End of \setprintlines.

```
2919 }%
2920 %
```

\setistwofollowinglines   The \ifistwofollowinglines boolean, used by the \Xtwolines and related setting, is set to true by \setistwofollowinglines. This command takes the following arguments:

- #1 First page number.

- #2 First line number.

- #3 Last page number.

- #4 Last line number.

If #3-#2 = 1, then that means the two lines are subsequent, and consequently \ifistwofollowinglines is set to true. However, if we use lineation by page, two given lines can be subsequent if:

- The first line number is equal to the last line number of the first page.

- The last line number is equal to 1.

- #3-#1 is equal to 1.

```
2921 \newif\ifistwofollowinglines@%
2922 \newcommand{\setistwofollowinglines}[4]{%
2923     \ifcsdef{lastlinenumberon@#1}%
2924       {\numdef{\tmp}{\csuse{lastlinenumberon@#1}}}%
2925       {\numdef{\tmp}{0}}%
2926     \istwofollowinglines@false%
2927     \ifnumequal{#4-#2}{1}%
2928     {\istwofollowinglines@true}%
2929     {\ifbypage@%
2930       \ifnumequal{#3-#1}{1}%
2931       {%
2932       \ifnumequal{#2}{\tmp}%
2933         {\ifnumequal{#4}{1}{\istwofollowinglines@true}{}}%
2934         {}%
2935       }%
2936       {}%
2937      \fi%
2938     }%
2939 }%
2940 %
```

\printlines   So, we have decided which part of line number sets will be printed depending of these
value. Now we are ready to print them. If the lineation is by pstart, we print the pstart.

```
2941 \def\printlines#1|#2|#3|#4|#5|#6|#7|{%
2942     \begingroup%
2943 %
```

If we use LuaTeX, ensure we use good text's direction.

```
2944     \ifluatex%
2945       \luatextextdir TLT%
2946     \fi%
2947 %
```

Decide which part of line number components we will print.

```
2948     \setprintlines{#1}{#2}{#3}{#4}{#5}{#6}%
2949 %
```

One subtlety left here is when to print a period between numbers. But the only instance
in which this is tricky is for the ending sub-line number: it could come after the starting
sub-line number (in which case we want only the dash) or after an ending line number
(in which case we need to insert a period). So, first, print the start line number.

```
2950     \ifdimequal{\csuse{Xboxstartlinenum@\@currentseries}}{0pt}%
2951       {\bgroup}%
```

```
2952     {\leavevmode\hbox to \csuse{Xboxstartlinenum@\@currentseries}\bgroup\
     hfill}%
2953   \ifl@d@pnum #1\fullstop\fi
2954   \linenumrep{#2}
2955   \ifl@d@ssub \fullstop \sublinenumrep{#3}\fi
2956   \egroup%
2957 %
```

Then print the dash + end line number, or the range symbol.

```
2958   \ifdimequal{\csuse{Xboxendlinenum@\@currentseries}}{0pt}%
2959     {\bgroup}%
2960     {\hbox to \csuse{Xboxendlinenum@\@currentseries}\bgroup}%
2961   \ifl@d@Xtwolines%
2962     \ifl@d@Xmorethantwolines%
2963       \csuse{Xmorethantwolines@\@currentseries}%
2964     \else%
2965       \csuse{Xtwolines@\@currentseries}%
2966     \fi%
2967   \else%
2968     \ifl@d@dash \endashchar\fi%
2969     \ifl@d@pnum #4\fullstop\fi%
2970     \ifl@d@elin \linenumrep{#5}\fi%
2971     \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumrep{#6}\fi%
2972   \fi%
2973   \ifdimequal{\csuse{Xboxendlinenum@\@currentseries}}{0pt}%
2974     {}%
2975     {\hfill}%Prevent underfull hbox
2976   \egroup%
2977   \endgroup%
2978 }%
2979 %
```

XIII   Familiar footnotes

XIII.1   Adjacent footnotes

The original edmac provided users with five series of critical footnotes (\Afootnote \Bfootnote \Cfootnote \Dfootnote \Efootnote), and LaTeX provides a single numbered footnote. The reledmac package uses the edmac mechanism to provide six 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 seemed to Peter Wilson such a useful ability that it was provided automatically by eledmac.

Maïeul Rouquette has maintained this feature in reledmac, despite he thought that is not directly in relationship with the aim of reledmac.

\multiplefootnotemarker These macros may have been defined by the memoir class, are provided by the footmisc
\multfootsep package and perhaps by other footnote packages. That is why we use \providecommand

and not \newcommand.

```
2980 \providecommand*{\multiplefootnotemarker}{3sp}
2981 \providecommand*{\multfootsep}{\textsuperscript{\normalfont,}}
2982
2983 %
```

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

```
2984 \providecommand*{\m@mmf@prepare}{%
2985   \kern-\multiplefootnotemarker
2986   \kern\multiplefootnotemarker\relax}
2987 %
```

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

```
2988 \providecommand*{\m@mmf@check}{%
2989   \ifdim\lastkern=\multiplefootnotemarker\relax
2990     \edef\@x@sf{\the\spacefactor}%
2991     \unkern
2992     \multfootsep
2993     \spacefactor\@x@sf\relax
2994   \fi}
2995
2996 %
```

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

```
2997 \@ifclassloaded{memoir}{}{%
2998 %
```

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

```
2999 \apptocmd{\@footnotetext}{\m@mmf@prepare}{}{}
3000 %
```

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

```
3001
3002 \patchcmd{\@footnotemark}
3003   {\nobreak}
3004   {\m@mmf@check
3005   \nobreak
3006   }
3007   {}{}
3008 \patchcmd{\@footnotemark}
3009   {\@makefnmark}
```

```
3010    {\@makefnmark
3011    \m@mmf@prepare
3012    }
3013    {}{}
3014 %
```

Finished the modifications for the non-memoir case.

```
3015 }
3016
3017 %
```

XIII.2   Regular footnotes for numbered texts

\l@doldold@footnotetext   In order to enable the regular \footnotes in numbered text we have to play around
\@footnotetext   with its \@footnotetext, using different forms for when in numbered or regular text.

```
3018 \pretocmd{\@footnotetext}{%
3019    \ifnumberedpar@
3020       \edtext{}{\l@dbfnote{#1}}%
3021    \else
3022    }{}{}
3023 \apptocmd{\@footnotetext}{\fi}{}{}%
3024 %
```

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

```
3025
3026 \newcommand{\l@dbfnote}[1]{%
3027    \ifnumberedpar@
3028    \gdef\@tag{#1\relax}%
3029       \ifledRcol%
3030          \xright@appenditem{\noexpand\vl@dbfnote{{\expandonce\@tag}}{\
     @thefnmark}}%
3031                              \to\inserts@listR
3032          \global\advance\insert@countR \@ne%
3033       \else%
3034          \xright@appenditem{\noexpand\vl@dbfnote{{\expandonce\@tag}}{\
     @thefnmark}}%
3035                              \to\inserts@list
3036          \global\advance\insert@count \@ne%
3037       \fi
3038    \fi\ignorespaces}
3039
3040 \newcommand{\vl@dbfnote}[2]{%
3041    \def\@thefnmark{#2}%
3042    \@footnotetext{#1}%
```

```
3043    }%
3044 %
```

XIII.3   Footnote formats

Some of the code for the various formats is remarkably similar to that in section ??.
     The following macros generally set things up for the 'standard' footnote format.

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

```
3045 \newcommand*{\prebodyfootmark}{%
3046    \leavevmode
3047    \ifhmode
3048      \edef\@x@sf{\the\spacefactor}%
3049      \m@mmf@check
3050      \nobreak
3051    \fi}
3052 \newcommand{\postbodyfootmark}{%
3053    \m@mmf@prepare
3054    \ifhmode\spacefactor\@x@sf\fi\relax}
3055
3056 %
```

XIII.4   Footnote arrangement

XIII.4.1   User level macro

\arrangementX   \arrangementX[⟨s⟩]{⟨*arrangement*⟩} command calls, for each series, a specific com-
     mand which set many counters and commands in order to define specific arrangement.

```
3057 \newcommandx{\arrangementX}[2][1,usedefault]{%
3058    \def\do##1{%
3059      \csname arrangementX@#2\endcsname{##1}%
3060    }%
3061    \ifstrempty{#1}%
3062      {%
3063      \dolistloop{\@series}%
3064      }%
3065      {
3066      \docsvlist{#1}%
3067      }%
3068 }%
3069 %
```

XIII.4.2   Normal footnotes

\normal@footnotemarkX   \normal@footnotemarkX{⟨*series*⟩} sets up the typesetting of the marker at the point
     where the footnote is called for.

```
3070 \newcommand*{\normal@footnotemarkX}[1]{%
3071   \prebodyfootmark
3072   \@nameuse{bodyfootmark#1}%
3073   \postbodyfootmark}
3074
3075 %
```

\normalbodyfootmarkX   The \normalbodyfootmarkX{⟨*series*⟩} *really* typesets the in-text marker. The style is
the normal superscript.

```
3076 \newcommand*{\normalbodyfootmarkX}[1]{%
3077   \hbox{\textsuperscript{\normalfont\@nameuse{@thefnmark#1}}}}
3078 %
```

\normalvfootnoteX   \normalvfootnoteX{⟨*series*⟩}{⟨*text*⟩} does the \insert for the ⟨*series*⟩ and calls the
series' \footfmt... to format the ⟨*text*⟩.

```
3079 \notbool{parapparatus@}{\newcommand*}{\newcommand}{\normalvfootnoteX}[2]{%
3080   \insert\@nameuse{footins#1}\bgroup
3081     \csuse{bhooknoteX@#1}
3082     \csuse{notefontsizeX@#1}
3083     \footsplitskips
3084     \ifl@dpairing\ifl@dpaging\else%
3085       \setnoteswidthliketwocolumnsX@{#1}%
3086     \fi\fi%
3087     \setnotesXpositionliketwocolumns@{#1}%
3088     \spaceskip=\z@skip \xspaceskip=\z@skip
3089     \csuse{\csuse{footnote@dir}}\@nameuse{footfmt#1}{#1}{#2}\egroup}
3090
3091 %
```

\mpnormalvfootnoteX   The minipage version.

```
3092 \newcommand*{\mpnormalvfootnoteX}[2]{%
3093   \global\setbox\@nameuse{mpfootins#1}\vbox{%
3094     \unvbox\@nameuse{mpfootins#1}
3095     \csuse{bhooknoteX@#1}
3096     \csuse{notefontsizeX@#1}
3097     \hsize\columnwidth
3098     \@parboxrestore
3099     \color@begingroup
3100     \@nameuse{footfmt#1}{#1}{#2}\color@endgroup}}
3101
3102 %
```

\normalfootfmtX   \normalfootfmtX{⟨*series*⟩}{⟨*text*⟩} typesets the footnote text, prepended by the marker.

```
3103 \notbool{parapparatus@}{\newcommand*}{\newcommand}{\normalfootfmtX}[2]{%
3104  \ifluatex%
3105     \luatextextdir\footnote@luatextextdir%
```

```
3106        \luatexpardir\footnote@luatexpardir%
3107        \par%
3108     \fi%
3109       \protected@edef\@currentlabel{%
3110           \@nameuse{@thefnmark#1}%
3111       }%
3112     \ledsetnormalparstuffX{#1}%
3113     \hangindent=\csuse{hangindentX@#1}%
3114     {{\csuse{notenumfontX@#1}\@nameuse{footfootmark#1}}\strut%
3115        #2\strut\par}}
3116
3117  %
```

\normalfootfootmarkX   \normalfootfootmarkX{⟨*series*⟩} is called by \normalfootfmtX to typeset the foot-
note marker in the footer before the footnote text.

```
3118  \newcommand*{\normalfootfootmarkX}[1]{%
3119    \textsuperscript{\@nameuse{@thefnmark#1}}}
3120
3121  %
```

\normalfootstartX   \normalfootstartX{⟨*series*⟩} is the ⟨*series*⟩ footnote starting macro used in the output
routine.

```
3122  \newcommand*{\normalfootstartX}[1]{%
3123      \ifdimequal{0pt}{\prenotesX@}{}%
3124         {%
3125         \iftoggle{prenotesX@}{%
3126             \togglefalse{prenotesX@}%
3127             \skip\csname footins#1\endcsname=%
3128                \dimexpr\csuse{prenotesX@}+\csuse{afterruleX@#1}\relax%
3129             }%
3130            {}%
3131         }%
3132     \vskip\skip\csname footins#1\endcsname%
3133     \leftskip=\z@
3134     \rightskip=\z@
3135     \ifl@dpairing\else%
3136        \hsize=\old@hsize%
3137     \fi%
3138     \setnoteswidthliketwocolumnsX@{#1}%
3139     \setnotesXpositionliketwocolumns@{#1}%
3140     \print@footnoteXrule{#1}%
3141  }%
3142
3143  %
```

\normalfootnoteruleX   The rule drawn before the footnote series group.

```
3144 \let\normalfootnoteruleX=\footnoterule
3145
3146 %
```

\normalfootgroupX	\normalfootgroupX{⟨series⟩} sends the contents of the ⟨series⟩ insert box to the output page without alteration.

```
3147 \newcommand*{\normalfootgroupX}[1]{%
3148   \unvbox\@nameuse{footins#1}%
3149   \hsize=\old@hsize%
3150   }%
3151
3152 %
```

\mpnormalfootgroupX	The minipage version.

```
3153 \newcommand*{\mpnormalfootgroupX}[1]{%
3154   \vskip\skip\@nameuse{mpfootins#1}
3155   \ifl@dpairing\ifparledgroup%
3156     \leavevmode\marks\parledgroup@{begin}%
3157     \marks\parledgroup@series{#1}%
3158     \marks\parledgroup@type{footnoteX}%
3159   \fi\fi\normalcolor
3160   \ifparledgroup%
3161     \ifl@dpairing%
3162     \else%
3163       \setnoteswidthliketwocolumnsX@{#1}%
3164       \setnotesXpositionliketwocolumns@{#1}%
3165       \print@footnoteXrule{#1}%
3166     \fi%
3167   \else%
3168     \setnoteswidthliketwocolumnsX@{#1}%
3169     \setnotesXpositionliketwocolumns@{#1}%
3170     \print@footnoteXrule{#1}%
3171   \fi%
3172   \unvbox\@nameuse{mpfootins#1}}
3173
3174 %
```

\normalbfnoteX

```
3176 \newcommand{\normalbfnoteX}[2]{%
3177   \ifnumberedpar@
3178     \ifledRcol%
3179       \ifluatex
3180         \footnotelang@lua[R]%
3181       \fi
3182       \@ifundefined{xpg@main@language}%if polyglossia
3183         {}%
3184         {\footnotelang@poly[R]}%
```

```
3185        \protected@xdef\thisfootnote{\csuse{thefootnote#1}}%
3186        \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\expandonce\
      thisfootnote}}%
3187                          \to\inserts@listR
3188        \global\advance\insert@countR \@ne%
3189      \else%
3190        \ifluatex
3191            \footnotelang@lua%
3192        \fi
3193        \@ifundefined{xpg@main@language}%if polyglossia
3194          {}%
3195          {\footnotelang@poly}%
3196        \protected@xdef\thisfootnote{\csuse{thefootnote#1}}%
3197        \xright@appenditem{\noexpand\vbfnoteX{#1}{#2}{\expandonce\
      thisfootnote}}%
3198                          \to\inserts@list
3199        \global\advance\insert@count \@ne%
3200      \fi
3201    \fi\ignorespaces}
3202
3203 %
```

\vbfnoteX
```
3204 \newcommand{\vbfnoteX}[3]{%
3205    \@namedef{@thefnmark#1}{#3}%
3206    \@nameuse{regvfootnote#1}{#1}{#2}}
3207
3208 %
```

\vnumfootnoteX
```
3209 \newcommand{\vnumfootnoteX}[2]{%
3210    \ifnumberedpar@
3211      \edtext{}{\normalbfnoteX{#1}{#2}}%
3212    \else
3213      \@nameuse{regvfootnote#1}{#1}{#2}%
3214    \fi}
3215
3216 %
```

arrangementX@normal \arrangementX@normal{⟨*series*⟩} initialises the settings for the ⟨*series*⟩ footnotes. This should always be called for each series.

```
3217 \newcommand*{\arrangementX@normal}[1]{%
3218    \csgdef{series@displayX#1}{normal}
3219    \expandafter\let\csname footstart#1\endcsname=\normalfootstartX
3220    \expandafter\newcount\csname prevpage#1@num\endcsname
3221    \@namedef{@footnotemark#1}{\normal@footnotemarkX{#1}}
3222    \@namedef{bodyfootmark#1}{\normalbodyfootmarkX{#1}}
3223    \expandafter\let\csname regvfootnote#1\endcsname=\normalvfootnoteX
3224    \expandafter\let\csname vfootnote#1\endcsname=\vnumfootnoteX
```

```
3225    \expandafter\let\csname footfmt#1\endcsname=\normalfootfmtX
3226    \@namedef{footfootmark#1}{\normalfootfootmarkX{#1}}
3227    \expandafter\let\csname footgroup#1\endcsname=\normalfootgroupX
3228    \expandafter\let\csname footnoterule#1\endcsname=\normalfootnoteruleX
3229    \count\csname footins#1\endcsname=1000
3230    \csxdef{default@footins#1}{1000}%Use to have note only for one side
3231    \dimen\csname footins#1\endcsname=\csuse{maxhnotesX@#1}
3232    \skip\csname footins#1\endcsname=\csuse{beforenotesX@#1}%
3233    \advance\skip\csname footins#1\endcsname by\csuse{afterruleX@#1}%
3234 %
```

Additions for minipages.

```
3235    \ifnoledgroup@\else%
3236      \expandafter\let\csname mpvfootnote#1\endcsname=\mpnormalvfootnoteX
3237      \expandafter\let\csname mpfootgroup#1\endcsname=\mpnormalfootgroupX
3238      \count\csname mpfootins#1\endcsname=1000
3239      \dimen\csname mpfootins#1\endcsname=\csuse{maxhnotesX@#1}
3240      \skip\csname mpfootins#1\endcsname=\csuse{beforenotesX@#1}%
3241      \advance\skip\csname mpfootins#1\endcsname by\csuse{afterruleX@#1}%
3242    \fi
3243 }
3244
3245 %
```

XIII.4.3   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.

\arrangementX@twocol
```
3246 \newcommand*{\arrangementX@twocol}[1]{%
3247    \csgdef{series@displayX#1}{twocol}
3248    \expandafter\let\csname regvfootnote#1\endcsname=\twocolvfootnoteX
3249    \expandafter\let\csname footfmt#1\endcsname=\twocolfootfmtX
3250    \expandafter\let\csname footgroup#1\endcsname=\twocolfootgroupX
3251    \dimen\csname footins#1\endcsname=\csuse{maxhnotesX@#1}%
3252    \skip\csname footins#1\endcsname=\csuse{beforenotesX@#1}%
3253    \advance\skip\csname footins#1\endcsname by \csuse{afterruleX@#1}\relax%
3254    \twocolfootsetupX{#1}
3255    \ifnoledgroup@\else%
3256      \expandafter\let\csname mpvfootnote#1\endcsname=\mpnormalvfootnoteX
3257      \expandafter\let\csname mpfootgroup#1\endcsname=\mptwocolfootgroupX
3258      \skip\csname mpfootins#1\endcsname=\csuse{beforenotesX@#1}%
3259      \advance\skip\csname mpfootins#1\endcsname by\csuse{afterruleX@#1}
3260      \mptwocolfootsetupX{#1}
3261    \fi%
3262 }
3263
3264 %
```

`\twocolfootsetupX`
`\mptwocolfootsetupX`

`\twocolfootsetupX{⟨series⟩}`

```
3265 \newcommand*{\twocolfootsetupX}[1]{%
3266   \count\csname footins#1\endcsname 500
3267   \csxdef{default@footins#1}{500}%Use this to confine the  notes to one
      side only
3268   \multiply\dimen\csname footins#1\endcsname by \tw@}
3269 \newcommand*{\mptwocolfootsetupX}[1]{%
3270   \count\csname mpfootins#1\endcsname 500
3271   \multiply\dimen\csname mpfootins#1\endcsname by \tw@}
3272
3273 %
```

`\twocolvfootnoteX`

`\twocolvfootnoteX{⟨series⟩}`

```
3274 \notbool{parapparatus@}{\newcommand*}{\newcommand}{\twocolvfootnoteX}[2]{%
3275   \insert\csname footins#1\endcsname\bgroup
3276     \csuse{notefontsizeX@#1}
3277     \footsplitskips
3278     \spaceskip=\z@skip \xspaceskip=\z@skip
3279     \@nameuse{footfmt#1}{#1}{#2}\egroup}
3280
3281 %
```

`\twocolfootfmtX`

`\twocolfootfmtX{⟨series⟩}`

```
3282 \notbool{parapparatus@}{\newcommand*}{\newcommand}{\twocolfootfmtX}[2]{%
3283     \protected@edef\@currentlabel{%
3284         \@nameuse{@thefnmark#1}%
3285     }%
3286   \normal@pars
3287   \hangindent=\csuse{hangindentX@#1}%
3288   \hsize \csuse{hsizetwocolX@#1}
3289  \nottoggle{parindentX@#1}{\parindent=\z@}{}
3290   \tolerance=5000\relax
3291   \leavevmode
3292  \csuse{colalignX@#1}%
3293   {\csuse{notenumfontX@#1}\@nameuse{footfootmark#1}\strut%
3294     #2\strut\par}\allowbreak}
3295
3296 %
```

`\twocolfootgroupX`
`\mptwocolfootgroupX`

`\twocolfootgroupX{⟨series⟩}`

```
3297 \newcommand*{\twocolfootgroupX}[1]{{\csuse{notefontsizeX@#1}
3298   \splittopskip=\ht\strutbox
3299   \expandafter
3300   \rigidbalance\csname footins#1\endcsname \tw@ \splittopskip}}
3301 \newcommand*{\mptwocolfootgroupX}[1]{{%
3302   \vskip\skip\@nameuse{mpfootins#1}
```

```
3303    \ifl@dpairing\ifparledgroup%
3304      \leavevmode\marks\parledgroup@{begin}%
3305      \marks\parledgroup@series{#1}%
3306      \marks\parledgroup@type{footnoteX}%
3307    \fi\fi\normalcolor
3308    \ifparledgroup%
3309      \ifl@dpairing%
3310      \else%
3311        \setnoteswidthliketwocolumnsX@{#1}%
3312        \setnotesXpositionliketwocolumns@{#1}%
3313        \print@footnoteXrule{#1}%
3314      \fi%
3315    \else%
3316      \setnoteswidthliketwocolumnsX@{#1}%
3317      \setnotesXpositionliketwocolumns@{#1}%
3318      \print@footnoteXrule{#1}%
3319    \fi%
3320    \splittopskip=\ht\strutbox
3321    \expandafter
3322    \rigidbalance\csname mpfootins#1\endcsname \tw@ \splittopskip}}
3323
3324  %
```

XIII.4.4   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.

`\arrangementX@threecol`
```
3325  \newcommand*{\arrangementX@threecol}[1]{%
3326    \csgdef{series@displayX#1}{threecol}
3327    \expandafter\let\csname regvfootnote#1\endcsname=\threecolvfootnoteX
3328    \expandafter\let\csname footfmt#1\endcsname=\threecolfootfmtX
3329    \expandafter\let\csname footgroup#1\endcsname=\threecolfootgroupX
3330    \dimen\csname footins#1\endcsname=\csuse{maxnotesX@#1}%
3331    \skip\csname footins#1\endcsname=\csuse{beforenotesX@#1}%
3332    \advance\skip\csname footins#1\endcsname by \csuse{afterruleX@#1}\relax%
3333    \threecolfootsetupX{#1}
3334    \ifnoledgroup@\else%
3335      \expandafter\let\csname mpvfootnote#1\endcsname=\mpnormalvfootnoteX
3336      \expandafter\let\csname mpfootgroup#1\endcsname=\mpthreecolfootgroupX
3337      \skip\csname mpfootins#1\endcsname=\csuse{beforenotesX@#1}%
3338      \advance\skip\csname mpfootins#1\endcsname by\csuse{afterruleX@#1}
3339      \mpthreecolfootsetupX{#1}
3340    \fi%
3341  }
3342
3343  %
```

`\threecolfootsetupX`   \threecolfootsetupX{⟨*series*⟩}
`\mpthreecolfootsetupX`

```
3344  \newcommand*{\threecolfootsetupX}[1]{%
3345    \count\csname footins#1\endcsname 333
3346    \csxdef{default@footins#1}{333}%Use this to confine the  notes to one
      side only
3347    \multiply\dimen\csname footins#1\endcsname by \thr@@}
3348  \newcommand*{\mpthreecolfootsetupX}[1]{%
3349    \count\csname mpfootins#1\endcsname 333
3350    \multiply\dimen\csname mpfootins#1\endcsname by \thr@@}
3351
3352  %
```

\threecolvfootnoteX   \threecolvfootnoteX{⟨*series*⟩}{⟨*text*⟩}

```
3353  \notbool{parapparatus@}{\newcommand*}{\newcommand}{\threecolvfootnoteX}[2]{
      %
3354    \insert\csname footins#1\endcsname\bgroup
3355      \csuse{notefontsizeX@#1}
3356      \footsplitskips
3357      \@nameuse{footfmt#1}{#1}{#2}\egroup}
3358
3359  %
```

\threecolfootfmtX   \threecolfootfmtX{⟨*series*⟩}

```
3360  \notbool{parapparatus@}{\newcommand*}{\newcommand}{\threecolfootfmtX}[2]{%
3361    \protected@edef\@currentlabel{%
3362        \@nameuse{@thefnmark#1}%
3363     }%
3364    \hangindent=\csuse{hangindentX@#1}%
3365    \normal@pars
3366    \hsize \csuse{hsizethreecolX@#1}
3367    \nottoggle{parindentX@#1}{\parindent=\z@}{} %
3368    \tolerance=5000\relax
3369    \leavevmode
3370    \csuse{colalignX@#1}%
3371    {\csuse{notenumfontX@#1}\@nameuse{footfootmark#1}\strut%
3372      #2\strut\par}\allowbreak}
3373
3374  %
```

\threecolfootgroupX   \threecolfootgroupX{⟨*series*⟩}
\mpthreecolfootgroupX

```
3375  \newcommand*{\threecolfootgroupX}[1]{{\csuse{notefontsizeX@#1}
3376    \splittopskip=\ht\strutbox
3377    \expandafter
3378    \rigidbalance\csname footins#1\endcsname \thr@@ \splittopskip}}
3379  \newcommand*{\mpthreecolfootgroupX}[1]{{%
3380    \vskip\skip\@nameuse{mpfootins#1}
3381    \ifl@dpairing\ifparledgroup
3382      \leavevmode\marks\parledgroup@{begin}%
```

```
3383         \marks\parledgroup@series{#1}%
3384         \marks\parledgroup@type{footnoteX}%
3385       \fi\fi\normalcolor
3386       \ifparledgroup%
3387         \ifl@dpairing%
3388         \else%
3389           \setnoteswidthliketwocolumnsX@{#1}%
3390           \setnotesXpositionliketwocolumns@{#1}%
3391           \print@footnoteXrule{#1}%
3392         \fi%
3393       \else%
3394         \setnoteswidthliketwocolumnsX@{#1}%
3395         \setnotesXpositionliketwocolumns@{#1}%
3396         \print@footnoteXrule{#1}%
3397       \fi%
3398       \splittopskip=\ht\strutbox
3399       \expandafter
3400       \rigidbalance\csname mpfootins#1\endcsname \thr@@ \splittopskip}}
3401
3402  %
```

XIII.4.5   Paragraphed footnotes

The following macros set footnotes as one paragraph.

