`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 tools in order to typeset scholarly editions. It is based on the `eledmac` package, which was based on the `ledmac` package, which was based on the `edmac` TeX package.

It can be used in combination with `reledpar` in order to typeset two texts in parallel, like an original text and its translation in a 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 every possible case). Examples starting with "1-" are for basic uses, those starting with "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 at:

`http://geekographie.maieul.net/146`

Contents

---

[*]This file (`reledmac.dtx`) has version number v2.13.1, last revised 2016/06/18.

[†]`maieul at maieul dot net`

1

1   Introduction

1.1   Aim of the package

The `reledmac` 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 to both prose and verse;

- multiple series of footnotes and endnotes;

- block or columnar formatting of the footnotes;

- simple tabular material may be line numbered;

- indexing keyed to page and line numbers.

`reledmac` 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 other LaTeX packages for typesetting critical editions. However, the aim of `reledmac` is to provide an "all in one" and flexible tool in the field of critical editions.

Any suggestions for new features are welcome.

This manual contains a general description of how to use `reledmac` followed by the complete source code and its extensive documentation (in sections I and following,

enumerated with Roman numerals). It ends with a list of actions to do when migrating from one version to other, a change history and an index to the source code.

You do not 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 earlier sections. But no documentation, however thorough, can cover every question that comes up and many can be answered quickly by consulting the code. On a first reading, we suggest that you 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 the 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.

As of 1994, we were pleased to be able to say that `edmac` was being used for the real-life book production of several interesting editions, such as the Latin texts of Euclid's *Ele-*

---

[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).

ments,[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.

1.2.2 `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. It is increasingly powerful and flexible, but it also has become increasingly divergent from the original TeX macro.

1.2.3 `eledmac`

Important changes were put in version 1.0, to make `ledmac` more easily extensible (see 7 p. 33). These changes can trigger small problems with the old customization. That is why a new name was selected: `eledmac` (extended `ledmac`).

---

[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.

To migrate from `ledmac` to `eledmac`, please read Appendix A.2 p. 335.

1.2.4 `reledmac`

`eledmac` has facilitated the creation of customized critical editions. However, the changes made to allow such customization were made in a non-systematic way. Many deprecated commands were kept and many technical 'debts' were accumulated, hindering the future evolution of the package.

For these reasons, Maïeul Rouquette decided on a spring cleaning of the code. As some commands name were changed, the resulting compatibility was broken (a little).

A new name was selected: `reledmac` (extended renewed `eledmac`). To migrate from `eledmac` to `reledmac`, please read Appendix A.9 p. 338.

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

A collaborative list of works edited with (r)(e)ledmac is available at `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 on the first run, it takes two more passes through LaTeX to be sure that everything is correctly placed. If you make any subsequent changes altering the number of lines or notes, the input file may similarly require three passes to get everything to the right place. `reledmac` will tell you that you need to make more runs when it detects changes, 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.

3   Compatibility warning

However, the best way to be sure that one has made the right number of runs is to use some of LaTeX's run scripts like *latexmk*.

If you use other classes than `\article` or `\book`, or modify the layout with `geometry`, some settings should be made to have correct height for the blocks of notes.

Please read 7.13.5 p. 43.

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.

4   Options

The package can be loaded with a number of global options which are listed here. There are two types of options: 1) options which provide specific features, and, 2) options which optimize the package's performance. It is advisable for you to read the relevant parts of the handbook, before reading about the first type of option (specific features), but you can look at the second type (package optimization) in your first reading of the manual.

4.1   Specific features

draft underlines lemmas in the main text.

auxdir `reledmac` generates auxiliary files. It could be useful to store them in a specific directory. You can set it using auxdir=⟨*folder*⟩ option. Note the two following point:

1. TEX is not able to create folder. You should create it yourself.
2. The option does not change the default LATEXauxiliary files (.aux, .toc, ...).

eledmac-compat help to migrate from `eledmac` to `reledmac` (see Appendix A.9.5 p. 339).

nopenalties must be called in some cases when using paragraphed endnotes (**??** p. **??**

nopbinverse prevents page break within verse environment.

noquotation by default, the quotation environment is redefined within numbered text. You can disable this redefinition with `noquotation` (see 17 p. 62).

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

widthliketwocolumns set the width of the text printed in a single column to be the same as the width of the text printed in two parallel columns with `reledpar`. This is useful when alternating between normal and parallel typesetting.

xindy and `xindy+hyperref` select `xindy` as the index processor (13.4 p. 56).

4.2   Optimizing package 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 (16.2 p. 61).

noend disables tools for endnotes (`\Aendnote`, `\Bendnote` etc.). If you do not need endnotes, this option lets `reledmac` 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 use of a series of critical notes and a new series of normal
notes inside minipage and ledgroup environments (see 10 p. 49). 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 con-
sumes memory space and processing speed. This is why, if your work does not
require the entire 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.

5   Text lines and paragraphs numbering

5.1   Text lines numbering

\beginnumbering   Each section of numbered text must be preceded by \beginnumbering and followed by
\endnumbering   \endnumbering, as in the following example.

```
\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 com-
mands. 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).

5.2   Paragraphs

5.2.1   Basics

\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
```

5.2.2   Automatically producing \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 com-
mand 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]

5.2.3   Content before specific \pstart and after specific \pend

Both \pstart and \pend can take a optional argument in brackets. Its content will
be printed before the beginning of \pstart / after the end of \pend instead of the
argument of \AtEveryPstart / \AtEveryPend. If you need to start a \pstart with
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 9 p. 45) or reledpar (see
19.1.8 p. 66).

A \noindent is automatically added before this argument.

5.2.4   Content before every \pstart and after every \pend

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

5.2.5   Numbering paragraphs (\pstart)

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

    \setcounter{pstart}{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.

5.2.6   Languages written in Right to Left

If you use languages written right to left with LuaLATEX or X LATEX, you must switch text
direction *before* the \pstart command.

5.2.7   Memory limits

**This paragraph is kept for history, but the problems described below should not
appear with the most recent version of LATEX.**
\pausenumbering   reledmac stores a lot of information about line numbers and footnotes in memory
\resumenumbering   as it goes through a numbered section. But at the end of such a section, it empties its

---

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

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 solution 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 type,

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

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

5.3 Lineation commands

5.3.1 Disabling lineation

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

5.3.2 Setting lineation start and step

\firstlinenum   By default, reledmac numbers every 5th line. There are two counters that control
\linenumincrement   this behaviour: firstlinenum and linenumincrement. They can be changed using
\firstlinenum{⟨*num*⟩} and \linenumincrement{⟨*num*⟩}. \firstlinenum specifies the first line that will have a printed number, and \linenumincrement is the difference between 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 integer 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 empty 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.

5.3.3   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.

5.3.4   Setting line number margin

\linenummargin   The command \linenummargin{⟨*location*⟩} specifies the margin where the line (or
pstart) numbers will be printed. The permissable values for ⟨*location*⟩ are left, right,
inner, or outer: for example, \linenummargin{inner}. The package's default set-
ting 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 the value in effect at the
\pend of a numbered paragraph. Apart from an initial setting for \linenummargin,
only change \linenummargin 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 of the current paragraph).

5.3.5   Other settings

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

5.4   Changing the line numbers

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

5.4.1   Sublineation

`\startsub`
`\endsub`
You insert the `\startsub` and `\endsub` commands in your text to turn sub-lineation on and off. For example, stage directions in plays 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 it changes in the middle.

You can change the separator between line number and subline number or using `\Xsublinesep` without any option argument (7.2.8 p. 36 or using `\Xsublinesepside`. But in the second case, it will change the separator only for line number in side, not for the footnotes.

5.4.2   Locking lineation

`\startlock`
`\endlock`
The `\startlock` command, used in running text, locks the line number at its current value, until you insert `\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. But in this case you may use the `\stanza` mechanism, see 9 p. 45.

`\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, assuming that the settings of the previous parameters requires the display of 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}`.

5.4.3   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 ex-
ample between a \pend and a \pstart. It sets the line number to ⟨*num*⟩. It has no
effect if used within a \pstart...\pend group.

5.4.4   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.

5.4.5   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 num-
ber 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.

\hidenumberingonleftpage   \hidenumberingonleftpage is like hidenumbering, but is applied only on left
page. \hidenumberingonrightpage is applied on right page. They can be useful if
the position of the line number is depending of the position of the page, but the position
of marginal note is fixed.

5.4.6   Execute code at each line

\dolinehook   reledmac provides an advanced feature for users. 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
cases, the latter is more useful than the former. The file examples/2-line_numbers_in_header.tex
provides an example for printing the first and last line numbers of a page in the header.

6 Apparatus commands

6.1 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 familiar footnote.

So :

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

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

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

6.2 Critical notes

6.2.1 The lemma

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

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

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; an \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.)

6.2.2   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 6.6.1 p. 32.

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. 7.2.4 p. 34).

- nonum disables line numbering for this note. A horizontal blank space is added instead. You can use \Xinplaceoflemmaseparator to set it (7.5.1 p. 38).

- nosep to disable the lemma separator for this note.

- linerangesep=⟨*c*⟩ to change to ⟨*c*⟩ the separator between start line and end line for this particular note.

Example: \Afootnote[nonum]{⟨*text*⟩}.

6.2.3   Endnotes

\Aendnote   **Inserting endnotes**   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   reledmac.
\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. 7.2.4 p. 34).

- nonum to disable line number for this particular note.

- nosep to disable the lemma separator for this particular note. A horizontal blank space is added instead. You can use \Xendinplaceoflemmaseparator to set it (7.5.2 p. 39).

- linerangesep=$\langle c \rangle$ to change to $\langle c \rangle$ the separator between start line and end line for this particular note.

\doendnotes **Printing endnotes**   Normally, endnotes are not printed: you must use the `\doendnotes{`$\langle s \rangle$`}`, where $\langle s \rangle$ 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 $\langle s \rangle$ series are printed, for all numbered sections.

\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{`$\langle s \rangle$`}`. For each value of $\langle s \rangle$, 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 (7.5.2 p. 39).

As endnotes may be printed at any point in the document they always start with the page number where they are called.

toendnotes **Code between endnotes**   Sometimes, it is useful to insert content between endnotes
Xtoendnotes of the same series: for example to separate endnotes of different sections of the same text. In this case, you could use *inside numbered text* the command:
`\toendnotes[`$\langle series \rangle$`]{`$\langle content \rangle$`}` where $\langle series \rangle$ is a comma-separated list of the series of endnotes where $\langle content \rangle$ must be inserted. If $\langle series \rangle$ is empty, then $\langle content \rangle$ is inserted to all the series.

For example:

```
\toendnotes{\section{Section's title}}
```

Alternatively, you can use `\Xtoendnotes{`$\langle content \rangle$`}`, where "X" must be replaced by a series letter.

Remember that the endnotes are temporarily stored in an auxiliary file. That means in general you want to write the $\langle content \rangle$ in the auxiliary file *without expanding it*, that is without interpreting TeX content.

However, in some case, you may want to write once-expanded[14] version of the ⟨*content*⟩, that is the version where the commands are expanded on the first level. This can be, for example, to get a counter value. Use the starred version in this case. For example:

```
\Atoendnotes*{\string\section{Letter 1 (chap. \thechapter)}}
```

6.2.4   Paragraph in critical apparatus

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

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

Note that you *cannot* use paragraphs (e.g. blank lines or \par) inside of notes, when they are set to `paragraph` arrangement!

6.2.5   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 and 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. ⟨*arg*⟩ actually consist of 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). I.e.
.\linenum{⟨*start page*⟩ | ⟨*s. line*⟩ | ⟨*s. sub-l.*⟩ | ⟨*end p.*⟩ | ⟨*e. l.*⟩ | ⟨*e. sub-l.*⟩ | ⟨*font*⟩ | }
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 only 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 notes. Its use comes in situations that \edtext has trouble dealing with for whatever reason. If you need notes for overlapping passages that aren't

————————

[14]The expansion mechanism' of TEXis a quite complex problem, but fundamental. We have no place to explain it fully here. Read introduction to TEXto understand well.

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 (11 p. 49) 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.

6.2.6   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 type 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:[15]

```
\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}}
```

6.3   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.

6.3.1   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.

---

[15]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.

6.3.2   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, "ᾳ" 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 configuration (the computer-input system).

Inside `reledmac`, the `\sameword` command considers these two unicodes (code positions) 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[16] 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` command.

6.3.3   Use with `\lemma` command

If you use the `\lemma` command, `reledmac` 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}
            and other \sameword{sw}
            and again \sameword{sw}
            it is all}%
    {\lemma{\sameword{sw} \ldots all}\Afootnote{critical note}}.%
```

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

Consequently, you must tell `reledmac` to which instance of `\sameword` you are referring in the first argument of `\edtext`:

---

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

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

- In the first argument of \edtext, use \sameword with the optional argument [⟨*X*⟩]. ⟨*X*⟩ is the depth of the \edtext where the \lemma is used. So if the \lemma is called in a \edtext inside another \edtext, ⟨*X*⟩ is equal to 2. If the \lemma is called in a \edtext "of first level", ⟨*X*⟩ is equal to 1. If the lemma is called in both 1 and 2 \edtext depth, ⟨*X*⟩ is 1,2. If that word is referenced in the lemma of every \edtext depth, ⟨*X*⟩ 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 ⟨*X*⟩ 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 example figure, 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.

$$1_{\text{inlemma}} \boxed{2} \; 3_2 \;^{1\ldots3} \; 4 \; 5_1 \;^{1\ldots5}$$

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.

6.3.4   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}%
}
```

6.4   Apparatus of Manuscripts

The critical notes mostly refer to textual variants between manuscripts which contain the text to be edited. It may so happen that the manuscripts only contain parts of the text. Depending on one's wishes, `reledmac` can generate lists of relevant manuscripts for any delimited portion of text. Such lists are referred to as "apparatuses of manuscripts".

To produce an apparatus of manuscripts with `reledmac`, you have to insert specific commands that are used to mark the sections for which only part of the manuscripts are relevant. These commands will be processed, and **after the second TEX run**, corresponding apparatuses of manuscripts will be inserted in the first (viz. 'A' series) level of footnotes.

As the insertion of this apparatus can change the page breaks, you may have to run TEX two or more times. We strongly recommend to use tools like *latexmk* to do that.

6.4.1   Marking sections of text

\msdata   \msdata{⟨*text*⟩} must be inserted at the point where a section for which only part of the manuscripts are relevant starts. ⟨*text*⟩ can be any arbitrary text, viz. a list of the manuscripts that are used for the section that starts. The command must be attached right at the point where the section starts, with no space, like so:

```
\msdata{ABC}Lorem ipsum
```

Which means that the section of text starting by "Lorem ipsum" is witnessed by manuscripts A, B and C.

\stopmsdata   \stopmsdata must be inserted at the point where the section of text previously marked by \msdata ends. The command must be attached right to the end of the section, with no space. As \stopmsdata is a LATEX argumentless macro, it will gobble the following space. To keep that space, you have to either append a backslash followed by a space or {} to \stopmsdata, like so:

```
\msdata{ABC}Lorem ipsum dolor
[…]
amet\stopmsdata{} \msdata{ABCD}sic transit […]
```

Which means that the part of text containing "Lorem ipsum dolor … amet" is witnessed by manuscripts A, B and C, while the part of text starting by "sic transit" is witnessed by manuscripts A, B, C and D.

\stopmsdata is also automatically inserted by \msdata.

Note that in most cases, any \stopmsdata is followed by \msdata. However, as these two command are usually separated by a space, it may happen that a line break be automatically inserted between them. This is why it is advised to always insert \stopmsdata, even if \msdata inserts it in case it is forgotten.

6.4.2   Layout of the apparatus of manuscripts

On every page, the apparatus of manuscripts marks the corresponding section with starting and ending line numbers. However, the following rules will be applied:

- If the section does not start on the current page, the starting line number will be the line number of the first line on the page.

- If the section does not stop on the current page, the ending line number will be the line number of the last line on the page.

- If the section neither starts nor ends on the current page, no line number will be printed. The same is true in case both \msdata is called at the very beginning of the page and \endmsdata is called at the very end of the page.

6.4.3   Settings

As the apparatus of manuscripts technically consists of first-level critical notes ('A' series), any setting available for critical notes can be applied (7 p. 33). However, the following *additional* commands are available.

\setmsdataseries   The series used by default for the apparatus of manuscripts is series A. However, you can change it with \setmsdataseries{⟨*series*⟩}.

\setmsdatalabel   As the apparatus of manuscripts consists of regular critical footnotes, a lemma is associated to them. By default, it is "Ms.". You can change it using \setmsdatalabel{⟨*txt*⟩}.

6.5   Familiar notes

6.5.1   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.

6.5.2   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 defined as:
\newcommand*{\bodyfootmarkA}{%
   \hbox{\textsuperscript{\normalfont\@nameuse{@thefnmarkA}}}}
The command \footfootmarkA controls the appearance of the mark at the start of the footnote text. It is defined as:
\newcommand*{\footfootmarkA}{\textsuperscript{\@nameuse{@thefnmarkA}}}
   There are similar command triples for the other series.

6.5.3   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 `reledmac` provides this automatically.

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

`\providecommand*{\multfootsep}{\textsuperscript{\normalfont,}}`
and can be changed if necessary.

6.6   Changing series

6.6.1   Create a new series

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

6.6.2   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–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}

6.6.3   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.

6.7   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}

7 Critical apparatus appearance

Some commands can be used to change the display of the footnotes. All can have an optional argument $[\langle s \rangle]$, 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 apply to the entire series.

When a length, noted $\langle l \rangle$, is used, it can be stretchable: `a plus b minus c`. The final length `m` is calculated by LaTeX to have: $a - c \leq m \leq a + b$. If you use some relative unit[17], it will be relative to font size of the footnote, except for commands concerning the place kept by the notes — including blank space.

Some commands are boolean, indicating when an option is enabled. If you want to disable the option after enabling it, you must use [false] as the second optional argument. For example:

- `\XX[A][false]` to disable the 'XX' option for the series A.

- `\XX[][false]` to disable it for all series.

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.

7.1 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 $\langle s \rangle$ series of critical footnotes and `\arrangementX[⟨s⟩]{⟨a⟩}` to change the arrangement of the $\langle s \rangle$ series of familiar footnotes.

The value of $\langle a \rangle$ can be one of the following

- `paragraph` formats all the footnotes of a series as a single paragraph. If you use this arrangement, you are strongly encouraged to read 19.1.6 p. 65.

- `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]Like `em` which is the width of an 'm' in a given font.

Note that you *cannot* use paragraphs (e.g. blank lines or \par) or line breaks (\break or \linebreak or \newline etc.) inside of notes, when they are set to paragraph arrangement!

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 note arrangement again.

\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 the arrangement macro again afterwards to take account of the new value.

7.2 Control line number printing

7.2.1 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⟩].

\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⟩], a distinction is made.

\Xsymlinenum
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 non-robust command.

\Xendnumberonlyfirstinline
\Xendnumberonlyfirstintwolines
\Xendsymlinenum
For endnotes, \Xendnumberonlyfirstinline; \Xendnumberonlyfirstintwolines and \Xendsymlinenum are the equivalents of \Xnumberonlyfirstinline; \Xnumberonlyfirstintwolines and \Xsymlinenum.

7.2.2 Arbitrary text before line number

\Xbeforenumber
\Xbeforenumber[⟨s⟩]{⟨txt⟩} allow to insert ⟨txt⟩ before the line number, only when the line number is printed, so taking into accout \Xnumberonlyfirstinline and similar.

7.2.3 Separator for line range

\Xlinerangeseparator
\Xendlinerangeseparator
By default, the separator between the begin line and the end line in a lines' range is an endash in a normal font (\textnormal{--}). You can change it for critical footnotes with \Xlinerangeseparator[⟨s⟩]{⟨text⟩}, and with \Xendlinerangeseparator[⟨s⟩]{⟨text⟩} for critical endnotes.

7.2.4 Abbreviate line range

\Xtwolines
\Xmorethantwolines
If a lemma is printed on two subsequent lines, reledmac will print the first and the last

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.

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*⟩].

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*⟩].

You can disable \Xtwolines and related for a specific note by using the '[fulllines]' argument in the note macro cf. 6.2.2 p. 24.

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

7.2.5   Disable line number

\Xnonumber   You can use \Xnonumber[⟨*s*⟩] if you do not want to have the line number in a footnote.
\Xendnonumber   \Xendnonumber[⟨*s*⟩] is the same for endnote.

7.2.6   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. 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.

7.2.7   Printing stanza number

\Xstanza   You can use \Xstanza[⟨s⟩] if you want to print the stanza number in the footnote,
before the line and subline number.
          Of course the stanza number is printed only when you use \numberstanza

\Xstanzaseparator

          When using \Xstanza, you can use \Xstanzaseparator[⟨s⟩]{⟨text⟩} to print
⟨text⟩ after the stanza number. Default value is empty.

7.2.8   Separator between line and subline numbers

\Xsublinesep   \Xsublinesep[⟨s⟩]{⟨txt⟩} changes the separator between line and subline in foot-
notes.
          **Employed without optional argument, it also change separator in side num-
ber**.
\Xendsublinesep   \Xendsublinesep[⟨s⟩]{⟨txt⟩} does the same thing for endnotes.
          **However, it does not change anything for the separator in side number. Use
\Xsublinesep without optional argument or \Xsublinesepside{⟨txt⟩} to do it.**
          The default value is \textnormal{.}.

7.2.9   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.
\Xendbeforenumber   \Xendbeforenumber and \Xendafternumber are the equivalents of \Xbeforenumber
\Xendafternumber   and \afterenumber for endnotes.
\Xnonbreakableafternumber   By default, the space defined by \Xafternumber is breakable. With \Xnonbreakableafternumber[⟨s⟩]
it becomes nonbreakable.

7.2.10   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.
\Xendbeforesymlinenum   \Xendbeforesymlinenum and \Xendaftersymlinenum are the equivalents of
\Xendaftersymlinenum   \Xbeforesymlinenum and \Xaftersymlinenum for the endnotes.

7.2.11   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.

7.2.12   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.

\Xendboxsymlinenum   \Xendboxsymlinenum[⟨s⟩]{⟨l⟩} is the same as \Xboxsymlinenum but for end-
notes.

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

7.3   For endnotes

\Xendbeforepagenumber   \Xendbeforepagenumber[⟨*s*⟩]{⟨*text*⟩} defines the text before the page number in end-
notes. Default value is p. ("p" followed by a dot).

\Xendafterpagenumber   \Xendafterpagenumber[⟨*s*⟩]{⟨*text*⟩} defines the text after the page number in
\Xendlineprefixsingle   endnotes. Default value is) (open parenthesis followed by a single space). \Xendlineprefixsingle[⟨*s*⟩]
defines the text before the line number in endnotes, when there is only one line. Default
value is empty. \Xendlineprefixmore[⟨*s*⟩]{⟨*text*⟩} defines the text before the line
\Xendlineprefixmore   number in endnotes, when there is more than one line. Default value is empty. If you
don't define it, use the value defined by \Xendlineprefixsingle.

7.4   Arbitrary code around line number

\Xendbhooklinenumber   \Xendbhooklinenumber[⟨*s*⟩]{⟨*code*⟩} is used to execute code before line number in
endnotes. The code is executed before the \Xendbeforelinenumber space and before
the \Xendnotenumfont font setting.

\Xendahooklinenumber   \Xendahooklinenumber[⟨*s*⟩]{⟨*code*⟩} is used to execute code after line number in
endnotes. The code is executed after the \Xendafternumber space.

\Xendbhookinplaceofnumber   \Xendbhookinplaceofnumber[⟨*s*⟩]{⟨*code*⟩} is used to execute code before space
or symbol which replace line number in endnotes. The code is executed before the
\Xendbeforesymlinenum space and before the \Xendnotenumfont font setting.

\Xendahookinplaceofnumber   \Xendahookinplaceofnumber[⟨*s*⟩]{⟨*code*⟩} is used to execute code after space
or symbol which replace line number in endnotes. The code is executed after the
\Xendaftersymlinenum space.

7.5   Separator between the lemma and the note

7.5.1   For footnotes

\Xlemmaseparator   By default, in a footnote, the separator between the lemma and the note is a right bracket
(\rbracket)[18]. 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 following text.

\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 sep-
arator 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.

---

[18]For polyglossia, when the lemma is RTL, the bracket automatically switches to a left bracket.

7.5.2 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. A 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 following text.

\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.

7.6 Font style

7.6.1 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.

7.6.2 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.

\Xlemmafont Use \Xlemmafont[⟨*s*⟩]{⟨*cmd*⟩} to apply a TEX font command to the lemma. For
\Xendlemmafont example, to have boldface lemma:

\Xlemmafont{\bfseries}

\Xendlemmafont[⟨*s*⟩]{⟨*cmd*⟩} is the same for endnotes.

7.6.3 For all notes

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

\notefontsizeX   \notefontsizeX[⟨*s*⟩]{⟨*command*⟩} is used to define the font size of familiar 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.

7.7 Wrapping notes

7.7.1 Wrapping lemmas

\Xwraplemma   \Xwraplemma[⟨*s*⟩]{⟨*cmd*⟩} is used to wrap, in the footnote, the lemma in a LATEX com-
mand. For example, with the bidi package, to ensure having a lemma written right to
left, use \Xwraplemma{\RL}.

\Xwrapendlemma   \Xendwraplemma[⟨*s*⟩]{⟨*cmd*⟩} is the same for endnotes.

7.7.2 Wrapping contents

\Xwrapcontent   \Xwrapcontent[⟨*s*⟩]{⟨*cmd*⟩} is used to wrap the footnote contents — excluding the
lemma — in a LATEXcommand.

For example, if the language of your note is not the same as the language of
the lemma, use \Xwrapcontent{\foreignlanguage{⟨*language*⟩}} (with babel) or
\Xwrapcontent{\text⟨*language*⟩} (for babel).

\Xendwrapcontent   \Xendwrapcontent[⟨*s*⟩]{⟨*cmd*⟩} is the same for endnotes.

\wrapcontentX   \wrapcontentX[⟨*s*⟩]{⟨*cmd*⟩} is the same for critical footnotes.

7.8 Indent of notes content

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

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

\Xendhangindent   For critical endnotes NOT paragraphed you can define an indentation with \Xendhangindent[⟨*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.

7.9   Arbitrary code at the beginning of notes

The three next commands add 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[⟨*s*⟩]{⟨*code*⟩} is to be used at the beginning of the critical footnotes.
\bhooknoteX   \bhooknoteX[⟨*s*⟩]{⟨*code*⟩} is to be used at the beginning of the familiar footnotes.
\Xendbhooknote   \Xendbhooknote[⟨*s*⟩]{⟨*code*⟩} is to be used at the beginning of the endnotes.

7.10   Arbitrary code before inserting note

\Xbeforeinserting   \Xbeforeinserting[⟨*s*⟩]{⟨*code*⟩} and \beforeinsertingX[⟨*s*⟩]{⟨*code*⟩} are very
beforeinsertingX   technical commands.

They allow one to add any arbitrary code just before the footnotes are added in the list of footnotes. The main use is to insert text direction code. For example, if you edit right-to-left text with `bidi`, but want your critical footnote be left-to-right, use \Xbeforeinserting\LTR. You should also use \Xwraplemma to ensure your lemmas are right-to-left in a left-to-right paragraph (7.7.1 p. 40)).

Note that the changes are local to the footnote.

7.11   Options for footnotes in columns

7.11.1   Alignment

\Xcolalign   By default, text in footnotes of two or three columns are flush left and without hy-
\colalignX   phenation. 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 `reledmac` setting.

\RaggedRight to have text left aligned *with hyphenation* (requires `ragged2e`).

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

\RaggedLeft to have text right aligned *with hyphenation* (requires `ragged2e`).

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

\Centering to have text centered *with hyphenation* (requires `ragged2e`).

7.11.2 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. Default value is .45 \hsize.

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

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

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

7.12 Options for paragraphed footnotes

7.12.1 Mark separation of notes

\Xafternote   You can add some horizontal space after a note by using \Xafternote[⟨s⟩]{⟨l⟩} (for
\afternoteX   critical 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 \Xparafootsep{$\parallel$}.

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.

7.12.2 Ragged text

\Xragged   Text in paragraphed critical notes is justified, but you can use \Xragged[⟨s⟩]{L} if you want it to be ragged left (i.e., right justified), or \Xragged[⟨s⟩]{R} if you want it to be ragged right (i.e., left justified).

\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.

7.13 Options for block of notes

7.13.1 Text before notes

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

7.13.2 Code before notes

\Xbhookgroup   While \Xtxtbeforenotes is for typesetting code before notes, \Xbhookgroup and
\bhookgroupX   \bhookgroupX (respectively for critical and familiar) are for executing code before a groups of notes, between the rules and the printing of the notes.

7.13.3 Spacing

\Xbeforenotes You can change the vertical space 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 used by reledmac decreases by 3pt. This 3pt decrease is not changed by this command.

\beforenotesX You can change the vertical space printed before the rule of the familiar notes with \beforenotesX[⟨s⟩]{⟨l⟩}. The default value is 1.2em plus .6em minus .6em.

Be careful, the standard LATEX footnote rule, which is used by reledmac, decreases 3pt. These 3pt are not changed by this command.

\Xprenotes You can set the space before the first series of critical notes printed on each page and set a different amount of space for each subsequent series on the page. You can do it with \Xprenotes{⟨l⟩}. The default value is 0pt. You can disable this feature by setting the length to 0pt.

\prenotesX You can set the space before the first printed (in a page) series of familiar notes to be different from the space before other series. The default value is 0pt. You can do this with \prenotesX{⟨l⟩}. You can disable this feature by setting the length to 0pt.

7.13.4 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 reledmac, adds 2.6pt. These 2.6pt are not changed by this command.

\afterruleX You can change the vertical space printed after the rule of the familiar notes with \afterruleX[⟨s⟩]{⟨l⟩}. The default value is 0pt.

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

7.13.5 Maximum height

\Xmaxhnotes By default, one series of critical notes can take up to 80% of \vsize, before being broken to the next page. If you want to change the size use \Xmaxhnotes[⟨s⟩]{⟨l⟩}. Be careful : the length can't be flexible, and is relative to the the current font. For example, if you want the note to take, at most, 33% of the text height, do \Xmaxhnotes{.33\textheight}.

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

Note that in many cases, you should call these commands in the begin of the document, because the \vsize in the preamble is not the same as \vsize after the preamble. That why we recommend to you to add in your preamble

```
\AtBeginDocument{
  \maxhnotesX{0.8\textheight}
  \Xmaxhnotes{0.8\textheight}
}
```

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

7.13.6 Width

\Xwidth   \Xwidth[⟨*s*⟩]{⟨*l*⟩} sets the total width of critical footnotes. \widthX[⟨*s*⟩]{⟨*l*⟩} does
\widthX   the same for familiar footnotes.

⟨*l*⟩ can be a length expression, parsable with \dimexpr. For example:

```
\Xwidth{\columnwidth+\marginparsep+\ledrsnotewidth}
\widthX{\columnwidth+\marginparsep+\ledrsnotewidth}
```

Note that changes the with of the block of notes. If you want to change the width of each column when typesetting notes in columns, use \Xhsizetwocol, \Xhsizethreecol, \hsizetwocolX, \hsizethreecolX, see 7.11.2 p. 42.

7.14 Footnotes and the `reledpar` columns

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

- \Xnoteswidthliketwocolumns[⟨*s*⟩] to create critical notes with a two-column size width.

- \noteswidthliketwocolumnsX[⟨*s*⟩] to create familiar notes with a two-column size width.

7.15 Endnotes in one paragraph

\Xendparagraph   By default, any new endnote starts a new paragraph. Use \Xendparagraph[⟨*s*⟩] 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 common separator is the double pipe (‖), which you can set by using \Xendsep{$\parallel$}.

8 Fonts

One of the most important features of the appearance of the notes, and indeed of your whole document, will be the fonts used. We will first describe the commands that give you control over the use of fonts in the different structural elements of the document, especially within the notes, and then in subsequent sections specify how these commands are used.

For those who are setting up for a large job, here is a list of the complete set of `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.

\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.

9 Verse

9.1 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.

9.2 Define stanza indents

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

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

      To specify these multiples one invokes, for example

\setstanzaindents{3,1,2,1,2}.

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

If it is known that each stanza line will fit in 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. 9.6 p. 47.

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.

9.3   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 indicate that they are repeated, defining the value of the stanzaindentsrepetition counter at *n*. For example:

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

   is like

```
\setstanzaindents{5,1,0,1,0,1,0,1,0,1,0}
```

   Be careful: the feature is changed in eledmac 1.5.1. See Appendix A.3 p. 336.

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.

9.4   Manual stanza indent

\stanzaindent
\stanzaindent*

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

If you use the mechanism of indent repetition, the next verse will be printed as it should be even if the current verse would have its normal indent value. In other 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.

9.5   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.

9.6   Hanging symbol

It is possible to insert a symbol in each line of hanging verse, as in French typography; for example, the opening bracket '['. To insert it in `reledmac`, use macro
\sethangingsymbol `\sethangingsymbol{⟨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}
```

9.7   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 18.2 p. 63 for further details.

9.8   Content before/after verses

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

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

  Use `\AtEveryStanza{⟨arg⟩}` to automatically add content at the begining of 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.

  Use \AtEveryPend{⟨*arg*⟩} to automatically add content after verses (including the final one) and \AtEveryPstart{⟨*arg*⟩} to automatically add content before verses (including the first one).

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

  Use \AtEveryStopStanza to automatically add content at the end of stanzas.

9.9   Numbering stanza

\numberstanzatrue   If you want to automatically number stanzas, use \numberstanzatrue. In this case,
\numberstanzafalse   the line number will restart at each \stanza.

 If you want to disable this feature again, use \numberstanzafalse.

 You can use this feature in combination with \Xstanza (7.2.7 p. 36).

thestanza   . You can redefine \thestanza to change the aspect of stanza number. Default value is:

```
\renewcommand{\thestanza}{%
  \textbf{\arabic{stanza}}%
}
```

 You can change the value of the stanza counter with the usual commands of LaTeX.

\stanzanumwrapper   You can redefine \stanzanumwrapper in order to modify the way the stanza number is inserted in the flow of text. Default value is:

```
\newcommand{\stanzanumwrapper}[1]{%
    \flagstanza{#1}%
}
```

9.10   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.

9.11   Notes on empty lines

Since v2.3.0 of reledmac, empty lines when typesetting verses no longer produce new paragraphs, and consequently, do not insert vertical spaces. Use optional argument of \stanza or \newverse to insert vertical space (9.8 p. 47).

10   Grouping

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

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

You can also put familiar footnotes (see section 6.5) 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` environment are typeset at the end of the environment, as with minipages, but the environment includes normal page breaks. The environment makes no change to the textwidth so it appears as normal text; it just might be that footnotes appear in the middle of a page, with text above and below.

`ledgroupsized`   The `ledgroupsized` environment is similar to `ledroup` except that you must specify 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}

11   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.

11.1   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 type \edlabel{toves-3}, for example.[19]

\edpageref   Elsewhere in the text, either before or after the \edlabel, you can refer to its location via \edpageref{⟨lab⟩}, or \edlineref{⟨lab⟩} will produce, respectively, the page, line, sub-line and pstart on which the \edlabel{⟨lab⟩} command occurred.

\edlineref
\sublineref
\pstartref   Note that the \edlineref command insert the side flag after the line number.

---

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

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

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

You will be warned if you use \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.)

11.2   Cross-referencing to a critical note

If you want to refer to a word which is a lemma word, the \edlabel command should
be in the first argument of \edtext command.

If you want to refer to the content of a a \Xfootnote, the line and subline number
printed will be the start line.

If you want to refer to starting and ending lines, you should use \appref and related
tools (11.6.2 p. 52).

11.3   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 6.2.5 p. 26).

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.)

- With reledpar, the \xlineref does not insert the right side flag, in order to
  obtain a line number. Use \xflagref to obtain the side flag, depending of your
  flag.

11.3.1 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., 6.2.5 p. 26 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.

11.4 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 type '\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.

11.5 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

11.6 References to start and end lines

11.6.1 Reference to main text lines

Many times, you may want to make a cross-reference to a passage that is defined by a start line and an end line. reledmac provides specific tools for this scenario.

\edlabelS   Use \edlabelS{⟨*label*⟩} to mark the start line of the passage.

\edlabelE   Use \edlabelE{⟨*label*⟩} to mark the end the end line of the passage. These two commands just create to label which are named ⟨*label*⟩:start and ⟨*label*⟩:end.

\edlabelSE   Use \edlabelSE{⟨*label*⟩} to mark just one location in the text. Contrary to a classical \edlabel, the ⟨*label*⟩ could be use with \SEref and \SErefwithpage.

\SEref   The main utility is to use them with three other commands. \SEref{⟨*label*⟩} will make a cross-reference printed as a reference in critical footnotes.

\SErefwithpage   \SErefwithpage will make a cross-reference printed as a reference in critical endnotes.

\SErefonlypage   \SErefonlypage will make a cross-reference printed only with page number.

11.6.2   References to lines that are commented on 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 (6.2.5 p. 26) 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.

11.6.3   Settings

\setapprefprefixsingle   **Specific to these tools**   If you use \apprefprefixsingle{⟨*prefix*⟩}, ⟨*prefix*⟩ will be
\setapprefprefixmore   printed before the line numbers of a \appref-reference. If you use \apprefprefixmore{⟨*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.

\setSErefprefixsingle   \setSErefprefixsingle and \setSErefprefixmore are similar for \SEref
setSErefprefixmore   command.
\setSErefonlypageprefixsingle   Use \setSErefonlypageprefixsingle{⟨*prefix*⟩} to set the page prefix for \SErefonlypage
\setSErefonlypageprefixmore   when there is only one page. Use \setSErefonlypageprefixmore{⟨*prefix*⟩} to set it
when there is more than one page. For example:

```
\setSErefonlypageprefixsingle{p.~}
\setSErefonlypageprefixmore{pp.~}
```

Note that if you do not use \setSErefonlypageprefixmore, the value of \setSErefonlypageprefixsi
is used instead.

   **Also note that \setSErefonlypageprefixsingle is only a shortcut for \Xendbeforepagenumber
(see 11.6.3 p. 52). So if you use \Xendbeforepagenumber without any optional ar-
gument, it will override this setting.**

Linked to setting of critical endnotes and footnotes   Some commands who set the
appearance of line numbers in critical footnotes also set the appearance of line numbers
in \appref and \SEref if you call them *without the optional series argument*.

   These commandes are the following:

   • \Xlineflag (for reledpar), enabled by default.

- `\Xlinerangeseparator`

- `\Xmorethantwolines`

- `\Xsublinesep`

- `\Xtwolines`

- `\Xtwolinesbutnotmore`

- `\Xtwolinesonlyinsamepage`

If you want to make settings specific to `\appref` or `\SEref`, just call them with an optional argument containing a comma-separated list of command names (for example `appref,SEref`) or with a suffix equal to the command name (for example `appref`).

The same principle is available for `\apprefwithpage`, `\SErefwithpage` and `\SErefonlypage` with the following commands:

- `\Xendafterpagenumber` (not for `\SErefonlypage`)

- `\Xendbeforepagenumber`

- `\Xendlineflag` (for `reledpar`), enabled by default.

- `\Xendlineprefixmore`

- `\Xendlineprefixsingle`

- `\Xendlinerangeseparator`

- `\Xendmorethantwolines`

- `\Xendsublinesep`

- `\Xendtwolines`

- `\Xendtwolinesbutnotmore`

- `\Xendtwolinesonlyinsamepage`

For one specific command When calling `\appref` and `\SEref`, you can use as a first optional argument, in brackets (`[]`), any optional argument which can be used for critical footnotes (6.2.2 p. 24).

When calling `\apprefwithpage`, `\SErefwithpage` or `\SErefonlypage` you can use as a first optional argument, in brackets (`[]`), any optional argument which can be used for critical endnotes (6.2.3 p. 24).

11.7   Compatibility with `xr` package

The `\externaldocument` command of the `\xr` package allows making cross-references from an external document, with the standard LaTeX commands `\label` and `\ref` (and related).

To use it with the `reledmac` cross-reference commands (i.e. `\edlabel` and related), you must do the following:

1. Load the `xr` package.

2. Load the `reledmac` package.

3. Use the `\externaldocument` document command.

12   Side notes

12.1   Basics

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 `\ledsidenote`s in the right hand margin. This is the opposite of 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 note commands for the same side are called in the same line, they will be appended and separated by a comma.

12.2   Setting

12.2.1   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`.

12.2.2   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).

12.2.3 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.

12.2.4 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*⟩}.

13 Indexing

13.1 Basics

\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 reledmac. That means you must first run
(Xe/Lua)LaTeX three times, then run makeindex, and then finally run (Xe/Lua)LaTeX again,
in order to 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 reledmac.

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

13.2 Referring to critical notes

If you want to refer to a word inside an \edtext{⟨*lemma*⟩}{⟨*app*⟩} command, \edindex
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 \edindex inside some \Xfootnote command, it will refer to that note, and a suffix *n* will be appended to the reference. You can redefine this suffix by redefining the command \ledinnotemark. Its actual definition is:

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

13.3   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[20].

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

Read the MAKEINDEX program's handbook about the .ist file.

13.4   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.[21]

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:

---

[20]For further detail, you can read http://tex.stackexchange.com/a/32783/7712.
[21]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.[22].

13.5   Advanced setting

`\edindexlab`   The `\edindex` process uses a `\label` and `\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.

14   Glossary

`reledmac` provides mechanism to make glossaries with the `glossaries` package, refering not to the page, but to the page and line.

14.1   Prealable setting

The standard compositor between page and line number in `reledmac` is a dash, while `glossaries` use, in standard, a dot. Consequently, you must:

- Or set `.` `glossaries`:
  `\glsSetCompositor{-}`

- Or set `reledmac`:
  `\renewcommand{\pagelinesep}{-}`
  In this case, the above will have consequences for your use of `\edindex` and you should set your `.ist` file (13.3 p. 56).

14.2   Commands

The `\gls`, `\Gls`, and related commands of `glossaries` packages have a prefixed version with `ed`, which refers to the page line. The argument are the same as for the standard commands. So for example:
`\edgls[⟨options⟩]{⟨label⟩}[⟨insert⟩]`

---

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

15   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.

<div style="float:left">edarrayl
edarrayc
edarrayr
edtabularl
edtabularc
edtabularr</div>

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

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

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.

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`
`\spreadtext`
`\spreadmath{`⟨*math*⟩`}` typesets `{`⟨*math*⟩`}` but the `{`⟨*math*⟩`}` has no effect on the 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}
```

$$
\begin{array}{llll}
1 & 2 & 3 & 4 \\
& F+G+C & & \\
a & bb & ccc & dddd
\end{array}
$$

`\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}
```

$$
\begin{array}{llllr}
1 & 2 & 3 & 4 & 5 \\
Q & & \text{fd} & \text{h} & \text{qwertziohg} \\
\text{v} & \text{wptz} & \text{x} & \text{y} & \text{vb} \\
\text{g} & \text{nnn} & \multicolumn{3}{c}{\overbrace{\qquad\qquad\qquad}} \\
\multicolumn{3}{c}{\underbrace{\qquad\qquad}} & \text{pq} & \text{dgh} \\
\text{k} & & \text{l} & \text{co} & \text{ghweropjklmnbvcxys} \\
1 & 2 & 3 & \multicolumn{2}{c}{\underline{\qquad\qquad}} &
\end{array}
$$

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}
```

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

\edatleft[⟨*math*⟩]{⟨*symbol*⟩}{⟨*halfheight*⟩} typesets the math ⟨*symbol*⟩ as \left{⟨*symbol*⟩}
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   \edbeforetab{⟨*text*⟩}{⟨*entry*⟩}, where ⟨*entry*⟩ is an entry in the leftmost column,
\edaftertab   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}
```

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

\edvertline   The macro \edvertline{⟨*height*⟩} draws a vertical line ⟨*height*⟩ high (contrast this
\edvertdots   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.

16   Sectioning commands

16.1   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` (5.2.3 p. 18):

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

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

16.2   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*`.

These 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
```

After the first run, you will see only the text. This is normal. After the second run, you will see the formatting. Finally, with the third run, you will see the table of contents.

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

16.3   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 files, save memory, and make reledmac run faster.

17   Quotation environments

The quotation and quote environments can be used so that the 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 \pstart … \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.

18   Page breaks

18.1   Control page breaking

reledmac and reledpar break pages automatically. However, you may sometimes
\ledpb   want to either force page breaks, or prevent them. The packages provide two macros:
\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, then l. 443 will be at the p. n, and the l. 444 \ledpbsetting at the p. $n + 1$. However, you can change the behavior and decide they will have effect after the end of the line, adding \ledpbsetting{after} at the beginning of your file (better: in your preamble). With the previous example, l. 444 will be on p. n and l. 445 will be on 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 be out of sync.

18.2 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 and no more. It works on the third run, or on the fourth run if using reledpar. By default, when a long verse runs 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.

19 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
```

19.1   Known and suspected limitations

19.1.1   Non-standard geometry

If you use classes other than `article` or `book`, or if you use the geometry package, you should use maxhnotesX and/or \Xmaxhnotes as explained in 7.13.5 p. 43 in order to prevent footnotes from overlapping the bottom margin.

19.1.2   `floatrow` package compatibility

The `floatrow` package must be loaded before the `reledmac`.

19.1.3   'No room for a new'

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

In order to prevent such problems, the first thing is to use the options to optimize `reledmac`. For example, if you need only two series of notes, use the `series={A,B}` option. Read 16.3 p. 62 in order to know which are the available options.

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

'no room for a new count' is often caused by `biblatex` being used at the same time. Load `reledmac` (and `reledpar`) *before* `biblatex`.

'no room for a new write' can be caused by 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` package. 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.

19.1.4   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.

19.1.5 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, insert
\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.

19.1.6 Paragraphed footnotes

The restriction on explicit line-breaking in paragraphed footnotes, mentioned on 7.1 p. 33, and described in more detail on XII.6.3 p. 162, 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.

If you use more than one series of paragraphed notes, it may happen, in some particular cases, that only the footnote rule, with no accompanying footnotes, be printed. In this case use reledmac package option nopenalties which should solve the problem, but also may produce widow or orphan lines. For the time being, we have no solution of this problem.

\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}
Note that you must call it *before* \Xarrangement{paragraph} or \arrangementX{paragraph}.

Any settings to 'geometry' must be made before \Xarrangement / \arrangementX.

Finally, in many cases you should use \Xmaxhnotes and / or \maxhnotesX (7.13.5 p. 43), in order to define the maximum height relative to \textheight and not to \vsize, because the \vsize value is not the same inside and outside of the preamble.

19.1.7 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[23] 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 throws away the first one.) The first incantation lets color show in both the main text and footnotes whereas the second one shows color in the main text but kills it in the lemma and footnotes. On the other hand if you use `\textcolor` instead, like

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

there is no need to fiddle with `\morenoexpands` as the color will naturally be displayed in both the text and footnotes. To kill the color in the lemma and footnotes, though, you can do:

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

It took 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.

19.1.8   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

---

[23]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 XXII) 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}[2016/06/18 v2.13.1 typeset critical editions]%
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  \newif\ifledfinal
36  \DeclareOptionX{final}{\ledfinaltrue}
37  \DeclareOptionX{draft}{\ledfinalfalse}
38  \ExecuteOptionsX{final}
```

```
39
40  \newif\ifparapparatus@
41  \DeclareOptionX{parapparatus}{\parapparatus@true}

42
43  \newif\iflednopbinverse
44  \DeclareOptionX{nopbinverse}{\lednopbinversetrue}

45
46  \newif\ifwidthliketwocolumns%
47  \DeclareOptionX{widthliketwocolumns}{\widthliketwocolumnstrue}%

48
49  \newif\ifcontinuousnumberingwithcolumns
50  \DeclareOptionX{continuousnumberingwithcolumns}{\
    continuousnumberingwithcolumnstrue}%

51
52  \newif\ifxindy@
53  \DeclareOptionX{xindy}[eledmac-markup-attr.xdy]{%
54    \AtBeginDocument{\immediate\openout\eledmac@xindy@out=#1}%
55    \newwrite\eledmac@xindy@out%
56    \xindy@true%
57    \gdef\eledmacmarkuplocrefdepth{:depth 1}%
58    \AtEndDocument{\immediate\closeout\eledmac@xindy@out}%
59  }%

60
61  \newif\ifxindyhyperref@
62  \DeclareOptionX{xindy+hyperref}{%
63    \xindyhyperref@true%
64  }%

65
66  \newif\ifeledmaccompat@%
67  \DeclareOptionX{eledmac-compat}{%
68    \eledmaccompat@true%
69  }%
70  \DeclareOptionX{nopenalties}{%
71    \AtBeginDocument{\let\add@penalties\relax}%
72  }
73  \def\l@auxdir{}%
74  \DeclareOptionX{auxdir}{%
75    \xdef\l@auxdir{#1/}%
76  }%
77  %
```

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.

```
78  \ProcessOptionsX*\relax

79
80  %
```

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 XaTeX is running, and ragged2e to manage ragged justification for paragraphed notes.

```
81 \RequirePackage{xargs}
82 \RequirePackage{etoolbox}
83 \@ifl@t@r\fmtversion{2015/10/01}
84  {}%
85  {\RequirePackage{etex}%
86  \csname reserveinserts\endcsname{32}%
87 }
88 \RequirePackage{suffix}
89 \RequirePackage{xstring}
90 \RequirePackage{ifluatex}
91 \RequirePackage{ragged2e}
92 \RequirePackage{ifxetex}%
93 %
```

II.5 Compatibility with LuaTeX

Here, we enable some primitives for LuaTeX.

```
94 \ifx\directlua\undefined\else%
95   \directlua{tex.enableprimitives("",{"textdir","pardir","bodydir"})}
96 \fi
97 %
```

II.6 Boolean flags

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

```
98 \newif\ifl@dmemoir
99 \@ifclassloaded{memoir}{\l@dmemoirtrue}{\l@dmemoirfalse}
100
101 %
```

\if@ledgroup   Flag set to true inside a ledgroup environment.

```
102 \newif\if@ledgroup%
103 %
```

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

```
104 \newif\ifl@imakeidx
105 \@ifpackageloaded{imakeidx}{\l@imakeidxtrue}{}%False is the default value
106 %
```

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

```
107 \newif\ifl@indextools%
108 \@ifpackageloaded{indextools}{%
109   \l@indextoolstrue%
110   \l@imakeidxtrue%
111   \let\imki@wrindexentry\indtl@wrindexentry%
112   }{}%
113 %
```

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.

```
114 \ifdef{\if@RTL}{}{\newif\if@RTL}
115 %
```

\if@firstlineofpage   `\if@firstlineofpage` is set to TRUE at the first line of every page. `\if@firstlineofpageR` is for the right side.

```
116 \newif\if@firstlineofpage%
117 \newif\if@firstlineofpageR%
118 %
```

II.7   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.

```
119 \newcommand{\reledmac@warning}[1]{\PackageWarning{reledmac}{#1}}
120 %
```

\reledmac@error   Write an error message.

```
121 \newcommand{\reledmac@error}[2]{\PackageError{reledmac}{#1}{#2}}
122 %
```

\led@err@NumberingStarted
\led@err@NumberingNotStarted
\led@err@NumberingShouldHaveStarted
```
123 \newcommand*{\led@err@NumberingStarted}{%
124   \reledmac@error{Numbering has already been started}{\@ehc}}
125 \newcommand*{\led@err@NumberingNotStarted}{%
126   \reledmac@error{Numbering was not started}{\@ehc}}
127 \newcommand*{\led@err@NumberingShouldHaveStarted}{%
128   \reledmac@error{Numbering should already have been started}{\@ehc}}
129 %
```

`\led@err@edtextoutsidepstart`₃₀ `\newcommand*{\led@err@edtextoutsidepstart}{%`

131   `\reledmac@error{\string\edtext\space outside numbered paragraph (\…pstart`
`\pend)}{\@ehc}}%`

132 `%`

`\led@mess@NotesChanged`₃₃ `\newcommand*{\led@mess@NotesChanged}{%`

134   `\typeout{reledmac reminder: }%`

135   `\typeout{ The number of the footnotes in this section`

136   `has changed since the last run.}%`

137   `\typeout{ You will need to run LaTeX two more times`

138   `before the footnote placement}%`

139   `\typeout{ and line numbering in this section are`

140   `correct.}}`

141 `%`

`\led@mess@SectionContinued`₄₂ `\newcommand*{\led@mess@SectionContinued}[1]{%`

143   `\message{Section #1 (continuing the previous section)}}`

144 `%`

`\led@err@LineationInNumbered`₄₅ `\newcommand*{\led@err@LineationInNumbered}{%`

146   `\reledmac@error{You can't use \string\lineation\space within`

147   `a numbered section}{\@ehc}}`

148 `%`

`\led@warn@BadLineation`₄₉ `\newcommand*{\led@warn@BadLineation}{%`

`\led@warn@BadLinenummargin`₅₀   `\reledmac@warning{Bad \string\lineation\space argument}}`

`\led@warn@BadLockdisp`₅₁ `\newcommand*{\led@warn@BadLinenummargin}{%`

`\led@warn@BadSublockdisp`₅₂   `\reledmac@warning{Bad \string\linenummargin\space argument}}`

153 `\newcommand*{\led@warn@BadLockdisp}{%`

154   `\reledmac@warning{Bad \string\lockdisp\space argument}}`

155 `\newcommand*{\led@warn@BadSublockdisp}{%`

156   `\reledmac@warning{Bad \string\sublockdisp\space argument}}`

157 `%`

`\led@warn@NoLineFile`₅₈ `\newcommand*{\led@warn@NoLineFile}[1]{%`

159   `\reledmac@warning{Can't find line-list file #1}}`

160 `%`

`\led@warn@LineFileObsolete`₆₁ `\newcommand*{\led@warn@Obsolete}[1]{%`

162   `\reledmac@warning{Line-list file #1 was obsolete. We have not read it.`
`Please run LaTeX again.}}`

163 `%`

arn@BadAdvancelineSubline₁₆₄ `\newcommand*{\led@warn@BadAdvancelineSubline}{%`
d@warn@BadAdvancelineLine₁₆₅ `  \reledmac@warning{\string\advanceline\space produced a sub-line`
₁₆₆ `                          number less than zero.}}`
₁₆₇ `\newcommand*{\led@warn@BadAdvancelineLine}{%`
₁₆₈ `  \reledmac@warning{\string\advanceline\space produced a line`
₁₆₉ `                          number less than zero.}}`
₁₇₀ `%`