\arrangementX@threecol   \footparagraphX{⟨series⟩}

```
3403  \newcommand*{\arrangementX@paragraph}[1]{%
3404    \csgdef{series@displayX#1}{paragraph}%
3405    \expandafter\newcount\csname #1prevpage@num\endcsname
3406    \expandafter\let\csname footstart#1\endcsname=\parafootstartX
3407    \expandafter\let\csname regvfootnote#1\endcsname=\para@vfootnoteX
3408    \expandafter\let\csname footfmt#1\endcsname=\parafootfmtX
3409    \expandafter\let\csname footgroup#1\endcsname=\para@footgroupX
3410    \expandafter\let\csname footnoterule#1\endcsname=\normalfootnoteruleX
3411    \count\csname footins#1\endcsname=1000
3412    \csxdef{default@footins#1}{1000}%Use this to confine the  notes to one
      side only
3413    \dimen\csname footins#1\endcsname=\csuse{maxhnotesX@#1}
3414    \skip\csname footins#1\endcsname=\csuse{beforenotesX@#1}%
3415    \advance\skip\csname footins#1\endcsname by\csuse{afterruleX@#1}%
3416    \para@footsetupX{#1}
3417    \ifnoledgroup@\else
3418      \expandafter\let\csname mpvfootnote#1\endcsname=\mppara@vfootnoteX
3419      \expandafter\let\csname mpfootgroup#1\endcsname=\mppara@footgroupX
3420      \count\csname mpfootins#1\endcsname=1000
3421      \dimen\csname mpfootins#1\endcsname=\csuse{maxhnotesX@#1}
3422      \skip\csname mpfootins#1\endcsname=\csuse{beforenotesX@#1}%
3423      \advance\skip\csname mpfootins#1\endcsname by\csuse{afterruleX@#1}%
3424    \fi
```

```
3425        }
3426
3427 %
```

\para@footsetupX   \para@footsetupX{⟨*series*⟩}

```
3428 \newcommand*{\para@footsetupX}[1]{{\csuse{notefontsizeX@#1}
3429    \setnoteswidthliketwocolumnsX@{#1}%
3430    \dimen0=\baselineskip
3431    \multiply\dimen0 by 1024
3432    \divide\dimen0 by \columnwidth \multiply\dimen0 by \footfudgefiddle\relax
     %
3433    \expandafter
3434    \xdef\csname footfudgefactor#1\endcsname{%
3435       \expandafter\strip@pt\dimen0 }}}
3436
3437 %
```

\parafootstartX   \parafootstartX{⟨*series*⟩}

```
3438 \newcommand*{\parafootstartX}[1]{%
3439     \ifdimequal{0pt}{\prenotesX@}{}%
3440        {%
3441        \iftoggle{prenotesX@}{%
3442             \togglefalse{prenotesX@}%
3443             \skip\csname footins#1\endcsname=%
3444                \dimexpr\csuse{prenotesX@}+\csuse{afterruleX@#1}\relax%
3445             }%
3446          {}%
3447        }%
3448    \vskip\skip\csname footins#1\endcsname%
3449    \leftskip=\z@
3450    \rightskip=\z@
3451    \parindent=\z@
3452    \vskip\skip\@nameuse{footins#1}%
3453    \setnoteswidthliketwocolumnsX@{#1}%
3454    \setnotesXpositionliketwocolumns@{#1}%
3455    \print@footnoteXrule{#1}%
3456    }
3457
3458 %
```

\para@vfootnoteX   \para@vfootnoteX{⟨*series*⟩}{⟨*text*⟩}
\mppara@vfootnoteX

```
3459 \newcommand*{\para@vfootnoteX}[2]{%
3460    \insert\csname footins#1\endcsname
3461    \bgroup
3462       \csuse{bhooknoteX@#1}
3463       \csuse{notefontsizeX@#1}
3464       \footsplitskips
```

```
3465        \setbox0=\vbox{\hsize=\maxdimen
3466          \noindent\@nameuse{footfmt#1}{#1}{#2}}%
3467        \setbox0=\hbox{\unvxhX{0}{#1}}%
3468        \dp0=\z@
3469        \ht0=\csname footfudgefactor#1\endcsname\wd0
3470        \box0
3471        \penalty0
3472      \egroup}
3473  \newcommand*{\mppara@vfootnoteX}[2]{%
3474      \global\setbox\@nameuse{mpfootins#1}\vbox{%
3475        \unvbox\@nameuse{mpfootins#1}
3476        \csuse{bhooknoteX@#1}
3477        \csuse{notefontsizeX@#1}
3478        \footsplitskips
3479        \setbox0=\vbox{\hsize=\maxdimen
3480          \noindent\color@begingroup\@nameuse{footfmt#1}{#1}{#2}\color@endgroup
      }%
3481        \setbox0=\hbox{\unvxhX{0}{#1}}%
3482        \dp0=\z@
3483        \ht0=\csname footfudgefactor#1\endcsname\wd0
3484        \box0
3485        \penalty0}}
3486
3487  %
```

```
3488  \newcommand*{\unvxhX}[2]{% 2th is optional for retro-compatibility
3489      \setbox0=\vbox{\unvbox#1%
3490        \global\setbox1=\lastbox}%
3491      \unhbox1
3492      \unskip              % remove \rightskip,
3493      \unskip              % remove \parfillskip,
3494      \unpenalty           % remove \penalty of 10000,
3495      \hskip\csuse{afternoteX@#2}}   % but add the glue to go between the notes
3496
3497  %
```

\parafootfmtX   \parafootfmtX{⟨series⟩}

```
3498  \newcommand*{\parafootfmtX}[2]{%
3499      \protected@edef\@currentlabel{%
3500          \@nameuse{@thefnmark#1}%
3501      }%
3502      \insertparafootsepX{#1}%
3503      \ledsetnormalparstuffX{#1}%
3504      {\csuse{notenumfontX@#1}%
3505        \csuse{notenumfontX@#1}%
3506        \@nameuse{footfootmark#1}%
3507        \strut%
3508        #2\penalty-10}}
```

```
3509
3510 %
```

\para@footgroupX   \para@footgroupX{⟨*series*⟩}
\mppara@footgroupX
```
3511 \newcommand*{\para@footgroupX}[1]{%
3512   \unvbox\csname footins#1\endcsname
3513   \ifcsstring{raggedX@#1}{L}{\RaggedLeft}{}%
3514   \ifcsstring{raggedX@#1}{R}{\RaggedRight}{}%
3515   \makehboxofhboxes
3516   \setbox0=\hbox{\unhbox0 \removehboxes}%
3517   \csuse{notefontsizeX@#1}
3518   \noindent\unhbox0\par}
3519 \newcommand*{\mppara@footgroupX}[1]{{%
3520   \setnoteswidthliketwocolumnsX@{#1}%
3521   \vskip\skip\@nameuse{mpfootins#1}
3522   \ifl@dpairing\ifparledgroup
3523     \leavevmode%
3524     \leavevmode\marks\parledgroup@{begin}%
3525     \marks\parledgroup@series{#1}%
3526     \marks\parledgroup@type{footnoteX}%
3527   \fi\fi\normalcolor
3528   \ifparledgroup%
3529     \ifl@dpairing%
3530     \else%
3531       \setnoteswidthliketwocolumnsX@{#1}%
3532       \setnotesXpositionliketwocolumns@{#1}%
3533       \print@footnoteXrule{#1}%
3534     \fi%
3535   \else%
3536       \setnoteswidthliketwocolumnsX@{#1}%
3537       \setnotesXpositionliketwocolumns@{#1}%
3538       \print@footnoteXrule{#1}%
3539   \fi%
3540   \unvbox\csname mpfootins#1\endcsname
3541   \ifcsstring{raggedX@#1}{L}{\RaggedLeft}{}%
3542   \ifcsstring{raggedX@#1}{R}{\RaggedRight}{}%
3543   \makehboxofhboxes
3544   \setbox0=\hbox{\unhbox0 \removehboxes}%
3545   \csuse{notefontsizeX@#1}
3546   \noindent\unhbox0\par}}
3547
3548 %
```

Insertion of the footnotes separator   The command \insertparafootsepX{⟨*series*⟩}
must be called at the beginning of \parafootftmX.

\prevpage@num
```
3549 \newcommand{\insertparafootsepX}[1]{%
```
\Xinsertparafootsep
```
3550     \ifnumequal{\csuse{prevpage#1@num}}{\page@num}%
```

```
3551        {\csuse{parafootsepX@#1}}%
3552        {}%
3553 }
3554 %
```

XIV   Code common to both critical and familiar footnote in normal arrangement

\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 num-
bered automatically (as set up by \autopar, for example).

In the case of footnote arranged in a "normal" way, we also must set some setting
for paragraph indent and text direction when using LuaLaTeX.

That why we have defined \ledsetnormalparstuff@common in order to make this
setting for both familiar and critical notes. This command is called by command to make
specific setting to critical or familiar footnote.

\ledsetnormalparstuff@common
\Xledsetnormalparstuff
\ledsetnormalparstuffX

```
3555 \newcommand*{\ledsetnormalparstuff@common}{%
3556   \ifluatex%
3557     \luatextextdir\footnote@luatextextdir%
3558     \luatexpardir\footnote@luatexpardir%
3559   \fi%
3560   \csuse{\csuse{footnote@dir}}%
3561   \normal@pars%
3562   \parfillskip \z@ \@plus 1fil}%
3563
3564 \newcommand*{\Xledsetnormalparstuff}[1]{%
3565   \ledsetnormalparstuff@common%
3566   \nottoggle{Xparindent@#1}{\noindent}{}%\noindent and and not \parindent=0
pt to avoid to break the (bad) change made when moving from ledmac to
eledmac
3567 }%
3568
3569 \newcommand*{\ledsetnormalparstuffX}[1]{%
3570   \ledsetnormalparstuff@common%
3571   \nottoggle{parindentX@#1}{\noindent}{}%\noindent and and not \parindent=0
pt to avoid to break the (bad) change made when moving from ledmac to
eledmac
3572 }%
3573 %
```

XV   Footnotes' width for two columns

We define here some commands which make sense only with reledpar, but must be
called when defining notes parameters. These commands change the width of block
notes to allow them to have the same size than two parallel columns.

`\old@hsize`
`noteswidthliketwocolumns@`
`oteswidthliketwocolumnsX@`

These two commands are called at the beginning of critical or familiar notes groups. They set, if the option is enabled, the `\hsize`. They are also called at the on the setup for paragraphed notes.

```
3574
3575 \newdimen\old@hsize%
3576 \AtBeginDocument{\old@hsize=\hsize}%
3577
3578 \newcommand{\setXnoteswidthliketwocolumns@}[1]{%
3579   \global\let\hsize@fornote=\hsize%
3580   \global\old@hsize=\hsize%
3581   \iftoggle{Xnoteswidthliketwocolumns@#1}%
3582     {%
3583     \csuse{setwidthliketwocolumns@\columns@position}%
3584     \global\let\hsize@fornote=\hsize%
3585     }%
3586     {}%
3587   \let\hsize=\hsize@fornote%
3588   \let\columnwidth=\hsize@fornote%
3589 }%
3590
3591 \newcommand{\setnoteswidthliketwocolumnsX@}[1]{%
3592   \global\let\hsize@fornote=\hsize%
3593   \global\old@hsize=\hsize%
3594   \iftoggle{noteswidthliketwocolumnsX@#1}%
3595     {%
3596     \csuse{setwidthliketwocolumns@\columns@position}%
3597     \global\let\hsize@fornote=\hsize%
3598     }%
3599     {}%
3600   \let\hsize=\hsize@fornote%
3601   \let\columnwidth=\hsize@fornote%
3602 }%
3603
3604 %
```

`espositionliketwocolumns@`
`sXpositionliketwocolumns@`

These two commands set the position of the critical / familiar footnotes, depending on the hooks Xnoteswidthliketwocolumns and noteswidthliketwocolumnsX. They call commands which are defined only in reledpar, because this feature has no sens without reledpar.

```
3605 \newcommand{\setXnotespositionliketwocolumns@}[1]{%
3606   \iftoggle{Xnoteswidthliketwocolumns@#1}{%
3607     \csuse{setnotespositionliketwocolumns@\columns@position}%
3608   }{}%
3609 }%
3610
3611 \newcommand{\setnotesXpositionliketwocolumns@}[1]{%
3612   \iftoggle{noteswidthliketwocolumnsX@#1}{%
3613     \csuse{setnotespositionliketwocolumns@\columns@position}%
```

```
3614    }{}%
3615  }%
3616
3617  %
```

XVI   Footnotes' order

\fnpos
\mpfnpos
\@fnpos
\@mpfnpos

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

```
3618  \def\@fnpos{familiar-critical}
3619  \def\@mpfnpos{critical-familiar}
3620  \newcommand{\fnpos}[1]{\xdef\@fnpos{#1}}
3621  \newcommand{\mpfnpos}[1]{\xdef\@mpfnpos{#1}}
3622  %
```

XVII   Footnotes' rule

Because the footnotes' rules can be shifted to the right when footnotes are set like two columns, we do not print them directly, but we put them in a \vbox.

\print@Xfootnoterule
\print@footnoteXrule

```
3623  \newcommand{\print@Xfootnoterule}[1]{%
3624    \vskip-\csuse{Xafterrule@#1}%Because count in \dimen\csuse{#1footins}
3625    \nointerlineskip%
3626    \moveleft-\leftskip\vbox{\csuse{#1footnoterule}}%
3627    \nointerlineskip%
3628    \vskip\csuse{Xafterrule@#1}%
3629  }%
3630
3631  \newcommand{\print@footnoteXrule}[1]{%
3632    \vskip-\csuse{afterruleX@#1}%Because count in \dimen\csuse{footins#1}
3633    \nointerlineskip%
3634    \moveleft-\leftskip\vbox{\csuse{footnoterule#1}}%
3635    \nointerlineskip%
3636    \vskip\csuse{afterruleX@#1}%
3637  }%
3638
3639  %
```

XVIII   Specific skip for first series of footnotes

XVIII.0.1   Overview

\Xbeforenotes inserts a specific skip for the first series of notes in a page. As we can't know in advance which series will be the first, we call \prepare@preXnotes before inserting any critical notes, in order to prevent page number overlapping.

1. If it is the first note of the current page, it changes the footnote skip for the series to the value specified to `\Xbeforenotes`. It also keeps the series of the note as the first one of the current page.

2. If it is not the first note of the current page:

   - If the current series is printed after the series kept as the first of the current page, then nothing happens.

   - If the current series is printed before the series kept as the first of the current page, then it changes the footnote skip of the current series to the value normally used by the series which was marked as the first of the page. It also keeps the current series as the new first one of the current page.

For example, suppose the series order is A,B. We call first a `\Bfootnote` and a `\Afootnote`. The only skips used are, finally, the skip specific to the first series of the page, and the skip for the B series. If we have not called `\Afootnote`, the only skip used is the skip specific to the first series of the page.

That is perfect.

The series skip and the first series of the current page are reset before the footnotes are printed. Then, the footstart macros manage the problem of the first series of the page.

After the rule, the space which is defined by `\Xafterrule` does not depend on whether the series is the first one of the page or not. So we use its normal value for each series.

And now, implementation !

XVIII.0.2   User level command

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

```
3640 \newtoggle{preXnotes@}
3641 \toggletrue{preXnotes@}
3642 \newcommand{\preXnotes@}{0pt}
3643 \newcommand*{\preXnotes}[1]{\renewcommand{\preXnotes@}{#1}}
3644 %
```

The same, but for familiar footnotes.

\preXnotes   `3645` `\newtoggle{prenotesX@}`
\preXnotes@ `3646` `\toggletrue{prenotesX@}`
```
3647 \newcommand{\prenotesX@}{0pt}
3648 \newcommand*{\prenotesX}[1]{\renewcommand{\prenotesX@}{#1}}
3649 %
```

XVIII.0.3 Internal commands

```
firstXseries@3650 \gdef\firstXseries@{}
prepare@preXnotes3651 \newcommand{\prepare@preXnotes}[1]{%
3652   \ifdimequal{0pt}{\preXnotes@}%
3653   {}%
3654   {%
3655     \IfStrEq{\firstXseries@}{}{%
3656       \global\skip\csuse{#1footins}=\preXnotes@%
3657       \global\advance\skip\csname #1footins\endcsname by\csuse{Xafterrule@
#1}%
3658       \gdef\firstXseries@{#1}%
3659     }%
3660     {%
3661      \ifseriesbefore{#1}{\firstXseries@}%
3662        {%
3663        \global\skip\csuse{#1footins}=\csuse{Xbeforenotes@\firstXseries@}%
3664        \global\advance\skip\csname #1footins\endcsname by\csuse{Xafterrule@
#1}%
3665        \gdef\firstXseries@{#1}%
3666        }%
3667        {}%
3668     }%
3669   }%
3670 }
3671 %
```

The same thing is required for familiar notes and \prenotesX.

```
firstseriesX@3672 \gdef\firstseriesX@{}
prepare@prenotesX3673 \newcommand{\prepare@prenotesX}[1]{%
3674   \ifdimequal{0pt}{\prenotesX@}%
3675   {}%
3676   {%
3677     \IfStrEq{\firstseriesX@}{}{%
3678       \global\skip\csuse{footins#1}=\prenotesX@%
3679       \global\advance\skip\csname footins#1\endcsname by\csuse{afterruleX@
#1}%
3680       \gdef\firstseriesX@{#1}%
3681     }%
3682     {%
3683      \ifseriesbefore{#1}{\firstseriesX@}%
3684        {%
3685        \global\skip\csuse{footins#1}=\csuse{beforenotesX@\firstseriesX@}%
3686        \global\advance\skip\csname footins#1\endcsname by\csuse{afterruleX@
#1}%
3687        \gdef\firstXseries@{#1}%
3688        }%
3689        {}%
```

```
3690        }%
3691      }%
3692  }
3693  %
```

XIX   Endnotes

First, check the noend option.

```
3694  \ifbool{noend@}{}{%Used instead of \ifnoend@ to prevent expansion problem
3695  %
```

\l@dend@open \l@dend@open and \l@dend@close are the macros that are used to open and close the
\l@dend@close endnote file. Note that all our writing to this file is \immediate: all page and line num-
bers for the endnotes are generated by the same mechanism we use for the footnotes, so
that there is no need to defer any writing to catch information from the output routine.
The argument of these two command is the series letter.

```
3696  \newcommand{\l@dend@open}[1]{%
3697    \global\booltrue{l@dend@#1}%
3698    \expandafter\immediate%
3699      \expandafter\openout%
3700        \csname l@d@#1end\endcsname%
3701        =\jobname.#1end\relax%
3702  }%
3703  \newcommand{\l@dend@close}[1]{%
3704    \global\boolfalse{l@dend@#1}%
3705    \expandafter\immediate%
3706      \expandafter\closeout\csname l@d@#1end\endcsname%
3707  }%
3708
3709  %
```

\l@dend@stuff \l@dend@stuff is used by \beginnumbering to do everything that is 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.

```
3710  \newcommand{\l@dend@stuff}{%
3711  \def\do##1{%
3712    \ifbool{l@dend@##1}{}%
3713      {\l@dend@open{##1}}%
3714    \expandafter\immediate\expandafter\write\csname l@d@##1end\endcsname{\
  string\l@d@section{\the\section@num}}%
3715    }%
3716  \dolistloop{\@series}%
3717  }%
3718
3719  %
```

\endprint The \endprint here is nearly identical in its functioning to \normalfootfmt.
\l@d@section   The endnote file also contains \l@d@section commands, which supply the section
numbers from the main text; standard reledmac does nothing with this information, but
it is there if you want to write custom macros to do something with it.   Arguments
are:

- #1 Line numbers and font selection.

- #2 Lemma.

- #3 Note content.

- #4 Series.

- #5 Optional argument of \Xendnote.

```
3720  \global\notbool{parapparatus@}{}{\long}\def\endprint#1#2#3#4#5{{%
3721    \ifXendinsertsep@%
3722      \hskip\csuse{Xendafternote@#4}%
3723      \csuse{Xendsep@#4}%
3724    \else%
3725      \iftoggle{Xendparagraph@#4}%
3726        {\global\Xendinsertsep@true}%
3727        {}%
3728    \fi%
3729    \xdef\@currentseries{#4}%
3730    \def\do##1{%
3731     \toggletrue{##1@}%
3732    }%
3733    \notblank{#5}{\docsvlist{#5}}{}%
3734    \csuse{Xendbhooknote@#4}%
3735    \csuse{Xendnotefontsize@#4}%
3736    \ifboolexpr{%
3737      togl {nonum@}%
3738      or togl {Xendnonumber@#4}%
3739      }%
3740      {\hspace{\csuse{Xendinplaceofnumber@#4}}}%
3741      {\printlineendnotearea{#1}{#4}}%
3742      \nottoggle{Xendlemmadisablefontselection@#4}%
3743        {\select@lemmafont#1|#2}%
3744        {#2}%
3745    \ifboolexpr{%
3746      togl {nosep@}%
3747      or test{\ifcsempty{Xendlemmaseparator@#4}}%
3748      }%
3749      {\hskip\csuse{Xendinplaceoflemmaseparator@#4}}%
3750      {\nobreak%
3751       \hskip\csuse{Xendbeforelemmaseparator@#4}%
3752       \csuse{Xendlemmaseparator@#4}%
3753       \hskip\csuse{Xendafterlemmaseparator@#4}%
```

```
3754        }%
3755      #3%
3756      \nottoggle{Xendparagraph@#4}{\par}{}%
3757    \def\do##1{%
3758        \togglefalse{##1@}%
3759      }%
3760      \notblank{#5}{\docsvlist{#5}}{}%
3761 }}%
3762
3763 \let\l@d@section=\@gobble
3764
3765 %
```

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

```
3766 \newcommand{\printlineendnotearea}[2]{%
3767   \bgroup%
3768      \csuse{Xendnotenumfont@#2}%
3769      \ifdimequal{\csuse{Xendboxlinenum@#2}}{0pt}%
3770        {\printendlines#1|}%
3771        {\leavevmode%
3772         \hbox to \csuse{Xendboxlinenum@#2}%
3773         {%
3774         \IfSubStr{RC}{\csuse{Xendboxlinenumalign@#2}}{\hfill}{}%
3775         \printendlines#1|%
3776         \IfSubStr{LC}{\csuse{Xendboxlinenumalign@#2}}{\hfill}{}%
3777        }}%
3778   \egroup%
3779   \enspace%
3780 }%
3781 %
```

\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.

```
3782 \newcommand*{\setprintendlines}[6]{%
3783   \l@d@pnumfalse \l@d@dashfalse
3784   \ifnum#4=#1 \else
3785     \l@d@pnumtrue
```

```
3786        \l@d@dashtrue
3787     \fi
3788 %
```

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

```
3789     \ifl@d@pnum \l@d@elintrue \else \l@d@elinfalse \fi
3790     \ifnum#2=#5 \else
3791         \l@d@elintrue
3792         \l@d@dashtrue
3793     \fi
3794 %
```

We print the starting sub-line if it is nonzero.

```
3795     \l@d@ssubfalse
3796     \ifnum#3=0 \else
3797         \l@d@ssubtrue
3798     \fi
3799 %
```

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

```
3800     \l@d@eslfalse
3801     \ifnum#6=0 \else
3802         \ifnum#6=#3
3803             \ifl@d@elin \l@d@esltrue \else \l@d@eslfalse \fi
3804         \else
3805             \l@d@esltrue
3806             \l@d@dashtrue
3807         \fi
3808     \fi%
3809 %
```

```
3810     \ifl@d@dash%
3811       \ifboolexpr{togl{fulllines@} or test{\ifcsempty{Xendtwolines@\
     @currentseries}}}%
3812         {}%
3813         {%
3814         \setistwofollowinglines{#1}{#2}{#4}{#5}%
3815           \ifboolexpr{%
3816             (%
3817                 togl {Xendtwolinesbutnotmore@\@currentseries}%
3818                 and not%
3819                   (%
3820                    bool {istwofollowinglines@}%
3821                   )%
3822             )%
3823             or%
3824             (%
```

```
3825              (not test{\ifnumequal{#1}{#4}})%
3826                and togl{Xendtwolinesonlyinsamepage@\@currentseries}%
3827              )%
3828          }%
3829          {}%
3830          {%
3831          \l@d@dashfalse%
3832          \l@d@Xtwolinestrue%
3833          \l@d@elinfalse%
3834          \l@d@eslfalse%
3835          \ifcsempty{Xendmorethantwolines@\@currentseries}%
3836            {}%
3837            {\ifistwofollowinglines@\else%
3838              \l@d@Xmorethantwolinestrue%
3839            \fi%
3840            }%
3841          }%
3842        }%
3843    \fi%
3844 %
```

End of \setprintendlines.

```
3845 }%
3846 %
```

\printendlines Now we are ready to print it all.

```
3847 \def\printendlines#1|#2|#3|#4|#5|#6|#7|{\begingroup
3848   \setprintendlines{#1}{#2}{#3}{#4}{#5}{#6}%
3849 %
```

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).

So, first, print the start lines.

```
3850   \ifdimequal{\csuse{Xendboxstartlinenum@\@currentseries}}{0pt}%
3851     {\bgroup}%
3852     {\leavevmode\hbox to \csuse{Xendboxstartlinenum@\@currentseries}\bgroup
\hfill}%
3853   \printnpnum{#1}%
3854   \linenumrep{#2}%
3855   \ifl@d@ssub \fullstop \sublinenumrep{#3}\fi
3856   \egroup%
3857 %
```

And now, print the dash + the end line number, or the line number range symbol.

```
3858   \ifdimequal{\csuse{Xendboxendlinenum@\@currentseries}}{0pt}%
3859     {\bgroup}%
```

```
3860        {\hbox to \csuse{Xendboxendlinenum@\@currentseries}\bgroup}%
3861      \ifl@d@Xtwolines%
3862        \ifl@d@Xmorethantwolines%
3863          \csuse{Xendmorethantwolines@\@currentseries}%
3864        \else%
3865          \csuse{Xendtwolines@\@currentseries}%
3866        \fi%
3867      \else%
3868        \ifl@d@dash \endashchar\fi%
3869        \ifl@d@pnum \printnpnum{#4}\fi%
3870        \ifl@d@elin \linenumrep{#5}\fi%
3871        \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumrep{#6}\fi%
3872      \fi%
3873      \ifdimequal{\csuse{Xendboxendlinenum@\@currentseries}}{0pt}%
3874        {}%
3875        {\hfill}%Prevent underfull hbox
3876      \egroup%
3877      \endgroup%
3878   }%
3879
3880   %
```

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

```
3881  \newcommand*{\printnpnum}[1]{p.#1) }
3882
3883  %
```

\doendnotes   \doendnotes is the command you use to print one series of endnotes; it takes one
\ifXendinsertsep@   argument: the series letter of the note series you want to print. \Xendinsertsep@ is
                     set to true at the first note of the series, and to false at the last one.

```
3884  \newif\ifXendinsertsep@%
3885  \newcommand*{\doendnotes}[1]{%
3886    \l@dend@close{#1}%
3887      \begingroup
3888        \makeatletter
3889        \expandafter\let\csname #1end\endcsname=\endprint
3890        \input\jobname.#1end%
3891        \global\Xendinsertsep@false%
3892      \endgroup}
3893  %
```

\doendnotesbysection   \doendnotesbysection is a variant of the previous macro. While \doendnotes print
                       endnotes for all of numbered sections \doendnotesbysection print the endnotes for
                       the first numbered section at its first call for a series, then for the second section at its
                       second call for the same series, then for the third section at its third call for the same
                       series, and so on.

```
3894 \newcommand*{\doendnotesbysection}[1]{%
3895 \l@dend@close{#1}%
3896 \global\expandafter\advance\csname #1end@bysection\endcsname by 1%
3897 \begingroup%
3898    \makeatletter%
3899    \def\l@d@section##1{%
3900      \ifnumequal{##1}{\csname #1end@bysection\endcsname}%
3901        {\cslet{#1end}{\endprint}}%
3902        {\cslet{#1end}{\@gobblefive}}%
3903    }%
3904    \input\jobname.#1end%
3905    \global\Xendinsertsep@false%
3906 \endgroup%
3907 }%
3908 %
```

End of section for end notes

```
3909 }%
3910 %
```

XX   Generate series of notes

In this section, X means the name of the series (A, B etc.)

\series   \series\series creates one more new series. It is a public command, which just loops on the private command \newseries@.

```
3911 \newcommand{\newseries}[1]{%
3912    \def\do##1{\newseries@{##1}}%
3913    \docsvlist{#1}
3914 }
3915 %
```

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

```
3916 \newcommand{\@series}{}
3917 %
```

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

\newseries@₁₈ \newcommand{\newseries@}[1]{

```
3919 %
```

XX.1   Test if series is still existing

```
3920      \xifinlist{#1}{\@series}{\led@warn@SeriesStillExist{#1}}%
3921      {%
3922 %
```

XX.2   Init specific to `reledpar`

When calling \newseries@ after having loaded `reledpar`, we need to load specific setting.

```
3923        \ifdefined\newseries@par%
3924          \newseries@par{#1}%
3925        \fi%
3926 %
```

XX.3   For critical footnotes

Critical footnotes are those which start with letters. We look for the \nocritical option of `reledmac`.

```
3927        \unless\ifnocritical@
3928 %
```

XX.3.1   Options

```
3929        \newtoggle{Xparindent@#1}
3930        \newtoggle{Xlemmadisablefontselection@#1}
3931        \csgdef{Xhangindent@#1}{0pt}%
3932        \csgdef{Xragged@#1}{}%
3933        \csgdef{Xhsizetwocol@#1}{0.45 \hsize}%
3934        \csgdef{Xhsizethreecol@#1}{.3 \hsize}%
3935        \csgdef{Xcolalign@#1}{\raggedright}%
3936        \csgdef{Xnotenumfont@#1}{\normalfont}%
3937        \csgdef{Xnotefontsize@#1}{\footnotesize}%
3938        \csgdef{Xbhooknote@#1}{}%
3939
3940        \csgdef{Xboxlinenum@#1}{0pt}%
3941        \csgdef{Xboxlinenumalign@#1}{L}%
3942
3943        \csgdef{Xboxstartlinenum@#1}{0pt}%
3944        \csgdef{Xboxendlinenum@#1}{0pt}%
3945
3946        \csgdef{Xboxsymlinenum@#1}{0pt}%
3947        \newtoggle{Xnumberonlyfirstinline@#1}%
3948        \newtoggle{Xnumberonlyfirstintwolines@#1}%
3949        \csgdef{Xtwolines@#1}{}%
3950        \csgdef{Xmorethantwolines@#1}{}%
3951        \newtoggle{Xtwolinesbutnotmore@#1}%
```

```
3952        \newtoggle{Xtwolinesonlyinsamepage@#1}%
3953        \newtoggle{Xonlypstart@#1}%
3954        \newtoggle{Xpstarteverytime@#1}%
3955        \newtoggle{Xpstart@#1}%
3956        \csgdef{Xsymlinenum@#1}{}%
3957        \newtoggle{Xnonumber@#1}%
3958        \csgdef{Xbeforenumber@#1}{0pt}%
3959        \csgdef{Xafternumber@#1}{0.5em}%
3960        \newtoggle{Xnonbreakableafternumber@#1}%
3961        \csgdef{Xbeforesymlinenum@#1}{\csuse{Xbeforenumber@#1}}%
3962        \csgdef{Xaftersymlinenum@#1}{\csuse{Xafternumber@#1}}%
3963        \csgdef{Xinplaceofnumber@#1}{1em}%
3964        \global\cslet{Xlemmaseparator@#1}{\rbracket}%
3965        \csgdef{Xbeforelemmaseparator@#1}{0em}%
3966        \csgdef{Xafterlemmaseparator@#1}{0.5em}%
3967        \csgdef{Xinplaceoflemmaseparator@#1}{1em}%
3968        \csgdef{Xbeforenotes@#1}{1.2em \@plus .6em \@minus .6em}
3969        \csgdef{Xafterrule@#1}{0pt}
3970        \csgdef{Xtxtbeforenotes@#1}{}
3971        \csgdef{Xmaxhnotes@#1}{0.8\vsize}
3972        \newtoggle{Xnoteswidthliketwocolumns@#1}%
3973        \csgdef{Xparafootsep@#1}{}%
3974        \csgdef{Xafternote@#1}{1em plus.4em minus.4em}
3975 %
```

XX.3.2   Create inserts, needed to add notes in foot

As regards inserts, see chapter 15 of *The TeXbook* by D. Knuth.

```
3976        \expandafter\newinsert\csname #1footins\endcsname%
3977        \unless\ifnoledgroup@%
3978          \expandafter\newinsert\csname mp#1footins\endcsname%
3979        \fi%
3980 %
```

XX.3.3   Create commands for critical apparatus, \Afootnote, \Bfootnote etc.