\led@warn@BadSetline₁₇₁ `\newcommand*{\led@warn@BadSetline}{%`
\led@warn@BadSetlinenum₁₇₂ `  \reledmac@warning{Bad \string\setline\space argument}}`
₁₇₃ `\newcommand*{\led@warn@BadSetlinenum}{%`
₁₇₄ `  \reledmac@warning{Bad \string\setlinenum\space argument}}`
₁₇₅ `%`

led@err@PstartNotNumbered₁₇₆ `\newcommand*{\led@err@PstartNotNumbered}{%`
\led@err@PstartInPstart₁₇₇ `  \reledmac@error{\string\pstart\space must be used within a`
\led@err@PendNotNumbered₁₇₈ `                    numbered section %`
\led@err@PendNoPstart₁₇₉ `                      (\string\…beginnumbering\string\endnumbering)}{\@ehc}}%`
ed@err@AutoparNotNumbered₁₈₀ `\newcommand*{\led@err@PstartInPstart}{%`
rr@NumberingWithoutPstart₁₈₁ `  \reledmac@error{\string\pstart\space encountered while another`
₁₈₂ `                    \string\pstart\space was in effect}{\@ehc}}`
₁₈₃ `\newcommand*{\led@err@PendNotNumbered}{%`
₁₈₄ `  \reledmac@error{\string\pend\space must be used within a`
₁₈₅ `                    numbered section}{\@ehc}}`
₁₈₆ `\newcommand*{\led@err@PendNoPstart}{%`
₁₈₇ `  \reledmac@error{\string\pend\space must follow a \string\pstart}{\@ehc}}`
₁₈₈ `\newcommand*{\led@err@AutoparNotNumbered}{%`
₁₈₉ `  \reledmac@error{\string\autopar\space must be used within a`
₁₉₀ `                    numbered section}{\@ehc}}`
₁₉₁ `\newcommand*{\led@err@NumberingWithoutPstart}{%`
₁₉₂ `  \reledmac@error{\string\beginnumbering...\string\endnumbering\space`
`without \string\pstart}{\@ehc}}%`
₁₉₃ `%`

\led@warn@BadAction₁₉₄ `\newcommand*{\led@warn@BadAction}{%`
₁₉₅ `  \reledmac@warning{Bad action code, value \next@action.}}`
₁₉₆ `%`

\led@warn@DuplicateLabel₁₉₇ `\newcommand*{\led@warn@DuplicateLabel}[1]{%`
ppLabelOutSecondArgEdtext₁₉₈ `  \reledmac@warning{Duplicate definition of label `#1'\@gobble}%`
\led@warn@RefUndefined₁₉₉ `  \@latex@warning@no@line{Label `#1' multiply defined}%`
\led@warn@RefUndefined₂₀₀ `}%`
₂₀₁ `\newcommand*{\led@warn@AppLabelOutSecondArgEdtext}[1]{%`
₂₀₂ `  \reledmac@warning{\string\applabel\space outside of the second argument`
`of an \string\edtext\space `#1' on page \the\pageno.}}%`

```
203 \newcommand*{\led@warn@RefUndefined}[1]{%
204   \G@refundefinedtrue%
205   \reledmac@warning{Reference `#1' on page \the\pageno\space undefined.%
206                     Using `000'.}%
207   \@latex@warning{Reference `#1' undefined\on@line}%
208 }%
209 \newcommand*{\led@warn@pairRefUndefined}[1]{%
210   \G@refundefinedtrue%
211   \reledmac@warning{Reference `#1:start' and/or `#1:end' on page \the\
pageno\space undefined.
212                     Using `??'.}%
213   \@latex@warning{Reference `#1:start' and/or `#1:end' undefined\on@line}%
214 }
215 %
```

```
\led@warn@NoMarginpars 216 \newcommand*{\led@warn@NoMarginpars}{%
217   \reledmac@warning{You can't use \string\marginpar\space in numbered text
}}
218 %
```

```
\led@warn@BadSidenotemargin 219 \newcommand*{\led@warn@BadSidenotemargin}{%
220   \reledmac@warning{Bad \string\sidenotemmargin\space argument}}
221 %
```

```
\led@warn@NoIndexFile 222 \newcommand*{\led@warn@NoIndexFile}[1]{%
223   \reledmac@warning{Undefined index file #1}}
224 %
```

```
\led@warn@SeriesStillExist 225 \newcommand{\led@warn@SeriesStillExist}[1]{%
226   \reledmac@warning{Series #1 is still existing !}%
227 }%
228 %
```

```
\led@err@ManySidenotes 229 \newcommand{\led@err@ManySidenotes}{%
\led@err@ManyLeftnotes 230   \ifledRcol@%
\led@err@ManyRightnotes 231      \reledmac@warning{\itemcount@\space sidenotes on line \the\line@numR\
space p. \the\page@numR}%
232   \else%
233      \reledmac@warning{\itemcount@\space sidenotes on line \the\line@num\
space p. \the\page@num}%
234   \fi%
235 }%
236 \newcommand{\led@err@ManyLeftnotes}{%
237   \ifledRcol@%
```

```
238     \reledmac@warning{\itemcount@\space leftnotes on line \the\line@numR\
   space p. \the\page@numR}%
239   \else%
240     \reledmac@warning{\itemcount@\space leftnotes on line \the\line@num\
   space p. \the\page@num}%
241   \fi%
242 }%
243 \newcommand{\led@err@ManyRightnotes}{%
244   \ifledRcol@%
245     \reledmac@warning{\itemcount@\space rightnotes on line \the\line@numR\
   space p. \the\page@numR}%
246   \else%
247     \reledmac@warning{\itemcount@\space rightnotes on line \the\line@num\
   space p. \the\page@num}%
248   \fi%
249 }%
250 %
```

```
\led@err@TooManyColumns  251 \newcommand*{\led@err@TooManyColumns}{%
\led@err@UnequalColumns  252   \reledmac@error{Too many columns}{\@ehc}}
\led@err@LowStartColumn  253 \newcommand*{\led@err@UnequalColumns}{%
 \led@err@HighEndColumn  254   \reledmac@error{Number of columns is not equal to the number
\led@err@ReverseColumns  255                   in the previous row (or \protect\\ \space forgotten?)}{\
   @ehc}}
                         256 \newcommand*{\led@err@LowStartColumn}{%
                         257   \reledmac@error{Start column is too low}{\@ehc}}
                         258 \newcommand*{\led@err@HighEndColumn}{%
                         259   \reledmac@error{End column is too high}{\@ehc}}
                         260 \newcommand*{\led@err@ReverseColumns}{%
                         261   \reledmac@error{Start column is greater than end column}{\@ehc}}
                         262 %
```

```
endnotes@outsidenumbering  263 \newcommand{\led@err@toendnotes@outsidenumbering}{%
                           264   \reledmac@error{\string\toendnotes\space and related commands must be
                              called inside a numbered texte (\string\…beginnumbering\string\endnumbering
                              )}{\@ehc}%
                           265 }%
                           266 %
```

```
err@EdtextWithoutFootnote  267 \newcommand{\led@err@EdtextWithoutFootnote}{%
                           268   \reledmac@error{edtext without Xfootnote. Check syntaxis}{\@ehc}%
                           269 }%
                           270 %
```

```
\led@err@FootnoteNotInSecondArgEdtext₂₇₁ \newcommand{\led@err@FootnoteNotInSecondArgEdtext}[1]{%
                                     272   \reledmac@error{#1footnote outside of the second argument of an edtext.
                                         Check syntax}{\@ehc}%
                                     273 }%
                                     274 %


              \led@error@ImakeidxAfterEledmac₂₇₅ \newcommand{\led@error@ImakeidxAfterEledmac}{%
                                     276   \reledmac@error{Imakeidx must be loaded before reledmac.}{\@ehc}%
                                     277 }%
                                     278 %


           \led@error@IndextoolsAfterEledmac₂₇₉ \newcommand{\led@error@IndextoolsAfterEledmac}{%
                                     280   \reledmac@error{Indextools must be loaded before reledmac.}{\@ehc}%
                                     281 }%
                                     282 %


              \led@error@fail@patch@@makecol₂₈₃ \newcommand{\led@error@fail@patch@@makecol}{%
                                     284   \reledmac@error{Fail to patch \string\@makecol\space command.}{\@ehc}%
                                     285 }%
                                     286 %


           \led@error@fail@patch@@reinserts₂₈₇ \newcommand{\led@error@fail@patch@@reinserts}{%
                                     288   \reledmac@error{Fail to patch \string\@reinserts\space command.}{\@ehc}%
                                     289 }%
                                     290 %


        \led@error@fail@patch@@doclearpage₂₉₁ \newcommand{\led@error@fail@patch@@doclearpage}{%
                                     292   \reledmac@error{Fail to patch \string\@doclearpage\space command.}{\@ehc}
                                         %
                                     293 }%
                                     294 %


          \led@error@fail@patch@@iiiminipage₂₉₅ \newcommand{\led@error@fail@patch@@iiiminipage}{%
                                     296   \reledmac@error{Fail to patch \string\@iiiminipage\space command.}{\@ehc}
                                         %
                                     297 }%
                                     298 %


        \led@error@fail@patch@endminipage₂₉₉ \newcommand{\led@error@fail@patch@endminipage}{%
                                     300   \reledmac@error{Fail to patch \string\endminipage\space command.}{\@ehc}%
                                     301 }%
                                     302 %
```

```
warning@hsizeX@deprecated 303 \newcommand{\led@warning@hsizeX@deprecated}{%
                          304   \reledmac@warning{\string\hsizeX\space command deprecated, use \string\
                              widthX\space instead.}%
                          305 }%
                          306 %
```

```
warning@Xhsize@deprecated 307 \newcommand{\led@warning@Xhsize@deprecated}{%
                          308   \reledmac@warning{\string\Xhsize\space command deprecated, use \string\
                              Xwidth\space instead.}%
                          309 }%
                          310 %
```

```
warning@msdatawithoutstop 311 \newcommand{\led@warning@msdatawithoutstop}{%
                          312   \reledmac@warning{\string\msdata\space without corresponding \string\
                              stopmsdata}%
                          313 }%
                          314 %
```

```
ning@preXnotes@deprecated 315 \newcommand{\led@warning@preXnotes@deprecated}{%
                          316   \reledmac@warning@preXnotes@deprecated%
                          317 }%
                          318 %
```

II.8   Gobbling

Here, we define some commands which gobble their arguments.

```
\@gobblethree 319 \providecommand*{\@gobblethree}[3]{}
 \@gobblefour 320 \providecommand*{\@gobblefour}[4]{}
 \@gobblefive 321 \providecommand*{\@gobblefive}[5]{}
             322 %
```

II.9   Miscellaneous commands

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

```
323 \ifledfinal
324   \newcommand*\showlemma[1]{#1}
325 \else
326   \newcommand*\showlemma[1]{\underline{#1}}
327 \fi
328
329 %
```

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

    Initialize it as \empty.

330 \let\linenumberlist=\empty

331

332 %

\@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).

333 \newcount\@l@dtempcnta \newcount\@l@dtempcntb

334 %

II.10   Prepare `reledpar`

\ifl@dpairing In preparation for the reledpar package, these are related to the 'right' text of paral-
\ifl@dpaging lel texts (when \ifl@dpairing is TRUE). They are explained in the eledpar manual.
\ifl@dprintingpages
\ifl@dprintingcolumns
\ifpst@rtedL   335 \newif\ifl@dpairing
\l@dnumpstartsL 336 \newif\ifl@dpaging%
337 \newif\ifl@dprintingpages%
338 \newif\ifl@dprintingcolumns%
339 \newif\ifpst@rtedL
340 \newcount\l@dnumpstartsL
341 %

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

342 \newif\ifledRcol

343 \newif\ifledRcol@

344 %

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

345 \newif\ifnumberingR

346 %

The \ifXnote@ macro is set to true when we are typesetting a critical footnote.

347 \newif\ifXnote@%

348 %

II.11 Booleans provided by other optional packages which are required in any case

`\ifindtl@innote`
`\ifindtl@notenumber`

The `\ifindtl@innote` and `\ifindtl@notenumber` are required even if `indextools` is not used.

```
349 \providebool{indtl@innote}%
350 \providebool{indtl@notenumber}%
351 %
```

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.

```
352 \newcount\section@num
353 \section@num=0
354 \let\extensionchars=\empty
355 %
```

`\ifnumbering`
`\numberingtrue`
`\numberingfalse`

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

```
356 \newif\ifnumbering
357 %
```

`\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.

```
358 \newcommand*{\beginnumbering}{%
359   \ifnumbering
360     \led@err@NumberingStarted
361     \endnumbering
362   \fi
363   \global\numberingtrue
364   \global\advance\section@num \@ne
365   \initnumbering@reg
366   \message{Section \the\section@num }%
367   \line@list@stuff{\jobname.\extensionchars\the\section@num}%
368   \l@dend@stuff
369   \setcounter{pstart}{1}
370   \ifl@dpairing
371     \global\l@dnumpstartsL \z@
372     \global\pst@rtedLfalse
373 %
```

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.

```
374   \else
375     \begingroup
376     \global\@afterindenttrue%In order to retablish normal feature if the \
    begingroup was not here
377     \initnumbering@quote
378     \ifwidthliketwocolumns%
379       \csuse{setwidthliketwocolumns@\columns@position}%
380       \csuse{setpositionliketwocolumns@\columns@position}%
381     \fi%
382   \fi
383   \gdef\eled@sections@@{}%
384   \if@noeled@sec\else%
385     \makeatletter\InputIfFileExists{\l@auxdir\jobname.eledsec\the\
    section@num}{}{}\makeatother%
386     \immediate\openout\eled@sectioning@out=\l@auxdir\jobname.eledsec\the\
    section@num\relax%
```

```
387    \fi%
388 }
389 \newcommand*{\initnumbering@reg}{%
390    \global\pst@rtedLfalse
391    \global\l@dnumpstartsL \z@
392    \global\absline@num \z@
393    \gdef\normal@page@break{}
394    \gdef\l@prev@pb{}
395    \gdef\l@prev@nopb{}
396    \global\line@num \z@
397    \global\subline@num \z@
398    \global\@lock \z@
399    \global\sub@lock \z@
400    \global\sublines@false
401    \global\let\next@page@num=\relax
402    \global\let\sub@change=\relax
403    \resetprevline@
404    \resetprevpage@num
405    \global\stopmsdata@inserted@true%
406    }
407
408 %
```

\endnumbering   \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.

```
409 \def\endnumbering{%
410    \ifnumbering
411      \global\numberingfalse
412      \normal@pars
413      \ifnum\l@dnumpstartsL=0%
414        \led@err@NumberingWithoutPstart%
415      \fi%
416      \ifl@dpairing
417        \global\pst@rtedLfalse
418      \else
419        \ifx\insertlines@list\empty\else
420          \global\noteschanged@true
421        \fi
422        \ifx\line@list\empty\else
423          \global\noteschanged@true
424        \fi
425      \fi
426      \ifnoteschanged@
427        \led@mess@NotesChanged
428      \fi
429    \else
430      \led@err@NumberingNotStarted
431    \fi
```

```
432    \autoparfalse
433    \if@noeled@sec\else%
434      \immediate\closeout\eled@sectioning@out%
435    \fi%
436    \ifl@dpairing\else
437      \global\l@dnumpstartsL=\z@%
438      \endgroup
439    \fi
440 }
441 %
```

`\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.[24]

```
442 \newcommand{\pausenumbering}{%
443    \ifautopar\global\autopar@pausetrue\fi%
444    \endnumbering\global\numberingtrue}
445 %
```

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.

```
446 \newcommand*{\resumenumbering}{%
447    \ifnumbering
448      \ifautopar@pause\autopar\fi
449      \global\pst@rtedLtrue
450      \global\advance\section@num \@ne
451      \led@mess@SectionContinued{\the\section@num}%
452      \line@list@stuff{\jobname.\extensionchars\the\section@num}%
453      \l@dend@stuff
454      \ifl@dpairing\else%
455        \begingroup%
456        \initnumbering@quote%
457        \ifwidthliketwocolumns%
458          \csuse{setwidthliketwocolumns@\columns@position}%
459          \csuse{setpositionliketwocolumns@\columns@position}%
460        \fi%
461      \fi%
462      \ifcontinuousnumberingwithcolumns%
463        \ifdefined\line@numR%
464          \ifnum\line@numR>\line@num%
465            \expandafter\setlinenum\expandafter{\the\line@numR}%
466          \fi%
467          \ifnum\last@page@numR>\last@page@num%
468            \global\last@page@num=\last@page@numR%
469          \fi%
470        \fi%
```

---

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

```
471    \fi%
472  \else
473    \led@err@NumberingShouldHaveStarted
474    \endnumbering
475    \beginnumbering
476  \fi}
477
478
479 %
```

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.

```
480 \newcommand*{\list@create}[1]{%
481   \global\let#1=\empty%
482 }%
483 %
```

\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 .

```
484 \newcommand*{\list@clear}[1]{%
485   \global\let#1=\empty%
486 }
487 %
```

\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.

```
488  \newtoks\led@toksa \newtoks\led@toksb
489  \global\led@toksa={\\}
490  \long\def\xright@appenditem#1\to#2{%
491    \global\led@toksb=\expandafter{#2}%
492    \xdef#2{\the\led@toksb\the\led@toksa\expandafter{#1}}%
493    \global\led@toksb={}}
494  %
```

\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.

```
495  \long\def\xleft@appenditem#1\to#2{%
496    \global\led@toksb=\expandafter{#2}%
497    \xdef#2{\the\led@toksa\expandafter{#1}\the\led@toksb}%
498    \global\led@toksb={}}
499  %
```

\gl@p   The \gl@p macro removes the leftmost item from a list and places it in a control se-
quence. You type \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.

```
500  \def\gl@p#1\to#2{\expandafter\gl@poff#1\gl@poff#1#2}
501  \long\def\gl@poff\\#1#2\gl@poff#3#4{\gdef#4{#1}\gdef#3{#2}}
502
503  %
```

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   • line-of-page: bypstart@ = false and bypage@ = true.
\ifbypage@
\bypage@true   • line-of-pstart: bypstart@ = true and bypage@ = false.
\bypage@false   reledmac will use the line-of-section system unless instructed otherwise.

```
504  \newif\ifbypage@
505  \newif\ifbypstart@
506  %
```

   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`₀₇ `\newif\ifbypage@R`
`\ifbypstart@R`₀₈ `\newif\ifbypstart@R`
₅₀₉ `%`

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

₅₁₀ `\newcommand*{\lineation}[1]{{`
₅₁₁ `%`

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

₅₁₂   `\ifnumbering`
₅₁₃   `\led@err@LineationInNumbered`
₅₁₄   `\else`
₅₁₅ `%`

If the argument is `page`.

₅₁₆   `\def\@tempa{#1}\def\@tempb{page}%`
₅₁₇   `\ifx\@tempa\@tempb`
₅₁₈   `\global\bypage@true`
₅₁₉   `\global\bypstart@false`
₅₂₀   `\unless\ifnocritical@%`
₅₂₁   `\Xpstart[][false]%`
₅₂₂   `\fi%`
₅₂₃ `%`

If the argument is `pstart`.

₅₂₄   `\else`
₅₂₅   `\def\@tempb{pstart}%`
₅₂₆   `\ifx\@tempa\@tempb`
₅₂₇   `\global\bypage@false`
₅₂₈   `\global\bypstart@true`
₅₂₉   `\unless\ifnocritical@%`
₅₃₀   `\Xpstart%`
₅₃₁   `\fi%`
₅₃₂ `%`

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

₅₃₃   `\else`
₅₃₄   `\def\@tempb{section}`
₅₃₅   `\ifx\@tempa\@tempb`
₅₃₆   `\global\bypage@false`
₅₃₇   `\global\bypstart@false`
₅₃₈   `\unless\ifnocritical@%`
₅₃₉   `\Xpstart[][false]%`
₅₄₀   `\fi%`
₅₄₁ `%`

In other case, it is an error.

```
542        \else
543                \led@warn@BadLineation
544          \fi
545        \fi
546      \fi
547    \fi}}
548 %
```

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.

```
549 \newcount\line@margin
550
551 \newcommand*{\linenummargin}[1]{{%
552   \l@dgetline@margin{#1}%
553   \ifnum\@l@dtempcntb>\m@ne
554     \ifledRcol
555       \global\line@marginR=\@l@dtempcntb
556       \led@warn@setting@in@rightside{\linenummargin}%
557     \else
558       \global\line@margin=\@l@dtempcntb
559     \fi
560   \fi}}
561
562 \newcommand*{\l@dgetline@margin}[1]{%
563   \def\@tempa{#1}\def\@tempb{left}%
564   \ifx\@tempa\@tempb
565     \@l@dtempcntb \z@
566   \else
567     \def\@tempb{right}%
568     \ifx\@tempa\@tempb
569       \@l@dtempcntb \@ne
570     \else
571       \def\@tempb{outer}%
572       \ifx\@tempa\@tempb
573         \@l@dtempcntb \tw@
574       \else
575         \def\@tempb{inner}%
576         \ifx\@tempa\@tempb
577           \@l@dtempcntb \thr@@
578         \else
579           \led@warn@BadLinenummargin
580           \@l@dtempcntb \m@ne
581         \fi
582       \fi
```

```
583        \fi
584      \fi}
585
586  %
```

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.

```
587  \newcounter{firstlinenum}
588    \setcounter{firstlinenum}{5}
589  \newcounter{linenumincrement}
590    \setcounter{linenumincrement}{5}
591  %
```

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

```
592  \newcounter{firstsublinenum}
593    \setcounter{firstsublinenum}{5}
594  \newcounter{sublinenumincrement}
595    \setcounter{sublinenumincrement}{5}
596
597  %
```

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

```
598
599  \newcommand*{\firstlinenum}[1]{%
600    \ifledRcol%
601      \setcounter{firstlinenumR}{#1}%
602      \led@warn@setting@in@rightside{\firstlinenum}%
603    \else%
604      \setcounter{firstlinenum}{#1}%
605    \fi%
606  }
607  \newcommand*{\linenumincrement}[1]{%
608    \ifledRcol%
609      \setcounter{linenumincrementR}{#1}%
610      \led@warn@setting@in@rightside{\linenumincrement}%
611    \else%
612      \setcounter{linenumincrement}{#1}%
613    \fi%
614  }
615  \newcommand*{\firstsublinenum}[1]{%
```

```
616  \ifledRcol%
617    \setcounter{firstsublinenumR}{#1}%
618    \led@warn@setting@in@rightside{\firstsublinenum}%
619  \else%
620    \setcounter{firstsublinenum}{#1}%
621  \fi%
622 }
623 \newcommand*{\sublinenumincrement}[1]{%
624  \ifledRcol%
625    \setcounter{sublinenumincrementR}{#1}%
626    \led@warn@setting@in@rightside{\sublinenumincrement}%
627  \else%
628    \setcounter{sublinenumincrement}{#1}%
629  \fi%
630 }
631
632 %
```

V.4   Line number locking

\lockdisp
\lock@disp
\l@dgetlock@disp When line locking is being used, the \lockdisp{⟨*word*⟩} macro specifies whether a line number—if one is due to appear—should be printed on the first printed line or on the last, or by all of them. Its argument is a word, either first, last, or all. Initially, it is set to first.

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

```
633 \newcount\lock@disp
634 \newcommand{\lockdisp}[1]{{%
635  \l@dgetlock@disp{#1}%
636  \ifnum\@l@dtempcntb>\m@ne
637    \global\lock@disp=\@l@dtempcntb
638  \else
639    \led@warn@BadLockdisp
640  \fi}}
641 \newcommand*{\l@dgetlock@disp}[1]{
642  \def\@tempa{#1}\def\@tempb{first}%
643  \ifx\@tempa\@tempb
644    \@l@dtempcntb \z@
645  \else
646    \def\@tempb{last}%
647    \ifx\@tempa\@tempb
648      \@l@dtempcntb \@ne
649    \else
650      \def\@tempb{all}%
651      \ifx\@tempa\@tempb
652        \@l@dtempcntb \tw@
653      \else
654        \@l@dtempcntb \m@ne
655      \fi
```

```
656        \fi
657    \fi}
658
659 %
```

\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.

```
660 \newcount\sublock@disp
661 \newcommand{\sublockdisp}[1]{{%
662   \l@dgetlock@disp{#1}%
663   \ifnum\@l@dtempcntb>\m@ne
664     \global\sublock@disp=\@l@dtempcntb
665   \else
666     \led@warn@BadSublockdisp
667   \fi}}
668
669 %
```

V.5   Line number style

\linenumberstyle   We provide a mechanism for using different representations of the line numbers, not
\linenumrep   just the normal arabic.
\linenumr@p   NOTE: In v0.7 \linenumrep and \sublinenumrep replaced the internal \linenumr@p
\sublinenumberstyle   and \sublinenumr@p.
\sublinenumrep   \linenumberstyle and \sublinenumberstyle are user level macros for setting
\sublinenumr@p   the number representation (\linenumrep and \sublinenumrep) for line and sub-line
                      numbers.

```
670 \newcommand*{\linenumberstyle}[1]{%
671   \def\linenumrep##1{\@nameuse{@#1}{##1}}}
672 \newcommand*{\sublinenumberstyle}[1]{%
673   \def\sublinenumrep##1{\@nameuse{@#1}{##1}}}
674 %
```

Initialise the number styles to arabic.

```
675 \linenumberstyle{arabic}
676   \let\linenumr@p\linenumrep
677 \sublinenumberstyle{arabic}
678   \let\sublinenumr@p\sublinenumrep
679
680 %
```

V.6   Line number printing

\leftlinenum   \leftlinenum and \rightlinenum are the macros that are called to print marginal
\rightlinenum   line numbers on a page, for left- and right-hand margins respectively. They are made
\linenumsep   easy to access and change, since you may want to change the styling in some way. These
\numlabfont
\ledlinenum

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.

```
681  \newlength{\linenumsep}
682    \setlength{\linenumsep}{1pc}
683  \newcommand*{\numlabfont}{\normalfont\scriptsize}
684  \newcommand*{\ledlinenum}{%
685    \bgroup%
686    \ifluatex%
687      \textdir TLT%
688    \fi%
689    \numlabfont\linenumrep{\line@num}%
690    \ifsublines@
691      \ifnum\subline@num>0\relax
692        \unskip%
693        \Xsublinesep@side%
694        \sublinenumrep{\subline@num}%
695      \fi
696    \fi%
697    \egroup%
698  }%
699
700  \newcommand*{\leftlinenum}{%
701    \ledlinenum
702    \kern\linenumsep}
703  \newcommand*{\rightlinenum}{%
704    \kern\linenumsep
705    \ledlinenum}
706
707  %
```

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`.

```
708  \newcount\line@num
709  %
```

`\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.

```
710  \newcount\subline@num
711  %
```

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

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.

```
712  \newif\ifsublines@
713  %
```

`\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.

```
714  \newcount\absline@num
715  %
```

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.

```
716 \newcount\@lock
717 \newcount\sub@lock
718 %
```

V.9   Line number associated to lemma

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

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

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

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

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

- \insertlines@list: the line numbers of lines that have footnotes or other in-
  sertions. These are the absolute numbers where the corresponding lemmas begin.
  This list contains one entry for every footnote in the section; one lemma may con-
  tribute 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:

```
719  \list@create{\line@list}
720  \list@create{\insertlines@list}
721  \list@create{\actionlines@list}
722  \list@create{\actions@list}
723
724  %
```

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

We will need some counts while we read the line-list, for the page number and the ending page, line, and sub-line numbers. Some of these will be used again later on, when we are acting on the data in our list macros.

```
725  \newcount\page@num
726  \newcount\endpage@num
727  \newcount\endline@num
728  \newcount\endsubline@num
729  %
```

\ifnoteschanged@
\noteschanged@true
\noteschanged@false

If the number of the footnotes in a section is different from what it was during the last run, or if this is the very first time you've run LaTeX, on this file, the information from the line-list used to place the notes will be wrong, and some notes will probably be misplaced. When this happens, we prefer to give a single error message for the whole section rather than messages at every point where we notice the problem, because we 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.

```
730  \newif\ifnoteschanged@
731  %
```

\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@  732  \newcommand*{\resetprevline@}{%
733      \def\do##1{\global\csundef{prevline##1}}%
734      \dolistloop{\@series}%
735  }
736  %
```

\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@  737 \newcommand*{\resetprevpage@num}{%
             738     \def\do##1{%
             739       \ifcsdef{prevpage##1@num}{%
             740         \global\csname prevpage##1@num\endcsname=\z@%
             741         \global\csname prevpage##1@numR\endcsname=\z@%
             742         }%
             743         {}%
             744       \ifcsdef{##1prevpage@num}{%
             745         \global\csname ##1prevpage@num\endcsname=\z@%
             746         \global\csname ##1prevpage@numR\endcsname=\z@%
             747         }%
             748         {}%
             749       }%
             750     \dolistloop{\@series}%
             751 }
             752 %
```

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.

```
753 \newread\@inputcheck
754 \newcommand*{\read@linelist}[1]{%
755   \ifledRcol%
756     \list@clearing@regR%
757   \else%
758     \list@clearing@reg%
759   \fi%
760 %
```

    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).

```
761   \list@clear{\maxlinesinpar@list}
762 %
```

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 executed.