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

```
3981        \global\notbool{parapparatus@}{\expandafter\newcommand\expandafter
     *}{\expandafter\newcommand}\csname #1footnote\endcsname[2][]{%
3982            \ifnum\@edtext@level>0%
3983              \begingroup%
3984              \newcommand{\content}{##2}%
3985              \ifnumberedpar@%
3986                  \ifledRcol%
3987                      \ifluatex%
3988                          \footnotelang@lua[R]%
3989                      \fi%
3990                      \@ifundefined{xpg@main@language}%if polyglossia
```

```
3991                          {}%
3992                          {\footnotelang@poly[R]}%
3993                      \footnoteoptions@[R]{##1}{true}%
3994                      \xright@appenditem{%
3995                        \noexpand\prepare@preXnotes{#1}%
3996                        \noexpand\prepare@edindex@fornote{\l@d@nums}%
3997                        \unexpanded{\def\sw@list@inedtext}{\expandafter\
      unexpanded\expandafter{\sw@inthisedtext}}%The value of the \sw@inthisedtext
       of current \edtext will be pushed to  \sw@list@inedtext when the notes are
       expanded.
3998                        \noexpand\csuse{v#1footnote}{#1}%
3999                          {{\l@d@nums}{\expandonce\@tag}{\expandonce\content}}
      %
4000                      }\to\inserts@listR
4001                      \footnoteoptions@[R]{##1}{false}%
4002                      \global\advance\insert@countR \@ne%
4003                  \else%
4004                      \ifluatex%
4005                        \footnotelang@lua%
4006                      \fi%
4007                    \@ifundefined{xpg@main@language}%if polyglossia
4008                          {}%
4009                          {\footnotelang@poly}%
4010                      \footnoteoptions@{##1}{true}%
4011                      \xright@appenditem{%
4012                        \noexpand\prepare@preXnotes{#1}%
4013                        \noexpand\prepare@edindex@fornote{\l@d@nums}%
4014                        \unexpanded{\def\sw@list@inedtext}{\expandafter\
      unexpanded\expandafter{\sw@inthisedtext}}%The value of the \sw@inthisedtext
       of current edtext will be pushed to  \sw@list@inedtext when the notes are
       expanded.
4015                        \noexpand\csuse{v#1footnote}{#1}%
4016                          {{\l@d@nums}{\expandonce\@tag}{\expandonce\content}}
      %
4017                      }\to\inserts@list
4018                      \global\advance\insert@count \@ne%
4019                      \footnoteoptions@{##1}{false}%
4020                  \fi
4021              \else
4022                  \csuse{v#1footnote}{#1}{{0|0|0|0|0|0|0|0}{}{##1}}%
4023              \fi%
4024              \endgroup%
4025          \else%
4026              \led@err@FootnoteWithoutEdtext%
4027          \fi%
4028      \ignorespaces%
4029              }
4030 %
```

We need to be able to modify reledmac's footnote macros and restore their

```
4031        \global\csletcs{#1@@footnote}{#1footnote}
4032 %
```

XX.3.4   Set standard display

```
4033        \Xarrangement@normal{#1}%
4034 %
```

End of for critical footnotes.

```
4035     \fi
4036 %
```

XX.4   For familiar footnotes

Familiar footnotes are those which end with letters. We look for the `nofamiliar` option of `reledmac`.

```
4037     \unless\ifnofamiliar@
4038 %
```

XX.4.1   Options

```
4039        \newtoggle{parindentX@#1}
4040        \csgdef{hangindentX@#1}{0pt}%
4041        \csgdef{raggedX@#1}{}%
4042        \csgdef{hsizetwocolX@#1}{0.45 \hsize}%
4043        \csgdef{hsizethreecolX@#1}{.3 \hsize}%
4044        \csgdef{colalignX@#1}{\raggedright}%
4045        \csgdef{notenumfontX@#1}{\normalfont}%
4046        \csgdef{notefontsizeX@#1}{\footnotesize}%
4047        \csgdef{bhooknoteX@#1}{}%
4048        \csgdef{afterruleX@#1}{0pt}
4049        \csgdef{beforenotesX@#1}{1.2em \@plus .6em \@minus .6em}
4050        \csgdef{maxhnotesX@#1}{0.8\vsize}%
4051        \newtoggle{noteswidthliketwocolumnsX@#1}%
4052        \csgdef{parafootsepX@#1}{}%
4053        \csgdef{afternoteX@#1}{1em plus.4em minus.4em}
4054 % End of for familiar footnotes.
4055 % \subsubsection{Create inserts, needed to add notes in foot}
4056 % As regards inserts, see chapter 15 of the TeXBook by D. Knuth.
4057 %     \begin{macrocode}
4058        \expandafter\newinsert\csname footins#1\endcsname%
4059        \unless\ifnoledgroup@%
4060          \expandafter\newinsert\csname mpfootins#1\endcsname%
4061        \fi%
4062 %
```

XX.4.2   Create tools for familiar footnotes (`\footnoteX`)

First, create the `\footnoteX` command. Note the double # in command: it is because a
command is called inside another command.

```
4063
4064        \global\expandafter\newcommand\csname footnote#1\endcsname[1]{%
4065            \begingroup%
4066                \prepare@prenotesX{#1}%
4067                \newcommand{\content}{##1}%
4068                \stepcounter{footnote#1}%
4069                \protected@csxdef{@thefnmark#1}{\csuse{thefootnote#1}}%
4070                \nottoggle{nomk@}%Nomk is set to true when using \
footnoteXnomk with \parpackage
4071                    {\csuse{@footnotemark#1}}%
4072                    {}%
4073                \ifluatex%
4074                  \xdef\footnote@luatextextdir{\the\luatextextdir}%
4075                  \xdef\footnote@luatexpardir{\the\luatexpardir}%
4076                \fi%
4077                \csuse{vfootnote#1}{#1}{\expandonce\content}\m@mmf@prepare%
4078            \endgroup%
4079        }
4080 %
```

Then define the counters.

```
4081        \newcounter{footnote#1}
4082          \global\expandafter\renewcommand\csname thefootnote#1\endcsname{\
arabic{footnote#1}}
4083 %
```

Do not forget to initialize series

```
4084   \arrangementX@normal{#1}%
4085   \fi
4086 %
```

XX.5   The endnotes

Endnotes are commands like `\Xendnote`, where X is a series letter. First, we check for
the noend options.

```
4087   \unless\ifnoend@
4088 %
```

XX.5.1   The auxiliary file

`\l@d@Xend`   Endnotes of all varieties are saved up in a file, one by series, typically named ⟨*jobname*⟩.Xend.
`\ifl@dend@X`   `\l@d@end` is the output stream number for this file, and `\ifl@dend@X` is a flag that is
`\l@dend@Xtrue`   true when the file is open.
`\l@dend@Xfalse`

```
4089        \expandafter\newwrite\csname l@d@#1end\endcsname%
4090        \expandafter\newif\csname ifl@dend@#1\endcsname%
4091 %
```

XX.5.2   The main macro

The `\Xendnote` macro functions to write one endnote to the `.Xend` 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 does not exceed restrictions on the length of lines in files.

```
4092
4093        \global\expandafter\newcommandx\csname #1endnote\endcsname[2][1,
     usedefault]{%
4094            \bgroup%
4095            \newlinechar='40%
4096            \global\@noneed@Footnotetrue%
4097            \newcommand{\content}{##2}%
4098            \expandafter\immediate\expandafter\write\csname l@d@#1end\
     endcsname{%
4099                \expandafter\string\csname #1end\endcsname%
4100                {\ifnumberedpar@\l@d@nums\fi}%
4101                {\ifnumberedpar@\expandonce\@tag\fi}%
4102                {\expandonce\content}%
4103                {#1}%
4104                {##1}%
4105                \@percentchar%
4106            }%
4107            \egroup%
4108            \ignorespaces%
4109        }%
4110 %
```

`\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 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 `\@gobblefive`, which just skips over its five arguments.

```
4111
4112        \global\cslet{#1end}{\@gobblefive}
4113 %
```

We need to store the number of times `\doendnotesbysection` is called for one series.

```
4114    \global\expandafter\newcount\csname #1end@bysection\endcsname%
4115 %
```

XX.5.3   The options

```
4116        \csgdef{Xendtwolines@#1}{}%
4117        \csgdef{Xendmorethantwolines@#1}{}%
4118        \newtoggle{Xendtwolinesbutnotmore@#1}{}%
4119        \newtoggle{Xendtwolinesonlyinsamepage@#1}{}%
4120        \newtoggle{Xendlemmadisablefontselection@#1}%
4121        \csgdef{Xendnotenumfont@#1}{\normalfont}%
4122        \csgdef{Xendnotefontsize@#1}{\footnotesize}%
4123        \csgdef{Xendbhooknote@#1}{}%
4124
4125        \csgdef{Xendboxlinenum@#1}{0pt}%
4126        \csgdef{Xendboxlinenumalign@#1}{L}%
4127
4128        \csgdef{Xendboxstartlinenum@#1}{0pt}%
4129        \csgdef{Xendboxendlinenum@#1}{0pt}%
4130
4131        \csgdef{Xendlemmaseparator@#1}{}%
4132        \csgdef{Xendbeforelemmaseparator@#1}{0em}%
4133        \csgdef{Xendafterlemmaseparator@#1}{0.5em}%
4134        \csgdef{Xendinplaceoflemmaseparator@#1}{0.5em}%
4135
4136        \newtoggle{Xendparagraph@#1}%
4137        \csgdef{Xendafternote@#1}{1em plus.4em minus.4em}%
4138        \csgdef{Xendsep@#1}{}%
4139
4140        \csgdef{Xendinplaceofnumber@#1}{0pt}%
4141        \newtoggle{Xendnonumber@#1}%
4142 %
```

End of endnotes declaration

```
4143    \fi%
4144 %
```

Dump series in `\@series`

```
4145      \listxadd{\@series}{#1}
4146    }
4147 }% End of \newseries
4148 %
```

XX.6   Init standards series (A,B,C,D,E)

```
4149 \expandafter\newseries\expandafter{\default@series}
4150 %
```

XXI   Setting series display

XXI.1   Change series order

\seriesatbegin   \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.

```
4151  \newcommand{\seriesatbegin}[1]{%
4152      \StrDel{\@series}{#1}[\@series]%
4153      \edef\@new{}%
4154      \listeadd{\@new}{#1}%
4155      \listeadd{\@new}{\@series}%
4156      \xdef\@series{\@new}%
4157  }
4158  %
```

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

```
4159  \newcommand{\seriesatend}[1]{%
4160      \StrDel{\@series}{#1}[\@series]%
4161      \edef\@new{}%
4162      \listeadd{\@new}{\@series}%
4163      \listeadd{\@new}{#1}%
4164      \xdef\@series{\@new}%
4165  }
4166  %
```

XXI.2   Test series order

\ifseriesbefore   \ifseriesbefore{⟨*seriesA*⟩}{⟨*seriesB*⟩}{⟨*true*⟩}{⟨*false*⟩} expands ⟨*true*⟩ if ⟨*seriesA*⟩ is printed before ⟨*seriesB*⟩, expands ⟨*false*⟩ otherwise.

```
4167  \newcommand{\ifseriesbefore}[4]{%
4168   \StrPosition{\@series}{#1}[\@first]%
4169   \StrPosition{\@series}{#2}[\@second]%
4170   \ifnumgreater{\@second}{\@first}{#3}{#4}%
4171  }
4172  %
```

XXI.2.1   Get the first series

In some specific case, we need to know the first series of the list of series.

\@getfirstseries   ```
4173 \newcommand{\@getfirstseries}{%
4174   \ifdefempty{\@series}%
4175   {\xdef\@firstseries{}}%
4176   {\StrChar{\@series}{1}[\@firstseries]}%
4177 }%
4178 %
```

XXI.3   Series setting

XXI.3.1   General way of working

The setting's command (like \numberonlyfirstinline), also called "hooks" can be divided in two categories: those which require a string values and those which require

a boolean value. The first category includes those which require a length value, because we store the length's expression send by user and we evaluate it only in the commands which requires to know the setting. The second category require boolean value only when it is set to FALSE. Otherwise, we understand the insinuated value is TRUE.

For each "hook" command, we store the value in commands (first category) or a `etoolbox`'s toggle (second category) which names are in the form `\<hook>@<series>`. For example when calling `\twolines{⟨sq.⟩}`, we store `sq.` in commands `\twolines@A`, `\twolines@B`, `\twolines@C`...for each series defined for use with `reledmac`, or, if the [⟨*series*⟩] optional argument was send, for each series of this argument.

These values are tested in some specific places, scattered in all the code, depending of their effects. The default values are defined by the `\newseries@` command.

In order to prevent code duplication, we have created some generic commands. Some of them change the value of any hook send as argument. Some other, getting a hook name, generate the user level commands.

XXI.3.2   Tools to set options

`\settoggle@series`   `\settoggle@series{⟨series⟩}{⟨toggle⟩}{⟨value⟩}` is a generic command to switch toggles for some series. The arguments are:

- #1 (mandatory): the series for which the hooks should be set. If empty, all the series will be affected.

- #2 (mandatory): the name of the hook.

- #3 (mandatory): the new value of toggle (true or false).

- #4 (optional): if equal to `reload`, reload the footnote setting (call again `\Xarrangement` or `\arrangementX` or ... depending of the footnote display).

- #5 (optional): if not empty, and if #1 is empty, change the hook setting for pseudo-series, as `appref`.

```
4179  \newcommandx{\settoggle@series}[5][4,5,usedefault]{%
4180      \def\do##1{%
4181        \global\settoggle{#2@##1}{#3}%
4182        \ifstrequal{#4}{reload}%
4183          {%
4184          \csuse{Xarrangement@\csuse{series@display##1}}{##1}%
4185          \csuse{arrangementX@\csuse{series@displayX##1}}{##1}%
4186          }%
4187          {}%
4188        }%
4189      \ifstrempty{#1}{%
4190              \dolistloop{\@series}%
4191              \ifstrempty{#5}{}{%
4192                 \docsvlist{#5}%
4193               }
4194            }%
```

```
4195              {%
4196                  \docsvlist{#1}%
4197              }%
4198 }
4199 %
```

\setcommand@series   \setcommand@series{⟨*series*⟩}{⟨*command*⟩}{⟨*value*⟩} is a generic command to store hook's value into commands specific to some series. The arguments are:

- #1 (mandatory): the series for which the hooks should be set. If empty, all the series will be affected.

- #2 (mandatory): the name of the hook.

- #3 (mandatory): the new value of the hook/command.

- #4 (optional): if equal to reload, reload the footnote setting (call \footnormal or \footparagraph or ... depending of the footnote display).

- #5 (optional): if not empty, and if #1 is empty, change the hook setting for pseudo-series, as appref.

```
4200 \newcommandx{\setcommand@series}[5][4,5,usedefault]{%
4201     \def\do##1{
4202         \csgdef{#2@##1}{#3}
4203     \ifstrequal{#4}{reload}{
4204         \csuse{Xarrangement@\csuse{series@display##1}}{##1}%
4205         \csuse{arrangementX@\csuse{series@displayX##1}}{##1}%
4206     }{}}
4207     \ifstrempty{#1}{%
4208             \dolistloop{\@series}%
4209             \ifstrempty{#5}{}{%
4210                \docsvlist{#5}
4211              }
4212     }%
4213     {%
4214             \docsvlist{#1}%
4215     }%
4216 }%
4217 %
```

XXI.3.3   Tools to generate options commands

\newhookcommand@series   \newhookcommand@series\command names is a generic command to add new commands for hooks, like \Xhsizetwocol. The first argument is the name of the hook, the second a comma separated list of pseudo-series where the hook can be used, like appref in the case of \Xtwolines. The second argument is also used to create commands named \<hookname><pseudoseries>, like \Xtwolinesappref.

```
4218 \newcommandx{\newhookcommand@series}[2][2,usedefault]{%
4219   \global\expandafter\newcommand\expandafter*\csname #1\endcsname[2][]{%
4220     \setcommand@series{##1}{#1}{##2}[][#2]%
4221   }%
4222   \ifstrempty{#2}{}{%
4223     \def\do##1{%
4224       \global\expandafter\newcommand\expandafter*\csname #1##1\endcsname
     [1]{%
4225         \csuse{#1}[##1]{####1}%
4226       }%
4227     }%
4228     \docsvlist{#2}%
4229   }%
4230 }
4231 %
```

`\newhooktoggle@series`   \newhooktoggle@series\command names is a generic command to add new commands for a new toggle hook, like \Xnumberonlyfirstinline. The second argument is also used to create commands named \<hookname><pseudoseries>, like \Xtwolinesbutnotmoreappref.

```
4232 \newcommandx{\newhooktoggle@series}[2][2,usedefault]{%
4233   \global\expandafter\newcommandx\expandafter*\csname #1\endcsname[2][1,2={
     true},usedefault]{%
4234     \settoggle@series{##1}{#1}{##2}[][#2]%
4235   }%
4236   \ifstrempty{#2}{}{%
4237     \def\do##1{%
4238       \global\expandafter\newcommand\expandafter*\csname #1##1\endcsname{%
4239         \csuse{#1}[##1]%
4240       }%
4241     }%
4242     \docsvlist{#2}%
4243   }%
4244 }
4245 %
```

`\newhooktoggle@series@reload`   \newhookcommand@toggle@reload does the same thing as \newhooktoggle@series but the commands created by this macro also reload the series arrangement.

```
4246 \newcommand{\newhooktoggle@series@reload}[1]{%
4247   \global\expandafter\newcommandx\expandafter*\csname #1\endcsname[2][1,2={
     true},usedefault]{%
4248     \settoggle@series{##1}{#1}{##2}[reload]%
4249   }%
4250 }%
4251 %
```

`\newhookcommand@series@reload`   \newhookcommand@series@reload does the same thing as \newhookcommand@series but the commands created by this macro also reload the series' arrangement.

```
4252  \newcommand{\newhookcommand@series@reload}[1]{%
4253    \global\expandafter\newcommand\expandafter*\csname #1\endcsname[2][]{%
4254      \setcommand@series{##1}{#1}{##2}[reload]%
4255    }%
4256  }
4257  %
```

XXI.3.4   Options for critical notes

Before generating the commands that are used to set the critical notes, such as \Xnumberonlyfirstinline, \Xlemmaseparator and the like, we check the nocritical option.

```
4258  \unless\ifnocritical@
4259    \newhooktoggle@series{Xparindent}
4260    \newhookcommand@series{Xtwolines}[appref]
4261    \newhookcommand@series{Xmorethantwolines}[appref]
4262    \newhooktoggle@series{Xtwolinesbutnotmore}[appref]
4263    \newhooktoggle@series{Xtwolinesonlyinsamepage}[appref]
4264    \newhookcommand@series{Xhangindent}
4265    \newhookcommand@series{Xragged}
4266    \newhookcommand@series{Xhsizetwocol}
4267    \newhookcommand@series{Xhsizethreecol}
4268    \newhookcommand@series{Xcolalign}%
4269    \newhookcommand@series{Xnotenumfont}
4270    \newhookcommand@series{Xbhooknote}
4271    \newhookcommand@series{Xboxsymlinenum}%
4272    \newhookcommand@series{Xsymlinenum}
4273    \newhookcommand@series{Xbeforenumber}
4274    \newhookcommand@series{Xafternumber}
4275    \newhookcommand@series{Xbeforesymlinenum}
4276    \newhookcommand@series{Xaftersymlinenum}
4277    \newhookcommand@series{Xinplaceofnumber}
4278    \newhookcommand@series{Xlemmaseparator}
4279    \newhookcommand@series{Xbeforelemmaseparator}
4280    \newhookcommand@series{Xafterlemmaseparator}
4281    \newhookcommand@series{Xinplaceoflemmaseparator}
4282    \newhookcommand@series{Xtxtbeforenotes}
4283    \newhookcommand@series@reload{Xafterrule}
4284    \newhooktoggle@series{Xnumberonlyfirstinline}
4285    \newhooktoggle@series{Xnumberonlyfirstintwolines}
4286    \newhooktoggle@series{Xnonumber}
4287    \newhooktoggle@series{Xpstart}
4288    \newhooktoggle@series{Xpstarteverytime}%
4289    \newhooktoggle@series{Xonlypstart}
4290    \newhooktoggle@series{Xnonbreakableafternumber}
4291    \newhooktoggle@series{Xlemmadisablefontselection}
4292    \newhookcommand@series@reload{Xmaxhnotes}
4293    \newhookcommand@series@reload{Xbeforenotes}
4294    \newhooktoggle@series@reload{Xnoteswidthliketwocolumns}%
```

```
4295    \newhookcommand@series{Xnotefontsize}
4296
4297    \newhookcommand@series{Xboxlinenum}%
4298    \newhookcommand@series{Xboxlinenumalign}%
4299
4300    \newhookcommand@series{Xboxstartlinenum}%
4301    \newhookcommand@series{Xboxendlinenum}%
4302
4303    \newhookcommand@series{Xafternote}%
4304    \newhookcommand@series{Xparafootsep}
4305
4306  \fi
4307  %
```

XXI.3.5   Options for familiar notes

Before generating the optional commands for familiar notes, we check the `\nofamiliar`
option.

```
4308  \unless\ifnofamiliar@
4309    \newhooktoggle@series{parindentX}
4310    \newhookcommand@series{hangindentX}
4311    \newhookcommand@series{raggedX}
4312    \newhookcommand@series{hsizetwocolX}
4313    \newhookcommand@series{hsizethreecolX}
4314    \newhookcommand@series{colalignX}%
4315    \newhookcommand@series{notenumfontX}
4316    \newhookcommand@series{bhooknoteX}
4317    \newhookcommand@series@reload{beforenotesX}
4318    \newhookcommand@series@reload{maxhnotesX}
4319    \newhooktoggle@series@reload{noteswidthliketwocolumnsX}%
4320    \newhookcommand@series@reload{afterruleX}
4321    \newhookcommand@series{notefontsizeX}
4322    \newhookcommand@series{afternoteX}
4323    \newhookcommand@series{parafootsepX}
4324  \fi
4325  %
```

XXI.3.6   Options for endnotes

Before generating the commands that are used to set the endnotes, such as `\Xnumberonlyfirstinline`,
`\Xlemmaseparator+` and the like, we check the noend option.

```
4326  \unless\ifnoend@
4327    \newhookcommand@series{Xendtwolines}[apprefwithpage]
4328    \newhookcommand@series{Xendmorethantwolines}[apprefwithpage]
4329    \newhooktoggle@series{Xendtwolinesbutnotmore}[apprefwithpage]
4330    \newhooktoggle@series{Xendtwolinesonlyinsamepage}[apprefwithpage]
4331    \newhookcommand@series{Xendnotenumfont}
```

```
4332    \newhookcommand@series{Xendbhooknote}
4333
4334    \newhookcommand@series{Xendboxlinenum}%
4335    \newhookcommand@series{Xendboxlinenumalign}%
4336
4337    \newhookcommand@series{Xendboxstartlinenum}%
4338    \newhookcommand@series{Xendboxendlinenum}%
4339
4340    \newhookcommand@series{Xendnotefontsize}
4341    \newhooktoggle@series{Xendlemmadisablefontselection}
4342    \newhookcommand@series{Xendlemmaseparator}
4343    \newhookcommand@series{Xendbeforelemmaseparator}
4344    \newhookcommand@series{Xendafterlemmaseparator}
4345    \newhookcommand@series{Xendinplaceoflemmaseparator}
4346
4347    \newhooktoggle@series{Xendparagraph}
4348    \newhookcommand@series{Xendafternote}
4349    \newhookcommand@series{Xendsep}
4350
4351    \newhookcommand@series{Xendinplaceofnumber}%
4352    \newhooktoggle@series{Xendnonumber}%
4353 \fi
4354 %
```

XXI.4 Hooks for a particular footnote

\fulllines@ \fulllines@ toggle is used to print the full lines references, and not the abbreviated form defined by \Xtwolines and \Xmorethantwolines.

```
4355 \newtoggle{fulllines@}%
4356 %
```

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

```
4357 \newtoggle{nonum@}
4358 %
```

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

```
4359 \newtoggle{nosep@}
4360 %
```

\nomk@ \nomk@ toggle is used by reledpar to remove the footnote mark in the text when using \footnoteXmk. Read reledpar handbook.

```
4361 \newtoggle{nomk@}%
4362 %
```

XXI.5   Alias

\Xnolemmaseparator   \Xnolemmaseparator[⟨*series*⟩] is just an alias for \Xlemmaseparator[⟨*series*⟩]{}.

4363 \newcommandx*{\Xnolemmaseparator}[1][1]{\Xlemmaseparator[#1]{}}
4364 %

XXII   Output routine

Now we begin the output routine and associated things.

XXII.0.1   Page number management

\pageno   \pageno is a page number, starting at 1, and \advancepageno increments the number.
\advancepageno
4365 \countdef\pageno=0 \pageno=1
4366 \newcommand*{\advancepageno}{\ifnum\pageno<\z@ \global\advance\pageno\m@ne
4367   \else\global\advance\pageno\@ne\fi}
4368
4369 %

XXII.0.2   Extra footnotes output

With luck we might only have to change \@makecol and \@reinserts of the LaTeX's
kernel. Since reledmac, we use etoolbox's patching commands instead of overriding.
It should provides better compatibility with other package which modify these com-
mands

\doxtrafeet   \doxtrafeet is the code extending \@makecol to cater for the extra reledmac feet.
                 We have two categories of extra footnotes. By default, we order the footnote inserts so
                 that the regular footnotes of LaTeX are first, then familiar familiar footnotes and finally
                 the critical footnotes.

4370 \newcommand*{\l@ddoxtrafeet}{%
4371   \IfStrEq{familiar-critical}{\@fnpos}
4372   {\do@feetX\Xdo@feet}%
4373   {%
4374   \IfStrEq{critical-familiar}{\@fnpos}%
4375   {\Xdo@feet\do@feetX}%
4376   {\do@feetX\Xdo@feet}%
4377   }%
4378 }%
4379
4380 %

\Xdo@feet   \Xdo@feet is the code extending \@makecol to cater for the extra critical feet.

```
4381 \newcommand*{\Xdo@feet}{%
4382   \setbox\@outputbox \vbox{%
4383     \unvbox\@outputbox
4384     \@opXfeet}}
4385 %
```

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

```
4386 \newcommand\print@Xnotes[1]{%
4387   \csuse{#1footstart}{#1}%
4388   \csuse{#1footgroup}{#1}%%
4389 }%
4390 %
```

We print all series of notes by looping on them. We check before printing them that
they are not voided.

```
4391 \newcommand*{\@opXfeet}{%
4392   \unless\ifnocritical@%
4393     \gdef\firstXseries@{}%
4394     \def\do##1{%
4395       \ifvoid\csuse{##1footins}\else%
4396         \global\skip\csuse{##1footins}=\csuse{Xbeforenotes@##1}%
4397         \global\advance\skip\csuse{##1footins} by\csuse{Xafterrule@##1}%
4398         \print@Xnotes{##1}%
4399       \fi%
4400     }%
4401     \dolistloop{\@series}%
4402   \fi%
4403 }%
4404 %
```

\l@ddodoreinxtrafeet \l@ddodoreinxtrafeet is the code for catering for the extra footnotes within \@reinserts.
We use the same category and ordering as in \l@ddoxtrafeet.

```
4405 \newcommand*{\l@ddodoreinxtrafeet}{%
4406   \IfStrEq{familiar-critical}{\@fnpos}
4407     {\@doreinfeetX\X@doreinfeet}%
4408     {%
4409     \IfStrEq{critical-familiar}{\@fnpos}%
4410       {\X@doreinfeet\@doreinfeetX}%
4411       {\@doreinfeetX\X@doreinfeet}%
4412     }%
4413 }
4414
4415 %
```

\X@doreinfeet \X@doreinfeet is the code for catering for the extra critical footnotes within \@reinserts.

```
4416 \newcommand*{\X@doreinfeet}{%
4417   \unless\ifnocritical@%
4418     \def\do##1{%
4419       \ifvoid\csuse{##1footins}\else%
4420         \insert\csuse{##1footins}{\unvbox\csuse{##1footins}}%
4421       \fi}%
4422     \dolistloop{\@series}
4423   \fi%
4424 }
4425
4426 %
```

\print@notesX   We have to add all the new kinds of familiar footnotes to the output routine. The normal
\do@feetX   way to add one series. \print@Xnotes is replaced by reledpar when using \Pages.
\@doreinfeetX

```
4427 \newcommand\print@notesX[1]{%
4428   \csuse{footstart#1}{#1}%
4429   \csuse{footgroup#1}{#1}%
4430 }%
4431 %
```

We print all the series of notes by looping on them. We check before printing them that
they are not voided.

```
4432 \newcommand*{\do@feetX}{%
4433   \unless\ifnofamiliar@%
4434     \gdef\firstseriesX@{}%
4435     \setbox\@outputbox \vbox{%
4436       \unvbox\@outputbox%
4437       \def\do##1{%
4438       \ifvoid\csuse{footins##1}\else%
4439         \global\skip\csuse{footins##1}=\csuse{beforenotesX@##1}%
4440         \global\advance\skip\csuse{footins##1} by\csuse{afterruleX@##1}%
4441         \print@notesX{##1}%
4442       \fi%
4443       }%
4444       \dolistloop{\@series}}%
4445   \fi%
4446 }%
4447
4448 \newcommand{\@doreinfeetX}{%
4449   \unless\ifnofamiliar@%
4450     \def\do##1{%
4451       \ifvoid\csuse{footins##1}\else
4452         \insert%
4453           \csuse{footins##1}
4454           {\unvbox\csuse{footins##1}}%
4455       \fi%
4456     }%
4457     \dolistloop{\@series}%
4458   \fi%
```

209

```
4459 }%
4460
4461 %
```

XXII.0.3   Standard output's commands patching

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.)

```
4462 \@ifclassloaded{memoir}{%
4463 %
```

memoir is loaded so we use memoir's built in hooks.

```
4464   \g@addto@macro{\m@mdoextrafeet}{\l@ddoxtrafeet}%
4465   \g@addto@macro{\m@mdodoreinextrafeet}{\l@ddodoreinxtrafeet}%
4466   }{%
4467 %
```

memoir has not been loaded, so patch \@makecol and \@reinserts.

```
4468   \@ifpackageloaded{fancyhdr}{%
4469     \patchcmd%
4470       {\latex@makecol}%
4471       {\xdef\@freelist{\@freelist\@midlist}}%
4472       {\xdef\@freelist{\@freelist\@midlist}\l@ddoxtrafeet}%
4473       {}%
4474       {\led@error@fail@patch@@makecol}%
4475     }{%
4476     \patchcmd%
4477       {\@makecol}%
4478       {\xdef\@freelist{\@freelist\@midlist}}%
4479       {\xdef\@freelist{\@freelist\@midlist}\l@ddoxtrafeet}%
4480       {}%
4481       {\led@error@fail@patch@@makecol}%
4482     }%
4483
4484   \patchcmd%
4485     {\@reinserts}%
4486     {\ifvbox}%
4487     {\l@ddodoreinxtrafeet\ifvbox}%
4488     {}%
4489     {\led@error@fail@patch@@reinserts}%
4490 }
4491
4492 %
```

It turns out that \@doclearpage also needs modifying.

\if@led@nofoot   We have to check if there are any leftover feet.

```
4493  \newif\if@led@nofoot
4494
4495  %
```

```
4496  \@ifclassloaded{memoir}{%
4497  %
```

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

`\@mem@extranofeet`
```
4498  \g@addto@macro{\@mem@extranofeet}{%%
4499    \def\do#1{%
4500      \unless\ifnocritical@%
4501        \ifvoid\csuse{#1footins}\else\@mem@nofootfalse\fi%
4502      \fi%
4503      \unless\ifnofamiliar@%
4504        \ifvoid\csuse{footins#1}\else\@mem@nofootfalse\fi%
4505      \fi%
4506      }
4507    \dolistloop{\@series}%
4508    }%
4509  }{%
4510  %
```

As memoir is not loaded we have patch \@doclearpage.

`\@led@testifnofoot`
`\@doclearpage`
```
4511  \newcommand*{\@led@testifnofoot}{%
4512    \@led@nofoottrue%
4513    \ifvoid\footins\else%
4514      \@led@nofootfalse%
4515    \fi%
4516    \def\do##1{%
4517      \unless\ifnocritical@%
4518        \ifvoid\csuse{##1footins}\else%
4519          \@led@nofootfalse%
4520        \fi%
4521      \fi%
4522      \unless\ifnofamiliar@%
4523        \ifvoid\csuse{footins##1}\else%
4524          \@led@nofootfalse%
4525        \fi%
4526      \fi%
4527    }%
4528    \dolistloop{\@series}%
4529  }%
4530
4531  \pretocmd%
4532    {\@doclearpage}%
4533    {\@led@testifnofoot}%
4534    {}%
```

```
4535  {\led@error@fail@patch@@doclearpage}%
4536
4537  \patchcmd%
4538  {\@doclearpage}%
4539  {\ifvoid\footins}%
4540  {\if@led@nofoot}%
4541  {}%
4542  {\led@error@fail@patch@@doclearpage}%
4543
4544  }
4545
4546  %
```

XXIII   Cross referencing

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*⟩} or \pstartref. These reference commands will produce, respectively, the page, line sub-line and pstart on which the \edlabel{⟨*foo*⟩} command occurred.

The reference macros warn you if a reference is made to an undefined label. If {⟨*foo*⟩} has been used as a label before, the \edlabel{⟨*foo*⟩} command will issue a complaint; subsequent \edpageref and \edlineref commands will refer to the latest occurrence of \edlabel{⟨*foo*⟩}.

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

```
4547  \list@create{\labelref@list}
4548  %
```

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

```
4549  \newcommand*{\zz@@@}{000|000} % set two counters to zero in one go
4550
4551  %
```

\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.[31]

This version of the original edmac \label uses \@bsphack and \@esphack to eliminate extra space problems and also use the LATEX write methods for the .aux file.

---

[31] The remaining macros in this section were kindly revised by Wayne Sullivan, who substantially improved their efficiency and flexibility.

Jesse Billett[32] found that the original code could be off by several pages. This version, hopefully cures that, and also allows for non-arabic page numbering.

```
4552 \newcommand*{\edlabel}[1]{%
4553   \ifl@dpairing\ifautopar%
4554     \strut%
4555   \fi\fi%
4556   \@bsphack%
4557   \ifledRcol%
4558     \write\linenum@outR{\string\@lab}%
4559     \ifx\labelref@listR\empty%
4560       \xdef\label@refs{\zz@@@}%
4561     \else%
4562       \gl@p\labelref@listR\to\label@refs%
4563     \fi%
4564     \ifvmode%
4565         \advancelabel@refs%
4566     \fi%
4567 %
```

Use code from the kernel \label command to write the correct page number. Also define an hypertarget if hyperref package is loaded.

```
4568     \protected@write\@auxout{}%
4569       {\string\l@dmake@labelsR\space\thepage|\label@refs|\the\c@pstartR
|{#1}}%
4570     \ifdef{\hypertarget}{\Hy@raisedlink{\hypertarget{#1}{}}}{}%
4571   \else%
4572     \write\linenum@out{\string\@lab}%
4573     \ifx\labelref@list\empty%
4574       \xdef\label@refs{\zz@@@}%
4575     \else%
4576       \gl@p\labelref@list\to\label@refs%
4577     \fi%
4578     \ifvmode%
4579         \advancelabel@refs%
4580     \fi%
4581   \protected@write\@auxout{}%
4582     {\string\l@dmake@labels\space\thepage|\label@refs|\the\c@pstart|{#1}}%
4583     \ifdef{\hypertarget}{\Hy@raisedlink{\hypertarget{#1}{}}}{}%
4584   \fi%
4585 \@esphack}%
4586
4587 %
```

\advancelabel@refs   In cases where \edlabel is the first element in a paragraph, we have a problem with
\labelrefsparseline   line counts, because line counts change only at the first horizontal box of the paragraph.
\labelrefsparsesubline Hence, we need to test \edlabel if it occurs at the start of a paragraph. To do so, we

---

[32](jdb43@cam.ac.uk) via the ctt thread 'ledmac cross referencing', 25 August 2003.

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.

```
4588 \newcounter{line}%
4589 \newcounter{subline}%
4590 \newcommand{\advancelabel@refs}{%
4591     \setcounter{line}{\expandafter\labelrefsparseline\label@refs}%
4592     \stepcounter{line}%
4593     \ifsublines@%
4594         \setcounter{subline}{\expandafter\labelrefsparsesubline\label@refs}%
4595         \stepcounter{subline}{1}%
4596         \def\label@refs{\theline|\thesubline}%
4597     \else%
4598         \def\label@refs{\theline|0}%
4599     \fi%
4600 }
4601 \def\labelrefsparseline#1|#2{#1}
4602 \def\labelrefsparsesubline#1|#2{#2}
4603 %
```

`\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).

#1 page number, #2 line number, #3 sub-line number, #4 pstart number, #5 label.

```
4604 \newcommand*{\l@dmake@labels}{}
4605 \def\l@dmake@labels#1|#2|#3|#4|#5{%
4606   \expandafter\ifx\csname the@label#5\endcsname \relax\else
4607     \led@warn@DuplicateLabel{#5}%
4608   \fi
4609   \expandafter\gdef\csname the@label#5\endcsname{#1|#2|#3|#4}%
4610   \ignorespaces}
4611
4612 %
```

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.

```
4613 \AtBeginDocument{%
4614   \def\l@dmake@labels#1|#2|#3|#4|#5{}%
4615 }
4616
4617 %
```

`\@lab` The `\@lab` command, which appears in the `\linenum@out` file, appends the current values of page, line and sub-line to the `\labelref@list`. These values are defined by

the earlier \@page, \@nl, and the \sub@on and \sub@off commands appearing in the
\linenum@out file.

LaTeX uses the page counter for page numbers. However, it appears that this is
not the right place to grab the page number. That task is now done in the \edlabel
macro. This version of \@lab appends just the current line and sub-line numbers to
\labelref@list.

```
4618
4619  \newcommand*{\@lab}{%
4620    \ifledRcol
4621      \xright@appenditem{\linenumr@p{\line@numR}|%
4622        \ifsublines@ \sublinenumr@p{\subline@numR}\else 0\fi}%
4623      \to\labelref@listR
4624    \else
4625      \xright@appenditem{\linenumr@p{\line@num}|%
4626        \ifsublines@ \sublinenumr@p{\subline@num}\else 0\fi}%
4627      \to\labelref@list
4628    \fi}
4629  %
```

\applabel   \applabel, if called in \edtext will insert automatically both a start and an end label
              for the current edtext lines.

```
4630  \newcommand*{\applabel}[1]{%
4631    \ifnum\@edtext@level>0%
4632  %
```

Label should not be already defined.

```
4633        \ifcsundef{the@label#1}{%
4634          \csdef{the@label#1}{applabel}%
4635        }%
4636        {%
4637          \led@warn@DuplicateLabel{#1 (applabel)}%
4638        }%
4639  %
```

Parse the \edtext line numbers.

```
4640        \expandafter\l@dp@rsefootspec\l@d@nums|%
4641  %
```

Use the LaTeX standard hack for label.

```
4642        \@bsphack%
4643  %
```

And now, write the data in the auxiliary file.

```
4644        \ifledRcol%
4645          \protected@write\@auxout{}%
4646            {\string\l@dmake@labelsR\space\l@dparsedstartpage|\
      l@dparsedstartline|\l@dparsedstartsub|\the\c@pstartR|{#1:start}}%
```

```
4647          \ifdef{\hypertarget}{\Hy@raisedlink{\hypertarget{#1:start}{}}}{}%
4648          \protected@write\@auxout{}%
4649            {\string\l@dmake@labelsR\space\l@dparsedendpage|\l@dparsedendline
       |\l@dparsedendsub|\the\c@pstartR|{#1:end}}%
4650        \else%
4651          \protected@write\@auxout{}%
4652            {\string\l@dmake@labels\space\l@dparsedstartpage|\
       l@dparsedstartline|\l@dparsedstartsub|\the\c@pstart|{#1:start}}%
4653          \ifdef{\hypertarget}{\Hy@raisedlink{\hypertarget{#1:start}{}}}{}%
4654          \protected@write\@auxout{}%
4655            {\string\l@dmake@labels\space\l@dparsedendpage|\l@dparsedendline
       |\l@dparsedendsub|\the\c@pstart|{#1:end}}%
4656        \fi%
4657  %
```

Use the LaTeX standard hack for label.

```
4658          \@esphack%
4659  %
```

Warning if \applabel is called outside of \edtext.

```
4660    \else%
4661      \led@warn@AppLabelOutEdtext{#1}%
4662    \fi%
4663  %
```

End of \applabel

```
4664  }%
4665  %
```

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

```
4666  \newrobustcmd{\wrap@edcrossref}[2]{%
4667    \ifdef{\hyperlink}%
4668      {\hyperlink{#1}{#2}}%
4669      {#2}%
4670  }
4671  %
```

\edpageref   If the specified label exists, \edpageref gives its page number.
\xpageref   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 do not print error mes-
             sages 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.

```
4672  \newcommand*{\edpageref}[1]{\l@dref@undefined{#1}\wrap@edcrossref{#1}{\
      l@dgetref@num{1}{#1}}}
4673  \newcommand*{\xpageref}[1]{\l@dgetref@num{1}{#1}}
4674
4675  %
```

\edlineref   If the specified label exists, \lineref gives its line number.
\xlineref
```
4676  \newcommand*{\edlineref}[1]{\l@dref@undefined{#1}\wrap@edcrossref{#1}{\
      l@dgetref@num{2}{#1}}}%
4677  \newcommand*{\xlineref}[1]{\l@dgetref@num{2}{#1}}%
4678
4679  %
```

\sublineref   If the specified label exists, \sublineref gives its sub-line number.
\xsublineref
```
4680  \newcommand*{\sublineref}[1]{\l@dref@undefined{#1}\wrap@edcrossref{#1}{\
      l@dgetref@num{3}{#1}}}
4681  \newcommand*{\xsublineref}[1]{\l@dgetref@num{3}{#1}}
4682
4683  %
```

\pstarteref   If the specified label exists, \pstartref gives its pstart number.
\xpstartref
```
4684  \newcommand*{\pstartref}[1]{\l@dref@undefined{#1}\wrap@edcrossref{#1}{\
      l@dgetref@num{4}{#1}}}
4685  \newcommand*{\xpstartref}[1]{\l@dgetref@num{4}{#1}}
4686
4687  %
```

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.

```
4688  \newcommand*{\l@dref@undefined}[1]{%
4689    \expandafter\ifx\csname the@label#1\endcsname\relax
4690      \led@warn@RefUndefined{#1}%
4691    \fi}
4692
4693  %
```

\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), sub-line (3) or (4) pstart 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.

```
4694 \newcommand*{\l@dgetref@num}[2]{%
4695   \expandafter
4696   \ifx\csname the@label#2\endcsname \relax
4697     000%
4698   \else
4699     \expandafter\expandafter\expandafter
4700     \l@dlabel@parse\csname the@label#2\endcsname|#1%
4701   \fi}
4702
4703 %
```

\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, 3 or 4) which defines which of the earlier three numbers to pick out. (It was earlier given as the first argument of \l@dgetref@num.)

```
4704 \newcommand*{\l@dlabel@parse}{}
4705 \def\l@dlabel@parse#1|#2|#3|#4|#5{%
4706   \ifcase #5%
4707   \or #1%
4708   \or #2%
4709   \or #3%
4710   \or #4%
4711   \fi}
4712 %
```

\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 does not, 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 {elephant}. The point of this is to be able to manufacture footnote line references to passages which can not be specified in the normal way as the first argument to \edtext for one reason or another. Using \xxref in the second argument of \edtext lets you set things up at least semi-automatically.

```
4713 \newcommand*{\xxref}[2]{%
4714   {%
4715     \expandafter\ifx\csname the@label#1\endcsname \relax%
4716       \expandafter\let\csname the@@label#1\endcsname\zz@@@%
4717     \else%
4718       \expandafter\def\csname the@@label#1\endcsname{\l@dgetref@num
4719 {1}{#1}|\l@dgetref@num{2}{#1}|\l@dgetref@num{3}{#1}}%
4719     \fi%
4720     \expandafter\ifx\csname the@label#2\endcsname \relax%
4721       \expandafter\let\csname the@@label#2\endcsname\zz@@@%
4722     \else%
```

```
4723        \expandafter\def\csname the@@label#2\endcsname{\l@dgetref@num
      {1}{#2}|\l@dgetref@num{2}{#2}|\l@dgetref@num{3}{#2}}%
4724      \fi%
4725      \ifdefined\@Rlineflag%
4726      \StrDel{\csuse{the@@label#1}}{\@Rlineflag}[\@tempa]%
4727        \StrDel{\csuse{the@@label#2}}{\@Rlineflag}[\@tempb]%
4728    \else%
4729      \letcs{\@tempa}{the@@label#1}%
4730      \letcs{\@tempb}{the@@label#2}%
4731  \fi%
4732      \linenum{\@tempa|%
4733        \@tempb}}}%
4734
4735  %
```

\appref
\apprefwithpage
\@apprefprefixsingle
\@apprefprefixmore
\setapprefprefixsingle
\setapprefprefixmore

\appref prints a crossref to some lines of the apparatus defined by \applabel. It prints the lines as they should be printed in the apparatus.

If \@apprefprefixsingle is not empty, it prints it before the line number. If \@apprefprefixmore is not empty, it prints it before the line numbers when the first line is not the same as the last line. \apprefwithpage prints a crossref to some lines of the apparatus defined by \applabel. It always prints the page number, as it should be printed in the end notes. The \Xtwolinesappref and \Xmorethantwolinesappref are similar to the footnote hooks and \Xtwolines \Xmorethantwolines.

So, first declare the default value of the hooks for the pseudo-series appref. Also declare the internal toggle which are switch by reledmac.

```
4736  \xdef\Xtwolines@appref{}%
4737  \xdef\Xmorethantwolines@appref{}%
4738  \newtoggle{Xtwolinesbutnotmore@appref}%
4739  \newtoggle{Xtwolinesonlyinsamepage@appref}%
4740
4741  \xdef\Xendtwolines@apprefwithpage{}%
4742  \xdef\Xendmorethantwolines@apprefwithpage{}%
4743  \newtoggle{Xendtwolinesbutnotmore@apprefwithpage}%
4744  \newtoggle{Xendtwolinesonlyinsamepage@apprefwithpage}%
4745
4746  %
```

Note that some of these hooks are declared but no user command can change their values. Such hooks are not pertinent for appref and apprefwithpage pseudo-series, but their values are nonetheless tested in some macros.

```
4747
4748  \xdef\Xboxstartlinenum@appref{0pt}
4749  \xdef\Xboxendlinenum@appref{0pt}
4750
4751  \xdef\Xendboxstartlinenum@apprefwithpage{0pt}
4752  \xdef\Xendboxendlinenum@apprefwithpage{0pt}
4753
4754  %
```

Now, declare the default value of `\@apprefprefixsingle` and `\@apprefprefixmore`, and the commands which defines them

```
4755 \newcommand\@apprefprefixsingle{}%
4756 \newcommand\@apprefprefixmore{}%
4757
4758 \newcommand{\apprefprefixsingle}[1]{%
4759   \gdef\@apprefprefixsingle{#1}%
4760 }
4761
4762 \newcommand{\setapprefprefixmore}[1]{%
4763   \gdef\@apprefprefixmore{#1}%
4764 }
4765
4766 %
```

And now, the main commands: `\appref` and `\apprefwithpage`. These commands call `\printlines` and `\printendlines`. That is why we have previously declared all hooks values tested inside these last commands.

```
4767 \newcommandx{\appref}[2][1,usedefault]{%
4768   \IfStrEq{#1}{fulllines}%
4769     {\toggletrue{fulllines@}}%
4770     {}%
4771   \xdef\@currentseries{appref}%
4772   \ifdefempty{\@apprefprefixmore}%
4773     {\@apprefprefixsingle}%
4774     {%
4775       \IfEq{\xlineref{#2:start}}{\xlineref{#2:end}}%
4776         {\@apprefprefixsingle}%
4777         {\@apprefprefixmore}%
4778     }%
4779   \printlines\xpageref{#2:start}|\xlineref{#2:start}|\xsublineref{#2:start
    }|\xpageref{#2:end}|\xlineref{#2:end}|\xsublineref{#2:end}||%
4780   \togglefalse{fulllines@}%
4781 }%
4782
4783 % \changes{v1.23.0}{2015/05/18}{Debug \protect\cs{Xendtwolines}, \protect\
    cs{Xendmorethantwolines}, \protect\cs{Xendtwolinesbutnotmore} and \protect\
    cs{Xendtwolinesonlyinsamepage} when using \protect\cs{apprefwithpage}.}
4784 \newcommandx{\apprefwithpage}[2][1,usedefault]{%
4785   \IfStrEq{#1}{fulllines}%
4786     {\toggletrue{fulllines@}}%
4787     {}%
4788   \xdef\@currentseries{apprefwithpage}%
4789   \printendlines\xpageref{#2:start}|\xlineref{#2:start}|\xsublineref{#2:
    start}|\xpageref{#2:end}|\xlineref{#2:end}|\xsublineref{#2:end}||%
4790   \togglefalse{fulllines@}%
4791 }%
4792 %
```

\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, fol-
                lowed by a page and a line number(s) as arguments. LaTeX defines a \makelabel macro
                which is used in lists. Peter Wilson has changed the name to \edmakelabel.

```
4793 \newcommand*{\edmakelabel}[2]{\expandafter\xdef\csname the@label#1\
     endcsname{#2}}
4794
4795 %
```

(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 VI.3 p. 109 and V.9 p. 80), since \xxref makes
a call to \linenum in order to do its work.)

XXIV   Side notes

Regular \marginpars do not work inside numbered text — they do not produce any
note but do put an extra unnumbered blank line into the text.

\@xympar   Changing \@xympar a little at least ensures that \marginpars in numbered text do not
            disturb the flow.

```
4796 \pretocmd{\@xympar}%
4797   {\ifnumberedpar@
4798     \led@warn@NoMarginpars
4799     \@esphack
4800   \else}%
4801   {}%
4802   {}%
4803
4804 \apptocmd{\@xympar}%
4805   {\fi}%
4806   {}
4807   {}
4808
4809 %
```

We provide side notes as replacement for \marginpar in numbered text.

\sidenote@margin   These are the sidenote equivalents to \line@margin and \linenummargin for spec-
\sidenotemargin   ifying which margin. The default is the right margin (opposite to the default for line
\l@dgetsidenote@margin numbers). \l@dgetsidenote@margin returns the number associated to side note mar-
                        gin:

left : 0

right : 1

outer : 2

inner : 3

```
4810  \newcount\sidenote@margin
4811  \newcommand*{\sidenotemargin}[1]{{%
4812    \l@dgetsidenote@margin{#1}%
4813    \ifnum\@l@dtempcntb>\m@ne
4814      \ifledRcol
4815        \global\sidenote@marginR=\@l@dtempcntb
4816      \else
4817        \global\sidenote@margin=\@l@dtempcntb
4818      \fi
4819    \fi}}
4820  \newcommand*{\l@dgetsidenote@margin}[1]{%
4821    \def\@tempa{#1}\def\@tempb{left}%
4822    \ifx\@tempa\@tempb
4823      \@l@dtempcntb \z@
4824    \else
4825      \def\@tempb{right}%
4826      \ifx\@tempa\@tempb
4827        \@l@dtempcntb \@ne
4828      \else
4829        \def\@tempb{outer}%
4830        \ifx\@tempa\@tempb
4831          \@l@dtempcntb \tw@
4832        \else
4833          \def\@tempb{inner}%
4834          \ifx\@tempa\@tempb
4835            \@l@dtempcntb \thr@@
4836          \else
4837            \led@warn@BadSidenotemargin
4838            \@l@dtempcntb \m@ne
4839          \fi
4840        \fi
4841      \fi
4842    \fi}
4843  \sidenotemargin{right}
4844
4845  %
```

\l@dlp@rbox We need two boxes to store sidenote texts.
\l@drp@rbox
```
4846  \newbox\l@dlp@rbox
4847  \newbox\l@drp@rbox
4848
4849  %
```

\ledlsnotewidth   These specify the width of the left/right boxes (initialised to \marginparwidth), their
\ledrsnotewidth   distance from the text (initialised to \linenumsep), and the fonts used.
\ledlsnotesep
\ledrsnotesep
\ledlsnotefontsetup
\ledrsnotefontsetup

```
4850 \newdimen\ledlsnotewidth \ledlsnotewidth=\marginparwidth
4851 \newdimen\ledrsnotewidth \ledrsnotewidth=\marginparwidth
4852 \newdimen\ledlsnotesep \ledlsnotesep=\linenumsep
4853 \newdimen\ledrsnotesep \ledrsnotesep=\linenumsep
4854 \newcommand*{\ledlsnotefontsetup}{\raggedleft\footnotesize}
4855 \newcommand*{\ledrsnotefontsetup}{\raggedright\footnotesize}
4856
4857 %
```

\ledleftnote   \ledleftnote, \ledrightnote, \ledinnernote, \ledouternote are the user com-
\ledrightnote   mands for left, right, inner and outer sidenotes. The two last one are just alias for the
\ledinnernote   two first one, depending of the page number. \ledsidenote{⟨*text*⟩} is the command
\ledouterote   for a moveable sidenote.
\ledsidenote

```
4858 \newcommand*{\ledleftnote}[1]{\edtext{}{\l@dlsnote{#1}}}
4859 \newcommand*{\ledrightnote}[1]{\edtext{}{\l@drsnote{#1}}}
4860
4861 \newcommand*{\ledinnernote}[1]{%
4862   \ifodd\c@page% Do not use \page@num, because it is not yet calculated
when command is called
4863     \ledleftnote{#1}%
4864   \else%
4865     \ledrightnote{#1}%
4866   \fi%
4867 }
4868
4869 \newcommand*{\ledouternote}[1]{%
4870   \ifodd\c@page% Do not use \page@num, because it is not yet calculated
when command is called
4871     \ledrightnote{#1}%
4872   \else%
4873     \ledleftnote{#1}%
4874   \fi%
4875 }
4876
4877 \newcommand*{\ledsidenote}[1]{\edtext{}{\l@dcsnote{#1}}}
4878 %
```

\l@dlsnote   . The 'footnotes' for left, right, and moveable sidenotes. The whole scheme is reminiscent
\l@drsnote   of the critical footnotes code.
\l@dcsnote

```
4879 \newif\ifrightnoteup
4880   \rightnoteuptrue
4881
4882 \newcommand*{\l@dlsnote}[1]{%
4883   \begingroup%
4884   \newcommand{\content}{#1}%
4885   \ifnumberedpar@
4886     \ifledRcol%
4887       \xright@appenditem{\noexpand\vl@dlsnote{\expandonce\content}}%
```

```
4888                                     \to\inserts@listR
4889          \global\advance\insert@countR \@ne%
4890        \else%
4891          \xright@appenditem{\noexpand\vl@dlsnote{\expandonce\content}}%
4892                                     \to\inserts@list
4893          \global\advance\insert@count \@ne%
4894        \fi
4895    \fi\ignorespaces\endgroup}
4896
4897 \newcommand*{\l@drsnote}[1]{%
4898    \begingroup%
4899    \newcommand{\content}{#1}%
4900    \ifnumberedpar@
4901      \ifledRcol%
4902        \xright@appenditem{\noexpand\vl@drsnote{\expandonce\content}}%
4903                                     \to\inserts@listR
4904          \global\advance\insert@countR \@ne%
4905      \else%
4906        \xright@appenditem{\noexpand\vl@drsnote{\expandonce\content}}%
4907                                     \to\inserts@list
4908          \global\advance\insert@count \@ne%
4909      \fi
4910    \fi\ignorespaces\endgroup}
4911
4912 \newcommand*{\l@dcsnote}[1]{%
4913    \begingroup%
4914    \newcommand{\content}{#1}%
4915    \ifnumberedpar@
4916      \ifledRcol%
4917        \xright@appenditem{\noexpand\vl@dcsnote{\expandonce\content}}%
4918                                     \to\inserts@listR
4919          \global\advance\insert@countR \@ne%
4920      \else%
4921        \xright@appenditem{\noexpand\vl@dcsnote{\expandonce\content}}%
4922                                     \to\inserts@list
4923          \global\advance\insert@count \@ne%
4924      \fi
4925    \fi\ignorespaces\endgroup}
4926
4927 %
```

\vl@dlsnote Put the left/right text into boxes, but just save the moveable text. \l@dcsnotetext,
\vl@drsnote \l@dcsnotetext@l and \l@dcsnotetext@r are etoolbox's lists which will store the
\vl@dcsnote content of side notes. We store the content in lists, because we need to loop later on
             them, in case many sidenote co-exist for the same line. That is there some special test
             to do, in order to:

- Store the content of \ledsidenote to \l@dcsnotetext in any cases.

- Store the content of \rightsidenote to:

- – \l@dcsnotetext if \ledsidenote is to be put on right.

- – \l@dcsnotetext@r if \ledsidenote is to be put on left.

- • Store the content of \leftsidenote to:

  - – \l@dcsnotetext if \ledsidenote is to be put on left.

  - – \l@dcsnotetext@l if \ledsidenote is to be put on right.