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

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

```
763    \get@linelistfile{#1}%
764    \ifcontinuousnumberingwithcolumns
765        \global\page@numR=\page@numR\relax
766        \global\last@page@numR=\last@page@numR\relax
767        \global\page@num=\page@num\relax
768        \global\last@page@num=\last@page@num\relax
769    \fi
770    \@stopmsd%Security if last \endms{} is forgotten
771    \unless\ifledRcol%Get the last line of the last page
772      \csnumgdef{@lastabsline@forpage@\the\page@num}{\the\absline@num}%
773      \csnumgdef{@lastline@forpage@\the\page@num}{\the\line@num}%
774    \else%
775      \csnumgdef{@lastabsline@forpageR@\the\page@numR}{\the\absline@numR}%
776      \csnumgdef{@lastline@forpageR@\the\page@numR}{\the\line@numR}%
777    \fi%
778    \endgroup
779  %
```

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

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

```
780    \ifledRcol
781      \global\page@numR=\m@ne
782      \ifx\actionlines@listR\empty
783        \gdef\next@actionlineR{1000000}%
784      \else
785        \gl@p\actionlines@listR\to\next@actionlineR
786        \gl@p\actions@listR\to\next@actionR
787      \fi
788    \else
789      \global\page@num=\m@ne
790      \ifx\actionlines@list\empty
791        \gdef\next@actionline{1000000}%
792      \else
793        \gl@p\actionlines@list\to\next@actionline
794        \gl@p\actions@list\to\next@action
795      \fi
796    \fi
```

```
797  }
798  %
```

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

```
799  \newcommand*{\list@clearing@reg}{%
800    \list@clear{\line@list}%
801    \list@clear{\insertlines@list}%
802    \list@clear{\actionlines@list}%
803    \list@clear{\actions@list}%
804    \list@clear{\linesinpar@listL}%
805    \list@clear{\linesonpage@listL}%
806    }%
807  %
```

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

```
808  \newcommand*{\get@linelistfile}[1]{%
809    \InputIfFileExists{\l@auxdir#1}{%
810      \global\noteschanged@false
811      \begingroup
812        \catcode`\[=1 \catcode`\]=2
813        \makeatletter \catcode`\^^M=9}{%
814      \led@warn@NoLineFile{\l@auxdir#1}%
815      \global\noteschanged@true
816      \begingroup}%
817  }
818
819  %
```

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 5.2.7 p. 18 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 deliber-
ately do not use \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 com-
mands 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.

```
820 \newcommand{\line@list@version}[1]{%
821   \IfStrEq{#1}{\this@line@list@version}%
822     {}%
823     {\ifledRcol%
824       \led@warn@Obsolete{\jobname.\extensionchars\the\section@num}%
825     \else%
826       \led@warn@Obsolete{\jobname.\extensionchars\the\section@num}%
827     \fi%
828     \endinput%
829     }%
830 }%
831 %
```

\@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.

```
832
833 \newcommand*{\@nl}[2]{%
834   \fix@page{#1}%
835   \ifledRcol%
836     \@nl@regR%
837   \else%
838     \@nl@reg%
839   \fi%
```

```
840  }
841  \newcommand*{\@nl@reg}{%
842    \ifx\l@dchset@num\relax \else
843      \advance\absline@num \@ne
844      \set@line@action
845      \let\l@dchset@num=\relax
846      \advance\absline@num \m@ne
847      \advance\line@num \m@ne
848    \fi
849  %
```

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

```
850    \advance\absline@num \@ne
851          \ifx\next@page@num\relax \else
852              \page@action
853              \let\next@page@num=\relax
854          \fi
855          \ifx\sub@change\relax \else
856              \ifnum\sub@change>\z@
857                  \sublines@true
858              \else
859                  \sublines@false
860              \fi
861              \sub@action
862              \let\sub@change=\relax
863          \fi
864  %
```

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

```
865          \ifcase\@lock
866              \or
867                  \@lock \tw@
868              \or \or
869                  \@lock \z@
870          \fi
871          \ifcase\sub@lock
872              \or
873                  \sub@lock \tw@
874              \or \or
875                  \sub@lock \z@
876          \fi
877  %
```

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

```
878          \ifsublines@
879              \ifnum\sub@lock<\tw@
880                  \advance\subline@num \@ne
```

```
881              \fi
882          \else
883              \ifnum\@lock<\tw@
884                \advance\line@num \@ne \subline@num \z@
885              \fi
886          \fi}
887
888 %
```

\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.

```
889 \newcount\last@page@num
890   \last@page@num=-10000
891
892 \newcommand*{\fix@page}[1]{%
893   \ifledRcol
894     \ifnum #1=\last@page@numR
895     \else
896       \csnumgdef{@lastabsline@forpageR@\the\page@numR}{\the\absline@numR}%
897       \csnumgdef{@lastline@forpageR@\the\page@numR}{\the\line@numR}%
898       \ifbypage@R
899         \line@numR \z@ \subline@numR \z@
900       \fi
901       \page@numR=#1\relax
902       \last@page@numR=#1\relax
903       \def\next@page@numR{#1}%
904     \fi
905   \else
906     \ifnum #1=\last@page@num
907     \else
908       \csnumgdef{@lastabsline@forpage@\the\page@num}{\the\absline@num}%
909       \csnumgdef{@lastline@forpage@\the\page@num}{\the\line@num}%
910       \ifbypage@
911         \line@num \z@ \subline@num \z@
912       \fi
913       \page@num=#1\relax
914       \last@page@num=#1\relax
915       \def\next@page@num{#1}%
916       \listxadd{\normal@page@break}{\the\absline@num}
917     \fi
918   \fi}
919 %
```

\@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
```
920 \newcommand*{\@pend}[1]{}
921 \newcommand*{\@pendR}[1]{}
```

```
922 \newcommand*{\@lopL}[1]{}
923 \newcommand*{\@lopR}[1]{}
924
925 %
```

\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.

```
926 \newcommand*{\sub@on}{\ifsublines@
927     \let\sub@change=\relax
928   \else
929     \def\sub@change{1}%
930   \fi}
931 \newcommand*{\sub@off}{\ifsublines@
932     \def\sub@change{-1}%
933   \else
934     \let\sub@change=\relax
935   \fi}
936
937 %
```

\@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.

```
938
939 \newcommand*{\@adv}[1]{%
940   \ifsublines@
941     \ifledRcol
942       \advance\subline@numR by #1\relax
943       \ifnum\subline@numR<\z@
944         \led@warn@BadAdvancelineSubline
945         \subline@numR \z@
946       \fi
947     \else
948       \advance\subline@num by #1\relax
949       \ifnum\subline@num<\z@
950         \led@warn@BadAdvancelineSubline
951         \subline@num \z@
952       \fi
953     \fi
954   \else
955     \ifledRcol
956       \advance\line@numR by #1\relax
957       \ifnum\line@numR<\z@
958         \led@warn@BadAdvancelineLine
959         \line@numR \z@
960       \fi
961     \else
962       \advance\line@num by #1\relax
```

```
963        \ifnum\line@num<\z@
964           \led@warn@BadAdvancelineLine
965           \line@num \z@
966        \fi
967      \fi
968    \fi
969    \set@line@action}
970
971 %
```

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

```
972
973 \newcommand*{\@set}[1]{%
974    \ifledRcol
975      \ifsublines@
976         \subline@numR=#1\relax
977      \else
978         \line@numR=#1\relax
979      \fi
980      \set@line@action
981    \else
982      \ifsublines@
983         \subline@num=#1\relax
984      \else
985         \line@num=#1\relax
986      \fi
987      \set@line@action
988    \fi}
989
990 %
```

`\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.

```
991
992 \newcommand*{\l@d@set}[1]{%
993    \ifledRcol
994      \line@numR=#1\relax
995      \advance\line@numR \@ne
996      \def\l@dchset@num{#1}
997    \else
998      \line@num=#1\relax
999      \advance\line@num \@ne
1000     \def\l@dchset@num{#1}
1001   \fi}
1002 \let\l@dchset@num\relax
```

```
1003
1004 %
```

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

```
1005
1006 \newcommand*{\page@action}{%
1007   \ifledRcol
1008     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
1009     \xright@appenditem{\next@page@numR}\to\actions@listR
1010   \else
1011     \xright@appenditem{\the\absline@num}\to\actionlines@list
1012     \xright@appenditem{\next@page@num}\to\actions@list
1013   \fi}
1014 %
```

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

```
1015
1016 \newcommand*{\set@line@action}{%
1017   \ifledRcol
1018     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
1019     \ifsublines@
1020       \@l@dtempcnta=-\subline@numR
1021     \else
1022       \@l@dtempcnta=-\line@numR
1023     \fi
1024     \advance\@l@dtempcnta by -5000\relax
1025     \xright@appenditem{\the\@l@dtempcnta}\to\actions@listR
1026   \else
1027     \xright@appenditem{\the\absline@num}\to\actionlines@list
1028     \ifsublines@
1029       \@l@dtempcnta=-\subline@num
1030     \else
1031       \@l@dtempcnta=-\line@num
1032     \fi
1033     \advance\@l@dtempcnta by -5000\relax
1034     \xright@appenditem{\the\@l@dtempcnta}\to\actions@list
1035   \fi}
1036 %
```

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

```
1037
1038 \newcommand*{\sub@action}{%
1039   \ifledRcol
1040     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
1041     \ifsublines@
```

```
1042        \xright@appenditem{-1001}\to\actions@listR
1043      \else
1044        \xright@appenditem{-1002}\to\actions@listR
1045      \fi
1046   \else
1047      \xright@appenditem{\the\absline@num}\to\actionlines@list
1048      \ifsublines@
1049        \xright@appenditem{-1001}\to\actions@list
1050      \else
1051        \xright@appenditem{-1002}\to\actions@list
1052      \fi
1053   \fi}
1054 %
```

\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.

```
1055 \newcommand*{\lock@on}{\futurelet\next\do@lockon}
1056
1057 \newcommand*{\do@lockon}{%
1058   \ifx\next\lock@off
1059      \global\let\lock@off=\skip@lockoff
1060   \else
1061      \ifledRcol
1062        \do@lockonR
1063      \else
1064        \do@lockonL
1065      \fi
1066   \fi}
1067
1068
1069 \newcommand*{\do@lockonL}{%
1070   \xright@appenditem{\the\absline@num}\to\actionlines@list
1071   \ifsublines@
1072      \xright@appenditem{-1005}\to\actions@list
1073      \ifnum\sub@lock=\z@
1074        \sub@lock \@ne
1075      \else
1076        \ifnum\sub@lock=\thr@@
1077          \sub@lock \@ne
1078        \fi
1079      \fi
1080   \else
1081      \xright@appenditem{-1003}\to\actions@list
```

```
1082     \ifnum\@lock=\z@
1083       \@lock \@ne
1084     \else
1085       \ifnum\@lock=\thr@@
1086         \@lock \@ne
1087       \fi
1088     \fi
1089   \fi}
1090
1091 %
```

\lock@off
\do@lockoff
\do@lockoffL
\skip@lockoff

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

```
1092 \newcommand*{\do@lockoffL}{%
1093   \xright@appenditem{\the\absline@num}\to\actionlines@list
1094   \ifsublines@
1095     \xright@appenditem{-1006}\to\actions@list
1096     \ifnum\sub@lock=\tw@
1097       \sub@lock \thr@@
1098     \else
1099       \sub@lock \z@
1100     \fi
1101   \else
1102     \xright@appenditem{-1004}\to\actions@list
1103     \ifnum\@lock=\tw@
1104       \@lock \thr@@
1105     \else
1106       \@lock \z@
1107     \fi
1108   \fi}
1109
1110 \newcommand*{\do@lockoff}{%
1111   \ifledRcol
1112     \do@lockoffR
1113   \else
1114     \do@lockoffL
1115   \fi}
1116 \newcommand*{\skip@lockoff}{\global\let\lock@off=\do@lockoff}
1117 \global\let\lock@off=\do@lockoff
1118
1119 %
```

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

```
1120 \newcommand*{\n@num}{%
1121   \ifledRcol%
1122     \xright@appenditem{\the\absline@numR}\to\actionlines@listR
1123     \xright@appenditem{-1007}\to\actions@listR
1124   \else%
```

```
1125        \xright@appenditem{\the\absline@num}\to\actionlines@list%
1126        \xright@appenditem{-1007}\to\actions@list%
1127     \fi%
1128 }%
1129
1130 %
```

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

```
1131 \newcommand*{\n@num@stanza}{%
1132     \ifledRcol%
1133        \xright@appenditem{\the\absline@numR}\to\actionlines@listR%
1134        \xright@appenditem{-1008}\to\actions@listR%
1135     \else%
1136        \xright@appenditem{\the\absline@num}\to\actionlines@list%%
1137        \xright@appenditem{-1008}\to\actions@list%
1138     \fi%
1139 }
1140 %
```

\ifl@dhidenumber \hidenumbering hides number in margin. It uses action code 1009. \hidenumberingonleftpage
\hidenumbering and \hidenumberingonrightpage ara variant, using action code only conditionnaly
\hidenumberingonleftpage
\hidenumberingonrigh**hh@nge**

```
1141 \newif\ifl@dhidenumber
1142 \newcommand*{\hidenumbering}{
1143     \ifledRcol%
1144        \write\linenum@outR{\string\hide@num}%
1145     \else%
1146        \write\linenum@out{\string\hide@num}%
1147     \fi%
1148 }%
1149 \newcommand*{\hide@num}{%
1150     \ifledRcol%
1151        \xright@appenditem{\the\absline@numR}\to\actionlines@listR%
1152        \xright@appenditem{-1009}\to\actions@listR%
1153     \else%
1154        \xright@appenditem{\the\absline@num}\to\actionlines@list%%
1155        \xright@appenditem{-1009}\to\actions@list%
1156     \fi%
1157 }
1158 \newcommand*{\hidenumberingonleftpage}{%
1159     \ifledRcol%
1160        \write\linenum@outR{\string\hide@num@left}%
1161     \else%
1162        \write\linenum@out{\string\hide@num@left}%
1163     \fi%
1164 }%
1165
```

```
1166 \newcommand*{\hide@num@left}{%
1167   \ifledRcol%
1168     \ifodd\page@numR\else%
1169       \xright@appenditem{\the\absline@numR}\to\actionlines@listR%
1170       \xright@appenditem{-1009}\to\actions@listR%
1171     \fi%
1172   \else%
1173     \ifodd\page@num\else%
1174       \xright@appenditem{\the\absline@num}\to\actionlines@list%%
1175       \xright@appenditem{-1009}\to\actions@list%
1176     \fi%
1177   \fi%
1178 }%
1179
1180 \newcommand*{\hidenumberingonrightpage}{%
1181   \ifledRcol%
1182     \write\linenum@outR{\string\hide@num@right}%
1183   \else%
1184     \write\linenum@out{\string\hide@num@right}%
1185   \fi%
1186 }%
1187
1188 \newcommand*{\hide@num@right}{%
1189   \ifledRcol%
1190     \ifodd\page@numR%
1191       \xright@appenditem{\the\absline@numR}\to\actionlines@listR%
1192       \xright@appenditem{-1009}\to\actions@listR%
1193     \fi%
1194   \else%
1195     \ifodd\page@num%
1196       \xright@appenditem{\the\absline@num}\to\actionlines@list%%
1197       \xright@appenditem{-1009}\to\actions@list%
1198     \fi%
1199   \fi%
1200 }%
1201
1202 %
```

\@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.

```
1203 \newcount\insert@count
1204 %
```

- #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.

```
1205 \newcommand*{\dummy@ref}[2]{#2}
1206 %
```

\@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.

```
1207 \newcommand*{\@ref}[2]{%
1208   \ifledRcol%
1209     \@ref@regR{#1}{#2}%
1210   \else%
1211     \@ref@reg{#1}{#2}%
1212   \fi%
1213 }%
1214 \newcommand*{\@ref@reg}[2]{%
1215   \global\insert@count=#1\relax
1216   \global\advance\@edtext@level by 1%
1217   \loop\ifnum\insert@count>\z@
1218     \xright@appenditem{\the\absline@num}\to\insertlines@list
1219     \global\advance\insert@count \m@ne
1220   \repeat
1221 %
```

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.

```
1222   \begingroup
1223     \let\@ref=\dummy@ref
1224     \let\@lopL\@gobble
1225     \let\page@action=\relax
1226     \let\sub@action=\relax
1227     \let\set@line@action=\relax
1228     \let\@lab=\relax
1229     \let\@lemma=\relax%
1230     \let\@sw\@gobblethree%
1231     #2
1232     \global\endpage@num=\page@num
1233     \global\endline@num=\line@num
1234     \global\endsubline@num=\subline@num
1235   \endgroup
1236 %
```

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

```
1237    \xright@appenditem%
1238      {\the\page@num|\the\line@num|%
1239       \ifsublines@ \the\subline@num \else 0\fi|%
1240       \the\endpage@num|\the\endline@num|%
1241       \ifsublines@ \the\endsubline@num \else 0\fi}\to\line@list
1242 %
```

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.

```
1243      \expandafter\list@create\expandafter{\csname sw@list@edtext@tmp@\the\
     @edtext@level\endcsname}%
1244      \providebool{lemmacommand@\the\@edtext@level}%
1245      \boolfalse{lemmacommand@\the\@edtext@level}%
1246 %
```

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

```
1247    #2%
1248 %
```

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.

```
1249  \ifnum\@edtext@level>0%
1250    \def\create@this@edtext@level{\expandafter\list@create\expandafter{\
     csname sw@list@edtext@\the\@edtext@level\endcsname}}%
1251    \ifcsundef{sw@list@edtext@\the\@edtext@level}{\create@this@edtext@level
     }{}%
1252    \letcs{\@tmp}{sw@list@edtext@\the\@edtext@level}%
1253    \letcs{\@tmpp}{sw@list@edtext@tmp@\the\@edtext@level}
1254    \xright@appenditem{\expandonce\@tmpp}\to\@tmp%
1255    \global\cslet{sw@list@edtext@\the\@edtext@level}{\@tmp}%
1256  \fi%
1257 %
```

Decrease edtext level counter.

```
1258  \global\advance\@edtext@level by -1%
1259 %

1260 }
1261
1262 %
```

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.

```
1263  \newwrite\linenum@out
1264  %
```

\iffirst@linenum@out@   Once any file is opened on this stream, we keep it open forever, or else switch to another
\first@linenum@out@true   file that we keep open. The reason is that we want the output routine to write the page
\first@linenum@out@false   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.

```
1265  \newif\iffirst@linenum@out@
1266     \first@linenum@out@true
1267  %
```

\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.

```
1268  \newcommand{\this@line@list@version}{5}%
1269  %
```

\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.

```
1270  \newcommand*{\line@list@stuff}[1]{%
1271  %
```

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

```
1272     \read@linelist{#1}%
1273  %
```

   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.

```
1274     \iffirst@linenum@out@
1275        \immediate\closeout\linenum@out%
1276        \global\first@linenum@out@false%
```

```
1277        \immediate\openout\linenum@out=\l@auxdir#1\relax%
1278        \immediate\write\linenum@out{\string\line@list@version{\
    this@line@list@version}}%
1279        \ifl@dpaging%
1280          \immediate\write\linenum@out{\string\@par@sync@option{\
    @par@this@sync@option}}%
1281        \fi%
1282      \else
1283 %
```

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.

```
1284        \if@minipage%
1285          \leavevmode%
1286        \fi%
1287        \closeout\linenum@out%
1288        \openout\linenum@out=\l@auxdir#1\relax%
1289        \write\linenum@out{\string\line@list@version{\this@line@list@version}}
    %
1290        \ifl@dpaging%
1291          \write\linenum@out{\string\@par@sync@option{\@par@this@sync@option}}
    %
1292        \fi%
1293      \fi}
1294
1295 %
```

\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.

```
1296 \newcommand*{\new@line}{%
1297   \IfStrEq{\led@pb@setting}{after}%
1298     {\xifinlist{\the\absline@num}{\l@prev@nopb}%
1299       {\xifinlist{\the\absline@num}{\normal@page@break}%
1300         {\numgdef{\@next@page}{\c@page+\@ne}%
1301         \write\linenum@out{\string\@nl[\@next@page][\@next@page]}%
1302         }%
1303         {\write\linenum@out{\string\@nl[\the\c@page][\thepage]}}%
1304       }%
1305       {\write\linenum@out{\string\@nl[\the\c@page][\thepage]}}}%
1306     {}%
1307   \IfStrEq{\led@pb@setting}{before}%
1308     {\numdef{\next@absline}{\the\absline@num+\@ne}%
1309     \xifinlist{\next@absline}{\l@prev@nopb}%
1310       {\xifinlist{\the\absline@num}{\normal@page@break}%
1311         {\numgdef{\nc@page}{\c@page+\@ne}%
1312         \write\linenum@out{\string\@nl[\nc@page][\nc@page]}%
1313         }%
1314         {\write\linenum@out{\string\@nl[\the\c@page][\thepage]}}%
1315       }%
```

```
1316        {\write\linenum@out{\string\@nl[\the\c@page][\thepage]}}%
1317      }%
1318      {}%
1319    \IfStrEqCase{\led@pb@setting}{{before}{\relax}{after}{\relax}}[\write\
       linenum@out{\string\@nl[\the\c@page][\thepage]}]%
1320  }
1321
1322  %
```

\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.

```
1323  \newif\if@noneed@Footnote%
1324
1325  \newcommand*{\flag@start}{%
1326    \ifledRcol%
1327      \edef\next{\write\linenum@outR{%
1328                      \string\@ref[\the\insert@countR][]}}%
1329      \next%
1330      \ifnum\insert@countR<1%
1331        \if@noneed@Footnote\else%
1332          \led@err@EdtextWithoutFootnote%
1333        \fi%
1334      \fi%
1335    \else%
1336      \edef\next{\write\linenum@out{%
1337                      \string\@ref[\the\insert@count][]}}%
1338      \next%
1339      \ifnum\insert@count<1%
1340        \if@noneed@Footnote\else%
1341          \led@err@EdtextWithoutFootnote%
1342        \fi%
1343      \fi%
1344    \fi}%
1345
1346  %
```

\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.

```
1347
1348
1349 \newcommand*{\startsub}{\dimen0\lastskip
1350   \ifdim\dimen0>0pt \unskip \fi
1351   \ifledRcol \write\linenum@outR{\string\sub@on}%
1352   \else       \write\linenum@out{\string\sub@on}%
1353   \fi
1354   \ifdim\dimen0>0pt \hskip\dimen0 \fi}
1355 \def\endsub{\dimen0\lastskip
1356   \ifdim\dimen0>0pt \unskip \fi
1357   \ifledRcol \write\linenum@outR{\string\sub@off}%
1358   \else       \write\linenum@out{\string\sub@off}%
1359   \fi
1360   \ifdim\dimen0>0pt \hskip\dimen0 \fi}
1361
1362 %
```

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

```
1363 \newcommand*{\advanceline}[1]{\leavevmode%
1364   \ifledRcol \write\linenum@outR{\string\@adv[#1]}%
1365   \else       \write\linenum@out{\string\@adv[#1]}%
1366   \fi%
1367 }
1368 %
```

\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.

```
1369
1370 \newcommand*{\setline}[1]{%
1371  \leavevmode%
1372   \ifnum#1<\z@
1373     \led@warn@BadSetline
1374   \else
1375     \ifledRcol \write\linenum@outR{\string\@set[#1]}%
1376     \else       \write\linenum@out{\string\@set[#1]}%
1377     \fi
1378   \fi}
1379
1380 %
```

\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.

```
1381
1382 \newcommand*{\setlinenum}[1]{%
1383   \ifnum#1<\z@
1384     \led@warn@BadSetlinenum
1385   \else
1386     \ifledRcol \write\linenum@outR{\string\l@d@set[#1]}
1387     \else       \write\linenum@out{\string\l@d@set[#1]} \fi
1388   \fi}
1389
1390 %
```

\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.

```
1391
1392 \newcommand*{\startlock}{%
1393   \ifledRcol \write\linenum@outR{\string\lock@on}%
1394   \else       \write\linenum@out{\string\lock@on}%
1395   \fi}
1396 \def\endlock{%
1397   \ifledRcol \write\linenum@outR{\string\lock@off}%
1398   \else       \write\linenum@out{\string\lock@off}%
1399   \fi}
1400 %
```

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

```
1401 \newif\ifl@dskipnumber
1402 \newif\ifl@dskipversenumber%
1403 \newcommand*{\skipnumbering}{%
1404   \leavevmode%
1405   \ifledRcol%
1406     \ifinstanza%
1407        \write\linenum@outR{\string\n@num@stanza}%
1408     \else%
1409        \write\linenum@outR{\string\n@num}%
1410     \fi%
1411     \advanceline{-1}%
1412   \else%
1413     \ifinstanza%
1414        \write\linenum@out{\string\n@num@stanza}%
1415     \else%
1416        \write\linenum@out{\string\n@num}%
1417     \fi%
1418     \advanceline{-1}%
1419   \fi%
1420 }%
1421
```

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 footnote macros directly within a paragraph whose lines are numbered (see comments to `\do@line`, VII.2.1 p. 135). 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.

```
1423 \list@create{\end@lemmas}
1424 %
```

\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.

```
1425 \newcommand{\dummy@edtext}[2]{#1}
1426 %
```

\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`.

```
1427 \newcommand{\dummy@edtext@showlemma}[2]{\showlemma{#1}}%
1428 %
```

   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.[25] 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.)

```
1429  \newcommand*{\no@expands}{%
1430    \let\select@@lemmafont=0%
1431    \let\startsub=\relax  \let\endsub=\relax
1432    \let\startlock=\relax \let\endlock=\relax
1433    \let\edlabel=\@gobble
1434    \let\setline=\@gobble \let\advanceline=\@gobble
1435    \let\sameword\sameword@inedtext%
1436    \let\edtext=\dummy@edtext
1437    \l@dtabnoexpands
1438    \morenoexpands}
```

---

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

1439 `\let\morenoexpands=\relax`

1440

1441 `%`

`\@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.

1442 `\newcommand{\@tag}{}`

1443 `%`

`\@edtext@level`   This counter is increased by 1 at each level of `\edtext`.

1444 `\newcount\@edtext@level%`

1445 `\@edtext@level=0%`

1446 `%`

`\if@edtext@secondarg@`   This boolean is set to TRUE before reading the second argument of a `\edtext`. It is tested on some macro which must be executed only inside a second argument.

1447 `\newif\if@edtext@secondarg@%`

1448 `%`

`\theedtext`   The edtext counter is increased at each `\edtext` command. It is used to add to insert hyperlinks between a notes and the lemma.

1449 `\newcounter{edtext}`

1450 `\renewcommand{\theedtext}{edtxt@\arabic{edtext}}%`

1451 `%`

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

1452 `\newcommand{\edtext}[2]{\leavevmode%`

1453 `%`

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

1454 `  \ifnumberedpar@%`

1455 `%`

we increment the `\@edtext@level` TEX counter to know in which level of `\edtext` we are.

1456 `    \global\advance\@edtext@level by 1%`

1457 `%`

We also increase the edtext LATEX counter to insert hypertarget if the `hyperref` package is loaded.

1458 `    \stepcounter{edtext}%`

1459 `%`

By default, we do not use `\lemma`

```
1460        \global\@lemmacommand@false%
1461 %
```

```
1462        \begingroup%
1463 %
```

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.

```
1464        \ifledRcol%
1465          \ifcsvoid{sw@list@edtextR@\the\@edtext@level}%
1466                {\global\let\sw@inthisedtext\empty}%
1467                {\expandafter\gl@p\csname sw@list@edtextR@\the\@edtext@level\
      endcsname\to\sw@inthisedtext}%
1468        \else%
1469          \ifcsvoid{sw@list@edtext@\the\@edtext@level}%
1470                {\global\let\sw@inthisedtext\empty}%
1471                {\expandafter\gl@p\csname sw@list@edtext@\the\@edtext@level\
      endcsname\to\sw@inthisedtext}%
1472        \fi%
1473 %
```

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

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

```
1474           \global\renewcommand{\@tag}{%
1475             \no@expands #1%
1476             }%
1477 %
```

\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.

```
1478           \set@line%
1479 %
```

\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.

```
1480           \ifledRcol \global\insert@countR \z@%
1481           \else      \global\insert@count \z@ \fi%
1482 %
```

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.

```
1483          \@edtext@secondarg@true%
1484          \ignorespaces #2\relax%
1485          \@edtext@secondarg@false%
1486 %
```

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

```
1487          \@ifundefined{xpg@main@language}{%if not polyglossia
1488            \flag@start}%
1489            {\if@RTL\flag@end\else\flag@start\fi%
1490            }%
1491 %
```

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

```
1492          \if@lemmacommand@%
1493           \ifledRcol%
1494             \write\linenum@outR{\string\@lemma}%
1495           \else%
1496             \write\linenum@out{\string\@lemma}%
1497           \fi%
1498         \fi%
1499 %
```

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.

```
1500         \endgroup%
1501         \ifdef{\hypertarget}%
1502            {%
1503            \csedef{thisedtext@\the\@edtext@level}{\theedtext}%We need one
      macro by level, as #1 can contain new \edtext
1504            \Hy@raisedlink@left{\hypertarget{\csuse{thisedtext@\the\
      @edtext@level}:start}{}}%
1505            \showlemma{#1}%
1506            \Hy@raisedlink{\hypertarget{\csuse{thisedtext@\the\@edtext@level}:
      end}{}}%
1507            }%
1508            {%
1509            \showlemma{#1}%
```

```
1510          }%
1511 %
```

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.

```
1512        \ifx\end@lemmas\empty \else%
1513          \gl@p\end@lemmas\to\x@lemma%
1514          \x@lemma%
1515          \global\let\x@lemma=\relax%
1516        \fi%
1517        \@ifundefined{xpg@main@language}{%if not polyglossia
1518            \flag@end}%
1519            {\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).
1520            }%
1521 %
```

We switch some flags to false.

- 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.

```
1522        \global\@noneed@Footnotefalse%
1523        \global\advance\@edtext@level by -1%
1524        \global\@lemmacommand@false%
1525 %
```

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

```
1526    \else%
1527    \showlemma{#1} (\textbf{\textsc{Edtext outside numbered paragraph}})\
    led@err@edtextoutsidepstart%
1528    \fi%
1529 }%
1530
1531 \newcommand*{\flag@end}{%
1532    \ifledRcol%
1533      \write\linenum@outR{]}%
1534    \else%
1535      \write\linenum@out{]}}%
1536    \fi}%
1537
1538 %
```

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

```
1539  \newif\ifnumberline
1540  \numberlinetrue
1541  %
```

\set@line   The \set@line macro is called by \edtext to put the line-reference field and font spec-
ifier 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 con-
vention for numerical references that have not yet been resolved.

```
1542  \newcommand*{\set@line}{%
1543    \ifledRcol
1544      \ifx\line@listR\empty
1545        \global\noteschanged@true
1546        \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
1547      \else
1548        \gl@p\line@listR\to\@tempb
1549        \xdef\l@d@nums{\@tempb|\edfont@info}%
1550        \global\let\@tempb=\undefined
1551      \fi
1552    \else
1553      \ifx\line@list\empty
1554        \global\noteschanged@true
1555        \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
1556      \else
1557        \gl@p\line@list\to\@tempb
1558        \xdef\l@d@nums{\@tempb|\edfont@info}%
1559        \global\let\@tempb=\undefined
1560      \fi
1561    \fi}
1562
1563  %
```

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

```
1564  \newcommand*{\edfont@info}{\f@encoding/\f@family/\f@series/\f@shape}
1565
1566  %
```

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. 119).

```
1567 \newcommand*{\lemma}[1]{%
1568   \global\@lemmacommand@true%
1569   \global\renewcommand{\@tag}{%
1570     \no@expands #1%
1571   }%
1572   \ignorespaces%
1573 }%
1574 %
```

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

```
1575 \newcommand{\@lemma}{%
1576   \booltrue{lemmacommand@\the\@edtext@level}%
1577 }%
1578 %
```

\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.

```
1579 \newif\if@lemmacommand@%
1580 %
```

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. 92): 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.

```
1581 \newcommand*{\linenum}[1]{%
1582   \xdef\@tempa{#1|||||||\noexpand\\\l@d@nums}%
1583   \global\let\l@d@nums=\empty
1584   \expandafter\line@set\@tempa|\\\ignorespaces}
1585 %
```

\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.

```
1586  \def\line@set#1|#2\\#3|#4\\{%
1587    \gdef\@tempb{#1}%
1588    \ifx\@tempb\empty
1589         \l@d@add{#3}%
1590    \else
1591         \l@d@add{#1}%
1592    \fi
1593    \gdef\@tempb{#4}%
1594    \ifx\@tempb\empty\else
1595       \l@d@add{|}\line@set#2\\#4\\%
1596    \fi}
1597  %
```

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

```
1598  \newcommand{\l@d@add}[1]{\xdef\l@d@nums{\l@d@nums#1}}
1599
1600  %
```

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 preced-
          ing 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 | | foo^1 foo^2 bar^2 foo^3 | (A)(B) foo^4 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`.[26]

    Because there is a bug with `\detokenize` and X͟ǝTEX when using non BMP characters[27], we detokenize only for not X͟ǝTEXengines. In any case, in X͟ǝTEX, 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.

```
1601 \newcommand{\get@sw@txt}[1]{%
1602   \ifxetex%
1603     \xdef\sw@txt{#1}%
1604   \else%
1605     \expandafter\xdef\expandafter\sw@txt\expandafter{\detokenize{#1}}%
1606   \fi%
1607 }%
1608 %
```

`\sameword` The hight level macro `\sameword`, used by the editor.

```
1609 \newcommandx{\sameword}[2][1,usedefault]{%
1610     \leavevmode%
1611     \get@sw@txt{#2}%
1612 %
```

Now, the real code. First, increment the counter corresponding to the argument.

---

[26]See `http://tex.stackexchange.com/q/244538/7712`.
[27]`http://sourceforge.net/p/xetex/bugs/108/`

```
1613    \unless\ifledRcol%
1614        \csnumgdef{sw@\sw@txt}{\csuse{sw@\sw@txt}+\@ne}%
1615 %
```

Then, write its value to the numbered file.

```
1616        \protected@write\linenum@out{}{\string\@sw{\sw@txt}{\csuse{sw@\sw@txt
}}{#1}}%
1617 %
```

Do the same thing if we are in the right columns.

```
1618    \else%
1619        \csnumgdef{sw@\sw@txt}{\csuse{sw@\sw@txt}+\@ne}%
1620        \protected@write\linenum@outR{}{\string\@sw{\sw@txt}{\csuse{sw@\sw@txt
}}{#1}}%
1621    \fi%
1622 %
```

And print the word.

```
1623    #2%
1624 }%
1625 %
```

A flag set to true if a `\@sw` relative rank must be added to the list of ranks for a specific `\edtext`.

```
\if@addsw 1626  \newif\if@addsw%
1627 %
```

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

```
1628 \newcommand{\@sw}[3]{%
1629    \get@sw@txt{#1}%
1630    \unless\ifledRcol%
1631 %
```

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

```
1632        \csxdef{sw@\sw@txt @\the\absline@num @\the\section@num}{#2}%
1633 %
```

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

```
1634        \numdef{\prev@line}{\the\absline@num-1}%
1635        \ifcsundef{sw@\sw@txt @\prev@line @\the\section@num}{%
1636          \csnumgdef{sw@\sw@txt @\prev@line @\the\section@num}{#2-1}%
1637        }{}%
1638 %
```

Then, calculate the position of the word in the line.

```
1639        \numdef{\the@sw}{#2-\csuse{sw@\sw@txt @\prev@line @\the\section@num}}%
1640 %
```

And do the same thing for the right side.

```
1641    \else%
1642      \csxdef{sw@\sw@txt @\the\absline@numR @\the\section@numR @R}{#2}%
1643      \numdef{\prev@line}{\the\absline@numR-1}%
1644      \ifcsundef{sw@\sw@txt @\prev@line @\the\section@numR @R}{%
1645        \csnumgdef{sw@\sw@txt @\prev@line @\the\section@numR @R}{#2-1}%
1646      }{}%
1647      \numdef{\the@sw}{#2-\csuse{sw@\sw@txt @\prev@line @\the\section@numR @R
}}%
1648    \fi%
1649 %
```

And now, add it to the list of `\@sw` for the current edtext, in all depth.

```
1650    \@tempcnta=\@edtext@level
1651    \@whilenum{\@tempcnta>0}\do{%
1652      \ifcsdef{sw@list@edtext@tmp@\the\@tempcnta}%
1653        {%
1654          \@addswfalse%
1655          \notbool{lemmacommand@\the\@tempcnta}%
1656            {\@addswtrue}%
1657            {\IfStrEq{#3}{inlemma}%
1658              {\@addswtrue}%
1659              {%
1660               \def\do##1{%
1661                  \ifnumequal{##1}{\the\@tempcnta}%
1662                    {\@addswtrue\listbreak}%
1663                    {}%
1664               }%
1665               \docsvlist{#3}%
1666              }%
1667            }%
1668          \if@addsw%
1669            \letcs{\@tmp}{sw@list@edtext@tmp@\the\@tempcnta}%
1670            \ifledRcol%
1671              \xright@appenditem{{\the@sw}{\the\absline@numR}}\to\@tmp%
1672            \else%
1673              \xright@appenditem{{\the@sw}{\the\absline@num}}\to\@tmp%
1674            \fi%
1675            \cslet{sw@list@edtext@tmp@\the\@tempcnta}{\@tmp}%
1676          \fi%
1677        }%
1678        {}%
1679      \advance\@tempcnta by -1%
1680    }%
1681 }%
1682 %
```

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

```
1683 \newcommandx{\sameword@inedtext}[2][1,usedefault]{%
1684   \get@sw@txt{#2}%
1685   \unless\ifledRcol@%
1686 %
```

Just a precaution.

```
1687       \ifx\sw@list@inedtext\empty%
1688         \def\the@sw{999}%
1689         \def\this@absline{-99}%
1690       \else%
1691 %
```

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

```
1692       \gl@p\sw@list@inedtext\to\@tmp%
1693       \edef\the@sw{\expandafter\@firstoftwo\@tmp}%
1694       \edef\this@absline{\expandafter\@secondoftwo\@tmp}%
1695     \fi%
1696 %
```

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

```
1697       \ifcsdef{sw@\sw@txt @\this@absline @\the\section@num}{%
1698       \numdef{\prev@line}{\this@absline-1}%
1699       \numdef{\sw@atthisline}{\csuse{sw@\sw@txt @\this@absline @\the\
      section@num}-\csuse{sw@\sw@txt @\prev@line @\the\section@num}}%
1700       }%
1701       {\numdef{\sw@atthisline}{0}}%
1702 %
```

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

```
1703       \ifnumgreater{\sw@atthisline}{1}%
1704         {\showwordrank{#2}{\the@sw}}%
1705         {#2}%
1706 %
```

And the same for right side.

```
1707   \else%
1708     \ifx\sw@list@inedtext\empty%
1709       \def\the@sw{999}%
1710       \def\this@absline{-99}%
1711     \else%
1712       \gl@p\sw@list@inedtext\to\@tmp%
1713       \edef\the@sw{\expandafter\@firstoftwo\@tmp}%
1714       \edef\this@absline{\expandafter\@secondoftwo\@tmp}%
1715     \fi%
1716     \ifcsdef{sw@\sw@txt @\this@absline @\the\section@numR @R}{%
```

```
1717          \numdef{\prev@line}{\this@absline-1}%
1718          \numdef{\sw@atthisline}{\csuse{sw@\sw@txt @\this@absline @\the\
        section@numR @R}-\csuse{sw@\sw@txt @\prev@line @\the\section@numR @R}}%
1719          }%
1720          {\numdef{\sw@atthisline}{0}}%
1721          \ifnumgreater{\sw@atthisline}{1}%
1722            {\showwordrank{#2}{\the@sw}}%
1723            {#2}%
1724      \fi%
1725  }%
1726  %
```

```
\showwordrank 1727  % Finally, the way the rank will be printed.
1728  \newcommand{\showwordrank}[2]{%
1729    #1\textsuperscript{#2}%
1730  }%
1731  %
```

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

```
1732  \newbox\raw@text
1733  \newif\ifnumberedpar@
1734  \newcount\num@lines
1735  \newbox\one@line
1736  \newcount\par@line
1737  %
```

\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.

```
1738
1739  \newcommand{\AtEveryPstart}[1]{%
1740    \ifstrempty{#1}%
1741      {\xdef\at@every@pstart{}}%
1742      {\gdef\at@every@pstart{\noindent#1}}%
1743  }%
1744  \xdef\at@every@pstart{}%
1745
1746  \newcounter{pstart}
1747  \renewcommand{\thepstart}{{\bfseries\@arabic\c@pstart}. }
1748  \newif\ifnumberpstart
1749  \numberpstartfalse
1750  \newif\iflabelpstart
1751  \labelpstartfalse
1752  \newcommandx*{\pstart}[1][1]{%
1753    \normal@pars%
1754    \ifstrempty{#1}{\at@every@pstart}{\noindent#1}%
1755    \ifluatex%
1756      \edef\l@luatextextdir@L{\the\textdir}%
1757    \fi%
1758    \@nobreaktrue%
1759    \ifnumbering \else%
1760      \led@err@PstartNotNumbered%
1761      \beginnumbering%
1762    \fi%
1763    \ifnumberedpar@%
1764      \led@err@PstartInPstart%
1765      \pend%
1766    \fi%
1767    \list@clear{\inserts@list}%
1768    \global\let\next@insert=\empty%
1769    \begingroup%
1770    \global\advance \l@dnumpstartsL\@ne
1771    \global\setbox\raw@text=\vbox\bgroup%
1772      \ifautopar\else%
1773      \ifnumberpstart%
1774        \ifinstanza\else%
1775         \ifsidepstartnum\else%
1776           \thepstart%
1777          \fi%
1778        \fi%
1779        \fi%
```

```
1780        \fi%
1781     \numberedpar@true%
1782     \iflabelpstart\protected@edef\@currentlabel%
1783          {\p@pstart\thepstart}
1784     \fi%
1785     \l@dzeropenalties%
1786     \ignorespaces%because not automatically ignored if an optional argument
     is used (classical TeX behavior for space after commands)
1787     }
1788 %
```

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

```
1789 \newcommandx*{\pend}[1][1]{\ifnumbering \else%
1790        \led@err@PendNotNumbered%
1791     \fi%
1792     \global\l@dskipversenumberfalse%
1793     \ifnumberedpar@ \else%
1794        \led@err@PendNoPstart%
1795     \fi%
1796 %
```

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.

```
1797     \l@dzeropenalties%
1798     \endgraf\global\num@lines=\prevgraf\egroup%
1799     \global\par@line=0%
1800 %
```

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.

```
1801     \csnumdef{pstartline}{0}%
1802     \loop\ifvbox\raw@text%
1803        \csnumdef{pstartline}{\pstartline+\@ne}%
1804        \do@line%
1805        \ifbypstart@%
1806         \ifnumequal{\pstartline}{1}{%
1807            \bgroup%
1808            \let\leavevmode\relax%
1809            \setline{1}%
1810            \egroup%
1811         \resetprevline@}{}%
1812        \fi%
```

```
1813    \repeat%
1814 %
```

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

```
1815    \flush@notes%
1816    \endgroup%
1817    \ignorespaces%
1818 %
```

Increase pstart counter.

```
1819    \ifnumberpstart%
1820        \pstartnumtrue%
1821    \fi%
1822    \addtocounter{pstart}{1}%
1823 %
```

Print the optional argument of \pend or the content printed after every \pend

```
1824    \normal@pars%
1825    \ifstrempty{#1}{\at@every@pend}{\noindent#1}%
1826 %
```

Restore standard nobreak setting and autopar setting. Normally, \if@nobreak is equal to true only immediately after a sectioning command (read latex.ltx file). As a \pstart...\pend structure can't contain any sectioning command, we set \if@nobreak to false.

```
1827    \@nobreakfalse%
1828    \ifautopar%
1829        \autopar%
1830    \fi%
1831 }
1832
1833 %
```

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    1834
\at@every@pend  1835 \newcommand{\AtEveryPend}[1]{%
                1836    \ifstrempty{#1}%
                1837        {\xdef\at@every@pend{}}%
                1838        {\gdef\at@every@pend{\noindent#1}}%
                1839 }%
                1840 \xdef\at@every@pend{}%
                1841
                1842 %
```

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

```
1843 \newcommand*{\l@dzeropenalties}{%
1844   \brokenpenalty \z@ \clubpenalty \z@
1845   \displaywidowpenalty \z@ \interlinepenalty \z@ \predisplaypenalty \z@
1846   \postdisplaypenalty \z@ \widowpenalty \z@}
1847
1848 %
```

\autopar   In most cases it is only an annoyance to have to label the paragraphs to be num-
bered 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, af-
ter \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 back-
wards 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.

```
1849 \newif\ifautopar
1850 \autoparfalse
1851 \newcommand*{\autopar}{
1852   \ifledRcol
1853     \ifnumberingR \else
1854     \led@err@AutoparNotNumbered
1855     \beginnumberingR
1856     \fi
1857     \else
1858     \ifnumbering \else
1859     \led@err@AutoparNotNumbered
1860     \beginnumbering
1861     \fi
1862   \fi
1863   \autopartrue
1864   \everypar{\setbox0=\lastbox
1865     \endgraf \vskip-\parskip
```

```
1866      \pstart \noindent \kern\wd0 \ifnumberpstart\ifinstanza\else\thepstart\
      fi\fi
1867      \let\par=\pend}%
1868    \ignorespaces}
1869 %
```

\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.

```
1870 \newcommand*{\normal@pars}{\ifautopar\everypar{}\fi\let\par\endgraf}
1871
1872 %
```

\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.

```
1873 \newif\ifautopar@pause
1874 %
```

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

```
1875 \newcommand*{\l@dunhbox@line}[1]{\unhbox #1}
1876 \newcommand*{\do@line}{%
1877  {\vbadness=10000
1878    \splittopskip=\z@
1879    \do@linehook
1880 \l@demptyd@ta
1881    \global\setbox\one@line=\vsplit\raw@text to\baselineskip}%
1882  \unvbox\one@line \global\setbox\one@line=\lastbox
1883  \getline@num
1884  \IfStrEq{\led@pb@setting}{before}{\led@check@pb\led@check@nopb}{}
1885  \ifnum\@lock>\@ne
1886    \inserthangingsymboltrue
1887  \else
1888    \inserthangingsymbolfalse
1889  \fi
1890  \check@pb@in@verse
1891  \ifl@dhidenumber%
1892    \global\l@dhidenumberfalse%
1893    \f@x@l@cks%
1894  \else%
1895    \affixline@num%
1896  \fi%
1897 %
```

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

```
1898    \xifinlist{\the\l@dnumpstartsL}{\eled@sections@@}%
1899       {\print@eledsection}%
1900       {\print@line}%
1901    \IfStrEq{\led@pb@setting}{after}{\led@check@pb\led@check@nopb}{}
1902    }%
1903 %
```

VII.2.2   Process for "normal" line

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

```
1904 \def\print@line{
1905 %
```

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

```
1906       \affixpstart@num%
1907 %
```

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

```
1908       \hb@xt@ \linewidth{%
1909 %
```

User hook.

```
1910          \do@insidelinehook%
1911 %
```

Left line number

```
1912          \l@dld@ta%
1913 %
```

Prepare text to be inserted before notes.

```
1914          \if@firstlineofpage%
1915             \set@Xtxtbeforenotes%
1916             \global\@firstlineofpagefalse%
1917          \fi%
1918 %
```

Insert footnotes made of manuscripts data.

```
1919          \insert@msdata%
1920 %
```

Restore marginal and footnotes.

```
1921          \add@inserts\affixside@note%
1922 %
```

Print left notes.

```
1923        \l@dlsn@te
1924 %
```

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

```
1925        {\ledllfill\hb@xt@ \wd\one@line{\new@line%
1926 %
```

If we use LuaLATEX then restore the direction.

```
1927        \ifluatex%
1928          \textdir\l@luatextextdir@L%
1929        \fi%
1930 %
```

Insert, if needed, the hanging symbol.

```
1931        \inserthangingsymbol%
1932 %
```

And so, print the line.

```
1933        \l@dunhbox@line{\one@line}}%
1934 %
```

Right line number

```
1935        \ledrlfill\l@drd@ta%
1936 %
```

Print right notes.

```
1937        \l@drsn@te
1938      }}%
1939 %
```

And reinsert penalties (for page breaking)…

```
1940     \add@penalties%
1941 }
1942 %
```

VII.2.3   Process for line containing \eledsection command

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

```
1943 \def\print@eledsection{%
1944     \if@firstlineofpage%
1945       \set@Xtxtbeforenotes%
1946       \global\@firstlineofpagefalse%
1947     \fi%
1948     \insert@msdata%
```

```
1949    \add@inserts\affixside@note%
1950    \numdef{\temp@}{\l@dnumpstartsL-1}%
1951    \xifinlist{\temp@}{\eled@sections@@}{\@nobreaktrue}{\@nobreakfalse}%
1952    \@eled@sectioningtrue%
1953    \csuse{eled@sectioning@\the\l@dnumpstartsL}%
1954    \@eled@sectioningfalse%
1955    \global\csundef{eled@sectioning@\the\l@dnumpstartsL}%
1956    \if@RTL%
1957      \hspace{-3\paperwidth}%
1958    {\hbox{\l@dunhbox@line{\one@line}} \new@line}%
1959    \else%
1960      \hspace{3\paperwidth}%
1961    {\new@line \hbox{\l@dunhbox@line{\one@line}}}%
1962    \fi%
1963    \vskip-\baselineskip%
1964  }
1965  %
```

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.

```
1966  \newcommand*{\do@linehook}{}
1967  \newcommand*{\do@insidelinehook}{}
1968  %
```

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

```
1969  \newcommand*{\dolinehook}[1]{\gdef\do@linehook{#1}}%
1970  \newcommand*{\doinsidelinehook}[1]{\gdef\do@insidelinehook{#1}}%
1971
1972  %
```

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

```
1973  \newcommand*{\l@demptyd@ta}{%
1974    \gdef\l@dld@ta{}%
1975    \gdef\l@drd@ta{}%
1976    \gdef\l@dcsnotetext@l{}%
1977    \gdef\l@dcsnotetext@r{}%
1978    \gdef\l@dcsnotetext{}}
1979
1980  %
```

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

```
1981 \newcommand{\l@dlsn@te}{%
1982   \hb@xt@ \z@{\hss\box\l@dlp@rbox\kern\ledlsnotesep}}
1983 \newcommand{\l@drsn@te}{%
1984   \hb@xt@ \z@{\kern\ledrsnotesep\box\l@drp@rbox\hss}}
1985
1986 %
```

`\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`.

```
1987 \newcommand*{\ledllfill}{\hfil}
1988 \newcommand*{\ledrlfill}{}
1989
1990 %
```

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.

```
1991 \newcommand*{\getline@num}{%
1992   \global\advance\absline@num \@ne%
1993   \do@actions
1994   \do@ballast
1995   \ifnumberline
1996       \ifsublines@
1997           \ifnum\sub@lock<\tw@
1998               \global\advance\subline@num \@ne
1999           \fi
2000       \else
2001         \ifnum\@lock<\tw@
2002             \global\advance\line@num \@ne
2003             \global\subline@num \z@
2004         \fi
2005       \fi
2006   \fi
2007 }
2008 %
```

`\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. 149).

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

```
2009 \newcount\ballast@count
2010 \newcounter{ballast}
2011   \setcounter{ballast}{0}
2012 %
```

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.

```
2013 \newcommand*{\do@ballast}{\global\ballast@count \z@
2014   \begingroup
2015     \advance\absline@num \@ne
2016     \ifnum\next@actionline=\absline@num
2017       \ifnum\next@action>-1001\relax
2018         \global\advance\ballast@count by -\c@ballast
2019       \fi
2020     \fi
2021   \endgroup}
2022 %
```

`\do@actions`
`\do@actions@next`   The `\do@actions` macro looks at the list of actions to take at particular absolute line 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`.

```
2023 \newcommand*{\do@actions}{%
2024   \global\let\do@actions@next=\relax
2025   \ifnum\absline@num<\next@actionline\else
2026 %
```

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.

```
2027     \ifnum\next@action>-1001
2028       \global\page@num=\next@action
2029       \global\@firstlineofpagetrue%
2030       \ifbypage@
2031         \global\line@num=\z@ \global\subline@num=\z@
2032         \resetprevline@
2033       \fi
2034       \add@msdata@firstlineofpage%
2035 %
```

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`.)

```
2036        \else
2037          \ifnum\next@action<-4999
2038            \@l@dtempcnta=-\next@action
2039            \advance\@l@dtempcnta by -5001
2040            \ifsublines@
2041                \global\subline@num=\@l@dtempcnta
2042            \else
2043                \global\line@num=\@l@dtempcnta
2044            \fi
2045 %
```

We rescale the value in `\@l@dtempcnta` so that we can use a case statement.

```
2046          \else
2047            \@l@dtempcnta=-\next@action
2048            \advance\@l@dtempcnta by -1000
2049            \do@actions@fixedcode
2050          \fi
2051        \fi
2052 %
```

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.

```
2053        \ifx\actionlines@list\empty
2054            \gdef\next@actionline{1000000}%
2055        \else
2056            \gl@p\actionlines@list\to\next@actionline
2057            \gl@p\actions@list\to\next@action
2058            \global\let\do@actions@next=\do@actions
2059        \fi
2060    \fi
2061 %
```

Make the recursive call, if necessary.

```
2062 \do@actions@next}
2063
2064 %
```

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

```
2065 \newcommand*{\do@actions@fixedcode}{%
2066    \ifcase\@l@dtempcnta
2067    \or%                    % 1001 = starting sublineation
2068        \global\sublines@true
2069    \or%                    % 1002 = ending sublineation
2070        \global\sublines@false
2071    \or%                    % 1003 = starting locking number
2072        \global\@lock=\@ne
```

```
2073    \or%                      % 1004 = ending locking number
2074      \ifnum\@lock=\tw@
2075        \global\@lock=\thr@@
2076      \else
2077        \global\@lock=\z@
2078      \fi
2079    \or%                      % 1005 = starting locking subnumber
2080      \global\sub@lock=\@ne
2081    \or%                      % 1006 = ending locking subnumber
2082      \ifnum\sub@lock=\tw@
2083        \global\sub@lock=\thr@@
2084      \else
2085        \global\sub@lock=\z@
2086      \fi
2087    \or%                      % 1007 = skipping numbering
2088      \l@dskipnumbertrue
2089    \or%                      % 1008 = skipping numbering in stanza
2090      \l@dskipversenumbertrue%
2091    \or%                      % 1009 = hiding number
2092      \l@dhidenumbertrue
2093    \or%                      % 1010 = inserting msdata
2094      \add@msdata%
2095    \else
2096      \led@warn@BadAction
2097    \fi}
2098
2099
2100 %
```

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:

$$n \quad = int((\mathit{linenum} - \mathit{firstlinenum})/\mathit{linenumincrement})$$
$$m \quad = \mathit{firstlinenum} + (n \times \mathit{linenumincrement})$$

(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.