```
4928 \newcommand*{\vl@dlsnote}[1]{%
4929   \ifledRcol@%
4930     \@l@dtempcntb=\sidenote@marginR%
4931     \ifnum\@l@dtempcntb>\@ne%
4932       \advance\@l@dtempcntb by\page@numR%
4933     \fi%
4934   \else%
4935     \@l@dtempcntb=\sidenote@margin%
4936     \ifnum\@l@dtempcntb>\@ne%
4937       \advance\@l@dtempcntb by\page@num%
4938     \fi%
4939   \fi%
4940   \ifodd\@l@dtempcntb%
4941     \listgadd{\l@dcsnotetext@l}{#1}%
4942   \else%
4943     \listgadd{\l@dcsnotetext}{#1}%
4944   \fi
4945 }
4946 \newcommand*{\vl@drsnote}[1]{%
4947   \ifledRcol@%
4948     \@l@dtempcntb=\sidenote@marginR%
4949     \ifnum\@l@dtempcntb>\@ne%
4950       \advance\@l@dtempcntb by\page@numR%
4951     \fi%
4952   \else%
4953     \@l@dtempcntb=\sidenote@margin%
4954     \ifnum\@l@dtempcntb>\@ne%
4955       \advance\@l@dtempcntb by\page@num%
4956     \fi%
4957   \fi%
4958   \ifodd\@l@dtempcntb%
4959     \listgadd{\l@dcsnotetext}{#1}%
4960   \else%
4961     \listgadd{\l@dcsnotetext@r}{#1}%
4962   \fi%
4963 }
4964 \newcommand*{\vl@dcsnote}[1]{\listgadd{\l@dcsnotetext}{#1}}
4965
4966 %
```

`\setl@dlp@rbox` `\setl@dlprbox{`⟨*lednums*⟩`}{`⟨*tag*⟩`}{`⟨*text*⟩`}` puts ⟨*text*⟩ into the `\l@dlp@rbox` box.
`\setl@drpr@box` And similarly for the right side box. It is these boxes that finally get displayed in the margins.

```
4967 \newcommand*{\setl@dlp@rbox}[1]{%
4968   {\parindent\z@\hsize=\ledlsnotewidth\ledlsnotefontsetup
4969     \global\setbox\l@dlp@rbox
4970     \ifleftnoteup
4971       =\vbox to\z@{\vss #1}%
4972     \else
4973       =\vbox to 0.70\baselineskip{\strut#1\vss}%
4974     \fi}}
4975 \newcommand*{\setl@drp@rbox}[1]{%
4976   {\parindent\z@\hsize=\ledrsnotewidth\ledrsnotefontsetup
4977     \global\setbox\l@drp@rbox
4978     \ifrightnoteup
4979       =\vbox to\z@{\vss#1}%
4980     \else
4981       =\vbox to0.7\baselineskip{\strut#1\vss}%
4982     \fi}}
4983 \newif\ifleftnoteup
4984   \leftnoteuptrue
4985 %
```

`\@sidenotesep` This macro is used to separate sidenotes of the same line.

```
4986 \newcommand{\setsidenotesep}[1]{\gdef\@sidenotesep{#1}}
4987 \newcommand{\@sidenotesep}{, }
4988 %
```

`\affixside@note` This macro puts any moveable sidenote text into the left or right sidenote box, depending on which margin it is meant to go in. It's a very much stripped down version of `\affixlin@num`.

Before do it, we concatenate all moveable sidenotes of the line, using `\@sidenotesep` as separator. It is the result that we put on the sidenote.

```
4989 \newcommand*{\affixside@note}{%
4990     \def\sidenotecontent@{}%
4991     \numgdef{\itemcount@}{0}%
4992     \def\do##1{%
4993         \ifnumequal{\itemcount@}{0}%
4994             {%
4995             \appto\sidenotecontent@{##1}}% Not print not separator before
the 1st note
4996             {\appto\sidenotecontent@{\@sidenotesep ##1}%
4997             }%
4998             \numgdef{\itemcount@}{\itemcount@+1}%
4999     }%
5000     \dolistloop{\l@dcsnotetext}%
5001     \ifnumgreater{\itemcount@}{1}{\led@err@ManySidenotes}{}%
5002 %
```

And we do the same for left and right notes (not movable).

```
5003    \gdef\@templ@d{}%
5004    \gdef\@templ@n{\l@dcsnotetext\l@dcsnotetext@l\l@dcsnotetext@r}%
5005    \ifx\@templ@d\@templ@n \else%
5006      \if@twocolumn%
5007        \if@firstcolumn%
5008          \setl@dlp@rbox{##1}{\sidenotecontent@}%
5009        \else%
5010          \setl@drp@rbox{\sidenotecontent@}%
5011        \fi%
5012      \else%
5013        \@l@dtempcntb=\sidenote@margin%
5014        \ifnum\@l@dtempcntb>\@ne%
5015          \advance\@l@dtempcntb by\page@num%
5016        \fi%
5017        \ifodd\@l@dtempcntb%
5018          \setl@drp@rbox{\sidenotecontent@}%
5019          \gdef\sidenotecontent@{}%
5020          \numgdef{\itemcount@}{0}%
5021          \dolistloop{\l@dcsnotetext@l}%
5022          \ifnumgreater{\itemcount@}{1}{\led@err@ManyLeftnotes}{}%
5023          \setl@dlp@rbox{\sidenotecontent@}%
5024        \else%
5025          \setl@dlp@rbox{\sidenotecontent@}%
5026          \gdef\sidenotecontent@{}%
5027          \numgdef{\itemcount@}{0}%
5028          \dolistloop{\l@dcsnotetext@r}%
5029          \ifnumgreater{\itemcount@}{1}{\led@err@ManyRightnotes}{}%
5030          \setl@drp@rbox{\sidenotecontent@}%
5031        \fi%
5032      \fi%
5033    \fi%
5034  }
5035  %
```

XXV   Minipages and such

We can put footnotes into minipages. The preparatory code has been set up earlier, all that remains is to ensure that it is available inside a minipage box. This requires some alteration to the kernel code, specifically the `\@iiiminipage` and `\endminipage` macros. We will arrange this so that additional series can be easily added.

`\l@dfeetbeginmini`   These will be the hooks in `\@iiiminpage` and `\endminipage`.
`\l@dfeetendmini`   They can be extended to handle other things if necessary.

```
5036  \ifnoledgroup@\else%
5037  \newcommand*{\l@dfeetbeginmini}{\l@dedbeginmini\l@dfambeginmini}
5038  \newcommand*{\l@dfeetendmini}{%
```

```
5039        \IfStrEq{critical-familiar}{\@mpfnpos}%
5040          {\l@dedendmini\l@dfamendmini}%
5041          {%
5042              \IfStrEq{familiar-critical}{\@mpfnpos}%
5043                  {\l@dfamendmini\l@dedendmini}%
5044                  {\l@dedendmini\l@dfamendmini}%
5045          }%
5046      }%
5047 %
```

\l@dedbeginmini These handle the initiation and closure of critical footnotes in a minipage environment.
\l@dedendmini

```
5048 \newcommand*{\l@dedbeginmini}{%
5049   \unless\ifnocritical@%
5050     \def\do##1{\csletcs{v##1footnote}{mpv##1footnote}}%
5051     \dolistloop{\@series}%
5052   \fi%
5053   }
5054 \newcommand*{\l@dedendmini}{%
5055   \unless\ifnocritical@%
5056     \ifl@dpairing%
5057       \ifledRcol%
5058         \flush@notesR%
5059       \else%
5060         \flush@notes%
5061       \fi%
5062     \fi
5063     \def\do##1{%
5064       \ifvoid\csuse{mp##1footins}\else%
5065         \ifl@dpairing\ifparledgroup%
5066           \ifledRcol%
5067             \dimgdef{\parledgroup@beforenotesR}{\parledgroup@beforenotesR+\
     skip\@nameuse{mp##1footins}}%
5068           \else%
5069             \dimgdef{\parledgroup@beforenotesL}{\parledgroup@beforenotesL
     +\skip\@nameuse{mp##1footins}}%
5070           \fi%
5071         \fi\fi%
5072         \csuse{mp##1footgroup}{##1}%
5073       \fi}%
5074     \dolistloop{\@series}%
5075   \fi%
5076 }%
5077
5078 %
```

\l@dfambeginmini These handle the initiation and closure of familiar footnotes in a minipage environment.
\l@dfamendmini

```
5079 \newcommand*{\l@dfambeginmini}{%
5080  \unless\ifnofamiliar@%
```

```
5081      \def\do##1{\csletcs{vfootnote##1}{mpvfootnote##1}}%
5082        \dolistloop{\@series}%
5083   \fi%
5084 }%

5085
5086 \newcommand*{\l@dfamendmini}{%
5087   \unless\ifnofamiliar@%
5088     \def\do##1{%
5089       \ifvoid\csuse{mpfootins##1}\else%
5090       \csuse{mpfootgroup##1}{##1}%
5091     \fi}%
5092     \dolistloop{\@series}%
5093   \fi%
5094 }%
5095 %
```

\@iiiminipage   This is our extended form of the kernel \@iiiminipage defined in ltboxes.dtx.

```
5096 \patchcmd%
5097   {\@iiiminipage}%
5098   {\let\@footnotetext\@mpfootnotetext}%
5099   {\let\@footnotetext\@mpfootnotetext\l@dfeetbeginmini}%
5100   {}%
5101   {\led@error@fail@patch@@iiiminipage}%
5102 %
```

\endminipage   This is our extended form of the kernel \endminipage defined in ltboxes.dtx.

```
5103 \patchcmd%
5104   {\endminipage}%
5105   {\footnoterule}%
5106   {\footnoterule\l@advance@parledgroup@beforenormalnotes}%
5107   {}%
5108   {\led@error@fail@patch@endminipage}
5109
5110 \patchcmd%
5111   {\endminipage}%
5112   {\@minipagefalse}%
5113   {\l@dfeetendmini\@minipagefalse}%
5114   {}%
5115   {\led@error@fail@patch@endminipage}
5116
5117 %
```

\l@dunboxmpfoot   \@ldunboxmpfoot insert normal footnotes for ledgroup.
\l@advance@parledgroup@beforenormalnotes
```
5118 \newcommand*{\l@dunboxmpfoot}{%
5119     \vskip\skip\@mpfootins
5120     \normalcolor
5121     \footnoterule
```

```
5122        \l@advance@parledgroup@beforenormalnotes
5123        \unvbox\@mpfootins%
5124 }
5125 %
```

When using parallel ledgroup, we need to store the vertical space added before footnote, in order to compensate them between left and right pages.

```
5126 \newcommand{\l@advance@parledgroup@beforenormalnotes}{%
5127   \ifparledgroup
5128     \ifl@dpairing
5129       \ifledRcol
5130           \dimgdef{\parledgroup@beforenotesR}{\parledgroup@beforenotesR+\
   skip\@mpfootins}
5131       \else
5132           \dimgdef{\parledgroup@beforenotesL}{\parledgroup@beforenotesL+\
   skip\@mpfootins}
5133       \fi
5134     \fi
5135   \fi
5136 }
5137 %
```

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.

```
5138 \newenvironment{ledgroup}{%
5139   \resetprevpage@num%
5140   \def\@mpfn{mpfootnote}\def\thempfn{\thempfootnote}\c@mpfootnote\z@%
5141   \let\@footnotetext\@mpfootnotetext
5142   \l@dfeetbeginmini%
5143 }{%
5144   \par
5145   \unskip
5146   \ifvoid\@mpfootins\else
5147     \l@dunboxmpfoot
5148   \fi
5149   \l@dfeetendmini%
5150 }
5151
5152 %
```

ledgroupsized \begin{ledgroupsized}[⟨pos⟩]{⟨width⟩}
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 ⟨width⟩ minipage. The optional ⟨pos⟩ controls the sideways position of numbered text.

```
5153 \newenvironment{ledgroupsized}[2][l]{%
5154 %
```

Set the various text measures.

```
5155   \hsize #2\relax
5156 %%  \textwidth #2\relax
5157 %%  \columnwidth #2\relax
5158 %
```

Initialize fills for centering.

```
5159   \let\ledllfill\hfil
5160   \let\ledrlfill\hfil
5161   \def\@tempa{#1}\def\@tempb{l}%
5162 %
```

Left adjusted numbered lines

```
5163       \ifx\@tempa\@tempb
5164     \let\ledllfill\relax
5165   \else
5166     \def\@tempb{r}%
5167     \ifx\@tempa\@tempb
5168 %
```

Right adjusted numbered lines

```
5169       \let\ledrlfill\relax
5170     \fi
5171   \fi
5172 %
```

Set up the footnoting.

```
5173   \def\@mpfn{mpfootnote}\def\thempfn{\thempfootnote}\c@mpfootnote\z@
5174   \let\@footnotetext\@mpfootnotetext
5175   \l@dfeetbeginmini%
5176 }{%
5177   \par
5178   \unskip
5179   \ifvoid\@mpfootins\else
5180     \l@dunboxmpfoot
5181   \fi
5182   \l@dfeetendmini%
5183 }
5184
5185 %
```

Close the \ifnoledgroup@\else.

```
5186 \fi%
5187 %
```

\ifledgroupnotesL@   These boolean tests check if we are in the notes of a ledgroup. If we are, we do not
\ifledgroupnotesR@   number the lines. It could be useful for parallel ledgroup of reledpar.

```
5188 \newif\ifledgroupnotesL@
5189 \newif\ifledgroupnotesR@
5190 %
```

XXVI   Indexing

Here is some code for indexing using page and line numbers.

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

```
5191 \AtBeginDocument{%
5192   \unless\ifl@imakeidx%
5193     \@ifpackageloaded{imakeidx}{\led@error@ImakeidxAfterEledmac}{}%
5194   \fi%
5195   \unless\ifl@indextools%
5196     \@ifpackageloaded{indextools}{\led@error@indextoolsAfterEledmac}{}%
5197   \fi%
5198 }
5199 %
```

\pagelinesep   In order to get a correct line number we have to use the label/ref mechanism. These
\edindexlab   macros are for that.
\c@labidx
```
5200 \newcommand{\pagelinesep}{-}
5201 \newcommand{\edindexlab}{$&}
5202 \newcounter{labidx}
5203 \setcounter{labidx}{0}
5204
5205 %
```

\doedindexlabel   This macro sets an \edlabel.

```
5206 \newcommand{\doedindexlabel}{%
5207   \stepcounter{labidx}%
5208   \edlabel{\edindexlab\thelabidx}%
5209 }
5210
5211 %
```

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

```
5212 \newcommand{\thepageline}{%
5213   \thepage%
5214   \pagelinesep%
5215   \xlineref{\edindexlab\thelabidx}%
5216 }
5217 %
```

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

```
5218 \newcommand{\thestartpageline}{%
5219   \l@dparsedstartpage%
5220   \pagelinesep%
5221   \l@dparsedstartline%
```

```
5222 }
5223 \newcommand{\theendpageline}{%
5224   \l@dparsedendpage%
5225   \pagelinesep%
5226   \l@dparsedendline%
5227 }
5228 %
```

\if@edindex@fornote@true   This boolean test is switching at the beginning of each critical note, to allow index re-
ferring to this note.

```
5229 \newif\if@edindex@fornote@
5230 %
```

\prepare@edindex@fornote   This macro is called at the beginning of each critical note. It switches some parameters,
to allow index referring to this note, with reference to page and line number. It also
defines \@ledinnote@command which will be printed as an encapsulating command
after the |.

```
5231 \newcommand{\prepare@edindex@fornote}[1]{%
5232     \l@dp@rsefootspec#1|%
5233     \@edindex@fornote@true%
5234 }
5235 %
```

\get@edindex@ledinnote@command   The \get@edindex@ledinnote@command macro defines a \@ledinnote@command
command which is added as an attribute (text inserted after |) of the next index entry.
    Consequently, we write the definition of the location reference attribute in the .xdy
file.

```
5236 \newcommand{\get@edindex@ledinnote@command}{%
5237   \ifxindy@%
5238     \gdef\@ledinnote@command{%
5239       ledinnote\thelabidx%
5240     }%
5241     \ifxindyhyperref@%
5242       \immediate\write\eledmac@xindy@out{%
5243         (define-attributes ("ledinnote\thelabidx"))^^J
5244         \space\space(markup-locref^^J
5245           \eledmacmarkuplocrefdepth^^J
5246           :open "\string\ledinnote[\edindexlab\thelabidx]{\@index@command
}{"^^J
5247           :close "}"^^J
5248           :attr "ledinnote\thelabidx"^^J
5249           )
5250         }%
5251     \else%
5252       \immediate\write\eledmac@xindy@out{%
5253         (define-attributes ("ledinnote\thelabidx"))^^J
```

```
5254        \space\space(markup-locref^^J
5255          \eledmacmarkuplocrefdepth^^J
5256          :open "\string\ledinnote{\@index@command}{"^^J
5257          :close "}"^^J
5258          :attr "ledinnote\thelabidx"^^J
5259          )
5260        }%
5261      \fi%
5262 %
```

If we do not use xindy option, \@ledinnote@command will produce something like
ledinnote{formatingcommand}.

```
5263    \else%
5264      \gdef\@ledinnote@command{%
5265          ledinnote[\edindexlab\thelabidx]{\@index@command}%
5266      }%
5267    \fi%
5268 }
5269 %
```

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

```
5270 \def\get@index@command#1|#2+{%
5271    \gdef\@index@txt{#1}%
5272    \gdef\@index@command{#2}%
5273    \xdef\@index@parenthesis{}%
5274    \IfBeginWith{\@index@command}{(}{%
5275      \StrGobbleLeft{\@index@command}{1}[\@index@command@]%
5276      \global\let\@index@command\@index@command@%
5277      \xdef\@index@parenthesis{(}%
5278      }{}%
5279    \IfBeginWith{\@index@command}{)}{%
5280      \StrGobbleLeft{\@index@command}{1}[\@index@command@]%
5281      \global\let\@index@command\@index@command@%
5282      \xdef\@index@parenthesis{)}%
5283      }{}%
5284 }
5285 %
```

\ledinnote   These macros are used to specify that an index reference points to a note. Arguments
\ledinnotehyperpage   of \ledinnote are: #1 (optional): the label for the hyperlink, #2: command applied to
\ledinnotemark   the number, #3: the number itself.

```
5286 \newcommandx\ledinnote[3][1,usedefault]{%
5287    \ifboolexpr{%
5288      test{\ifdefequal{\iftrue}{\ifHy@hyperindex}}%
5289      or%
5290      bool {xindyhyperref@}%
5291      }%
```

```
5292        {%
5293        \csuse{#2}{\hyperlink{#1}{\ledinnotemark{#3}}}%
5294        }%
5295        {%
5296        \csuse{#2}{\ledinnotemark{#3}}%
5297        }%
5298 }%
5299 \newcommand{\ledinnotehyperpage}[2]{\csuse{#1}{\ledinnotemark{\hyperpage
     {#2}}}}%
5300 \newcommand{\ledinnotemark}[1]{#1\emph{n}}%
5301 %
```

Eledmac and ledmac were using the specific indexing tools of the memoir in order to
allow multiple index. However, eledmac used imakeidx or indextools tools when
one these two package was loaded. This system forced to maintained a double code,
which was not very useful. Since reledmac, we use only the imakeidx or indextools
tools.

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

\edindex   Write the index information to the idx file.

\@wredindex
```
5302 \newcommandx{\@wredindex}[2][1=\expandonce\jobname,usedefault]{%#1 = the
     index name, #2 = the text
5303   \global\let\old@Rlineflag\@Rlineflag%
5304   \gdef\@Rlineflag{}%
5305   \ifl@imakeidx%
5306     \if@edindex@fornote@%
5307       \IfSubStr[1]{#2}{|}{\get@index@command#2+}{\get@index@command#2|+}%
5308       \get@edindex@ledinnote@command%
5309       \expandafter\imki@wrindexentry{#1}{\@index@txt|(\@ledinnote@command
     }{\thestartpageline}%
5310       \expandafter\imki@wrindexentry{#1}{\@index@txt|)\@ledinnote@command
     }{\theendpageline}%
5311     \else%
5312       \get@edindex@hyperref{#2}%
5313       \imki@wrindexentry{#1}{\@index@txt\@edindex@hyperref}{\thepageline}%
5314     \fi%
5315   \else%
5316     \if@edindex@fornote@%
5317       \IfSubStr[1]{#2}{|}{\get@index@command#2+}{\get@index@command#2|+}%
5318       \get@edindex@ledinnote@command%
5319       \expandafter\protected@write\@indexfile{}%
5320   {\string\indexentry{\@index@txt|(\@ledinnote@command}{\thestartpageline}
5321     }%
5322       \expandafter\protected@write\@indexfile{}%
5323   {\string\indexentry{\@index@txt|)\@ledinnote@command}{\theendpageline}
5324         }%
5325     \else%
```

```
5326        \protected@write\@indexfile{}%
5327   {\string\indexentry{#2}{\thepageline}
5328   }%
5329      \fi%
5330    \fi%
5331    \endgroup
5332    \global\let\@Rlineflag\old@Rlineflag%
5333    \@esphack%
5334 }
5335 %
```

Need to add the definition of \edindex to \makeindex, and initialise \edindex to do nothing.

```
5336 \pretocmd{\makeindex}{%
5337   \def\edindex{\@bsphack
5338   \doedindexlabel
5339   \begingroup
5340   \@sanitize
5341   \@wredindex}}{}{}
5342 \newcommand{\edindex}[1]{\@bsphack\@esphack}
5343 %
```

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

```
5344 \newcommand{\hyperlinkformat}[3]{%
5345   \ifstrempty{#1}%
5346     {\hyperlink{#2}{#3}}%
5347     {\csuse{#1}{\hyperlink{#2}{#3}}%
5348   }}
5349 %
```

\hyperlinkR \hyperlinkR command is to be used to create a internal hyperlink and \ledRflag, when indexing.

```
5350 \newcommand{\hyperlinkR}[2]{%
5351  \hyperlink{#1}{#2\@Rlineflag}%
5352 }%
5353
5354 %
```

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

```
5355 \newcommand{\hyperlinkformatR}[3]{%
5356   \hyperlinkformat{#1}{#2}{#3\@Rlineflag}%
5357 }%
5358
5359 %
```

`\get@edindex@hyperref`
`\@edindex@hyperref`

`\get@edindex@hyperref` is to be used to define the `\@edindex@hyperref` macro, which, in index, links to the point where the index was called (with `hyperref`.

```
5360 \newcommand{\get@edindex@hyperref}[1]{%
5361 %
```

We have to disable temporary spaces to work through a `xstring` bug (or feature?)

```
5362     \edef\temp@{%
5363     \catcode`\ =9 %space need for catcode
5364     #1%
5365     \catcode`\ =10 % space need for catcode
5366     }%
5367 %
```

Now, we define `\@edindex@hyperref` if the `hyperindex` of `hyperref` is enabled.

```
5368     \ifdefequal{\iftrue}{\ifHy@hyperindex}{%
5369       \IfSubStr{\temp@}{|}%
5370         {\get@index@command#1+%
5371         \ifledRcol%
5372           \gdef\@edindex@hyperref{|\@index@parenthesis %space kept
5373           hyperlinkformatR{\@index@command}%
5374           {\edindexlab\thelabidx}}%
5375         \else%
5376           \gdef\@edindex@hyperref{|\@index@parenthesis %space kept
5377           hyperlinkformat{\@index@command}%
5378           {\edindexlab\thelabidx}}%
5379         \fi%
5380         }%
5381         {\get@index@command#1|+%
5382         \ifledRcol%
5383          \gdef\@edindex@hyperref{|hyperlinkR{\edindexlab\thelabidx}}%
5384         \else%
5385          \gdef\@edindex@hyperref{|hyperlink{\edindexlab\thelabidx}}%
5386         \fi%
5387         }%
5388       }%
5389 %

5390 % If we use both xindy and hyperref, first get the \protect\cs{
     index@command} command.
5391 % Then define \protect\cs{@edindex@hyperref} in the form \verb+eledmacXXX+
5392 %     \begin{macrocode}
5393     {\ifxindyhyperref@%
5394       \IfSubStr{\temp@}{|}%
5395         {\get@index@command#1+}%
5396         {\get@index@command#1|+}%
5397       \gdef\@edindex@hyperref{|eledmac\thelabidx}%
5398 %
```

If we start a reference range by a opening parenthesis, store the `\thelabidx` for the current `\edindex`, then define `\@edindex@hyperref` in the form `|(eledmac\thelabidx`.

```
5399        \IfStrEq{\@index@parenthesis}{(}%
5400          {%
5401          \csxdef{xindyparenthesis@\@index@txt}{\thelabidx}%
5402          \gdef\@edindex@hyperref{|(eledmac\thelabidx}%
5403          }%
5404          {}%
5405 %
```

This \thelabidx will be called back at the closing parenthesis, to have the same number in \@edindex@hyperref command that we had at the opening parenthesis. \@edindex@hyperref start by a closing parenthesis, then followed by eledmacXXX where XXX is the \thelabidx of the opening \edindex.

```
5406        \IfStrEq{\@index@parenthesis}{)}%
5407          {%
5408          \xdef\@edindex@hyperref{|)eledmac\csuse{xindyparenthesis@\
@index@txt}}%
5409          \global\csundef{xindyparenthesis@\@index@txt}%
5410          }%
5411 %
```

Write in the .xdy file the attributes of the location.

```
5412            {%
5413          \immediate\write\eledmac@xindy@out{%
5414            (define-attributes ("eledmac\thelabidx"))^^J
5415            \space\space(markup-locref^^J
5416              \eledmacmarkuplocrefdepth^^J
5417            :open "\string\hyperlink%
5418                    \ifledRcol R\fi%
5419                    {\edindexlab\thelabidx}%
5420                    {\ifdefempty{\@index@command}%
5421                      {}%
5422                      {\@backslashchar\@index@command}%
5423                    {"^^J
5424          :close "}}"^^J
5425          :attr "eledmac\thelabidx"^^J
5426            )
5427          }%
5428        }%
5429 %
```

And now, in any other case.

```
5430      \else%
5431        \gdef\@index@txt{#1}%
5432        \gdef\@edindex@hyperref{}%
5433      \fi%
5434      }%
5435 }
5436 %
```

XXVII   Verse

The original code is principally Wayne Sullivan's code from `edstanza`. However, the
code has been many time modified by Maïeul Rouquette in order to obtain new features
and improved compatibility with `reledpar`.

XXVII.1   Hanging symbol management

`\@hangingsymbol` The macro `\@hangingsymbol` is used to insert a symbol on each hanging of verses. It
`\sethangingsymbol` is set by user level macro `\sethangingsymbol`.
`\ifinstanza`   For example, in french typographie the symbol is '['. We obtain it by the next code:

`\sethangingsymbol{[\,}`

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

```
5437 \def\@hangingsymbol{}
5438 \newcommand*{\sethangingsymbol}[1]{%
5439   \gdef\@hanginsymbol{#1}%
5440 }%
5441 \newif\ifinstanza
5442 %
```

`\inserthangingymbol` The boolean `\ifinserthangingsymbol` is set to TRUE when `\@lock` is greater than 1,
`\ifinserthangingsymbol` i.e. when we are not in the first line of a verse. The switch of `\ifinserthangingsymbol`
is made in `\do@line` before the printing of line but after the line number calculation.

```
5443 \newif\ifinserthangingsymbol
5444 \newcommand{\inserthangingsymbol}{%
5445 \ifinserthangingsymbol%
5446    \ifinstanza%
5447       \@hangingsymbol%
5448    \fi%
5449 \fi%
5450 }
5451 %
```

XXVII.2   Using & character

`\ampersand` Within a stanza the `\&` macro is going to be usurped. We need an alias in case an & needs
to be typeset in a stanza. Define it rather than letting it in case some other package has
already defined it.

```
5452 \newcommand*{\ampersand}{\char`\&}
5453
5454 %
```

XXVII.3   Code category setting

\stanza@count   Before we can define the main macros we need to save and reset some category codes.
\stanzaindentbase   To save the current values we use \next and \body from the \loop macro.

```
5455    \chardef\body=\catcode`\@
5456    \catcode`\@=11
5457    \chardef\next=\catcode`\&
5458    \catcode`\&=\active
5459
5460  %
```

XXVII.4   Stanza count and indent

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 indenta-
tions are multiples of this value. The default value of \stanzaindentbase is 20pt.

```
5461    \newcount\stanza@count
5462    \newlength{\stanzaindentbase}
5463    \setlength{\stanzaindentbase}{20pt}
5464
5465  %
```

\strip@szacnt   The indentations of stanza lines are non-negative integer multiples of the unit called
\setstanzavalues   \stanzaindentbase. To make it easier for the user to specify these numbers, some
list macros are defined. These take numerical values in a list separated by commas and
assign the values to special control sequences using \mathchardef. Though this does
limit the range from 0 to 32767, it should suffice for most applications, including *penal-
ties*, which will be discussed below.

```
5466  \def\strip@szacnt#1,#2|{\def\@tempb{#1}\def\@tempa{#2|}}
5467  \newcommand*{\setstanzavalues}[2]{\def\@tempa{#2,,|}%
5468          \stanza@count\z@
5469        \def\next{\expandafter\strip@szacnt\@tempa
5470              \ifx\@tempb\empty\let\next\relax\else
5471              \expandafter\mathchardef\csname #1@\number\stanza@count
5472              @\endcsname\@tempb\relax
5473              \advance\stanza@count\@ne\fi\next}%
5474        \next}
5475
5476  %
```

\setstanzaindents   In the original edmac, \setstanzavalues{sza}{⟨...⟩} had to be called to set the in-
\setstanzapenalties   dents, and similarly \setstanzavalues{szp}{⟨...⟩}. to set the penalties. \setstanzaindents
and \setstanzapenalties macros are a convenience to give the user one less thing
to worry about (misspelling the first argument).

```
5477  \newcommand*{\setstanzaindents}[1]{\setstanzavalues{sza}{#1}}
5478  \newcommand*{\setstanzapenalties}[1]{\setstanzavalues{szp}{#1}}
```

```
5479 %
5480 %
```

\managestanza@modulo   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.

```
5481 \newcounter{stanzaindentsrepetition}
5482 \newcount\stanza@modulo
5483
5484 \newcommand*{\managestanza@modulo}[0]{
5485     \advance\stanza@modulo\@ne
5486     \ifnum\stanza@modulo>\value{stanzaindentsrepetition}
5487       \stanza@modulo\@ne
5488     \fi
5489 }
5490 %
```

\stanzaindent   The macro \stanzaindent, when called at the beginning of a verse, changes the in
\stanzaindent*   dentation normally defined for this verse by \setstanzaindent. The starred version skips the current verse for the repetition of stanza indent.

```
5491 \newcommand{\stanzaindent}[1]{%
5492   \hspace{\dimexpr#1\stanzaindentbase-\parindent\relax}%
5493   \ignorespaces%
5494 }%
5495 \WithSuffix\newcommand\stanzaindent*[1]{%
5496   \stanzaindent{#1}%
5497   \global\advance\stanza@modulo-\@ne%
5498   \ifnum\stanza@modulo=0%
5499     \global\stanza@modulo=\value{stanzaindentsrepetition}%
5500   \fi%
5501   \ignorespaces%
5502 }%
5503 %
```

XXVII.5   Main work

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

```
5504  \newcommandx{\stanza@line}[1][1]{
5505      \ifnum\value{stanzaindentsrepetition}=0
5506          \parindent=\csname sza@\number\stanza@count
5507                  @\endcsname\stanzaindentbase
5508      \else
5509          \parindent=\csname sza@\number\stanza@modulo
5510                  @\endcsname\stanzaindentbase
5511          \managestanza@modulo
5512      \fi
5513      \pstart[#1]\stanza@hang\ignorespaces}
5514  \xdef\stanza@hang{\noexpand\leavevmode\noexpand\startlock
5515              \hangindent\expandafter
5516              \noexpand\csname sza@0@\endcsname\stanzaindentbase
5517              \hangafter\@ne}
5518  \def\sza@penalty{\count@\csname szp@\number\stanza@count @\endcsname
5519              \ifnum\count@>\@M\advance\count@-\@M\penalty-\else
5520              \penalty\fi\count@}
5521  %
```

<table>
<tr><td>\@startstanza</td><td rowspan="7">Now we have the components of the \stanza macro, which appears at the start of a group of lines. This macro initializes the count and checks to see if hanging indentation and penalties are to be included. Hanging indentation suspends the line count, so that the enumeration is by verse line rather than by print line. If the print line count is desired, invoke \let\startlock\relax and do the same for \endlock. Here and above we have used \xdef to make the stored macros take up a bit less space, but it also makes them more obscure to the reader. Lines of the stanza are delimited by ampersands &. The last line of the stanza must end with \&.</td></tr>
<tr><td>\stanza</td></tr>
<tr><td>\@stopstanza</td></tr>
<tr><td>\newverse</td></tr>
</table>

```
5522  \xdef\@startstanza[#1]{%
5523      \noexpand\instanzatrue\expandafter
5524      \begingroup%
5525      \catcode`\noexpand\&\active%
5526      \global\stanza@count\@ne\stanza@modulo\@ne
5527      \noexpand\ifnum\expandafter\noexpand
5528      \csname sza@0@\endcsname=\z@\let\noexpand\stanza@hang\relax
5529      \let\noexpand\endlock\relax\noexpand\else\interlinepenalty
5530      \@M\rightskip\z@ plus 1fil\relax\noexpand\fi\noexpand\ifnum
5531      \expandafter\noexpand\csname szp@0@\endcsname=\z@
5532      \let\noexpand\sza@penalty\relax\noexpand\fi%
5533      \def\noexpand&{%
5534              \noexpand\newverse[][]}%
5535      \def\noexpand\&{\noexpand\@stopstanza}%
5536      \noexpand\stanza@line[#1]}
5537
5538  \newcommandx{\stanza}[1][1,usedefault]{\@startstanza[#1]}
5539
5540  \newcommandx{\@stopstanza}[1][1,usedefault]{%
5541      \unskip%
5542      \endlock%
```

```
5543    \pend[#1]%
5544    \endgroup%
5545    \instanzafalse%
5546 }
5547
5548 \newcommandx*{\newverse}[2][1,2,usedefault]{%
5549    \unskip%
5550    \endlock\pend[#1]\sza@penalty\global%
5551    \advance\stanza@count\@ne\stanza@line[#2]%
5552    }
5553
5554 %
```

\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.

```
5555 \newcommand*{\flagstanza}[2][\stanzaindentbase]{%
5556    \hskip -#1\llap{#2}\hskip #1\ignorespaces}
5557
5558 %
```

XXVII.6   Restore catcode and penalties

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.

```
5559    \catcode`\&=\next
5560    \catcode`\@=\body
5561    \setstanzavalues{szp}{0}
5562
5563 %
```

XXVIII   Arrays and tables

XXVIII.1   Preamble: macro as environment

The following is borrowed, and renamed, from the amsmath package. See also the CTT
thread 'eeq and amstex', 1995/08/31, started by Keith Reckdahl and ended definitively
by David M. Jones.

Several of the [math] macros scan their body twice. This means we must collect all
text in the body of an environment form before calling the macro.

\@emptytoks This is actually defined in the amsgen package.

```
5564  \newtoks\@emptytoks
5565
5566  %
```

The rest is from `amsmath`.

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

```
5567  \newtoks\l@denvbody
5568
5569  %
```

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

```
5570  \newcommand{\addtol@denvbody}[1]{%
5571    \global\l@denvbody\expandafter{\the\l@denvbody#1}}
5572
5573  %
```

`\l@dcollect@body`   The macro `\l@dcollect@body` starts the scan for the `\end{⟨env⟩}` 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`.

```
5574  \newcommand{\l@dcollect@body}[1]{%
5575    \l@denvbody{\expandafter#1\expandafter{\the\l@denvbody}}%
5576    \edef\processl@denvbody{\the\l@denvbody\noexpand\end{\@currenvir}}%
5577    \l@denvbody\@emptytoks \def\l@dbegin@stack{b}%
5578    \begingroup
5579      \expandafter\let\csname\@currenvir\endcsname\l@dcollect@@body
5580      \edef\processl@denvbody{\expandafter\noexpand\csname\@currenvir\
      endcsname}%
5581      \processl@denvbody%
5582      }%
5583
5584  %
```

`\l@dpush@begins`   When adding a piece of the current environment's contents to `\l@denvbody`, we scan it to check for additional `\begin` tokens, and add a 'b' to the stack for any that we find.

```
5585  \def\l@dpush@begins#1\begin#2{%
5586    \ifx\end#2\else b\expandafter\l@dpush@begins\fi}
5587
5588  %
```

`\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.

```
5589  \def\l@dcollect@@body#1\end#2{%
5590    \edef\l@dbegin@stack{\l@dpush@begins#1\begin\end
5591                         \expandafter\@gobble\l@dbegin@stack}%
5592    \ifx\@empty\l@dbegin@stack
5593      \endgroup
5594      \@checkend{#2}%
5595      \addtol@denvbody{#1}%
5596    \else
5597      \addtol@denvbody{#1\end{#2}}%
5598    \fi
5599    \processl@denvbody % A little tricky! Note the grouping
5600  }
5601
5602  %
```

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>
```

XXVIII.2   Tabular environments

This is based on the work by Herbert Breger in developing `tabmac.tex`.

The original `tabmac.tex` file was void of comments or any explanatory text other than the above notice. The algorithm is Breger's. Peter Wilson have made some cosmetic changes to the original code and reimplemented some things so they are more LaTeX-like. All the commentary are from Peter Wilson, as are any mistake or errors.

However, Maïeul Rouquette has modified code in order to add new features of `eledmac` and `reledmac`.

XXVIII.2.1   Disabling and restoring commands

`\l@dtabnoexpands`   More no expansion for critical and familiar footnotes in tabular environment.

```
5603 \newcommand*{\l@dtabnoexpands}{%
5604   \let\rtab=0%
5605   \let\ctab=0%
5606   \let\ltab=0%
5607   \let\rtabtext=0%
5608   \let\ltabtext=0%
5609   \let\ctabtext=0%
5610   \let\edbeforetab=0%
5611   \let\edaftertab=0%
5612   \let\edatleft=0%
5613   \let\edatright=0%
5614   \let\edvertline=0%
5615   \let\edvertdots=0%
5616   \let\edrowfill=0%
5617 }
5618
5619 %
```

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

```
5620 \newcommand{\disable@familiarnotes}{%
5621   \unless\ifnofamiliar@%
5622     \def\do##1{%
5623         \csletcs{footnote@@##1}{footnote##1}%
5624         \expandafter\renewcommand \csname footnote##1\endcsname[1]{%
5625           \protected@csxdef{@thefnmark##1}{\csuse{thefootnote##1}}%
5626           \csuse{@footnotemark##1}%
5627         }%
5628     }%
5629     \dolistloop{\@series}%
5630   \fi%
5631 }%
5632 \newcommand{\restore@familiarnotes}{%
5633   \unless\ifnofamiliar@%
5634     \def\do##1{%
5635         \csletcs{footnote##1}{footnote@@##1}%
5636     }%
5637     \dolistloop{\@series}%
5638   \fi%
5639 }%
5640
5641 %
```

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

```
5642 \newcommand{\disable@sidenotes}{%
5643   \let\@@ledrightnote\ledrightnote%
5644   \let\@@ledleftnote\ledleftnote%
5645   \let\@@ledsidenote\ledsidenote%
5646   \let\ledrightnote\@gobble%
5647   \let\ledleftnote\@gobble%
5648   \let\ledsidenote\@gobble%
5649 }%
5650 \newcommand{\restore@sidenotes}{%
5651   \let\ledrightnote\@@ledrightnote%
5652   \let\ledleftnote\@@ledleftnote%
5653   \let\ledsidenote\@@ledsidenote%
5654 }%
5655 %
```

\disable@notes   Disable/restore side and familiar notes.
\restore@notes

```
5656 \newcommand{\disable@notes}{%
5657   \disable@sidenotes%
5658   \disable@familiarnotes%
5659 }%
5660 \newcommand{\restore@notes}{%
5661   \restore@sidenotes%
5662   \restore@familiarnotes%
5663 }%
```

```
5664 %
```

\EDTEXT   We need to be able to modify the \edtext macros and also restore their original defi-
\xedtext   nitions.

```
5665 \let\EDTEXT=\edtext
5666 \newcommand{\xedtext}[2]{\EDTEXT{#1}{#2}}
5667 %
```

\EDLABEL   We need to be able to modify and restore the \edlabel macro.
\xedlabel
```
5668 \let\EDLABEL=\edlabel
5669 \newcommand*{\xedlabel}[1]{\EDLABEL{#1}}
5670 %
```

\EDINDEX   Macros supporting modification and restoration of \edindex.
\xedindex
\nulledindex
```
5671 \let\EDINDEX=\edindex
5672 \newcommand{\xedindex}{\@bsphack%
5673   \@ifnextchar [{\l@d@index}{\l@d@index[\jobname]}}
5674 \newcommand{\nulledindex}[2][\jobname]{\@bsphack\@esphack}
5675
5676 %
```

\@line@@num   Macro supporting restoration of \linenum.

```
5677 \let\@line@@num=\linenum
5678 %
```

\l@dgobblearg   \l@dgobbleoptarg[⟨*arg*⟩]{⟨*arg*⟩} replaces these two arguments (first is optional) by
\relax.

```
5679 \newcommand*{\l@dgobbleoptarg}[2][]{\relax}%
5680
5681 %
```

\Relax   \let\Relax=\relax
\NEXT   \let\NEXT=\next
```
5682 \let\Relax=\relax
5683 \let\NEXT=\next
5684
5685 %
```

\l@dmodforedtext   Modify and restore various macros for when \edtext is used.
\l@drestoreforedtext
```
5686 \newcommand{\l@dmodforedtext}{%
5687   \let\edtext\relax
5688   \def\do##1{\global\csletcs{##1footnote}{l@dgobbleoptarg}}%
5689   \dolistloop{\@series}%
5690   \let\edindex\nulledindex
5691   \let\linenum\@gobble}
```

```
5692 \newcommand{\l@drestoreforedtext}{%
5693   \def\do##1{\global\csletcs{##1footnote}{##1@@footnote}}
5694   \dolistloop{\@series}%
5695   \let\edindex\xedindex}
5696 %
```

\l@dnullfills Nullify and restore some column fillers, etc.
\l@drestorefills

```
5697 \newcommand{\l@dnullfills}{%
5698   \def\edlabel##1{}%
5699   \def\edrowfill##1##2##3{}%
5700 }
5701 \newcommand{\l@drestorefills}{%
5702   \def\edrowfill##1##2##3{\@EDROWFILL@{##1}{##2}{##3}}%
5703 }
5704
5705 %
```

\letsforverteilen Gathers some lets and other code that is common to the *verteilen* macros.

```
5706 \newcommand{\letsforverteilen}{%
5707   \let\edtext\xedtext
5708   \let\edindex\xedindex
5709   \def\do##1{\global\csletcs{##1footnote}{##1@@footnote}}
5710   \dolistloop{\@series}%
5711   \let\linenum\@line@@num
5712   \hilfsskip=\l@dcolwidth%
5713   \advance\hilfsskip by -\wd\hilfsbox
5714   \def\edlabel##1{\xedlabel{##1}}}
5715
5716 %
```

\disablel@dtabfeet Declarations for using or using \edtext inside tabulars. The default at this point is for
\enablel@dtabfeet \edtext.

```
5717 \newcommand\disablel@dtabfeet{\l@dmodforedtext}%
5718 \newcommand\enablel@dtabfeet{\l@drestoreforedtext}%
5719 %
```

XXVIII.2.2   Counters, boxes and lengths

\l@dampcount \l@dampcount is a counter for the & column dividers and \l@dcolcount is a counter
\l@dcolcount for the columns.

```
5720 \newcount\l@dampcount
5721   \l@dampcount=1\relax
5722 \newcount\l@dcolcount
5723   \l@dcolcount=0\relax
5724
5725 %
```

`\hilfsbox` Some (temporary) helper items.
`\hilfsskip`
`\Hilfsbox`
`\hilfscount`

```
5726  \newbox\hilfsbox
5727  \newskip\hilfsskip
5728  \newbox\Hilfsbox
5729  \newcount\hilfscount
5730
5731  %
```

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).

```
5732  \newdimen\dcoli
5733  \newdimen\dcolii
5734  \newdimen\dcoliii
5735  \newdimen\dcoliv
5736  \newdimen\dcolv
5737  \newdimen\dcolvi
5738  \newdimen\dcolvii
5739  \newdimen\dcolviii
5740  \newdimen\dcolix
5741  \newdimen\dcolx
5742  \newdimen\dcolxi
5743  \newdimen\dcolxii
5744  \newdimen\dcolxiii
5745  \newdimen\dcolxiv
5746  \newdimen\dcolxv
5747  \newdimen\dcolxvi
5748  \newdimen\dcolxvii
5749  \newdimen\dcolxviii
5750  \newdimen\dcolxix
5751  \newdimen\dcolxx
5752  \newdimen\dcolxxi
5753  \newdimen\dcolxxii
5754  \newdimen\dcolxxiii
5755  \newdimen\dcolxxiv
5756  \newdimen\dcolxxv
5757  \newdimen\dcolxxvi
5758  \newdimen\dcolxxvii
5759  \newdimen\dcolxxviii
5760  \newdimen\dcolxxix
5761  \newdimen\dcolxxx
5762  \newdimen\dcolerr    % added for error handling
5763
5764  %
```

`\l@dcolwidth` This is a cunning way of storing the columnwidths indexed by the column number `\l@dcolcount`, like an array. (was `\Dimenzuordnung`)

```
5765  \newcommand{\l@dcolwidth}{\ifcase \the\l@dcolcount \dcoli %???
```

```
5766     \or \dcoli \or \dcolii \or \dcoliii
5767     \or \dcoliv \or \dcolv \or \dcolvi
5768     \or \dcolvii \or \dcolviii \or \dcolix \or \dcolx
5769     \or \dcolxi \or \dcolxii \or \dcolxiii
5770     \or \dcolxiv \or \dcolxv \or \dcolxvi
5771     \or \dcolxvii \or \dcolxviii \or \dcolxix \or \dcolxx
5772     \or \dcolxxi \or \dcolxxii \or \dcolxxiii
5773     \or \dcolxxiv \or \dcolxxv \or \dcolxxvi
5774     \or \dcolxxvii \or \dcolxxviii \or \dcolxxix \or \dcolxxx
5775     \else \dcolerr \fi}
5776
5777 %
```

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

```
5778 \newcommand*{\stepl@dcolcount}{\advance\l@dcolcount\@ne
5779     \ifnum\l@dcolcount>30\relax
5780         \led@err@TooManyColumns
5781     \fi}
5782
5783 %
```

`\l@dsetmaxcolwidth` Sets the column width to the maximum value seen so far.

```
5784 \newcommand{\l@dsetmaxcolwidth}{%
5785     \ifdim\l@dcolwidth < \wd\hilfsbox
5786         \l@dcolwidth = \wd\hilfsbox
5787     \else \relax \fi}
5788
5789 %
```

`\measuremcell` Measure (recursively) the width required for a math cell.

```
5790 \def\measuremcell #1&{%
5791     \ifx #1\\ \ifnum\l@dcolcount=0\let\NEXT\relax%
5792             \else\l@dcheckcols%
5793                     \l@dcolcount=0%
5794                     \let\NEXT\measuremcell%
5795             \fi%
5796     \else\setbox\hilfsbox=\hbox{$\displaystyle{#1}$}%
5797         \stepl@dcolcount%
5798         \l@dsetmaxcolwidth%
5799         \let\NEXT\measuremcell%
5800     \fi\NEXT}
5801
5802 %
```

`\measuretcell` Measure (recursively) the width required for a text cell.

```
5803 \def\measuretcell #1&{%
5804     \ifx #1\\ \ifnum\l@dcolcount=0\let\NEXT\relax%
5805              \else\l@dcheckcols%
5806                    \l@dcolcount=0%
5807                    \let\NEXT\measuretcell%
5808              \fi%
5809     \else\setbox\hilfsbox=\hbox{#1}%
5810       \stepl@dcolcount%
5811       \l@dsetmaxcolwidth%
5812       \let\NEXT\measuretcell%
5813     \fi\NEXT}
5814
5815 %
```

\measuremrow   Measure (recursively) the width required for a math row.

```
5816 \def\measuremrow #1\\{%
5817     \ifx #1&\let\NEXT\relax%
5818     \else\measuremcell #1&\\&\\&%
5819         \let\NEXT\measuremrow%
5820     \fi\NEXT}
5821 %
```

\measuretrow   Measure (recursively) the width required for a text row.

```
5822 \def\measuretrow #1\\{%
5823     \ifx #1&\let\NEXT\relax%
5824     \else\measuretcell #1&\\&\\&%
5825       \let\NEXT\measuretrow%
5826     \fi\NEXT}
5827
5828 %
```

\edtabcolsep   The length \edtabcolsep controls the distance between columns.

```
5829 \newskip\edtabcolsep
5830 \global\edtabcolsep=10pt
5831
5832 %
```

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

```
5834
5835 %
```

\l@dcheckcols   Check that the number of columns is consistent.

```
5836 \newcommand*{\l@dcheckcols}{%
5837   \ifnum\l@dcolcount=1\relax
```

```
5838      \else
5839        \ifnum\l@dampcount=1\relax
5840        \else
5841          \ifnum\l@dcolcount=\l@dampcount\relax
5842          \else
5843            \l@d@err@UnequalColumns
5844          \fi
5845        \fi
5846        \l@dampcount=\l@dcolcount
5847    \fi}
5848
5849  %
```

\edfilldimen   A length.

```
5850  \newdimen\edfilldimen
5851  \edfilldimen=0pt
5852
5853  %
```

\c@addcolcount   A counter to hold the number of a column. We use a roman number so that we can grab
\theaddcolcount the column dimension from \dcol.

```
5854  \newcounter{addcolcount}
5855    \renewcommand{\theaddcolcount}{\roman{addcolcount}}
5856  %
```

XXVIII.2.3   Tabular typesetting

\setmcellright   Typeset (recursively) cells of display math right justified.

```
5857  \def\setmcellright #1&{\def\edlabel##1{}%
5858          \let\edindex\nulledindex
5859          \ifx #1\\ \ifnum\l@dcolcount=0%\removelastskip
5860                      \let\Next\relax%
5861                  \else\l@dcolcount=0%
5862                          \let\Next=\setmcellright%
5863                  \fi%
5864          \else%
5865              \disablel@dtabfeet%
5866              \stepl@dcolcount%
5867              \disable@notes%
5868              \setbox\hilfsbox=\hbox{$\displaystyle{#1}$}%
5869              \restore@notes%
5870              \letsforverteilen%
5871              \hskip\hilfsskip$\displaystyle{#1}$%
5872              \hskip\edtabcolsep%
5873              \let\Next=\setmcellright%
5874          \fi\Next}
5875
5876  %
```

\settcellright Typeset (recursively) cells of text right justified.

```
5877  \def\settcellright #1&{\def\edlabel##1{}%
5878          \let\edindex\nulledindex
5879          \ifx #1\\ \ifnum\l@dcolcount=0%\removelastskip
5880                      \let\Next\relax%
5881                  \else\l@dcolcount=0%
5882                      \let\Next=\settcellright%
5883                  \fi%
5884          \else%
5885              \disablel@dtabfeet%
5886              \stepl@dcolcount%
5887              \disable@notes%
5888              \setbox\hilfsbox=\hbox{#1}%
5889              \restore@notes%
5890              \letsforverteilen%
5891              \hskip\hilfsskip#1%
5892              \hskip\edtabcolsep%
5893              \let\Next=\settcellright%
5894          \fi\Next}
5895  %
```

\setmcellleft Typeset (recursively) cells of display math left justified.

```
5896  \def\setmcellleft #1&{\def\edlabel##1{}%
5897      \let\edindex\nulledindex
5898      \ifx #1\\ \ifnum\l@dcolcount=0 \let\Next\relax%
5899              \else\l@dcolcount=0%
5900                  \let\Next=\setmcellleft%
5901              \fi%
5902      \else   \disablel@dtabfeet%
5903              \stepl@dcolcount%
5904              \disable@notes%
5905              \setbox\hilfsbox=\hbox{$\displaystyle{#1}$}%
5906              \restore@notes%
5907              \letsforverteilen%
5908              $\displaystyle{#1}$\hskip\hilfsskip\hskip\edtabcolsep%
5909              \let\Next=\setmcellleft%
5910      \fi\Next}
5911
5912  %
```

\settcellleft Typeset (recursively) cells of text left justified.

```
5913  \def\settcellleft #1&{\def\edlabel##1{}%
5914      \let\edindex\nulledindex
5915      \ifx #1\\ \ifnum\l@dcolcount=0 \let\Next\relax%
5916              \else\l@dcolcount=0%
5917                  \let\Next=\settcellleft%
5918              \fi%
```

```
5919        \else    \disablel@dtabfeet%
5920                 \stepl@dcolcount%
5921                 \disable@notes%
5922                 \setbox\hilfsbox=\hbox{#1}%
5923                 \restore@notes%
5924                 \letsforverteilen%
5925                 #1\hskip\hilfsskip\hskip\edtabcolsep%
5926                 \let\Next=\settcellleft%
5927        \fi\Next}
5928  %
```

\setmcellcenter   Typeset (recursively) cells of display math centered.

```
5929  \def\setmcellcenter #1&{\def\edlabel##1{}%
5930        \let\edindex\nulledindex
5931        \ifx #1\\ \ifnum\l@dcolcount=0\let\Next\relax%
5932                 \else\l@dcolcount=0%
5933                        \let\Next=\setmcellcenter%
5934                 \fi%
5935        \else    \disablel@dtabfeet%
5936                 \stepl@dcolcount%
5937                 \disable@notes%
5938                 \setbox\hilfsbox=\hbox{$\displaystyle{#1}$}%
5939                 \restore@notes%
5940                 \letsforverteilen%
5941                 \hskip 0.5\hilfsskip$\displaystyle{#1}$\hskip0.5\hilfsskip%
5942                 \hskip\edtabcolsep%
5943                 \let\Next=\setmcellcenter%
5944        \fi\Next}
5945
5946  %
```

\settcellcenter   Typeset (recursively) cells of text centered.

```
5947  \def\settcellcenter #1&{\def\edlabel##1{}%
5948        \let\edindex\nulledindex
5949        \ifx #1\\ \ifnum\l@dcolcount=0 \let\Next\relax%
5950                 \else\l@dcolcount=0%
5951                        \let\Next=\settcellcenter%
5952                 \fi%
5953        \else    \disablel@dtabfeet%
5954                 \stepl@dcolcount%
5955                 \disable@notes%
5956                 \setbox\hilfsbox=\hbox{#1}%
5957                 \restore@notes%
5958                 \letsforverteilen%
5959                 \hskip 0.5\hilfsskip #1\hskip 0.5\hilfsskip%
5960                 \hskip\edtabcolsep%
5961                 \let\Next=\settcellcenter%
5962        \fi\Next}
```

```
5963
5964  %
```

\NEXT
```
5965  \let\NEXT=\relax
5966
5967  %
```

\setmrowright   Typeset (recursively) rows of right justified math.

```
5968  \def\setmrowright #1\\{%
5969      \ifx #1& \let\NEXT\relax
5970      \else \centerline{\setmcellright #1&\\&\\&}
5971          \let\NEXT=\setmrowright
5972      \fi\NEXT}
5973  %
```

\settrowright   Typeset (recursively) rows of right justified text.

```
5974  \def\settrowright #1\\{%
5975      \ifx #1& \let\NEXT\relax
5976      \else \centerline{\settcellright #1&\\&\\&}
5977          \let\NEXT=\settrowright
5978      \fi\NEXT}
5979
5980  %
```

\setmrowleft   Typeset (recursively) rows of left justified math.

```
5981  \def\setmrowleft #1\\{%
5982      \ifx #1&\let\NEXT\relax
5983      \else \centerline{\setmcellleft #1&\\&\\&}
5984          \let\NEXT=\setmrowleft
5985      \fi\NEXT}
5986  %
```

\settrowleft   Typeset (recursively) rows of left justified text.

```
5987  \def\settrowleft #1\\{%
5988      \ifx #1& \let\NEXT\relax
5989      \else \centerline{\settcellleft #1&\\&\\&}
5990          \let\NEXT=\settrowleft
5991      \fi\NEXT}
5992
5993  %
```

\setmrowcenter   Typeset (recursively) rows of centered math.

```
5994 \def\setmrowcenter #1\\{%
5995     \ifx #1& \let\NEXT\relax%
5996     \else \centerline{\setmcellcenter #1&\\&\\&}
5997         \let\NEXT=\setmrowcenter
5998      \fi\NEXT}
5999 %
```

\settrowcenter   Typeset (recursively) rows of centered text.

```
6000 \def\settrowcenter #1\\{%
6001     \ifx #1& \let\NEXT\relax
6002     \else \centerline{\settcellcenter #1&\\&\\&}
6003         \let\NEXT=\settrowcenter
6004     \fi\NEXT}
6005
6006 %
```

\nullsetzen   `\newcommand{\nullsetzen}{%`

```
6007 \newcommand{\nullsetzen}{%
6008     \stepl@dcolcount%
6009     \l@dcolwidth=0pt%
6010     \ifnum\l@dcolcount=30\let\NEXT\relax%
6011         \l@dcolcount=0\relax
6012     \else\let\NEXT\nullsetzen%
6013     \fi\NEXT}
6014
6015 %
```