```
2101 \newcommand*{\affixline@num}{%
2102 %
```

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).

```
2103    \ifledgroupnotesL@\else
2104      \ifnumberline
2105        \ifl@dskipnumber
2106          \global\l@dskipnumberfalse
2107        \else
2108          \ifsublines@
2109            \@l@dtempcntb=\subline@num
2110            \ifnum\subline@num>\c@firstsublinenum
2111              \@l@dtempcnta=\subline@num
2112              \advance\@l@dtempcnta by-\c@firstsublinenum
2113              \divide\@l@dtempcnta by\c@sublinenumincrement
2114              \multiply\@l@dtempcnta by\c@sublinenumincrement
2115              \advance\@l@dtempcnta by\c@firstsublinenum
2116            \else
2117              \@l@dtempcnta=\c@firstsublinenum
2118            \fi
2119 %
```

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.

```
2120            \ch@cksub@l@ck
2121 %
```

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

```
2122          \else
2123            \@l@dtempcntb=\line@num
2124 %
```

Check on the \linenumberlist If it is \empty use the standard algorithm.

```
2125            \ifx\linenumberlist\empty
2126              \ifnum\line@num>\c@firstlinenum
2127                \@l@dtempcnta=\line@num
2128                \advance\@l@dtempcnta by-\c@firstlinenum
2129                \divide\@l@dtempcnta by\c@linenumincrement
2130                \multiply\@l@dtempcnta by\c@linenumincrement
2131                \advance\@l@dtempcnta by\c@firstlinenum
2132              \else
2133                \@l@dtempcnta=\c@firstlinenum
2134              \fi
2135            \else
2136 %
```

The `\linenumberlist` was not `\empty`, so here is Wayne's numbering mechanism.
This takes place in TEX's mouth.

```
2137                 \@l@dtempcnta=\line@num
2138                 \edef\rem@inder{,\linenumberlist,\number\line@num,}%
2139                 \edef\sc@n@list{\def\noexpand\sc@n@list
2140                   ####1,\number\@l@dtempcnta,####2|{\def\noexpand\rem@inder
     {####2}}}%
2141                 \sc@n@list\expandafter\sc@n@list\rem@inder|%
2142                 \ifx\rem@inder\empty%
2143                   \advance\@l@dtempcnta\@ne
2144                 \fi
2145               \fi
2146 %
```

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

```
2147                 \ch@ck@l@ck
2148               \fi
2149 %
```

The following tests are true if we need to print a line number.

```
2150             \ifnum\@l@dtempcnta=\@l@dtempcntb
2151               \ifl@dskipversenumber\else
2152 %
```

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`

```
2153             \if@twocolumn
2154               \if@firstcolumn
2155                 \gdef\l@dld@ta{\llap{{\leftlinenum}}}%
2156               \else
2157                 \gdef\l@drd@ta{\rlap{{\rightlinenum}}}%
2158               \fi
2159             \else
2160               \@l@dtempcntb=\line@margin
2161               \ifnum\@l@dtempcntb>\@ne
2162                 \advance\@l@dtempcntb \page@num
2163               \fi
2164               \ifodd\@l@dtempcntb
```

```
2165                    \gdef\l@drd@ta{\rlap{{\rightlinenum}}}%
2166                  \else
2167                    \gdef\l@dld@ta{\llap{{\leftlinenum}}}%
2168                  \fi
2169                \fi
2170              \fi
2171            \fi
2172 %
```

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

```
2173            \f@x@l@cks
2174          \fi
2175        \fi
2176      \fi
2177 }
2178
2179 %
```

`\ch@cksub@l@ck`
`\ch@ck@l@ck`
`\f@x@l@cks`

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

```
2180 \newcommand*{\ch@cksub@l@ck}{%
2181      \ifcase\sub@lock
2182        \or
2183          \ifnum\sublock@disp=\@ne
2184              \@l@dtempcntb=\z@ \@l@dtempcnta=\@ne
2185          \fi
2186        \or
2187          \ifnum\sublock@disp=\tw@ \else
2188              \@l@dtempcntb=\z@ \@l@dtempcnta=\@ne
2189          \fi
2190        \or
2191          \ifnum\sublock@disp=\z@
2192              \@l@dtempcntb=\z@ \@l@dtempcnta=\@ne
2193          \fi
2194      \fi}
2195 %
```

Similarly for line numbers.

```
2196 \newcommand*{\ch@ck@l@ck}{%
2197      \ifcase\@lock
2198        \or
2199          \ifnum\lock@disp=\@ne
2200              \@l@dtempcntb=\z@ \@l@dtempcnta=\@ne
2201          \fi
2202        \or
2203          \ifnum\lock@disp=\tw@ \else
```

```
2204                    \@l@dtempcntb=\z@ \@l@dtempcnta=\@ne
2205                \fi
2206            \or
2207            \ifnum\lock@disp=\z@
2208                    \@l@dtempcntb=\z@ \@l@dtempcnta=\@ne
2209            \fi
2210        \fi}
2211 %
```

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

```
2212 \newcommand*{\f@x@l@cks}{%
2213    \ifcase\@lock
2214    \or
2215        \global\@lock=\tw@
2216    \or \or
2217        \global\@lock=\z@
2218    \fi
2219    \ifcase\sub@lock
2220    \or
2221        \global\sub@lock=\tw@
2222    \or \or
2223        \global\sub@lock=\z@
2224    \fi}
2225
2226 %
```

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.

```
2227 \leftpstartnum
2228 \rightpstartnum  \newif\ifsidepstartnum
2229 \ifsidepstartnum  \newcommand*{\affixpstart@num}{%
2230        \ifsidepstartnum
2231            \if@twocolumn
2232                \if@firstcolumn
2233                    \gdef\l@dld@ta{\llap{{\leftpstartnum}}}%
```

```
2234          \else
2235              \gdef\l@drd@ta{\rlap{{\rightpstartnum}}}%
2236          \fi
2237      \else
2238          \@l@dtempcntb=\line@margin
2239          \ifnum\@l@dtempcntb>\@ne
2240              \advance\@l@dtempcntb \page@num
2241          \fi
2242          \ifodd\@l@dtempcntb
2243              \gdef\l@drd@ta{\rlap{{\rightpstartnum}}}%
2244          \else
2245              \gdef\l@dld@ta{\llap{{\leftpstartnum}}}%
2246          \fi
2247      \fi
2248    \fi
2249
2250 }
2251 %
2252
2253 \newif\ifpstartnum
2254 \pstartnumtrue
2255 \newcommand*{\leftpstartnum}{
2256    \ifpstartnum\thepstart
2257    \kern\linenumsep\fi
2258    \global\pstartnumfalse
2259 }
2260 \newcommand*{\rightpstartnum}{
2261    \ifpstartnum
2262    \kern\linenumsep
2263    \thepstart
2264    \fi
2265    \global\pstartnumfalse
2266 }
2267 %
```

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 paragraph.

```
2268 \list@create{\inserts@list}
2269 %
```

\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.

```
2270 \newcommand*{\add@inserts}{%
2271   \global\let\add@inserts@next=\relax
2272 %
```

If \inserts@list is empty, there are not any more notes or insertions for this
paragraph, and we need not waste our time.

```
2273   \ifx\inserts@list\empty \else
2274 %
```

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

```
2275   \ifx\next@insert\empty
2276     \ifx\insertlines@list\empty
2277       \global\noteschanged@true
2278       \gdef\next@insert{100000}%
2279     \else
2280       \gl@p\insertlines@list\to\next@insert
2281     \fi
2282   \fi
2283 %
```

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.

```
2284   \ifnum\next@insert=\absline@num
2285     \gl@p\inserts@list\to\@insert
2286     \@insert
2287     \global\let\@insert=\undefined
2288     \global\let\next@insert=\empty
2289     \global\let\add@inserts@next=\add@inserts
2290   \fi
2291 \fi
2292 %
```

Make the recursive call, if necessary.

```
2293  \add@inserts@next}
2294
2295  %
```

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 paragraph; these penalties get removed by the \vsplit operation. \displaywidowpenalty and \brokenpenalty are not restored, since we have no easy way to find out where we should insert them.

In this code, \num@lines is the number of lines in the whole paragraph, and \par@line is the line we 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. 139). Finally, the penalty is checked to see that it does not go below −10000.

```
2296  \newcommand*{\add@penalties}{\@l@dtempcnta=\ballast@count
2297    \ifnum\num@lines>\@ne
2298      \global\advance\par@line \@ne
2299      \ifnum\par@line=\@ne
2300        \advance\@l@dtempcnta \clubpenalty
2301      \fi
2302      \@l@dtempcntb=\par@line \advance\@l@dtempcntb \@ne
2303      \ifnum\@l@dtempcntb=\num@lines
2304        \advance\@l@dtempcnta \widowpenalty
2305      \fi
2306      \ifnum\par@line<\num@lines
2307        \advance\@l@dtempcnta \interlinepenalty
2308      \fi
2309    \fi
2310      \ifnum\@l@dtempcnta=\z@
2311        \relax
2312      \else
2313        \ifnum\@l@dtempcnta>-10000
2314          \penalty\@l@dtempcnta
2315        \else
2316          \penalty -10000
2317        \fi
2318      \fi}
2319
2320  %
```

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.

```
2321  \newcommand*{\flush@notes}{%
2322    \@xloop
2323      \ifx\inserts@list\empty \else
2324        \gl@p\inserts@list\to\@insert
2325        \@insert
2326        \global\let\@insert=\undefined
2327    \repeat}
2328
2329  %
```

\@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 types \@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.

```
2330  \def\@xloop#1\repeat{%
2331    \def\body{#1\expandafter\body\fi}%
2332    \body}
2333
2334  %
```

XI.4   Text before notes

\set@Xtxtbeforenotes   The \set@Xtxtbeforenotes macro resets the Xtxtbeforesnotes@⟨*series*⟩@typeset boolean to false. Just before the first note of the ⟨*series*⟩ in a page, the Xtextbeforenotes will be inserted.

```
2335  \newcommand{\set@Xtxtbeforenotes}{%
2336    \unless\ifnocritical@%
2337      \def\do##1{%
2338        \global\togglefalse{Xtxtbeforesnotes@##1@typeset}%
2339      }%
2340      \dolistloop{\@series}%
2341    \fi%
2342  }%
2343  %
```

`\insert@Xtxtbeforenotes` `\insert@Xtxtbeforenotes{⟨series⟩}`, called when inserting a note, will insert the text before the note if it is not already inserted. For paragraphed footnotes, it will insert it as a component of the first footnote. For other types of footnotes, it will insert it as a regular footnote.

```
2344  \newcommand{\insert@Xtxtbeforenotes}[1]{%
2345    \nottoggle{Xtxtbeforesnotes@#1@typeset}{%
2346      \global\toggletrue{Xtxtbeforesnotes@#1@typeset}%
2347      \ifcsvoid{Xtxtbeforenotes@#1}{}{%
2348        \ifcsstring{series@display#1}{paragraph}%
2349          {\noindent\csuse{Xtxtbeforenotes@#1}}%
2350          {\expandafter\insert\csname#1footins\endcsname%
2351            \bgroup%
2352              \noindent\strut\csuse{Xnotefontsize@#1}\csuse{Xtxtbeforenotes@
      #1}%
2353            \egroup%
2354          }%
2355      }%
2356    }%
2357    {}%
2358  }%
2359  %
```

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.

```
2360    \def\select@lemmafont#1|#2|#3|#4|#5|#6|#7|{\select@@lemmafont#7|}
2361    \def\select@@lemmafont#1/#2/#3/#4|%
2362      {\fontencoding{#1}\fontfamily{#2}\fontseries{#3}\fontshape{#4}%
2363      \selectfont}
2364
2365  %
```

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.

```
2366 \newcommand*{\footnoteoptions@}[3]{%
2367     \def\do##1{%
2368         \ifstrequal{#1}{L}{% In Leftside
2369             \xright@appenditem{\noexpand\setkeys[mac]{#3footnoteoption}{\
unexpanded{##1}}}\to\inserts@list%
2370             \global\advance\insert@count \@ne% Increment the left insert
counter.
2371         }%
2372         {%
2373             \xright@appenditem{\noexpand\setkeys[mac]{#3footnoteoption}{\
unexpanded{##1}}}\to\inserts@listR%
2374             \global\advance\insert@countR \@ne% Increment the right insert
counter insert.
2375         }%
2376     }%
2377     \notblank{#2}{\docsvlist{#2}}{}% Parsing all options
2378 }
2379 %
```

XII.3   Notes language

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

```
2380 \newcommandx*{\footnotelang@lua}[1][1=L,usedefault]{%
2381     \ifstrequal{#1}{L}{%
2382     \xright@appenditem{{\csxdef{footnote@luatextextdir}{\the\textdir}}}\to\
inserts@list%Know the dir of lemma
2383         \global\advance\insert@count \@ne%
2384         \xright@appenditem{{\csxdef{footnote@luatexpardir}{\the\pardir}}}\to\
inserts@list%Know the dir of lemma
2385         \global\advance\insert@count \@ne%
2386     }%
2387     {%
2388     \xright@appenditem{{\csxdef{footnote@luatextextdir}{\the\textdir}}}\to\
inserts@listR%Know the dir of lemma
2389         \global\advance\insert@countR \@ne%
2390         \xright@appenditem{{\csxdef{footnote@luatexpardir}{\the\pardir}}}\to\
inserts@listR%Know the dir of lemma
2391         \global\advance\insert@countR \@ne%
2392     }%
2393 }
2394 %
```

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

```
2395  \newcommandx*{\footnotelang@poly}[1][1=L,usedefault]{%
2396      \ifstrequal{#1}{L}{%
2397        \if@RTL%
2398            \xright@appenditem{{\csxdef{footnote@dir}{@RTLtrue}}}\to\
      inserts@list%Know the language used in the lemma
2399            \global\advance\insert@count \@ne%
2400        \else
2401            \xright@appenditem{{\csxdef{footnote@dir}{@RTLfalse}}}\to\
      inserts@list%Know the language of lemma
2402            \global\advance\insert@count \@ne%
2403        \fi%
2404      \xright@appenditem{{\csxdef{footnote@lang}{\expandonce\languagename}}}\
      to\inserts@list%Know the language of lemma
2405      \global\advance\insert@count \@ne%
2406      }%
2407      {%
2408        \if@RTL
2409            \xright@appenditem{{\csxdef{footnote@dir}{@RTLtrue}}}\to\
      inserts@listR%Know the language of lemma
2410            \global\advance\insert@countR \@ne%
2411        \else
2412            \xright@appenditem{{\csxdef{footnote@dir}{@RTLfalse}}}\to\
      inserts@listR%Know the language of lemma
2413            \global\advance\insert@countR \@ne%
2414        \fi
2415      \xright@appenditem{{\csxdef{footnote@lang}{\expandonce\languagename}}}\
      to\inserts@listR%Know the language of lemma
2416      \global\advance\insert@countR \@ne%
2417      }%
2418  }
2419  %
```

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", "paragraphed", "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 parallel pages in eledpar, in order to allow notes to flow from left to right pages and *vice-versa*. Otherwise, we let it to \@MM, which is the standard LaTeX \floatingpenalty.

```
2420 \newcommand*{\footsplitskips}{%
2421   \interlinepenalty=\interfootnotelinepenalty
2422   \unless\ifl@dprintingpages%
2423     \floatingpenalty=\@MM%
2424   \fi%
2425   \splittopskip=\ht\strutbox \splitmaxdepth=\dp\strutbox
2426   \leftskip=\z@skip \rightskip=\z@skip}
2427
2428 %
```

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

```
2429 \let\normalfootnoterule=\footnoterule
2430 %
```

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.

```
2431 \newcommandx{\Xarrangement}[2][1,usedefault]{%
2432   \def\do##1{%
2433     \csname Xarrangement@#2\endcsname{##1}%
2434   }%
```

```
2435    \ifstrempty{#1}%
2436       {%
2437       \dolistloop{\@series}%
2438       }%
2439       {
2440       \docsvlist{#1}%
2441       }%
2442 }%
2443 %
```

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 insert 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.

```
2444 \newcommand*{\Xarrangement@normal}[1]{%
2445    \csgdef{series@display#1}{normal}
2446    \expandafter\let\csname #1footstart\endcsname=\normalfootstart
2447    \expandafter\let\csname v#1footnote\endcsname=\normalvfootnote
2448    \expandafter\let\csname #1footfmt\endcsname=\normalfootfmt
2449    \expandafter\let\csname #1footgroup\endcsname=\normalfootgroup
2450    \expandafter\let\csname #1footnoterule\endcsname=%
2451                                        \normalfootnoterule
2452    \count\csname #1footins\endcsname=1000
2453    \dimen\csname #1footins\endcsname=\csuse{Xmaxhnotes@#1}
2454    \skip\csname #1footins\endcsname=\csuse{Xbeforenotes@#1}%
2455    \advance\skip\csname #1footins\endcsname by\csuse{Xafterrule@#1}%
2456 %
```

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.

```
2457    \csxdef{default@#1footins}{1000}%Use this to confine the  notes to one
     side only
2458 %
```

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).

```
2459    \ifnoledgroup@\else%
2460      \expandafter\let\csname mpv#1footnote\endcsname=\mpnormalvfootnote
2461      \expandafter\let\csname mp#1footgroup\endcsname=\mpnormalfootgroup
2462      \count\csname mp#1footins\endcsname=1000
2463      \dimen\csname mp#1footins\endcsname=\csuse{Xmaxhnotes@#1}
2464      \skip\csname mp#1footins\endcsname=\csuse{Xbeforenotes@#1}%
2465      \advance\skip\csname mp#1footins\endcsname by\csuse{Xafterrule@#1}%
2466    \fi
2467 }
2468
2469 %
```

\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.

```
2470 \notbool{parapparatus@}{\newcommand*}{\newcommand}{\normalvfootnote}[2]{%
2471    \insert@Xtxtbeforenotes{#1}%
2472    \csuse{Xbeforeinserting@#1}%
2473    \insert\csname #1footins\endcsname\bgroup
2474    \hsize=\expandafter\dimexpr\csuse{Xwidth@#1}\relax%
2475    \noindent\csuse{Xbhooknote@#1}%
2476    \csuse{Xnotefontsize@#1}%
2477    \footsplitskips
2478    \ifl@dpairing\ifl@dpaging\else%
2479      \setXnoteswidthliketwocolumns@{#1}%
2480    \fi\fi%
2481    \setXnotespositionliketwocolumns@{#1}%
2482    \spaceskip=\z@skip \xspaceskip=\z@skip
2483    \csname #1footfmt\endcsname #2{#1}\egroup}
2484 %
```

\mpnormalvfootnote   And a somewhat different version for minipages.

```
2485 \notbool{parapparatus@}{\newcommand*}{\newcommand}{\mpnormalvfootnote}[2]{%
2486    \global\setbox\@nameuse{mp#1footins}\vbox{%
2487      \insert@Xtxtbeforenotes{#1}%
2488      \unvbox\@nameuse{mp#1footins}
2489      \noindent\csuse{Xbhooknote@#1}%
2490      \csuse{Xnotefontsize@#1}%
2491      \hsize\columnwidth
```

```
2492        \@parboxrestore
2493        \color@begingroup
2494        \csname #1footfmt\endcsname #2{#1}\color@endgroup}}
2495
2496  %
```

`\normalfootfmt`   \normalfootfmt is a 'normal' macro to take the footnote line and page number infor-
mation (see V.9 p. 92), 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.

```
2497
2498
2499  \notbool{parapparatus@}{\newcommand*}{\newcommand}{\normalfootfmt}[4]{%
2500    \Xledsetnormalparstuff{#4}%
2501    \hangindent=\csuse{Xhangindent@#4}%
2502    \everypar{\hangindent=\csuse{Xhangindent@#4}}%
2503    \rule\z@\splittopskip%
2504    {\printlinefootnote{#1}{#4}}%
2505    \print@lemma{#1}{#2}{#4}%
2506    \csuse{Xwrapcontent@#4}{#3}%
2507    \strut\par}
2508  %
```

`\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 \Xprenotes@ 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.

```
2509  \newcommand*{\normalfootstart}[1]{%
2510  %
```

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. 205. Here is part of this algorithm, when the block of notes are ready to be printed.

```
2511    \ifdimequal{0pt}{\Xprenotes@}{}%
2512        {%
2513        \iftoggle{Xprenotes@}{%
2514            \togglefalse{Xprenotes@}%
2515            \skip\csname #1footins\endcsname=%
2516              \glueexpr\csuse{Xprenotes@}+\csuse{Xafterrule@#1}\relax%
2517            }%
2518        {}%
2519        }%
2520    \vskip\skip\csname #1footins\endcsname%
2521 %
```

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. 203 for the general description of the problem.

```
2522    \leftskip0pt \rightskip0pt
2523    \ifl@dpairing\else%
2524        \hsize=\old@hsize%
2525    \fi%
2526    \setXnoteswidthliketwocolumns@{#1}%
2527    \setXnotespositionliketwocolumns@{#1}%
2528 %
```

And now, print the footnote's rule to finish the footnote's introduction.

```
2529    \print@Xfootnoterule{#1}%
2530 }%
2531 %
```

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

```
2532 \newcommand*{\normalfootgroup}[1]{%
2533    \csuse{Xbhookgroup@#1}%
2534    \unvbox\csname #1footins\endcsname%
2535    \hsize=\old@hsize%
2536    }%
2537
2538 %
```

\mpnormalfootgroup A somewhat different version for minipages. Note that, in this case, we do not make distinctions between the \Xfootgroup and \Xfootstarts macros.

```
2539 \unless\ifnoledgroup@
2540 \newcommand*{\mpnormalfootgroup}[1]{{
2541    \vskip\skip\@nameuse{mp#1footins}
2542    \ifl@dpairing\ifparledgroup%
2543        \leavevmode\marks\parledgroup@{begin}%
```

```
2544      \marks\parledgroup@series{#1}%
2545      \marks\parledgroup@type{Xfootnote}%
2546   \fi\fi\normalcolor%
2547   \ifparledgroup%
2548      \ifl@dpairing%
2549      \else%
2550         \setXnoteswidthliketwocolumns@{#1}%
2551         \setXnotespositionliketwocolumns@{#1}%
2552         \print@Xfootnoterule{#1}%%
2553      \fi%
2554   \else%
2555      \setXnoteswidthliketwocolumns@{#1}%
2556      \setXnotespositionliketwocolumns@{#1}%
2557      \print@Xfootnoterule{#1}%%
2558   \fi%
2559   \setlength{\parindent}{0pt}
2560   \csuse{Xbhookgroup@#1}%
2561   \unvbox\csname mp#1footins\endcsname}}
2562 \fi
2563 %
```

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 foot-
notes 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 switch-
ing 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.

```
2564 \newcommand*{\Xarrangement@paragraph}[1]{%
2565   \csgdef{series@display#1}{paragraph}
2566   \expandafter\newcount\csname #1prevpage@num\endcsname
2567   \expandafter\newcount\csname #1prevpage@numR\endcsname%
2568   \expandafter\let\csname #1footstart\endcsname=\parafootstart
2569   \expandafter\let\csname v#1footnote\endcsname=\paravfootnote
2570   \expandafter\let\csname #1footfmt\endcsname=\parafootfmt
2571   \expandafter\let\csname #1footgroup\endcsname=\parafootgroup
2572   \count\csname #1footins\endcsname=1000
2573   \csxdef{default@#1footins}{1000}%Use this to confine the  notes to one
     side only
2574   \dimen\csname #1footins\endcsname=\csuse{Xmaxhnotes@#1}
```

```
2575    \skip\csname #1footins\endcsname=\csuse{Xbeforenotes@#1}%
2576    \advance\skip\csname #1footins\endcsname by\csuse{Xafterrule@#1}%
2577    \para@footsetup{#1}
2578 %
```

And the extra setup for minipages.

```
2579    \ifnoledgroup@\else
2580      \expandafter\let\csname mpv#1footnote\endcsname=\mpparavfootnote
2581      \expandafter\let\csname mp#1footgroup\endcsname=\mpparafootgroup
2582      \count\csname mp#1footins\endcsname=1000
2583      \dimen\csname mp#1footins\endcsname=\csuse{Xmaxhnotes@#1}
2584      \skip\csname mp#1footins\endcsname=\csuse{Xbeforenotes@#1}%
2585      \advance\skip\csname mp#1footins\endcsname by\csuse{Xafterrule@#1}%
2586    \fi
2587 }
2588 %
```

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

```
2589 \providecommand{\footfudgefiddle}{64}
2590 %
```

\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.

```
2591 \newcommand*{\para@footsetup}[1]{{\csuse{Xbhookgroup@#1}\csuse{
     Xnotefontsize@#1}
2592    \setXnoteswidthliketwocolumns@{#1}%
2593    \ifcsempty{Xwidth@#1}%
2594      {}%
2595      {\columnwidth=\expandafter\dimexpr\csuse{Xwidth@#1}\relax}%
2596    \dimen0=\baselineskip
2597    \multiply\dimen0 by 1024
2598    \divide \dimen0 by \columnwidth \multiply\dimen0 by \footfudgefiddle\
     relax
2599    \csxdef{#1footfudgefactor}{%
2600      \expandafter\strip@pt\dimen0 }}}
2601
2602 %
```

\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.

```
2603  \newcommand*{\parafootstart}[1]{%
2604    \rightskip=0pt \leftskip=0pt%
2605    \nottoggle{Xparindent@#1}{\parindent=\z@}{}%
2606      \ifdimequal{0pt}{\Xprenotes@}{}%
2607        {%
2608        \iftoggle{Xprenotes@}{%
2609              \togglefalse{Xprenotes@}%
2610              \skip\csname #1footins\endcsname=%
2611                \glueexpr\csuse{Xprenotes@}+\csuse{Xafterrule@#1}\relax%
2612              }%
2613          {}%
2614        }%
2615    \vskip\skip\csname #1footins\endcsname%
2616    \setXnoteswidthliketwocolumns@{#1}%
2617    \setXnotespositionliketwocolumns@{#1}%
2618    \print@Xfootnoterule{#1}%
2619    \let\bidi@RTL@everypar\@empty%
2620    \noindent\leavevmode}
2621  %
```

\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.[28]

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.[29] So
changes from one language to another will not invoke the proper hyphenation rules in

---

[28] Michael Downes, 'Line Breaking in `\unhboxed` Text', *TUGboat* **11** (1990), pp. 605–612.
[29] 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.[30] 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.[31] In other words, be very careful not to use \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. 157 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.

```
2622  \newcommand*{\paravfootnote}[2]{%
2623    \csuse{Xbeforeinserting@#1}%
2624    \insert\csname #1footins\endcsname
2625    \bgroup
2626      \csuse{Xnotefontsize@#1}
2627      \footsplitskips
2628      \setbox0=\vbox{\hsize=\maxdimen%
2629        \let\bidi@RTL@everypar\@empty%
2630        \insert@Xtxtbeforenotes{#1}%
2631        \noindent\csuse{Xbhooknote@#1}%
2632        \csname #1footfmt\endcsname #2{#1}}%
2633      \setbox0=\hbox{\Xunvxh{0}{#1}}%
2634      \dp0=0pt
2635      \ht0=\csname #1footfudgefactor\endcsname\wd0
2636  %
```

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

---

[30]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.

[31]'Line Breaking', p. 610.