\edatleft   \edatleft[⟨*math*⟩]{⟨*symbol*⟩}{⟨*len*⟩}. Left ⟨*symbol*⟩, 2⟨*len*⟩ high with prepended ⟨*math*⟩ vertically centered.

```
6016 \newcommand{\edatleft}[3][\@empty]{%
6017   \ifx#1\@empty
6018     \vbox to 10pt{\vss\hbox{$\left#2\vrule width0pt height #3
6019                         depth 0pt \right. $}\hss}\vfil}
6020   \else
6021     \vbox to 4pt{\vss\hbox{$#1\left#2\vrule width0pt height #3
6022               depth 0pt \right. $}\vfil}
6023   \fi}
6024 %
```

\edatright   \edatright[⟨*math*⟩]{⟨*symbol*⟩}{⟨*len*⟩}. Right ⟨*symbol*⟩, 2⟨*len*⟩ high with appended ⟨*math*⟩ vertically centered.

```
6025 \newcommand{\edatright}[3][\@empty]{%
6026   \ifx#1\@empty
6027     \vbox to 10pt{\vss\hbox{$\left.\vrule width0pt height #3
6028               depth 0pt \right#2 $}\hss}\vfil}
6029   \else
6030     \vbox to 4pt{\vss\hbox{$\left.\vrule width0pt height #3
```

```
6031                          depth 0pt \right#2 #1 $}\vfil}
6032     \fi}
6033
6034 %
```

\edvertline   \edvertline{⟨*len*⟩} vertical line ⟨*len*⟩ high.

```
6035 \newcommand{\edvertline}[1]{\vbox to 8pt{\vss\hbox{\vrule height #1}\vfil}}
6036
6037 %
```

\edvertdots   \edvertdots{⟨*len*⟩} vertical dotted line ⟨*len*⟩ high.

```
6038 \newcommand{\edvertdots}[1]{\vbox to 1pt{\vss\vbox to #1%
6039         {\cleaders\hbox{$\m@th\hbox{.}\vbox to 0.5em{ }$}\vfil}}}
6040
6041 %
```

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

```
6042 \newcommand{\l@dtabaddcols}[2]{%
6043   \l@dcheckstartend{#1}{#2}%
6044   \ifl@dstartendok
6045   \setcounter{addcolcount}{#1}%
6046   \@whilenum \value{addcolcount}<#2\relax \do
6047   {\advance\edfilldimen by \the \csname dcol\theaddcolcount\endcsname
6048    \advance\edfilldimen by \edtabcolsep
6049    \stepcounter{addcolcount}}%
6050   \advance\edfilldimen by \the \csname dcol\theaddcolcount\endcsname
6051   \fi
6052 }
6053
6054 %
```

\ifl@dstartendok   \l@dcheckstartend{⟨*startcol*⟩}{⟨*endcol*⟩} checks that the values of ⟨*startcol*⟩ and
\l@dcheckstartend   ⟨*endcol*⟩ are sensible. If they are then \ifl@dstartendok is set TRUE, otherwise it
is set FALSE.

```
6055 \newif\ifl@dstartendok
6056 \newcommand{\l@dcheckstartend}[2]{%
6057   \l@dstartendoktrue
6058   \ifnum #1<\@ne
6059     \l@dstartendokfalse
6060     \led@err@LowStartColumn
6061   \fi
6062   \ifnum #2>30\relax
6063     \l@dstartendokfalse
6064     \led@err@HighEndColumn
6065   \fi
```

```
6066    \ifnum #1>#2\relax
6067      \l@dstartendokfalse
6068      \led@err@ReverseColumns
6069    \fi
6070  }
6071
6072  %
```

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

```
6073  \newcommand*{\edrowfill}[3]{%
6074    \l@dtabaddcols{#1}{#2}%
6075    \hb@xt@ \the\l@dcolwidth{\hb@xt@ \the\edfilldimen{#3}\hss}}
6076  \let\@edrowfill@=\edrowfill
6077  \def\@EDROWFILL@#1#2#3{\@edrowfill@{#1}{#2}{#3}}
6078
6079  %
```

\edbeforetab   The macro \edbeforetab{⟨*text*⟩}{⟨*math*⟩} puts ⟨*text*⟩ at the left margin before
\edaftertab array cell entry ⟨*math*⟩. Conversely, the macro \edaftertab{⟨*math*⟩}{⟨*text*⟩} puts
⟨*text*⟩ at the right margin after array cell entry ⟨*math*⟩. \edbeforetab should be in the
first column and \edaftertab in the last column. The following macros support these.

\leftltab \leftltab{⟨*text*⟩} for \edbeforetab in \ltab.

```
6080  \newcommand{\leftltab}[1]{%
6081    \hb@xt@\z@{\vbox{\edtabindent%
6082    \moveleft\Hilfsskip\hbox{\ #1}}\hss}}
6083
6084  %
```

\leftrtab \leftrtab{⟨*text*⟩}{⟨*math*⟩} for \edbeforetab in \rtab.

```
6085  \newcommand{\leftrtab}[2]{%
6086    #2\hb@xt@\z@{\vbox{\edtabindent%
6087      \advance\Hilfsskip by\dcoli%
6088      \moveleft\Hilfsskip\hbox{\ #1}}\hss}}
6089
6090  %
```

\leftctab \leftctab{⟨*text*⟩}{⟨*math*⟩} for \edbeforetab in \ctab.

```
6091  \newcommand{\leftctab}[2]{%
6092        \hb@xt@\z@{\vbox{\edtabindent\l@dcolcount=\l@dampcount%
6093        \advance\Hilfsskip by 0.5\dcoli%
6094        \setbox\hilfsbox=\hbox{\def\edlabel##1{}%
```

```
6095            \disablel@dtabfeet$\displaystyle{#2}$}%
6096            \advance\Hilfsskip by -0.5\wd\hilfsbox%
6097            \moveleft\Hilfsskip\hbox{\ #1}}\hss}%
6098            #2}
6099
6100 %
```

`\rightctab`   `\rightctab{⟨math⟩}{⟨text⟩}` for `\edaftertab` in `\ctab`.

```
6101 \newcommand{\rightctab}[2]{%
6102            \setbox\hilfsbox=\hbox{\def\edlabel##1{}%
6103            \disablel@dtabfeet#2}\l@dampcount=\l@dcolcount%
6104            #1\hb@xt@\z@{\vbox{\edtabindent\l@dcolcount=\l@dampcount%
6105            \advance\Hilfsskip by 0.5\l@dcolwidth%
6106            \advance\Hilfsskip by -\wd\hilfsbox%
6107            \setbox\hilfsbox=\hbox{\def\edlabel##1{}%
6108            \disablel@dtabfeet$\displaystyle{#1}$}%
6109            \advance\Hilfsskip by -0.5\wd\hilfsbox%
6110            \advance\Hilfsskip by \edtabcolsep%
6111            \moveright\Hilfsskip\hbox{ #2}}\hss}%
6112            }
6113
6114 %
```

`\rightltab`   `\rightltab{⟨math⟩}{⟨text⟩}` for `\edaftertab` in `\ltab`.

```
6115 \newcommand{\rightltab}[2]{%
6116            \setbox\hilfsbox=\hbox{\def\edlabel##1{}%
6117            \disablel@dtabfeet#2}\l@dampcount=\l@dcolcount%
6118            #1\hb@xt@\z@{\vbox{\edtabindent\l@dcolcount=\l@dampcount%
6119            \advance\Hilfsskip by\l@dcolwidth%
6120            \advance\Hilfsskip by-\wd\hilfsbox%
6121            \setbox\hilfsbox=\hbox{\def\edlabel##1{}%
6122            \disablel@dtabfeet$\displaystyle{#1}$}%
6123            \advance\Hilfsskip by-\wd\hilfsbox%
6124            \advance\Hilfsskip by\edtabcolsep%
6125            \moveright\Hilfsskip\hbox{ #2}}\hss}%
6126            }
6127
6128 %
```

`\rightrtab`   `\rightrtab{⟨math⟩}{⟨text⟩}` for `\edaftertab` in `\rtab`.

```
6129 \newcommand{\rightrtab}[2]{%
6130            \setbox\hilfsbox=\hbox{\def\edlabel##1{}%
6131            \disablel@dtabfeet#2}%
6132            #1\hb@xt@\z@{\vbox{\edtabindent%
6133            \advance\Hilfsskip by-\wd\hilfsbox%
6134            \advance\Hilfsskip by\edtabcolsep%
6135            \moveright\Hilfsskip\hbox{ #2}}\hss}%
```

```
6136              }
6137
6138  %
```

\rtab \rtab{⟨*body*⟩} typesets ⟨*body*⟩ as an array with the entries right justified.

\edbeforetab   The process is first to measure the ⟨*body*⟩ to get the column widths, and then in a

\edaftertab second pass to typeset the body.

```
6139  \newcommand{\rtab}[1]{%
6140    \l@dnullfills
6141      \def\edbeforetab##1##2{\leftrtab{##1}{##2}}%
6142      \def\edaftertab##1##2{\rightrtab{##1}{##2}}%
6143      \measurembody{#1}%
6144    \l@drestorefills
6145      \variab
6146      \setmrowright #1\\&\\%
6147      \enablel@dtabfeet}
6148
6149  %
```

\measurembody \measurembody{⟨*body*⟩} measures the array ⟨*body*⟩.

```
6150  \newcommand{\measurembody}[1]{%
6151    \disablel@dtabfeet%
6152    \l@dcolcount=0%
6153    \nullsetzen%
6154    \l@dcolcount=0
6155    \measuremrow #1\\&\\%
6156    \global\l@dampcount=1}
6157
6158  %
```

\rtabtext \rtabtext{⟨*body*⟩} typesets ⟨*body*⟩ as a tabular with the entries right justified.

```
6159  \newcommand{\rtabtext}[1]{%
6160   \l@dnullfills
6161      \measuretbody{#1}%
6162   \l@drestorefills
6163      \variab
6164      \settrowright #1\\&\\%
6165      \enablel@dtabfeet}
6166
6167  %
```

\measuretbody \measuretbody{⟨*body*⟩} measures the tabular ⟨*body*⟩.

```
6168  \newcommand{\measuretbody}[1]{%
6169    \disable@notes%
6170    \disablel@dtabfeet%
6171    \l@dcolcount=0%
```

```
6172    \nullsetzen%
6173    \l@dcolcount=0
6174    \measuretrow #1\\&\\%
6175    \restore@notes%
6176    \global\l@dampcount=1}
6177
6178 %
```

\ltab   Array with entries left justified.

\edbeforetab
\edaftertab
```
6179 \newcommand{\ltab}[1]{%
6180   \l@dnullfills
6181       \def\edbeforetab##1##2{\leftltab{##1}{##2}}%
6182       \def\edaftertab##1##2{\rightltab{##1}{##2}}%
6183       \measurembody{#1}%
6184   \l@drestorefills
6185       \variab
6186       \setmrowleft #1\\&\\%
6187       \enablel@dtabfeet}
6188
6189 %
```

\ltabtext   Tabular with entries left justified.

```
6190 \newcommand{\ltabtext}[1]{%
6191    \l@dnullfills
6192       \measuretbody{#1}%
6193    \l@drestorefills
6194       \variab
6195       \settrowleft #1\\&\\%
6196       \enablel@dtabfeet}
6197
6198 %
```

\ctab   Array with centered entries.

\edbeforetab
\edaftertab
```
6199 \newcommand{\ctab}[1]{%
6200    \l@dnullfills
6201       \def\edbeforetab##1##2{\leftctab{##1}{##2}}%
6202       \def\edaftertab##1##2{\rightctab{##1}{##2}}%
6203       \measurembody{#1}%
6204    \l@drestorefills
6205       \variab
6206       \setmrowcenter #1\\&\\%
6207       \enablel@dtabfeet}
6208
6209 %
```

\ctabtext   Tabular with entries centered.

```
6210 \newcommand{\ctabtext}[1]{%
6211    \l@dnullfills
6212      \measuretbody{#1}%
6213    \l@drestorefills
6214      \variab
6215      \settrowcenter #1\\&\\%
6216      \enablel@dtabfeet}
6217
6218 %
```

\spreadtext
```
6219 \newcommand{\spreadtext}[1]{%\l@dcolcount=\l@dampcount%
6220    \hb@xt@ \the\l@dcolwidth{\hbox{#1}\hss}}
6221 %
```

\spreadmath
```
6222 \newcommand{\spreadmath}[1]{%
6223    \hb@xt@ \the\l@dcolwidth{\hbox{$\displaystyle{#1}$}\hss}}
6224
6225 %
```

\HILFSskip   More helpers.
\Hilfsskip
```
6226 \newskip\HILFSskip
6227 \newskip\Hilfsskip
6228
6229 %
```

\EDTABINDENT
```
6230 \newcommand{\EDTABINDENT}{%
6231     \ifnum\l@dcolcount=30\let\NEXT\relax\l@dcolcount=0%
6232     \else\stepl@dcolcount%
6233         \advance\Hilfsskip by\l@dcolwidth%
6234         \ifdim\l@dcolwidth=0pt\advance\hilfscount\@ne
6235         \else\advance\Hilfsskip by \the\hilfscount\edtabcolsep%
6236         \hilfscount=1\fi%
6237         \let\NEXT=\EDTABINDENT%
6238     \fi\NEXT}%
6239 %
```

\edtabindent   (was \tabindent)
```
6240 \newcommand{\edtabindent}{%
6241     \l@dcolcount=0\relax
6242     \Hilfsskip=0pt%
6243     \hilfscount=1\relax
6244     \EDTABINDENT%
6245     \hilfsskip=\hsize%
6246     \advance\hilfsskip -\Hilfsskip%
```

```
6247        \Hilfsskip=0.5\hilfsskip%
6248        }%
6249
6250 %
```

`\EDTAB` (was `\TAB`)

```
6251 \def\EDTAB #1|#2|{%
6252     \setbox\tabhilfbox=\hbox{$\displaystyle{#1}$}%
6253     \setbox\tabHilfbox=\hbox{$\displaystyle{#2}$}%
6254     \advance\tabelskip -\wd\tabhilfbox%
6255     \advance\tabelskip -\wd\tabHilfbox%
6256     \unhbox\tabhilfbox\hskip\tabelskip%
6257     \unhbox\tabHilfbox}%
6258
6259 %
```

`\EDTABtext` (was `\TABtext`)

```
6260 \def\EDTABtext #1|#2|{%
6261     \setbox\tabhilfbox=\hbox{#1}%
6262     \setbox\tabHilfbox=\hbox{#2}%
6263     \advance\tabelskip -\wd\tabhilfbox%
6264     \advance\tabelskip -\wd\tabHilfbox%
6265     \unhbox\tabhilfbox\hskip\tabelskip%
6266     \unhbox\tabHilfbox}%
6267 %
```

`\tabhilfbox` Further helpers.
`\tabHilfbox`
```
6268 \newbox\tabhilfbox
6269 \newbox\tabHilfbox
6270
6271 %
```

XXVIII.2.4   Environments

edarrayl edarrayc edarrayr The 'environment' forms for `\ltab`, `\ctab` and `\rtab`.

```
6272 \newenvironment{edarrayl}{\l@dcollect@body\ltab}{}
6273 \newenvironment{edarrayc}{\l@dcollect@body\ctab}{}
6274 \newenvironment{edarrayr}{\l@dcollect@body\rtab}{}
6275
6276 %
```

edtabularl edtabularc edtabularr The 'environment' forms for `\ltabtext`,
`\ctabtext` and `\rtabtext`.

```
6277 \newenvironment{edtabularl}{\l@dcollect@body\ltabtext}{}
6278 \newenvironment{edtabularc}{\l@dcollect@body\ctabtext}{}
```

```
6279  \newenvironment{edtabularr}{\l@dcollect@body\rtabtext}{}

6280

6281  %
```

XXIX   Quotation's commands

\initnumbering@quote   This macro, called at the begin of any numbered section, redefine locally the quotation and quote environment, in order to allow to use it inside numbered section.

\quotation   \initnumbering@quote defines quotation environment.
\endquotation
\quote
\endquote

```
6282  \newcommand{\initnumbering@quote}{
6283      \ifnoquotation@\else
6284      \renewcommand{\quotation}{\par\leavevmode%
6285                                \parindent=1.5em%
6286                                \skipnumbering%
6287                                \ifautopar%
6288                                    \vskip-\parskip%
6289                                \else%
6290                                    \vskip\topsep%
6291                                \fi%
6292                                \global\leftskip=\leftmargin%
6293                                \global\rightskip=\leftmargin%
6294                           }
6295      \renewcommand{\endquotation}{\par%
6296                                \global\leftskip=0pt%
6297                                \global\rightskip=0pt%
6298                                \leavevmode%
6299                                \skipnumbering%
6300                                \ifautopar%
6301                                    \vskip-\parskip%
6302                                \else%
6303                                    \vskip\topsep%
6304                                \fi%
6305                                }
6306      \renewcommand{\quote}{\par\leavevmode%
6307                                \parindent=0pt%
6308                                \skipnumbering%
6309                                \ifautopar%
6310                                    \vskip-\parskip%
6311                                \else%
6312                                    \vskip\topsep%
6313                                \fi%
6314                                \global\leftskip=\leftmargin%
6315                                \global\rightskip=\leftmargin%
6316      }
6317      \renewcommand{\endquote}{\par%
6318                                \global\leftskip=0pt%
6319                                \global\rightskip=0pt%
```

```
6320                                          \leavevmode%
6321                                          \skipnumbering%
6322                                          \ifautopar%
6323                                             \vskip-\parskip%
6324                                          \else%
6325                                             \vskip\topsep%
6326                                          \fi%
6327                                          }
6328        \fi
6329  }
6330  %
```

XXX   Section's title commands

XXX.1   Commands to disable some feature

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

```
6331  \newcommand{\ledsectnotoc}{\let\addcontentsline\@gobblethree}
6332  %
```

`\ledsectnomark`   The `\ledsectnomark` only disables the `\chaptermark`, `\sectionmark` and `\subsectionmark` macros.

```
6333  \newcommand{\ledsectnomark}{%
6334    \let\chaptermark\@gobble%
6335    \let\sectionmark\@gobble%
6336    \let\subsectionmark\@gobble%
6337  }
6338  %
```

XXX.2   General overview

The system of `\eledxxxx` commands to section text work like this:

1. When one of these commands is called, `reledmac` writes to an auxiliary files:

   - The section level.
   - The section title.
   - The side (when `eledpar` is used).
   - The pstart where the command is called.
   - If we have starred version or not.

2. `reledmac` adds the title of the section to pstart, as normal content. This is to enable critical notes.

3. When LATEX is run a other time, this file is read. That:

> - Adds the pstart number to a list of pstarts where a sectioning command is used.

> - Defines a command, the name of which contains the pstart number, and which calls the normal LATEX sectioning command.

4. This last command is called when the pstart is effectively printed.

XXX.3 \beforeeledchapter command

We do not define commands for \eledsection and related if the noeledsec option is loaded. We use etoolbox tests and not the \ifxxx...\else...\fi structure to prevent problem of expansions with command after the \ifxxx which contains \fi. As we patch command inside this test, we need to change the category code of # character *before* \notbool statement, because the second argument is read with the standard catcode (read *The TeXbook* to understand when the catcode's change has effect).

```
6339 \catcode`\#=12
6340 \notbool{@noeled@sec}{%
6341 %
```

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

```
6342 \ifl@dmemoir
6343   \newcommand\beforeeledchapter{%
6344   \clearforchapter%
6345 }
6346 \else
6347   \newcommand\beforeeledchapter{%
6348   \if@openright%
6349    \cleardoublepage%
6350   \else%
6351    \clearpage%
6352   \fi%
6353 }
6354 \fi
6355 %
```

XXX.4   Auxiliary commands

\if@eled@sectioning   The boolean \if@eled@sectioning is set to true when a sectioning command is called by a \eledxxx command, and set to false after. It is used to enable/disable line number printing.

```
6356 \newif\if@eled@sectioning
6357 %
```

`\print@leftmargin@eledsection`
`\print@rightmargin@eledsection`

`\print@leftmargin@eledsection` and `\print@rightmargin@eledsection` are added by reledmac inside the code of sectioning command, in order to affix lines numbers. They include tests for RTL languages.

```
6358 \def\print@rightmargin@eledsection{%
6359   \if@eled@sectioning%
6360     \begingroup%
6361     \if@RTL%
6362         \let\llap\rlap%
6363         \let\leftlinenum\rightlinenum%
6364         \let\leftlinenumR\rightlinenumR%
6365         \let\l@drd@ta\l@dld@ta%
6366         \let\l@drsn@te\l@dlsn@te%
6367     \fi%
6368     \hfill\l@drd@ta \csuse{LR}{\l@drsn@te}%
6369     \endgroup%
6370   \fi%
6371 }%
6372
6373 \def\print@leftmargin@eledsection{%
6374   \if@eled@sectioning%
6375     \leavevmode%
6376     \begingroup%
6377     \if@RTL%
6378         \let\rlap\llap%
6379         \let\rightlinenum\leftlinenum%
6380         \let\rightlinenumR\leftlinenumR%
6381         \let\l@dld@ta\l@drd@ta%
6382         \let\l@dlsn@te\l@drsn@te%
6383     \fi%
6384     \l@dld@ta\csuse{LR}{\l@dlsn@te}%
6385     \endgroup%
6386   \fi%
6387 }%
6388
6389 %
```

XXX.5   Patching standard commands

`\chapter`
`\M@sect`
`\@mem@old@ssect`
`\@makechapterhead`
`\@makechapterhead`
`\@makeschapterhead`
`\@sect`
`\@ssect`

We have to patch LaTeX, book and `memoir` sectioning commands in order to:

- Disable \edtext inside.

- Disable page breaking (for \chapter).

- Add line numbers and sidenotes.

Unfortunately, Maïeul Rouquette was not able to try if `memoir` is loaded. That is why `eledmac` tries to define for both standard class and `memoir` class.

```
6390 \AtBeginDocument{%
6391 \patchcmd{\chapter}{\clearforchapter}{%
6392   \if@eled@sectioning\else%
6393     \ifl@dprintingpages\else%
6394       \clearforchapter%
6395     \fi%
6396   \fi%
6397   }
6398   {}
6399   {}
6400
6401 \pretocmd{\M@sect}
6402   {\let\old@edtext=\edtext%
6403   \let\edtext=\dummy@edtext@showlemma%
6404   }
6405   {}
6406   {}
6407
6408 \apptocmd{\M@sect}
6409   {\let\edtext=\old@edtext}
6410   {}
6411   {}
6412
6413 \patchcmd{\M@sect}
6414   { #9}
6415   { #9%
6416   \print@rightmargin@eledsection%
6417   }
6418   {}
6419   {}
6420
6421 \patchcmd{\M@sect}
6422   {\hskip #3\relax}
6423   {\hskip #3\relax%
6424   \print@leftmargin@eledsection%
6425   }
6426   {}
6427   {}
6428
6429 \patchcmd{\@mem@old@ssect}
6430   {#5}
6431   {#5%
6432   \print@leftmargin@eledsection%
6433   }
6434   {}
6435   {}
6436
6437 \patchcmd{\@mem@old@ssect}
6438   {\hskip #1}
6439   {\hskip #1%
```

```
6440      \print@rightmargin@eledsection%
6441      }
6442      {}
6443      {}

6445  \patchcmd{\chapter}{\if@openright\cleardoublepage\else\clearpage\fi}{%
6446      \if@eled@sectioning\else%
6447        \ifl@dprintingpages\else%
6448          \if@openright\cleardoublepage\else\clearpage\fi%No clearpage inside a
      \eledsection: will keep critical notes from printing on the title page.
6449        \fi%
6450      \fi%
6451      }%
6452      {}%
6453      {}%

6455  \patchcmd{\@makechapterhead}
6456      {#1}
6457      {\print@leftmargin@eledsection%
6458        #1%
6459      \print@rightmargin@eledsection%
6460      }
6461      {}
6462      {}

6464  \patchcmd{\@makechapterhead}% For BIDI
6465      {\if@RTL\raggedleft\else\raggedright\fi}%
6466      {\if@eled@sectioning\else%
6467        \if@RTL\raggedleft\else\raggedright\fi%
6468      \fi%
6469      }%
6470      {}%
6471      {}%

6473  \patchcmd{\@makeschapterhead}
6474      {#1}
6475      {\print@leftmargin@eledsection%
6476        #1%
6477      \print@rightmargin@eledsection%
6478      }
6479      {}
6480      {}

6482  \pretocmd{\@sect}
6483      {\let\old@edtext=\edtext
6484      \let\edtext=\dummy@edtext@showlemma%
6485      }
6486      {}
6487      {}

6488
```

```
6489  \apptocmd{\@sect}
6490    {\let\edtext=\old@edtext}
6491    {}
6492    {}
6493
6494  \pretocmd{\@ssect}
6495    {\let\old@edtext=\edtext%
6496    \let\edtext=\dummy@edtext@showlemma%
6497    }
6498    {}
6499    {}
6500
6501  \apptocmd{\@ssect}
6502    {\let\edtext=\old@edtext}
6503    {}
6504    {}
6505
6506  %
```

hyperref also redefines \@sect. That is why, when manipulating arguments, we patch \@sect and the same only if hyperref is not used. If it is, we patch the \NR commands.

```
6507  \@ifpackageloaded{nameref}{
6508
6509    \patchcmd{\NR@sect}
6510      {#8}
6511      {#8%
6512      \print@rightmargin@eledsection%
6513      }
6514      {}
6515      {}
6516
6517    \patchcmd{\NR@sect}
6518      {\hskip #3\relax}
6519      {\hskip #3\relax%
6520      \print@leftmargin@eledsection%
6521      }
6522      {}
6523      {}
6524
6525    \patchcmd{\NR@ssect}
6526      {#5}
6527      {#5%
6528      \print@rightmargin@eledsection%
6529      }
6530      {}
6531      {}
6532
6533    \patchcmd{\NR@ssect}
6534      {\hskip #1}
6535      {\hskip #1%
```

```
6536        \print@leftmargin@eledsection%
6537        }
6538        {}
6539        {}
6540    }%
6541    {
6542    \patchcmd{\@sect}
6543        {#8}
6544        {#8%
6545        \print@rightmargin@eledsection%
6546        }
6547        {}
6548        {}

6550    \patchcmd{\@sect}
6551        {\hskip #3\relax}
6552        {\hskip #3\relax%
6553        \print@leftmargin@eledsection%
6554        }
6555        {}
6556        {}

6558    \patchcmd{\@ssect}
6559        {#5}
6560        {#5%
6561        \print@rightmargin@eledsection%
6562        }
6563        {}
6564        {}

6566    \patchcmd{\@ssect}
6567        {\hskip #1}
6568        {\hskip #1%
6569        \print@leftmargin@eledsection%
6570        }
6571        {}
6572        {}
6573    }%
6574 }
6575 %
```

Now, we have finished to patch the commands, using # with a catcode equals to 12. We close the \notbool{@noeled@sec} statement, restore the normal catcode for # and reopen a new \notbool{@noeled@sec} statement.

```
6576 {}}%
6577 \protect\catcode`\#=6 %Space NEEDS by \catcode
6578 \notbool{@noeled@sec}{%
6579 %
```

XXX.6   Main code of `\eledxxx` commands

`\eled@sectioning@out` `\eled@sectioning@out` is the output file, to dump the pstarts where a sectioning command is used.

```
6580 \newwrite\eled@sectioning@out
6581 %
```

`\eledchapter` And now, the user sectioning commands, which write to the file, and also add content
`\eledsection` as a "normal" line.
`\eledsubsection`
`\eledsubsubsection`
`\eledchapter*`
`\eledsection*`
`\eledsubsection*`
`\eledsubsubsection*`

```
6582 \newcommand{\eledchapter}[2][]{%
6583   #2%
6584   \ifledRcol%
6585     \immediate\write\eled@sectioningR@out{%
6586       \string\eled@chapter{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR}{}{R}
6587       }%
6588   \else%
6589     \immediate\write\eled@sectioning@out{%
6590       \string\eled@chapter{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL}{}{}
6591       }%
6592   \fi%
6593 }
6594
6595 \newcommand{\eledsection}[2][]{%
6596   #2%
6597   \ifledRcol%
6598     \immediate\write\eled@sectioningR@out{%
6599       \string\eled@section{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR}{}{R}
6600       }%
6601   \else%
6602     \immediate\write\eled@sectioning@out{%
6603       \string\eled@section{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL}{}{}
6604       }%
6605   \fi%
6606 }
6607
6608 \newcommand{\eledsubsection}[2][]{%
6609   #2%
6610   \ifledRcol%
6611     \immediate\write\eled@sectioningR@out{%
6612       \string\eled@subsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR}{}{R
}
6613       }%
6614   \else%
6615     \immediate\write\eled@sectioning@out{%
6616       \string\eled@subsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL}{}{}
6617       }%
6618   \fi%
6619 }
6620 \newcommand{\eledsubsubsection}[2][]{%
```

```
6621    #2%
6622    \ifledRcol%
6623      \immediate\write\eled@sectioningR@out{%
6624        \string\eled@subsubsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR
      }{}{R}
6625        }%
6626    \else%
6627      \immediate\write\eled@sectioning@out{%
6628        \string\eled@subsubsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL
      }{}{}
6629        }%
6630    \fi%
6631  }


6632

6633
6634  \WithSuffix\newcommand\eledchapter*[2][]{%
6635    #2%
6636    \ifledRcol%
6637      \immediate\write\eled@sectioningR@out{%
6638        \string\eled@chapter{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR}{*}{R}
6639        }%
6640    \else%
6641      \immediate\write\eled@sectioning@out{%
6642        \string\eled@chapter{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL}{*}{}
6643        }%
6644    \fi%
6645  }

6646
6647  \WithSuffix\newcommand\eledsection*[2][]{%
6648    #2%
6649    \ifledRcol%
6650      \immediate\write\eled@sectioningR@out{%
6651        \string\eled@section{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR}{*}{R}
6652        }%
6653    \else%
6654      \immediate\write\eled@sectioning@out{%
6655        \string\eled@section{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL}{*}{}
6656        }%
6657    \fi%
6658  }

6659
6660  \WithSuffix\newcommand\eledsubsection*[2][]{%
6661    #2%
6662    \ifledRcol%
6663      \immediate\write\eled@sectioningR@out{%
6664        \string\eled@subsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR}{*}{
      R}
6665        }%
6666    \else%
6667      \immediate\write\eled@sectioning@out{%
```

```
6668        \string\eled@subsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL
      }{*}{}
6669          }%
6670    \fi%
6671  }

6672
6673  \WithSuffix\newcommand\eledsubsubsection*[2][]{%
6674    #2%
6675    \ifledRcol%
6676        \immediate\write\eled@sectioningR@out{%
6677          \string\eled@subsubsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR
      }{*}{R}
6678          }%
6679    \else%
6680        \immediate\write\eled@sectioning@out{%
6681          \string\eled@subsubsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL
      }{*}{}
6682          }%
6683    \fi%
6684  }
6685  %
```

XXX.7   Macros written in the auxiliary file

`\eled@chapter`
`\eled@section`
`\eled@subsection`
`\eled@subsubsection`

The sectioning macros, called in the auxiliary file. They have five arguments:

1. Optional arguments of LaTeX sectioning command.

2. Mandatory arguments of LaTeX sectioning command.

3. Pstart number.

4. Side: R if right, nothing if left.

5. Starred or not.

```
6686  \def\eled@chapter#1#2#3#4#5{%
6687      \ifstrempty{#4}%
6688        {%
6689        \ifstrempty{#1}%
6690          {%
6691          \global\csdef{eled@sectioning@#3#5}{\let\edtext=\
      dummy@edtext@showlemma\chapter{#2}}%
6692          \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext{}\
      chaptermark{#2}}%
6693          }%Need for \pairs, because of using parbox.
6694          {%
6695          \global\csdef{eled@sectioning@#3#5}{\let\edtext=\
      dummy@edtext@showlemma\chapter[#1]{#2}}%
```

```
6696        \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext{}\
    chaptermark{#2}}%Need for \pairs, because of using parbox.
6697          }%
6698        }%
6699        {%
6700       \ifstrempty{#1}%
6701          {\global\csdef{eled@sectioning@#3#5}{\let\edtext=\
    dummy@edtext@showlemma\chapter*{#2}}}%
6702          {\global\csdef{eled@sectioning@#3#5}{\let\edtext=\
    dummy@edtext@showlemma\chapter*[#1]{#2}}}%Bug in LaTeX!
6703        }%
6704   \listcsgadd{eled@sections#5@@}{#3}%
6705      }
6706 \def\eled@section#1#2#3#4#5{%
6707     \ifstrempty{#4}%
6708       {\ifstrempty{#1}%
6709         {%
6710         \global\csdef{eled@sectioning@#3#5}{\section{#2}}%
6711         \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext{}\
    sectionmark{#2}}%Need for \pairs, because of using parbox.
6712         }%
6713         {%
6714         \global\csdef{eled@sectioning@#3#5}{\section[#1]{#2}}%
6715         \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext{}\
    sectionmark{#1}}%Need for \pairs, because of using parbox.
6716         }%
6717       }%
6718       {\ifstrempty{#1}%
6719         {\global\csdef{eled@sectioning@#3#5}{\section*{#2}}}%
6720         {\global\csdef{eled@sectioning@#3#5}{\section*[#1]{#2}}}%Bug in
    LaTeX!
6721       }
6722   \listcsgadd{eled@sections#5@@}{#3}%
6723      }
6724 \def\eled@subsection#1#2#3#4#5{%
6725     \ifstrempty{#4}%
6726       {\ifstrempty{#1}%
6727         {%
6728         \global\csdef{eled@sectioning@#3#5}{\subsection{#2}}%
6729         \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext{}\csuse
    {subsectionmark}{#2}}%Need for \pairs, because of using parbox. \csuse in
    case of \subsectionmark is not defined (book)
6730         }%
6731         {%
6732         \global\csdef{eled@sectioning@#3#5}{\subsection[#1]{#2}}%
6733         \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext{}\csuse
    {subsectionmark}{#1}}%Need for \pairs, because of using parbox. \csuse in
    case of \subsectionmark is not defined (book)
6734         }%
6735       }%
```

```
6736        {\ifstrempty{#1}%
6737           {\global\csdef{eled@sectioning@#3#5}{\subsection*{#2}}}%
6738           {\global\csdef{eled@sectioning@#3#5}{\subsection*[#1]{#2}}}}%Bug in
     LaTeX!
6739         }
6740    \listcsgadd{eled@sections#5@@}{#3}%
6741      }
6742 \def\eled@subsubsection#1#2#3#4#5{%
6743      \ifstrempty{#4}%
6744        {\ifstrempty{#1}%
6745           {\global\csdef{eled@sectioning@#3#5}{\subsubsection{#2}}}%
6746           {\global\csdef{eled@sectioning@#3#5}{\subsubsection[#1]{#2}}}%
6747        }%
6748        {\ifstrempty{#1}%
6749           {\global\csdef{eled@sectioning@#3#5}{\subsubsection*{#2}}}%
6750           {\global\csdef{eled@sectioning@#3#5}{\subsubsection*[#1]{#2}}}}%Bug
     in LaTeX!
6751         }
6752    \listcsgadd{eled@sections#5@@}{#3}%
6753      }
6754
6755 %
```

End of the conditional test about `noeledsec` option.

```
6756 }{}
6757 %
```

XXXI 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.

```
6758 \def\normal@page@break{}
6759 %
```

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

```
6760  \def\l@prev@pb{}
6761  \def\l@prev@nopb{}
6762  %
```

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

```
6763  \newcommand{\ledpb}{\write\linenum@out{\string\led@pb}}
6764  \newcommand{\ledpbnum}[1]{\write\linenum@out{\string\led@pbnum{#1}}}
6765  \newcommand{\lednopb}{\write\linenum@out{\string\led@nopb}}
6766  \newcommand{\lednopbnum}[1]{\write\linenum@out{\string\led@nopbnum{#1}}}
6767  %
```

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

```
6768  \newcommand{\led@pb}{\listxadd{\l@prev@pb}{\the\absline@num}}
6769  \newcommand{\led@pbnum}[1]{\listxadd{\l@prev@pb}{#1}}
6770  \newcommand{\led@nopb}{\listxadd{\l@prev@nopb}{\the\absline@num}}
6771  \newcommand{\led@nopbnum}[1]{\listxadd{\l@prev@nopb}{#1}}
6772  %
```

\ledpbsetting The \ledpbsetting macro only changes the value of \led@pb@macro, for which the
\led@pb@setting default value is before.

```
6773  \def\led@pb@setting{before}
6774  \newcommand{\ledpbsetting}[1]{\gdef\led@pb@setting{#1}}
6775  %
```

\led@check@pb The \led@check@pb and \led@check@nopb are called before or after each line. They
\led@check@nopb check if a page break must occur, depending on the current line and on the content of
\l@pb.

```
6776  \newcommand{\led@check@pb}{\xifinlist{\the\absline@num}{\l@prev@pb}{\
      pagebreak[4]}{}}
6777  \newcommand{\led@check@nopb}{%
6778      \IfStrEq{\led@pb@setting}{before}{%
6779          \xifinlist{\the\absline@num}{\l@prev@nopb}%
6780              {\numdef{\abs@prevline}{\the\absline@num-1}%
6781              \xifinlist{\abs@prevline}{\normal@page@break}%
6782                  {\nopagebreak[4]\enlargethispage{\baselineskip}}%
6783                  {}}%
6784              {}}%
6785          {}%
6786      {}%
6787      \IfStrEq{\led@pb@setting}{after}{%
```

```
6788        \xifinlist{\the\absline@num}{\l@prev@nopb}{%
6789          \xifinlist{\the\absline@num}{\normal@page@break}%
6790            {\nopagebreak[4]\enlargethispage{\baselineskip}}%
6791              {}%
6792 }%
6793          {}}%
6794        {}%
6795      {}%
6796 }
6797 %
```

XXXII Long verse: prevents being separated by a page break

\iflednopbinverse The \lednopbinverse boolean is set to false by default. If set to true, reledmac 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.

```
6798 \newcommand{\check@pb@in@verse}{%
6799    \ifinstanza\iflednopbinverse\ifinserthangingsymbol% Using stanzas and
    enabling page breaks in verse control, while on a hanging verse.
6800        \ifnum\page@num=\last@page@num\else%If we have change page
6801          \IfStrEq{\led@pb@setting}{before}{%
6802              \numgdef{\abs@line@verse}{\the\absline@num-1}%
6803              \ledpbnum{\abs@line@verse}%
6804            }{}%
6805          \IfStrEq{\led@pb@setting}{after}{%
6806              \numgdef{\abs@line@verse}{\the\absline@num-1}%
6807              \lednopbnum{\abs@line@verse}%
6808          }{}%
6809        \fi%
6810    \fi\fi\fi%
6811 }
6812 %
```

XXXIII Compatibility with eledmac

Here, we define some command for the eledmac-compat option.

```
6813  \ifeledmaccompat@%

6814
6815  \newcommand{\footnormalX}[1]{\arrangementX[#1]{normal}}%
6816  \newcommand{\footparagraphX}[1]{\arrangementX[#1]{paragraph}}%
6817  \newcommand{\foottwocolX}[1]{\arrangementX[#1]{twocol}}%
6818  \newcommand{\footthreecolX}[1]{\XarrangementX[#1]{threecol}}%

6819
6820  \unless\ifnocritical@
6821    \newcommand{\footnormal}[1]{\Xarrangement[#1]{normal}}%
6822    \newcommand{\footparagraph}[1]{\Xarrangement[#1]{paragraph}}%
6823    \newcommand{\foottwocol}[1]{\Xarrangement[#1]{twocol}}%
6824    \newcommand{\footthreecol}[1]{\Xarrangement[#1]{threecol}}%
6825    \let\hsizetwocol\Xhsizetwocol
6826    \let\hsizethreecol\Xhsizethreecol
6827    \let\bhookXnote\Xbhooknote
6828    \let\boxsymlinenum\Xboxsymlinenum
6829    \let\symlinenum\Xsymlinenum
6830    \let\beforenumberinfootnote\Xbeforenumber
6831    \let\afternumberinfootnote\Xafternumber
6832    \let\beforeXsymlinenum\Xbeforesymlinenum
6833    \let\afterXsymlinenum\Xaftersymlinenum
6834    \let\inplaceofnumber\Xinplaceofnumber
6835    \let\Xlemmaseparator\lemmaseparator
6836    \let\afterlemmaseparator\Xafterlemmaseparator
6837    \let\beforelemmaseparator\Xbeforelemmaseparator
6838    \let\inplaceoflemmaseparator\Xinplaceoflemmaseparator
6839    \let\txtbeforeXnotes\Xtxtbeforenotes
6840    \let\afterXrule\Xafterrule
6841    \let\numberonlyfirstinline\Xnumberonlyfirstinline
6842    \let\numberonlyfirstintwolines\Xnumberonlyfirstintwolines
6843    \let\nonumberinfootnote\Xnonumberinfootnote
6844    \let\pstartinfootnote\Xpstart
6845    \let\pstartinfootnoteeverytime\Xpstarteverytime
6846    \let\onlyXpstart\Xonlypstart
6847    \let\Xnonumberinfootnote\Xnonumber
6848    \let\nonbreakableafternumber\Xnonbreakableafternumber
6849    \let\maxhXnotes\Xmaxhnotes
6850    \let\beforeXnotes\Xbeforenotes
6851    \let\boxlinenum\Xboxlinenum
6852    \let\boxlinenumalign\Xboxlinenumaligm
6853    \let\boxstartlinenum\Xboxstartlinenum
6854    \let\boxendlinenum\Xboxendlinenum
6855    \let\twolines\Xtwolines
6856    \let\morethantwolines\Xmorethantwolines
6857    \let\twolinesbutnotmore\Xtwolinesbutnotmore
6858    \let\twolinesonlyinsamepage\Xtwolinesonlyinsamepage
6859    \fi

6860
6861    \unless\ifnofamiliar@
6862    \let\notesXwidthliketwocolumns\noteswidthliketwocolumnsX
```

```
6863    \fi
6864    \newcommandx{\parafootsep}[2][1,usedefault]{%
6865      \Xparafootsep[#1]{#2}%
6866      \parafootsepX[#1]{#2}
6867    }%

6868
6869    \newcommandx{\afternote}[2][1,usedefault]{%
6870      \Xafternote[#1]{#2}%
6871      \afternoteX[#1]{#2}%
6872    }%

6873
6874    \unless\ifnoend@
6875    \let\XendXtwolines\Xendtwolines
6876    \let\XendXmorethantwolines\Xendmorethantwolines
6877    \let\bhookXendnote\Xendbhooknote
6878    \let\boxXendlinenum\Xendboxlinenum%
6879    \let\boxXendlinenumalign\Xendboxlinenumalign%
6880    \let\boxXendstartlinenum\Xendboxstartlinenum%
6881    \let\boxXendendlinenum\Xendboxendlinenum%
6882    \let\XendXlemmaseparator\Xendlemmaseparator
6883    \let\XendXbeforelemmaseparator\Xendbeforelemmaseparator
6884    \let\XendXafterlemmaseparator\Xendafterlemmaseparator
6885    \let\XendXinplaceoflemmaseparator\Xendinplaceoflemmaseparator
6886    \fi

6887
6888    \AtBeginDocument{%
6889      \ifdef\lineref{}{\let\lineref\edlineref}%
6890    }%

6891

6892
6893    \fi%
6894  %
```

</code>

Appendix A   Some things to do when changing version

Appendix A.1   Migrating from `edmac` to `ledmac`

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 `ledmac`.

The package still provides the original `\text` command, but it is (a) deprecated, and (b) its name has been changed[33] to `\critext`; use the `\edtext` macro instead. However, if you do use `\critext` (the new name for `\text`), the following is a reminder.

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

> `\critext{`⟨*lemma*⟩`}`⟨*commands*⟩`/`

The ⟨*lemma*⟩ argument is the lemma in the main text: `\critext` both prints this as part of the text, and makes it available to the ⟨*commands*⟩ you specify to generate notes. The `/` at the end terminates the command; it is part of the macro's definition so that spaces after the macro will be treated as significant.

For example:

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

1 I saw my friend
2 Smith on Tuesday.
2 Smith] Jones C, D.

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

The ⟨*lemma*⟩ may contain further `\critext` commands. Nesting makes it possible to print an explanatory note on a long passage together with notes on variants for individual words within the passage. For example:

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

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

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

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:

---

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

```
\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 VI p. 101 to find out why.) Aliases of the second kind may be nested without any problem.

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

Appendix A.2   Migration from `ledmac` to `eledmac`

In `eledmac`, some changes were made in the code to allow easy customization. This may cause problems for people who have already made their own. The next sections explain how to handle this.

If you have created your own series using \addfootins and \addfootinsX, you must use instead the \newseries command (see 5.5.1 p. 28), and remove any \Xfootnote command.

If you have customized the \XXXXXfmt command, please check whether you can achieve the same by the commands documented for display options (6 p. 28) or \Xfootnote options (5.2.2 p. 22). Otherwise please add a new ticket on Github to request a new function for doing this.[34]

If for some reason you do not want to make the modifications to use the new functions of `eledmac`, you can continue using your own \XXXXXfmt command, but you must replace:

```
\renewcommand*{XXXXfmt}[3]
```

with

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

---

[34]`https://github.com/maieul/ledmac/issues`

If you do not make that, you will get 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. Otherwise the command after the \protect will be discarded.

Appendix A.3   Migration to eledmac 1.5.1

The version 1.5.1 corrects a bug in stanzaindentsrepetition (cf. 8.3 p. 39). This bug had two consequences:

1. stanzaindentsrepetition did not work when its value was greater than 2.

2. stanzaindentsrepetition worked wrong when its value was equal to 2.

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

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

This code, in versions prior to 1.5.1, made the first line have an indentation of 0, the second line of 1, the third verse of 0, the fourth verse of 1 and so forth.

But this code should have instead achieved quite the contrary: the first line would have an indentation of 1, the second line of 0, the third line of 1, the fourth line of 0 an so forth.

So version 1.5.1 corrected this bug. If you want to keep the former presentation, you must change:

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

to:

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

Appendix A.4   Migration to eledmac 1.12.0

The migration to eledmac 1.12.0 is easy:

• You must first delete all the auxiliary files, then compile your document three times as usual.

• If you have modified \l@reg, which is not advisable, you must rename it to \@nl@reg.

There is an additional problem. If you have put text into brackets just after \pstart or \pend, this text will be considered to be an optional argument of \pstart or \pend (see 4.2.3 p. 16). If so, add a \relax between \pstart/\pend and the first bracket.

The version 1.12.0 also introduce a better way to handle sectional divisions inside numbered text. Please read 14.2 p. 52.

Appendix A.5   Migration to eledmac 17.1

This version changes the default setting of `\Xpstart`. Henceforth, pstart numbers will be printed in footnotes within the section of text where you have called `\numberpstarttrue`.

We do not see any reason to print them in the other sections. However, if you want to print the pstart numbers in all of the footnotes, whatever the section, without having to use `\numberpstarttrue`, you can use `\Xpstarteverytime`.

Appendix A.6   Migration to eledmac 1.21.0

Appendix A.6.1   `\Xledsetnormalparstuff` and `\ledsetnormalparstuffX`

The `\ledsetnormalparstuff` has been split into two different commands:

- `\Xledsetnormalparstuff` for critical notes;
- `\ledsetnormalparstuffX` for familiar notes.

Both commands can take an optional argument which is the series letter. If you have redefined `\ledsetnormalparstuff` or any of the commands which call them, you must change them accordingly.

Appendix A.6.2   Endnotes

In any case, delete the `.end` file before the next run.

The previous version of `Eledmac` had a bug: there were two spaces between the starting page number and the starting line number, but only one space between the ending page number and the ending line number.

As a matter of fact, a spurious space was added after the first `\printnpnum`. This spurious space has been deleted. However, if you want to keep the previous spurious space, you may load the package with the `oldprintnpnumspace` option.

If you have redefined `\endprint`, you must:

- Contact us and ask for the feature that required your hack, in order to avoid such a hack in the future.
- Use the new fifth argument.
- Add `\xdef\@currentseries{#4}` at the beginning of your own command.

Appendix A.7   Migration to eledmac 1.22.0

The `\ledinnote` command now takes a first optional argument, which is the label for the hyperreference. If you have redefined it, change your redefinition, and check whether you can avoid this redefinition by only redefining `\ledinnotemark`.

Appendix A.8   Migration to eledmac 1.23.0

You must delete the numbered auxiliary files before compiling with the new version of `eledmac`.

Appendix A.9   Migration from `eledmac` to `reledmac`

There are many changes in `reledmac` which require the user to make modifications.

Appendix A.9.1   Risk of 'no room for a new'

The risk to obtain a 'no room for a new something' error is greater in `reledmac` than it is in `eledmac`. See 17.2 p. 54 in order to know how to limit it.

Appendix A.9.2   Multiple indices with memoir

`Eledmac` and `ledmac` used the specific indexing tools of the `memoir` class designed to produce multiple indices. However, `eledmac` could also use `imakeidx` or `indextools` tools independently of the `memoir` class. This system forced to maintain redundant code. Since `reledmac`, we use only the `imakeidx` or `indextools` tools.

Consequently: Users of `memoir` are invited to use `indextool` or `imakeidx` to produce multiple indices.

Appendix A.9.3   Deprecated commands and options

The table of deprecated commands and their alternatives follows. Note that the way some commands must be used may have changed. Please read the handbook.

Deprecated command	*Replaced with*
`\addfootins`	`\newseries`
`\addfootinsX`	`\newseries`
`\critext`	`\edtext`
`\falseverse`	`\newverse`
`\interparanoteglue`	`\Xafternote` and `\afternoteX`
`\ledchapter`	`\eledchapter`
`\ledsection`	`\eledsection`
`\ledsetnormalparstuff`	`\Xledsetnormalparstuff` and `\ledsetnormalparstuffX`
`\ledsubsection`	`\eledsubsection`
`\ledsubsubsection`	`\eledsubsubsection`
`\noeledsec`	Package option `noeledsec`
`\noendnotes`	Package option `noendnotes`
`\pageparbreak`	`\ledpb`

The `ledsecnolinenumber` option has been removed, because it was related to deprecated commands.

The `oldprintnpnumspace` option has been removed too, because it was related to a historical bug. The `\usingedtext` and `\usingcritext` commands are also deprecated.

Appendix A.9.4   `\renewcommand` **replaced by command**

Many uses of `\renewcommand` have been replaced with uses of specific commands. Please read handbook about specific commands.

Deprecated `\renewcommand`	*Replaced with*
`\@led@extranofeet`	`\newseries`
`\apprefprefixmore`	`\setapprefprefixmore`
`\apprefprefixsingle`	`\setapprefprefixsingle`
`\endstanzaextra`	Optional argument of `\&`
`\hangingsymbol`	`\sethangingsymbol`
`\ledfootinsdim`	`\Xmaxnotes` and `\maxnotesX`
`\parafootftmsep`	`\Xparafootsep` and `\parafootsepX`
`\notenumfont`	`\Xnotenumfont`, `\Xendnotenumfont` and `\notenumfontX`
`\notefontsetup`	`\Xnotefontsize`, `\Xendnotefontsize` and `\notefontsizeX`
`\sidenotesep`	`\setsidenotsep`
`\startstanzahook`	Optional argument of `\stanza`
`\symplinenum`	`\Xsymlinenum`

Appendix A.9.5   **Commands the names of which have been changed**

In order to help the migration from `eledmac` to `reledmac`, you may load `reledmac` with `eledmac-compat` option. However, it is advised not to, and to change the command names themselves instead. In many cases, you use only a few of them, except the `\footparagraph` command.

Old command	*New command*
`\footparagraph`	`\Xarrangement`
`\footnormal`	`\Xarrangement`
`\foottwocol`	`\Xarrangement`
`\footthreecol`	`\Xarrangement`
`\footparagraphX`	`\arrangementX`
`\footnormalX`	`\arrangementX`
`\foottwocolX`	`\arrangementX`
`\footthreecolX`	`\arrangementX`
`\afterlemmaseparator`	`\Xafterlemmaseparator`
`\afternote`	`\Xafternote` and `\afternoteX`
`\afternumberinfootnote`	`\Xafternumber`
`\afterXrule`	`\Xafterrule`
`\afterXsymlinenum`	`\Xaftersymlinenum`
`\beforelemmaseparator`	`\Xbeforelemmaseparator`
`\beforenumberinfootnote`	`\Xbeforenumber`
`\beforeXnotes`	`\Xbeforenotes`
`\beforeXsymlinenum`	`\Xbeforesymlinenum`

Old command	New command
\bhookXnote	\Xbhookendnote
\bhookXnote	\Xbooknote
\boxendlinenum	\Xboxendlinenum
\boxlinenum	\Xboxlinenum
\boxlinenumalign	\Xboxlinenumalign
\boxstartlinenum	\Xboxstartlinenum
\boxsymlinenum	\Xboxsymlinenum
\boxXendlinenum	\Xendboxlinenum
\boxXendlinenumalign	\Xendboxlinenumalign
\boxXendstartlinenum	\boxXendstartlinenum
\letboxXendendlinenum	\Xendletboxendlinenum
\hsizetwocol	\Xhsizetwocol
\hsizethreecol	\Xhsizethreecol
\inplaceoflemmaseparator	\Xinplaceoflemmaseparator
\inplaceofnumber	\Xinplaceofnumber
\lemmaseparator	\Xlemmaseparator
\maxhXnotes	\Xmaxhnotes
\morethantwolines	\Xmorethantwolines
\nonumberinfootnote	\Xnonumber
\notesXwidthliketwocolumns	\noteswidthliketwocolumnsX
\noXlemmaseparator	\Xnolemmaseparator
\numberonlyfirstinline	\Xnumberonlyfirstinline
\numberonlyfirstintwolines	\Xnumberonlyfirstintwolines
\nonbreakableafternumber	\Xnonbreakableafternumber
\onlyXpstart	\Xonlypstart
\parafootsep	\Xparafootsep and \parafootsepX
\pstartinfootnote	\Xpstart
\pstartinfootnoteeverytime	\Xpstarteverytime
\symlinenum	\Xsymlinenum
\twolines	\Xtwolines
\twolinesbutnotmore	\Xtwolinesbutnotmore
\twolinesonlyinsamepage	\Xtwolinesonlyinsamepage
\txtbeforeXnotes	\Xtxtbeforenotes
\XendXafterlemmaseparator	\Xendafterlemmaseparator
\XendXbeforelemmaseparator	\Xendbeforelemmaseparator
\XendXinplaceoflemmaseparator	\Xendinplaceoflemmaseparator
\XendXlemmaseparator	\Xendlemmaseparator
\XendXmorethantwolines	\Xendmorethantwolines
\XendXtwolines	\Xendtwolines
\Xnonumberinfootnote	\Xnonumber
\lineref	\edlineref

Appendix A.9.6   Endnotes

With `reledmac`, there is now one auxiliary file for every endnotes set (.Aend, .Bend, .Cend etc.). If you have overriden `\doendnotes` (which you would not have done) you must adapt your code.

Appendix A.9.7   Z Series

The 'Z' series of notes has been removed. Only five series are provided now by default: A, B, C, D, E.

Appendix A.9.8   Internal commands

Users who have overriden internal commands, which is wrong, must adapt according to the following. Or better, they should not override any of such commands and use `reledmac` options instead.

- If you have modified `\Xfootfmt`, note that the fourth argument is now mandatory.

- `\unvxh` has been replaced with `\Xunvxh` and `\unvxhX` with two mandatory arguments.

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

A

B

F

T

Change History

v1.12.0.

\@nl@reg: To ensure compatibility with \musixtex, \@l becomes \@l. Consequently, \@l@reg becomes \@nl@reg. 85

General: Add \ledinnernote and \ledouternote commands. 45

Add \Xendparagraph and related settings. 37

Add hyperlink to crossref (needs hyperref package). 42

Compatibility with musixtex. 1

Debug eledmac sectioning command after using \resumenumbering. 1

Ensure that imakeidx is loaded *before* eledmac 231

New hooks: \Xafterrule and \afterruleX 36

New options for ragged-paragraph notes 36

New sectioning commands. 51

Optional arguments for \pstart and \pend. 16

\AtEveryPstart: New optional argument for \pstart, to execute code before it. . . 116

\edindex: Use correctly default index when imakeidx is loaded. 234

\endquote: \ledxxx sectioning commands are deprecated and replaced by \eledxxx commands. 265

\ifledRcol@: Add \ifledRcol@ for eledpar 66

\initnumbering@reg: \beginnumbering is defined only on eledmac, not on eledpar. 68

\l@dcsnote: \l@dlsnote,\l@drsnote and \l@dcsnote defined only one time, in eledmac, including needs for eledpar case. 222

\l@dgetsidenote@margin: \sidenotemargin is now directly defined in eledmac to be able to manage eledpar. 221

\l@dunhbox@line: \do@line is split in more little commands. 121

\newhookcommand@series@reload: Debug \beforenotesX and \maxhnotesX which did not work when called after \footparagraphX. 202

Debug \Xbeforenotes and \Xmaxhnotes which did not work when called after \footparagraph. 202

\pend: New optional argument for \pend, to execute code after it. 118

\stanza: &can have an optional argument: content to be printed after. 241

\Stanza can have an optional argument: content to be printed before. 241

Add \newverse macro, \falseverse deprecated. 241

v1.12.1.

\wrap@edcrossref: Fix spurious spaces. 215

v1.12.2.

\l@dunhbox@line: Fix a bug with critical notes at the tops of pages (added by v12.0.0) 121

v1.12.3.

General: Add macros for new messages since v0.7 60

Correct bug with side and familiar notes in tabular environments. 1

Debug \eledxxx with some paper size 1

Debug \ledinnernote and \ledouternote commands in the top of pages. 45

Debug left and right notes (bugs added by 1.12.0) 1

Underline lemma in \eledxxx when using draft mode. 1

\flag@end: \flag@start and \flag@end are now defined only one time for eledmac and eledpar 98

\flag@start send a error message when a \edtext is done without insert (note) . 98

\reledmac@error: Replaced error messages 61

v1.12.4.

General: Debug spurious page breaks before \chapter (bug added in 1.12.0) 1