```
2637        \if@RTL\noindent \leavevmode\fi\box0%
2638        \penalty0
2639      \egroup}
2640
2641  %
```

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.

```
2642  \newcommand*{\mpparavfootnote}[2]{%
2643    \global\setbox\@nameuse{mp#1footins}\vbox{%
2644        \unvbox\@nameuse{mp#1footins}%
2645        \csuse{Xnotefontsize@#1}
2646        \footsplitskips
2647        \setbox0=\vbox{\hsize=\maxdimen%
2648          \let\bidi@RTL@everypar\@empty%
2649          \insert@Xtxtbeforenotes{#1}%
2650          \noindent\color@begingroup%
2651          \csuse{Xbhooknote@#1}%
2652          \csname #1footfmt\endcsname #2{#1}\color@endgroup}%
2653        \setbox0=\hbox{\Xunvxh{0}{#1}}%
2654        \dp0=\z@
2655        \ht0=\csname #1footfudgefactor\endcsname\wd0
2656        \box0
2657        \penalty0
2658  }}
2659
2660  %
```

\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.

```
2661  \newcommand*{\Xunvxh}[2]{%
2662    \setbox0=\vbox{\unvbox#1%
2663        \global\setbox1=\lastbox}%
2664    \unhbox1
```

```
2665    \unskip              % remove \rightskip,
2666    \unskip              % remove \parfillskip,
2667    \unpenalty           % remove \penalty of 10000,
2668    \hskip\csuse{Xafternote@#2}\relax}% but add the glue to go between the
        notes
2669
2670 %
```

\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).

```
2671 \newcommand*{\parafootfmt}[4]{%
2672    \Xinsertparafootsep{#4}%
2673    \ledsetnormalparstuff@common%
2674    \printlinefootnote{#1}{#4}%
2675    \print@lemma{#1}{#2}{#4}%
2676    \csuse{Xwrapcontent@#4}{#3}%
2677    \penalty-10 }
2678 %
```

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 \paravfootnote.
                       The call to \Xnotefontsize@⟨s⟩ is to ensure that the correct \baselineskip for
                  the footnotes is used. The argument is the note series letter.

```
2679 \newcommand*{\parafootgroup}[1]{%
2680    \hsize=\expandafter\dimexpr\csuse{Xwidth@#1}\relax%
2681    \unvbox\csname #1footins\endcsname
2682    \ifcsstring{Xragged@#1}{L}{\RaggedLeft}{}%
2683    \ifcsstring{Xragged@#1}{R}{\RaggedRight}{}%
2684    \makehboxofhboxes
2685    \setbox0=\hbox{\unhbox0 \removehboxes}%
2686    \csuse{Xbhookgroup@#1}%
2687    \csuse{Xnotefontsize@#1}%
2688    \unhbox0\par%
2689    \global\hsize=\old@hsize%
2690    }%
2691
2692 %
```

\mpparafootgroup   The minipage version.

```
2693 \newcommand*{\mpparafootgroup}[1]{{%
2694   \setXnoteswidthliketwocolumns@{#1}%
2695   \vskip\skip\@nameuse{mp#1footins}
2696   \ifl@dpairing\ifparledgroup%
2697     \leavevmode\marks\parledgroup@{begin}%
2698     \marks\parledgroup@series{#1}%
2699     \marks\parledgroup@type{Xfootnote}%
2700   \fi\fi\normalcolor
2701   \ifparledgroup%
2702     \ifl@dpairing%
2703     \else%
2704       \setXnoteswidthliketwocolumns@{#1}%
2705       \setXnotespositionliketwocolumns@{#1}%
2706       \print@Xfootnoterule{#1}%%
2707     \fi%
2708   \else%
2709       \setXnoteswidthliketwocolumns@{#1}%
2710       \setXnotespositionliketwocolumns@{#1}%
2711       \print@Xfootnoterule{#1}%
2712   \fi%
2713   \unvbox\csname mp#1footins\endcsname
2714   \ifcsstring{Xragged@#1}{L}{\RaggedLeft}{}%
2715   \ifcsstring{Xragged@#1}{R}{\RaggedRight}{}%
2716   \makehboxofhboxes
2717   \setbox0=\hbox{\unhbox0 \removehboxes}%
2718   \csuse{Xbhookgroup@#1}%
2719   \csuse{Xnotefontsize@#1}%
2720   \nottoggle{Xparindent@#1}{\parindent=\z@}{}%
2721   \unhbox0\par}}
2722
2723 %
```

And finally, the two macros which are required to transform the long horizontal box stored in the insert' box to a printable text.

```
\makehboxofhboxes 24 \newcommand*{\makehboxofhboxes}{\setbox0=\hbox{}%
    \removehboxes 25   \loop
2726       \unpenalty
2727       \setbox2=\lastbox
2728     \ifhbox2
2729       \setbox0=\hbox{\box2\unhbox0}%
2730     \repeat}
2731
2732 \newcommand*{\removehboxes}{\setbox0=\lastbox
2733   \ifhbox0{\removehboxes}\unhbox0 \fi}
2734
2735 %
```

Insertion of the footnotes separator   The command \Xinsertparafootsep{⟨*series*⟩} must be called at the beginning of \parafootftm.

```
\prevpage@num 2736  \newcommand{\Xinsertparafootsep}[1]{%
\Xinsertparafootsep 2737    \ifledRcol@%
               2738      \ifnumequal{\csuse{#1prevpage@numR}}{\page@numR}%
               2739          {\ifcsdef{prevline#1}% Be sur \prevline#1 exists.
               2740        {\ifnumequal{\csuse{prevline#1}}{\line@numR}%
               2741          {\ifcsempty{Xsymlinenum@#1}{\csuse{Xparafootsep@#1}}{}}%
               2742          {\csuse{Xparafootsep@#1}}%
               2743        }%
               2744        {\csuse{Xparafootsep@#1}}%
               2745      }%
               2746      {}%
               2747      \global\csname #1prevpage@numR\endcsname=\page@numR%
               2748    \else%
               2749      \ifnumequal{\csuse{#1prevpage@num}}{\page@num}%
               2750          {\ifcsdef{prevline#1}% Be sur \prevline#1 exists.
               2751        {\ifnumequal{\csuse{prevline#1}}{\line@num}%
               2752          {\ifcsempty{Xsymlinenum@#1}{\csuse{Xparafootsep@#1}}{}}%
               2753          {\csuse{Xparafootsep@#1}}%
               2754        }%
               2755        {\csuse{Xparafootsep@#1}}%
               2756      }%
               2757      {}%
               2758      \global\csname #1prevpage@num\endcsname=\page@num%
               2759    \fi%
               2760  }
               2761  %
```

XII.6.4   Columnar footnotes

Common tools

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

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

We do not call directly \rigidbalance, but we call \Xrigidbalance for critical notes and \rigidbalanceX for familiar notes. Both of them call \rigidbalance.

```
2762  \newcount\@k \newdimen\@h
2763  \newcommand*{\Xrigidbalance}[3]{%
2764      \hsize=\expandafter\dimexpr\csuse{Xwidth@\@currentseries}\relax%
```

```
2765        \rigidbalance{#1}{#2}{#3}%
2766 }%
2767
2768 \newcommand*{\rigidbalanceX}[3]{%
2769      \hsize=\expandafter\dimexpr\csuse{widthX@\@currentseries}\relax%
2770      \rigidbalance{#1}{#2}{#3}%
2771 }%
2772
2773 \newcommand*{\rigidbalance}[3]{%
2774   \setbox0=\box#1 \@k=#2 \@h=#3%
2775   \@@line{\splittopskip=\@h \vbadness=\@M \hfilneg
2776   \valign{##\vfil\cr\dosplits}}}
2777
2778 \newcommand*{\dosplits}{\ifnum\@k>0 \noalign{\hfil}\splitoff
2779   \global\advance\@k-1\cr\dosplits\fi}
2780
2781 \newcommand*{\splitoff}{\dimen0=\ht0
2782   \divide\dimen0 by\@k \advance\dimen0 by\@h
2783   \setbox2 \vsplit0 to \dimen0
2784   \unvbox2 }
2785
2786 %
```

Three columns

```
2787 \newcommand*{\Xarrangement@threecol}[1]{%
2788   \csgdef{series@display#1}{threecol}
2789   \expandafter\let\csname v#1footnote\endcsname=\threecolvfootnote
2790   \expandafter\let\csname #1footfmt\endcsname=\threecolfootfmt
2791   \expandafter\let\csname #1footgroup\endcsname=\threecolfootgroup
2792   \dimen\csname #1footins\endcsname=\csuse{Xmaxhnotes@#1}%
2793   \skip\csname #1footins\endcsname=\csuse{Xbeforenotes@#1}%
2794   \advance\skip\csname #1footins\endcsname by\csuse{Xafterrule@#1}%
2795   \threecolfootsetup{#1}
2796 %
```

The additional setup for minipages.

```
2797   \ifnoledgroup@\else
2798     \expandafter\let\csname mpv#1footnote\endcsname=\mpnormalvfootnote
2799     \expandafter\let\csname mp#1footgroup\endcsname=\mpthreecolfootgroup
2800     \skip\csname mp#1footins\endcsname=\csuse{Xbeforenotes@#1}%
2801     \advance\skip\csname mp#1footins\endcsname by\csuse{Xafterrule@#1}%
2802     \mpthreecolfootsetup{#1}
2803   \fi
2804 }
2805
2806 %
```

The \footstart and \footnoterule macros for these notes assume the normal values
(XII.6.2 p. 157 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.

```
2807 \newcommand*{\threecolfootsetup}[1]{%
2808   \count\csname #1footins\endcsname 333
2809   \csxdef{default@#1footins}{333}%Use this to confine the  notes to one
side only
2810   \multiply\dimen\csname #1footins\endcsname \thr@@}
2811 %
```

\mpthreecolfootsetup   The setup for minipages.

```
2812 \newcommand*{\mpthreecolfootsetup}[1]{%
2813   \count\csname mp#1footins\endcsname 333
2814   \multiply\dimen\csname mp#1footins\endcsname \thr@@}
2815
2816 %
```

\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).

```
2817 \notbool{parapparatus@}{\newcommand*}{\newcommand}{\threecolvfootnote}[2]{%
2818   \csuse{Xbeforeinserting@#1}%
2819   \insert\csname #1footins\endcsname\bgroup%
2820   \hsize=\expandafter\dimexpr\csuse{Xwidth@#1}\relax%
2821   \noindent\csuse{Xbhooknote@#1}%
2822   \csuse{Xnotefontsize@#1}%
2823   \footsplitskips%
```

```
2824    \csname #1footfmt\endcsname #2{#1}\egroup}
2825 %
```

\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.

```
2826 \notbool{parapparatus@}{\newcommand*}{\newcommand}{\threecolfootfmt}[4]{%
2827    \normal@pars%
2828    \hsize \csuse{Xhsizethreecol@#4}%
2829    \nottoggle{Xparindent@#4}{\parindent=\z@}{}%
2830    \tolerance=5000%
2831    \hangindent=\csuse{Xhangindent@#4}%
2832    \everypar{\hangindent=\csuse{Xhangindent@#4}}%
2833    \@tempdima=\parindent%
2834    \csuse{Xcolalign@#4}%
2835    \parindent=\@tempdima%
2836    \strut{%
2837    \hspace{\parindent}%
2838    \printlinefootnote{#1}{#4}%
2839    }%
2840    \print@lemma{#1}{#2}{#4}%
2841    \csuse{Xwrapcontent@#4}{#3}%
2842    \strut\par\allowbreak}
2843 %
```

\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.

```
2844 \newcommand*{\threecolfootgroup}[1]{%
2845    \csuse{Xbhookgroup@#1}\par%
2846    \splittopskip=\ht\strutbox
2847    \expandafter
2848    \Xrigidbalance\csname #1footins\endcsname \thr@@ \splittopskip}
2849 %
```

\mpthreecolfootgroup   The setup for minipages.

```
2850 \newcommand*{\mpthreecolfootgroup}[1]{{%
2851    \vskip\skip\@nameuse{mp#1footins}
2852    \ifl@dpairing\ifparledgroup%
2853       \leavevmode\marks\parledgroup@{begin}%
```

```
2854        \marks\parledgroup@series{#1}%
2855        \marks\parledgroup@type{Xfootnote}%
2856      \fi\fi\normalcolor
2857      \ifparledgroup%
2858        \ifl@dpairing%
2859        \else%
2860          \setXnoteswidthliketwocolumns@{#1}%
2861          \setXnotespositionliketwocolumns@{#1}%
2862          \print@Xfootnoterule{#1}%
2863        \fi%
2864      \else%
2865        \setXnoteswidthliketwocolumns@{#1}%
2866        \setXnotespositionliketwocolumns@{#1}%
2867        \print@Xfootnoterule{#1}%
2868      \fi%
2869      \csuse{Xbhookgroup@#1}\par%
2870      \splittopskip=\ht\strutbox
2871      \expandafter
2872      \Xrigidbalance\csname mp#1footins\endcsname \thr@@ \splittopskip}}
2873
2874 %
```

Two columns

```
\Xarrangement@twocol 2875 \newcommand*{\Xarrangement@twocol}[1]{%
2876      \csgdef{series@display#1}{twocol}
2877      \expandafter\let\csname v#1footnote\endcsname=\twocolvfootnote
2878      \expandafter\let\csname #1footfmt\endcsname=\twocolfootfmt
2879      \expandafter\let\csname #1footgroup\endcsname=\twocolfootgroup
2880      \dimen\csname #1footins\endcsname=\csuse{Xmaxhnotes@#1}%
2881      \skip\csname #1footins\endcsname=\csuse{Xbeforenotes@#1}%
2882      \advance\skip\csname #1footins\endcsname by\csuse{Xafterrule@#1}%
2883      \twocolfootsetup{#1}
2884 %
```

The additional setup for minipages.

```
2885      \ifnoledgroup@\else
2886        \expandafter\let\csname mpv#1footnote\endcsname=\mpnormalvfootnote
2887        \expandafter\let\csname mp#1footgroup\endcsname=\mptwocolfootgroup
2888        \skip\csname mp#1footins\endcsname=\csuse{Xbeforenotes@#1}%
2889        \advance\skip\csname mp#1footins\endcsname by\csuse{Xafterrule@#1}%
2890        \mptwocolfootsetup{#1}
2891      \fi
2892 }
2893
2894 %
```

\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

```
2895  \newcommand*{\twocolfootsetup}[1]{%
2896    \count\csname #1footins\endcsname 500
2897    \csxdef{default@#1footins}{500}%
2898    \multiply\dimen\csname #1footins\endcsname \tw@}
2899  %
```

```
2900  \notbool{parapparatus@}{\newcommand*}{\newcommand}{\twocolvfootnote}[2]{%
2901    \csuse{Xbeforeinserting@#1}%
2902    \insert\csname #1footins\endcsname\bgroup%
2903    \hsize=\expandafter\dimexpr\csuse{Xwidth@#1}\relax%
2904    \noindent\csuse{Xbhooknote@#1}%
2905    \csuse{Xnotefontsize@#1}%
2906    \footsplitskips%
2907    \csname #1footfmt\endcsname #2{#1}\egroup}
2908  %
```

```
2909  \notbool{parapparatus@}{\newcommand*}{\newcommand}{\twocolfootfmt}[4]{% 4th
      arg is optional, for backward compatibility
2910    \normal@pars%
2911    \hsize \csuse{Xhsizetwocol@#4}%
2912    \nottoggle{Xparindent@#4}{\parindent=\z@}{}%
2913    \tolerance=5000%
2914    \hangindent=\csuse{Xhangindent@#4}%
2915    \everypar{\hangindent=\csuse{Xhangindent@#4}}%
2916  \@tempdima=\parindent%
2917    \csuse{Xcolalign@#4}%
2918  \parindent=\@tempdima%
2919    \strut{%
2920  \hspace{\parindent}%
2921  \printlinefootnote{#1}{#4}%
2922  }%
2923  \print@lemma{#1}{#2}{#4}%
2924  \csuse{Xwrapcontent@#4}{#3}%
2925  \strut\par\allowbreak}
2926  %
```

```
2927  \newcommand*{\twocolfootgroup}[1]{%
2928    \csuse{Xbhookgroup@#1}\par%
2929    \splittopskip=\ht\strutbox
2930    \expandafter
2931    \Xrigidbalance\csname #1footins\endcsname \tw@ \splittopskip}
2932
2933  %
```

\mptwocolfootsetup The versions for minipages.
\mptwocolfootgroup

```
2934  \newcommand*{\mptwocolfootsetup}[1]{%
2935    \count\csname mp#1footins\endcsname 500
2936    \multiply\dimen\csname mp#1footins\endcsname \tw@}
2937  %

2938  \newcommand*{\mptwocolfootgroup}[1]{{%
2939    \vskip\skip\@nameuse{mp#1footins}
2940    \ifl@dpairing\ifparledgroup%
2941      \leavevmode\marks\parledgroup@{begin}%
2942      \marks\parledgroup@series{#1}%
2943      \marks\parledgroup@type{Xfootnote}%
2944    \fi\fi\normalcolor
2945    \ifparledgroup%
2946      \ifl@dpairing%
2947      \else%
2948        \setXnoteswidthliketwocolumns@{#1}%
2949        \setXnotespositionliketwocolumns@{#1}%
2950        \print@Xfootnoterule{#1}%
2951      \fi%
2952    \else%
2953      \setXnoteswidthliketwocolumns@{#1}%
2954      \setXnotespositionliketwocolumns@{#1}%
2955      \print@Xfootnoterule{#1}%
2956    \fi%
2957    \csuse{Xbhookgroup@#1}\par%
2958    \splittopskip=\ht\strutbox
2959    \expandafter
2960    \Xrigidbalance\csname mp#1footins\endcsname \tw@ \splittopskip}}
2961
2962  %
```

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   The fonts that are used for printing notes might not have the character mapping we
\fullstop   expect: for example, the Computer Modern font that contains old-style numerals does
\rbracket   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 foot-notes.

With polyglossia, each critical note has a \footnote@lang which shows the lan-guage of the lemma, and which can be used to switch the bracket from right to left.

```
2963  \def\endashchar{\textnormal{--}}
2964
2965  \newcommand*{\fullstop}{\textnormal{.}}
2966  \def\Xsublinesep@side{\fullstop}
2967
2968  \newcommand*{\rbracket}{\textnormal{%
2969      \csuse{text\csuse{footnote@lang}}{%
2970          \ifluatex%
2971          \ifdefstring{\footnote@luatextextdir}{TRT}{\thinspace[}{\thinspace
      ]}%
2972              \else%
2973          \thinspace]%
2974          \fi}%
2975      }%
2976  }
2977
2978  %
```

XII.7.2   Pstart number in footnote

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

```
2979  \newcommand{\printpstart}[0]{%
2980      \ifboolexpr{bool{l@dpairing} or bool{l@dprintingpages} or bool{
      l@dprintingcolumns}}{%
2981          \ifledRcol%
2982              \thepstartR%
2983          \else%
2984              \thepstartL%
2985          \fi%
2986      }{%
2987          \thepstart%
2988      }%
2989  }
2990  %
```

XII.7.3   Lemma printing

\print@lemma   \print@lemma is called inside critical footnotes to print the lemma and the lemma sep-arator (#1: line number and font information, #2: lemma, #3: series).

```
2991  %
2992  \newcommand{\print@lemma}[3]{%
2993    \bgroup%
2994      \nottoggle{Xlemmadisablefontselection@#3}%
```

```
2995          {\select@lemmafont#1|}%
2996          {}%
2997      \bgroup%
2998          \csuse{Xlemmafont@#3}%Deprecated
2999          \csuse{Xwraplemma@#3}{#2}%
3000      \egroup%
3001    \egroup%
3002    \iftoggle{nosep@}{%
3003      \hskip\csuse{Xinplaceoflemmaseparator@#3}%
3004      \relax%
3005      }%
3006      {\ifcsempty{Xlemmaseparator@#3}%
3007        {%
3008          \hskip\csuse{Xinplaceoflemmaseparator@#3}%
3009          \relax%
3010        }%
3011        {%
3012          \nobreak%
3013          \hskip\csuse{Xbeforelemmaseparator@#3}%
3014          \csuse{Xlemmaseparator@#3}%
3015          \hskip\csuse{Xafterlemmaseparator@#3}%
3016          \relax%
3017        }%
3018    }%
3019 }%
3020 %
```

XII.7.4   Line number printing

\printlinefootnote   The \printlinefootnote macro is called in each \<type>footfmt command. It con-
trols 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.

```
3021 \newcommand{\printlinefootnote}[2]{%
3022    \l@dp@rsefootspec#1|%
3023    \iftoggle{Xnumberonlyfirstintwolines@#2}{%
3024        \edef\lineinfo@{\l@dparsedstartline - \l@dparsedstartsub - \
       l@dparsedendline - \l@dparsedendsub}%
3025        }%
3026        {%
3027        \edef\lineinfo@{\l@dparsedstartline - \l@dparsedstartsub}%
3028        }%
3029    \iftoggle{nonum@}{%Try if the line number must printed for this specific
       not (by default, yes)
3030    \hspace{\csuse{Xinplaceofnumber@#2}}%
3031    }%
3032    {%
3033        {%
```

```
3034              \iftoggle{Xnonumber@#2}%Try if the line number must printed (by
      default, yes)
3035                {%
3036              \hspace{\csuse{Xinplaceofnumber@#2}}%
3037                }%
3038                {%
3039                  {\iftoggle{Xnumberonlyfirstinline@#2}% If for this series the
      line number must be printed only in the first time.
3040                    {%
3041                    \ifcsdef{prevline#2}%
3042                      {%Be sure the \prevline exists.
3043                      \ifcsequal{prevline#2}{lineinfo@}%Try it
3044                        {%
3045                        \ifcsempty{Xsymlinenum@#2}%
3046                         {%
3047                              \hspace{\csuse{Xinplaceofnumber@#2}}%
3048                         }%
3049                         {\printsymlinefootnotearea{#2}}%
3050                        }%
3051                        {%
3052                          \printlinefootnotearea{#1}{#2}%
3053                        }%
3054                      }%
3055                      {%
3056                      \printlinefootnotearea{#1}{#2}%
3057                      }%
3058                }%
3059                  {%
3060                  \printlinefootnotearea{#1}{#2}%
3061                  }%
3062                  \csxdef{prevline#2}{\lineinfo@}%
3063                }%
3064            }%
3065          }%
3066        }%
3067 }
3068 %
```

\printsymlinefootnotearea   This macro prints the space before the line symbol, changes the font, when prints the line symbol and the space after it.

```
3069 \newcommand{\printsymlinefootnotearea}[1]{%
3070   \hspace{\csuse{Xbeforesymlinenum@#1}}%
3071   \csuse{Xnotenumfont@#1}%
3072   \ifdimequal{\csuse{Xboxsymlinenum@#1}}{\z@}%
3073     {\csuse{Xsymlinenum@#1}}%
3074     {\hbox to \csuse{Xboxsymlinenum@#1}%
3075       {\csuse{Xsymlinenum@#1}\hfill}%
3076     }%
3077   \hspace{\csuse{Xaftersymlinenum@#1}}%
```

```
3078 }%
3079 %
```

\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.)

```
3080 \newcommand{\printlinefootnotearea}[2]{%
3081   \printXbeforenumber{#2}%
3082   \csuse{Xnotenumfont@#2}%
3083   \boxfootnotenumbers{#1}{#2}%
3084   \printXafternumber{#2}%
3085 }%
3086 %
```

\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.

```
3087 \newcommand{\boxfootnotenumbers}[2]{%
3088   \ifdimequal{\csuse{Xboxlinenum@#2}}{0pt}{%
3089       \printlinefootnotenumbers{#1}{#2}%
3090     }%
3091     {%
3092       \hbox to \csuse{Xboxlinenum@#2}%
3093         {%
3094         \IfSubStr{RC}{\csuse{Xboxlinenumalign@#2}}{\hfill}{}%
3095         \printlinefootnotenumbers{#1}{#2}%
3096         \IfSubStr{LC}{\csuse{Xboxlinenumalign@#2}}{\hfill}{}%
3097         }%
3098     }%
3099 }%
3100 %
```

\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.

```
3101 \newcommand{\printlinefootnotenumbers}[2]{%
3102   \xdef\@currentseries{#2}%
3103   \ifboolexpr{%
3104    (togl{Xpstart@#2} and bool{numberpstart})%
3105    or togl{Xpstarteverytime@#2}}%
3106   {\printpstart}{}%
3107   \iftoggle{Xstanza@#2}{%
3108     \ifnumberstanza%
3109       \printstanza%
3110       \csuse{Xstanzaseparator@#2}%
```

```
3111      \fi%
3112    }{}%
3113    \iftoggle{Xonlypstart@#2}{}{%
3114      \csuse{Xtxtbeforenumber@#2}%
3115      \printlines#1|\ifledRcol@\@Rlineflag\fi|}%
3116 }%
3117 %
```

\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.)

```
3118 \newcommand{\printXbeforenumber}[1]{%
3119    \hspace{\csuse{Xbeforenumber@#1}}%
3120 }%
3121 %
```

\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

```
3122 \newcommand{\printXafternumber}[1]{%
3123    \iftoggle{Xnonbreakableafternumber@#1}{\nobreak}{}%
3124    \hspace{\csuse{Xafternumber@#1}}%
3125 }%
3126 %
```

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 com-
plicated 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. 92: 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 27   \newif\ifl@d@pnum
\ifl@d@ssub 28   \newif\ifl@d@ssub
\ifl@d@elin 29   \newif\ifl@d@elin
 \ifl@d@esl 30   \newif\ifl@d@esl
\ifl@d@dash 31   \newif\ifl@d@dash
\ifl@d@Xtwolines 3132   \newif\ifl@d@Xtwolines%
\ifl@d@Xmorethantwolines 3133   \newif\ifl@d@Xmorethantwolines%
                 3134 %
```

\l@dp@rsefootspec   \l@dp@rsefootspec parses lines specification and defines macros which hold the nu-
\l@dparsedstartpage   meric values. Just a reminder of the arguments:
\l@dparsedstartline   \printlines   #1 | #2 | #3 | #4 | #5 | #6 | #7
\l@dparsedstartsub   \printlines start-page | line | subline | end-page | line | subline | fontflag
\l@dparsedendpage
\l@dparsedendline
\l@dparsedendsub

```
3135 \def\l@dp@rsefootspec#1|#2|#3|#4|#5|#6|#7|{%
3136   \gdef\l@dparsedstartpage{#1}%
3137   \gdef\l@dparsedstartline{#2}%
3138   \gdef\l@dparsedstartsub{#3}%
3139   \gdef\l@dparsedendpage{#4}%
3140   \gdef\l@dparsedendline{#5}%
3141   \gdef\l@dparsedendsub{#6}%
3142 }
3143 %
```

Initialise the several number value macros.

```
3144 \def\l@dparsedstartpage{0}%
3145 \def\l@dparsedstartline{0}%
3146 \def\l@dparsedstartsub{0}%
3147 \def\l@dparsedendpage{0}%
3148 \def\l@dparsedendline{0}%
3149 \def\l@dparsedendsub{0}%
3150
3151 %
```

\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.

```
3152 \newcommand*{\setprintlines}[6]{%
3153   \l@d@pnumfalse \l@d@dashfalse
3154 %
```

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

```
3155   \ifbypage@
3156     \ifnum#4=#1 \else
3157       \l@d@pnumtrue
3158       \l@d@dashtrue
3159     \fi
3160   \fi
3161 %
```

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.

```
3162    \ifl@d@pnum \l@d@elintrue \else \l@d@elinfalse \fi
3163    \ifnum#2=#5 \else
3164        \l@d@elintrue
3165        \l@d@dashtrue
3166    \fi
3167 %
```

We print the starting sub-line if it is nonzero.

```
3168    \l@d@ssubfalse
3169    \ifnum#3=0 \else
3170        \l@d@ssubtrue
3171    \fi
3172 %
```

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.

```
3173    \l@d@eslfalse
3174    \ifnum#6=0 \else
3175        \ifnum#6=#3
3176            \ifl@d@elin \l@d@esltrue \else \l@d@eslfalse \fi
3177        \else
3178            \l@d@esltrue
3179            \l@d@dashtrue
3180        \fi
3181    \fi%
3182 %
```

However, if the \Xtwolines is set for the current series, we do not print the last line number.

```
3183    \ifl@d@dash%
3184        \ifboolexpr{togl{fulllines@} or test{\ifcsempty{Xtwolines@\
@currentseries}}}%
3185            {}%
3186            {%
3187            \setistwofollowinglines{#1}{#2}{#4}{#5}%
3188                \ifboolexpr{%
3189                    (%
3190                        togl {Xtwolinesbutnotmore@\@currentseries}%
3191                        and not%
3192                        (%
3193                        bool {istwofollowinglines@}%
3194                        )%
3195                    )%
3196                    or%
3197                    (%
3198                        (not test{\ifnumequal{#1}{#4}})%
3199                        and togl{Xtwolinesonlyinsamepage@\@currentseries}%
```

```
3200              )%
3201            }%
3202            {}%
3203            {%
3204            \l@d@dashfalse%
3205            \l@d@Xtwolinestrue%
3206            \l@d@elinfalse%
3207            \l@d@eslfalse%
3208            \ifcsempty{Xmorethantwolines@\@currentseries}%
3209              {}%
3210              {\ifistwofollowinglines@\else%
3211                \l@d@Xmorethantwolinestrue%
3212              \fi%
3213            }%
3214          }%
3215        }%
3216    \fi%
3217 %
```

End of \setprintlines.

```
3218 }%
3219 %
```

\setistwofollowinglines   The \ifistwofollowinglines boolean, used by the \Xtwolines and related setting,
is set to true by \setistwofollowinglines. This command takes the following argu-
ments:

- #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.

```
3220 \newif\ifistwofollowinglines@%
3221 \newcommand{\setistwofollowinglines}[4]{%
3222    \ifcsdef{lastlinenumberon@#1}%
3223      {\numdef{\tmp}{\csuse{lastlinenumberon@#1}}}%
3224      {\numdef{\tmp}{0}}%
```

```
3225      \istwofollowinglines@false%
3226      \ifnumequal{#4-#2}{1}%
3227      {\istwofollowinglines@true}%
3228      {\ifbypage@%
3229        \ifnumequal{#3-#1}{1}%
3230        {%
3231        \ifnumequal{#2}{\tmp}%
3232          {\ifnumequal{#4}{1}{\istwofollowinglines@true}{}}%
3233          {}%
3234        }%
3235        {}%
3236      \fi%
3237      }%
3238 }%
3239 %
```

\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. Arguments are 1) start page number 2) start line number 3) start subline number 4) end page number 5) end line number 6) end subline number 7) font specification 8) side flag

```
3240 \def\printlines#1|#2|#3|#4|#5|#6|#7|#8|{%
3241    \begingroup%
3242 %
```

If we use LuaTeX, ensure we use good text's direction.

```
3243    \ifluatex%
3244      \edef\@tmp{\the\textdir}%
3245      \ifdefstring{\@tmp}{TLT}{}{\textdir TLT}%Test in order to prevent
    spurious space (bug #397)
3246    \fi%
3247 %
```

Decide which part of line number components we will print.

```
3248    \setprintlines{#1}{#2}{#3}{#4}{#5}{#6}%
3249 %
```

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.

```
3250    \ifdimequal{\csuse{Xboxstartlinenum@\@currentseries}}{0pt}%
3251      {\bgroup}%
3252      {\leavevmode\hbox to \csuse{Xboxstartlinenum@\@currentseries}\bgroup\
    hfill}%
3253    \ifl@d@pnum%
3254      \wrap@edcrossref{\@this@crossref@start}{#1}%
3255      \csuse{Xsublinesep@\@currentseries}%
3256    \fi%
```

```
3257    \wrap@edcrossref{\@this@crossref@start}{%
3258      \linenumrep{#2}%
3259      \iftoggle{Xlineflag@\@currentseries}{#8}{}%
3260    }%
3261    \ifl@d@ssub%
3262      \csuse{Xsublinesep@\@currentseries}%
3263      \wrap@edcrossref{\@this@crossref@start}{\sublinenumrep{#3}}%
3264    \fi
3265    \egroup%
3266 %
```

Then print the dash + end line number, or the range symbol.

```
3267    \ifdimequal{\csuse{Xboxendlinenum@\@currentseries}}{0pt}%
3268      {\bgroup}%
3269      {\hbox to \csuse{Xboxendlinenum@\@currentseries}\bgroup}%
3270    \ifl@d@Xtwolines%
3271      \ifl@d@Xmorethantwolines%
3272        \csuse{Xmorethantwolines@\@currentseries}%
3273      \else%
3274        \csuse{Xtwolines@\@currentseries}%
3275      \fi%
3276    \else%
3277      \ifl@d@dash%
3278        \ifdefined\linerangesep@%
3279          \linerangesep@%
3280        \else%
3281          \csuse{Xlinerangeseparator@\@currentseries}%
3282        \fi%
3283      \fi%
3284      \ifl@d@pnum%
3285        \wrap@edcrossref{\@this@crossref@end}{#4}%
3286        \csuse{Xsublinesep@\@currentseries}%
3287      \fi%
3288      \ifl@d@elin%
3289        \wrap@edcrossref{\@this@crossref@end}{%
3290          \linenumrep{#5}%
3291          \iftoggle{Xlineflag@\@currentseries}{#8}{}%
3292        }%
3293      \fi%
3294      \ifl@d@esl%
3295        \ifl@d@elin%
3296          \csuse{Xsublinesep@\@currentseries}%
3297        \fi%
3298        \wrap@edcrossref{\@this@crossref@end}{\sublinenumrep{#6}}%
3299      \fi%
3300    \fi%
3301    \ifdimequal{\csuse{Xboxendlinenum@\@currentseries}}{0pt}%
3302      {}%
3303      {\hfill}%Prevent underfull hbox
3304    \egroup%
```

```
3305    \endgroup%
3306  }%
3307  %
```

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 num-
bered 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.

```
3308  \providecommand*{\multiplefootnotemarker}{3sp}
3309  \providecommand*{\multfootsep}{\textsuperscript{\normalfont,}}
3310
3311  %
```

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

```
3312  \providecommand*{\m@mmf@prepare}{%
3313    \kern-\multiplefootnotemarker
3314    \kern\multiplefootnotemarker\relax}
3315  %
```

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

```
3316  \providecommand*{\m@mmf@check}{%
3317    \ifdim\lastkern=\multiplefootnotemarker\relax
3318      \edef\@x@sf{\the\spacefactor}%
3319      \unkern
3320      \multfootsep
3321      \spacefactor\@x@sf\relax
3322    \fi}
3323
3324  %
```

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

```
3325 \@ifclassloaded{memoir}{}{%
3326 %
```

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

```
3327 \apptocmd{\@footnotetext}{\m@mmf@prepare}{}{}
3328 %
```

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

```
3329
3330 \patchcmd{\@footnotemark}
3331   {\nobreak}
3332   {\m@mmf@check
3333   \nobreak
3334   }
3335   {}{}
3336 \patchcmd{\@footnotemark}
3337   {\@makefnmark}
3338   {\@makefnmark
3339   \m@mmf@prepare
3340   }
3341   {}{}
3342 %
```

Finished the modifications for the non-memoir case.

```
3343 }
3344
3345 %
```

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.

```
3346 \pretocmd{\@footnotetext}{%
3347   \ifnumberedpar@
3348     \edtext{}{\l@dbfnote{#1}}%
3349   \else
3350   }{}{}
3351 \apptocmd{\@footnotetext}{\fi}{}{}%
3352 %
```

\l@dbfnote   \l@dbfnote adds the footnote to the insert list, and \vl@dbfnote calls the original
\vl@dbfnote   \@footnotetext.   We also patch \footnote in order to get the correct footnote
\vl@dbfnote   numbers when typesetting parallel texts. This is moved into a \get@fnmark command.
\footnote
\get@fnmark
\get@thisfootnote

```
3353
3354 \patchcmd%
3355   {\footnote}%
3356   {\stepcounter\@mpfn}%
3357   {%
3358   \ifboolexpr{bool{l@dpairing} or bool{l@dprintingpages} or bool{
      l@dprintingcolumns}}{%
3359       \global\advance\footnote@reading by \@ne%
3360       \get@thisfootnote%
3361       \get@fnmark{\thisfootnote}%
3362       \ifcsdef{footnotereading\the\footnote@reading=typeset}%
3363         {\setcounter{\@mpfn}{\csuse{footnotereading\the\footnote@reading=
      typeset}}}%
3364         {\setcounter{\@mpfn}{\footnote@reading}}%
3365     }{%
3366       \stepcounter\@mpfn%
3367     }%
3368   }%
3369   {}
3370   {}
3371
3372 \newcommand{\get@thisfootnote}{%
3373     \ifboolexpr{bool{l@dpairing} or bool{l@dprintingpages} or bool{
      l@dprintingcolumns}}{%
3374       \protected@xdef\thisfootnote{\the\footnote@reading}%
3375     }{%
3376       \protected@xdef\thisfootnote{\the\c@footnote}%
3377     }%
3378 }%
3379
3380 \newcommand{\l@dbfnote}[1]{%
3381  \get@thisfootnote%
3382   \gdef\@tag{#1\relax}%
3383     \ifledRcol%
3384       \xright@appenditem{%
3385         \ifdefined\Hy@footnote@currentHref%
3386           \noexpand\def\noexpand\Hy@footnote@currentHref{\
      Hy@footnote@currentHref}%
3387         \fi%
3388         \noexpand\vl@dbfnote{{\expandonce\@tag}}{\thisfootnote}%
3389       }%
3390                         \to\inserts@listR
3391       \global\advance\insert@countR \@ne%
3392     \else%
3393       \xright@appenditem{%
```

```
3394              \ifdefined\Hy@footnote@currentHref%
3395                \noexpand\def\noexpand\Hy@footnote@currentHref{\
      Hy@footnote@currentHref}%
3396              \fi%
3397              \noexpand\vl@dbfnote{{\expandonce\@tag}}{\thisfootnote}%
3398          }%
3399                              \to\inserts@list
3400        \global\advance\insert@count \@ne%
3401      \fi
3402    \ignorespaces%
3403 }%
3404
3405 \newcommand{\get@fnmark}[1]{%
3406   \ifboolexpr{bool{l@dpairing} or bool{l@dprintingpages} or bool{
      l@dprintingcolumns}}%
3407      {%
3408        \stepcounter{footnote@typeset}%
3409        \setcounter{footnote}{\c@footnote@typeset}%
3410          \immediate\write\@mainaux{%
3411            \csgdef{footnotereading#1=typeset}{\the\c@footnote@typeset}%
3412          }%
3413        \def\@thefnmark{\thefootnote}%
3414      }%
3415      {%
3416        \@namedef{@thefnmark}{#1}%
3417      }%
3418 }%
3419
3420 \newcommand{\vl@dbfnote}[2]{%
3421   \get@fnmark{#2}%
3422   \@footnotetext{#1}%
3423 }%
3424 %
```

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

```
3425 \newcommand*{\prebodyfootmark}{%
3426   \leavevmode
3427   \ifhmode
3428     \edef\@x@sf{\the\spacefactor}%
3429     \m@mmf@check
3430     \nobreak
3431   \fi}
3432 \newcommand{\postbodyfootmark}{%
```

```
3433      \m@mmf@prepare
3434      \ifhmode\spacefactor\@x@sf\fi\relax}
3435
3436  %
```

XIII.4   Footnote arrangement

XIII.4.1   User level macro

\arrangementX \arrangementX[⟨*s*⟩]{⟨*arrangement*⟩} command calls, for each series, a specific command which set many counters and commands in order to define specific arrangement.

```
3437  \newcommandx{\arrangementX}[2][1,usedefault]{%
3438      \def\do##1{%
3439        \csname arrangementX@#2\endcsname{##1}%
3440      }%
3441      \ifstrempty{#1}%
3442        {%
3443        \dolistloop{\@series}%
3444        }%
3445        {
3446        \docsvlist{#1}%
3447        }%
3448  }%
3449  %
```

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.

```
3450  \newcommand*{\normal@footnotemarkX}[1]{%
3451      \prebodyfootmark
3452      \wrapped@bodyfootmarkX{#1}%
3453      \postbodyfootmark}
3454
3455  %
```

\normalbodyfootmarkX The \normalbodyfootmarkX{⟨*series*⟩} *really* typesets the in-text marker. The style is the normal superscript.

```
3456  \newcommand*{\normalbodyfootmarkX}[1]{%
3457      \hbox{\textsuperscript{\normalfont\@nameuse{@thefnmark#1}}}}}
3458  %
```

\normalvfootnoteX \normalvfootnoteX{⟨*series*⟩}{⟨*text*⟩} does the \insert for the ⟨*series*⟩ and calls the series' \footfmt... to format the ⟨*text*⟩.

```
3459 \notbool{parapparatus@}{\newcommand*}{\newcommand}{\normalvfootnoteX}[2]{%
3460   \csuse{beforeinsertingX@#1}%
3461   \insert\@nameuse{footins#1}\bgroup
3462     \hsize=\expandafter\dimexpr\csuse{widthX@#1}\relax%
3463     \noindent\csuse{bhooknoteX@#1}%
3464     \csuse{notefontsizeX@#1}%
3465     \footsplitskips
3466     \ifl@dpairing\ifl@dpaging\else%
3467       \setnoteswidthliketwocolumnsX@{#1}%
3468     \fi\fi%
3469     \setnotesXpositionliketwocolumns@{#1}%
3470     \spaceskip=\z@skip \xspaceskip=\z@skip
3471     \csuse{\csuse{footnote@dir}}\@nameuse{footfmt#1}{#1}{#2}\egroup}
3472
3473 %
```

\mpnormalvfootnoteX   The minipage version.

```
3474 \newcommand*{\mpnormalvfootnoteX}[2]{%
3475   \get@thisfootnoteX{#1}%
3476   \get@fnmarkX{#1}{\thisfootnote}%
3477   \edef\this@footnoteX@reading{\the\csname footnote#1@reading\endcsname}%
3478   \global\setbox\@nameuse{mpfootins#1}\vbox{%
3479     \unvbox\@nameuse{mpfootins#1}
3480     \noindent\csuse{bhooknoteX@#1}%
3481     \csuse{notefontsizeX@#1}%
3482     \hsize\columnwidth
3483     \@parboxrestore
3484     \color@begingroup
3485     \@nameuse{footfmt#1}{#1}{#2}\color@endgroup}}
3486
3487 %
```

\normalfootfmtX   \normalfootfmtX{⟨series⟩}{⟨text⟩} typesets the footnote text, prepended by the marker.

```
3488 \notbool{parapparatus@}{\newcommand*}{\newcommand}{\normalfootfmtX}[2]{%
3489  \ifluatex%
3490     \textdir\footnote@luatextextdir%
3491     \pardir\footnote@luatexpardir%
3492     \par%
3493  \fi%
3494    \protected@edef\@currentlabel{%
3495        \@nameuse{@thefnmark#1}%
3496     }%
3497   \ledsetnormalparstuffX{#1}%
3498   \hangindent=\csuse{hangindentX@#1}%
3499   \everypar{\hangindent=\csuse{hangindentX@#1}}%
3500   \rule\z@\splittopskip%
3501   {{\csuse{notenumfontX@#1}\wrapped@footfootmarkX{#1}}%
3502     \csuse{wrapcontentX@#1}{#2}%
```

```
3503    \strut\par}}
3504
3505 %
```

\normalfootfootmarkX   \normalfootfootmarkX{⟨*series*⟩} is called by \normalfootfmtX to typeset the foot-
note marker in the footer before the footnote text.

```
3506 \newcommand*{\normalfootfootmarkX}[1]{%
3507    \textsuperscript{\@nameuse{@thefnmark#1}}}
3508
3509 %
```

\normalfootstartX   \normalfootstartX{⟨*series*⟩} is the ⟨*series*⟩ footnote starting macro used in the output
routine.

```
3510 \newcommand*{\normalfootstartX}[1]{%
3511    \ifdimequal{0pt}{\prenotesX@}{}%
3512      {%
3513      \iftoggle{prenotesX@}{%
3514          \togglefalse{prenotesX@}%
3515          \skip\csname footins#1\endcsname=%
3516            \glueexpr\csuse{prenotesX@}+\csuse{afterruleX@#1}\relax%
3517        }%
3518          {}%
3519      }%
3520    \vskip\skip\csname footins#1\endcsname%
3521    \leftskip=\z@
3522    \rightskip=\z@
3523    \ifl@dpairing\else%
3524       \hsize=\old@hsize%
3525    \fi%
3526    \setnoteswidthliketwocolumnsX@{#1}%
3527    \setnotesXpositionliketwocolumns@{#1}%
3528    \print@footnoteXrule{#1}%
3529 }%
3530
3531 %
```

\normalfootnoteruleX   The rule drawn before the footnote series group.

```
3532 \let\normalfootnoteruleX=\footnoterule
3533
3534 %
```

\normalfootgroupX   \normalfootgroupX{⟨*series*⟩} sends the contents of the ⟨*series*⟩ insert box to the out-
put page without alteration.

```
3535 \newcommand*{\normalfootgroupX}[1]{%
3536    \csuse{bhookgroupX@#1}%
3537    \unvbox\@nameuse{footins#1}%
```

```
3538      \hsize=\old@hsize%
3539      }%
3540
3541 %
```

\mpnormalfootgroupX   The minipage version.

```
3542 \newcommand*{\mpnormalfootgroupX}[1]{%
3543   \vskip\skip\@nameuse{mpfootins#1}
3544   \ifl@dpairing\ifparledgroup%
3545     \leavevmode\marks\parledgroup@{begin}%
3546     \marks\parledgroup@series{#1}%
3547     \marks\parledgroup@type{footnoteX}%
3548   \fi\fi\normalcolor
3549   \ifparledgroup%
3550     \ifl@dpairing%
3551     \else%
3552       \setnoteswidthliketwocolumnsX@{#1}%
3553       \setnotesXpositionliketwocolumns@{#1}%
3554       \print@footnoteXrule{#1}%
3555     \fi%
3556   \else%
3557     \setnoteswidthliketwocolumnsX@{#1}%
3558     \setnotesXpositionliketwocolumns@{#1}%
3559     \print@footnoteXrule{#1}%
3560   \fi%
3561   \csuse{bhookgroupX@#1}%
3562   \unvbox\@nameuse{mpfootins#1}}
3563
3564 %
```

\normalbfnoteX 3565

```
3566 \newcommand{\normalbfnoteX}[2]{%
3567   \get@thisfootnoteX{#1}%
3568     \ifledRcol%
3569       \ifluatex
3570         \footnotelang@lua[R]%
3571       \fi
3572       \@ifundefined{xpg@main@language}%if polyglossia
3573         {}%
3574         {\footnotelang@poly[R]}%
3575       \xright@appenditem{%
3576         \noexpand\led@set@index@fornote{#1}%
3577         \unexpanded{\def\this@footnoteX@reading}{\the\csname footnote#1
    @reading\endcsname}%
3578         \noexpand\vbfnoteX{#1}{#2}{\expandonce\thisfootnote}%
3579         \noexpand\led@reinit@index@fornote%
3580       }%
3581                                  \to\inserts@listR
```

```
3582        \global\advance\insert@countR \@ne%
3583      \else%
3584        \ifluatex
3585            \footnotelang@lua%
3586        \fi
3587        \@ifundefined{xpg@main@language}%if polyglossia
3588          {}%
3589          {\footnotelang@poly}%
3590        \xright@appenditem{%
3591          \noexpand\led@set@index@fornote{#1}%
3592          \unexpanded{\def\this@footnoteX@reading}{\the\csname footnote#1
      @reading\endcsname}%
3593          \noexpand\vbfnoteX{#1}{#2}{\expandonce\thisfootnote}%
3594          \noexpand\led@reinit@index@fornote%
3595        }%
3596                          \to\inserts@list
3597        \global\advance\insert@count \@ne%
3598      \fi
3599    \ignorespaces}
3600
3601 %
```

\get@thisfootnoteX   The macro \get@thisfootnote command just saves the footnote number in the \thisfootnote macro, depending on the use of pairing environments.

```
3602 \newcommand{\get@thisfootnoteX}[1]{%
3603     \ifboolexpr{bool{l@dpairing} or bool{l@dprintingpages} or bool{
      l@dprintingcolumns}}{%
3604        \protected@xdef\thisfootnote{\the\csname footnote#1@reading\endcsname
      }%
3605     }{%
3606        \protected@xdef\thisfootnote{\the\csname c@footnote#1\endcsname}%
3607     }%
3608 }%
3609 %
```

\vbfnoteX   This command calls the correct footnote-inserting commands.

```
3610 \newcommand{\vbfnoteX}[3]{%
3611   \get@fnmarkX{#1}{#3}%
3612   \@nameuse{regvfootnote#1}{#1}{#2}%
3613 }%
3614
3615 %
```

\get@fnmarkX   This command gets the correct footnote number when typesetting parallel texts.

```
3616 \newcommand{\get@fnmarkX}[2]{%
3617   \ifboolexpr{bool{l@dpairing} or bool{l@dprintingpages} or bool{
      l@dprintingcolumns}}%
```

```
3618      {%
3619        \stepcounter{footnote#1@typeset}%
3620        \setcounter{footnote#1}{\value{footnote#1@typeset}}%
3621        \@namedef{@thefnmark#1}{\csuse{thefootnote#1}}%
3622          \immediate\write\@mainaux{%
3623            \csgdef{footnote#1reading#2=typeset}{\the\csname c@footnote#1
      @typeset\endcsname}%
3624          }%
3625      }%
3626      {%
3627        \@namedef{@thefnmark#1}{#2}%
3628      }%
3629 }
3630 %
3631 %
```

\vnumfootnoteX
```
3632 \newcommand{\vnumfootnoteX}[2]{%
3633    \ifnumberedpar@
3634      \edtext{}{\normalbfnoteX{#1}{#2}}%
3635    \else
3636      \def\this@footnoteX@reading{\the\csname footnote#1@reading\endcsname}%
3637      \get@thisfootnoteX{#1}%
3638      \get@fnmarkX{#1}{\expandonce\thisfootnote}%
3639      \@nameuse{regvfootnote#1}{#1}{#2}%
3640    \fi}
3641
3642 %
```

arrangementX@normal
\arrangementX@normal{⟨*series*⟩} initialises the settings for the ⟨*series*⟩ footnotes. This should always be called for each series.

```
3643 \newcommand*{\arrangementX@normal}[1]{%
3644    \csgdef{series@displayX#1}{normal}
3645    \expandafter\let\csname footstart#1\endcsname=\normalfootstartX
3646    \expandafter\newcount\csname prevpage#1@num\endcsname%
3647    \expandafter\newcount\csname prevpage#1@numR\endcsname%
3648    \@namedef{@footnotemark#1}{\normal@footnotemarkX{#1}}
3649    \@namedef{bodyfootmark#1}{\normalbodyfootmarkX{#1}}
3650    \expandafter\let\csname regvfootnote#1\endcsname=\normalvfootnoteX
3651    \expandafter\let\csname vfootnote#1\endcsname=\vnumfootnoteX
3652    \expandafter\let\csname footfmt#1\endcsname=\normalfootfmtX
3653    \@namedef{footfootmark#1}{\normalfootfootmarkX{#1}}
3654    \expandafter\let\csname footgroup#1\endcsname=\normalfootgroupX
3655    \expandafter\let\csname footnoterule#1\endcsname=\normalfootnoteruleX
3656    \count\csname footins#1\endcsname=1000
3657    \csxdef{default@footins#1}{1000}%Use to have note only for one side
3658    \dimen\csname footins#1\endcsname=\csuse{maxhnotesX@#1}
3659    \skip\csname footins#1\endcsname=\csuse{beforenotesX@#1}%
3660    \advance\skip\csname footins#1\endcsname by\csuse{afterruleX@#1}%
```

```
3661 %
```

Additions for minipages.

```
3662   \ifnoledgroup@\else%
3663     \expandafter\let\csname mpvfootnote#1\endcsname=\mpnormalvfootnoteX
3664     \expandafter\let\csname mpfootgroup#1\endcsname=\mpnormalfootgroupX
3665     \count\csname mpfootins#1\endcsname=1000
3666     \dimen\csname mpfootins#1\endcsname=\csuse{maxhnotesX@#1}
3667     \skip\csname mpfootins#1\endcsname=\csuse{beforenotesX@#1}%
3668     \advance\skip\csname mpfootins#1\endcsname by\csuse{afterruleX@#1}%
3669   \fi
3670 }
3671
3672 %
```

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`
```
3673 \newcommand*{\arrangementX@twocol}[1]{%
3674   \csgdef{series@displayX#1}{twocol}
3675   \expandafter\let\csname regvfootnote#1\endcsname=\twocolvfootnoteX
3676   \expandafter\let\csname footfmt#1\endcsname=\twocolfootfmtX
3677   \expandafter\let\csname footgroup#1\endcsname=\twocolfootgroupX
3678   \dimen\csname footins#1\endcsname=\csuse{maxhnotesX@#1}%
3679   \skip\csname footins#1\endcsname=\csuse{beforenotesX@#1}%
3680   \advance\skip\csname footins#1\endcsname by \csuse{afterruleX@#1}\relax%
3681   \twocolfootsetupX{#1}
3682   \ifnoledgroup@\else%
3683     \expandafter\let\csname mpvfootnote#1\endcsname=\mpnormalvfootnoteX
3684     \expandafter\let\csname mpfootgroup#1\endcsname=\mptwocolfootgroupX
3685     \skip\csname mpfootins#1\endcsname=\csuse{beforenotesX@#1}%
3686     \advance\skip\csname mpfootins#1\endcsname by\csuse{afterruleX@#1}
3687     \mptwocolfootsetupX{#1}
3688   \fi%
3689 }
3690
3691 %
```

`\twocolfootsetupX`
`\mptwocolfootsetupX`

`\twocolfootsetupX{`⟨*series*⟩`}`

```
3692 \newcommand*{\twocolfootsetupX}[1]{%
3693   \count\csname footins#1\endcsname 500
3694   \csxdef{default@footins#1}{500}%Use this to confine the  notes to one
      side only
3695   \multiply\dimen\csname footins#1\endcsname by \tw@}
3696 \newcommand*{\mptwocolfootsetupX}[1]{%
3697   \count\csname mpfootins#1\endcsname 500
```

```
3698     \multiply\dimen\csname mpfootins#1\endcsname by \tw@}
3699
3700 %
```

\twocolvfootnoteX   \twocolvfootnoteX{⟨*series*⟩}

```
3701 \notbool{parapparatus@}{\newcommand*}{\newcommand}{\twocolvfootnoteX}[2]{%
3702   \csuse{beforeinsertingX@#1}%
3703   \insert\csname footins#1\endcsname\bgroup%
3704     \hsize=\expandafter\dimexpr\csuse{widthX@#1}\relax%
3705     \noindent\csuse{bhooknoteX@#1}%
3706     \csuse{notefontsizeX@#1}%
3707     \footsplitskips%
3708     \spaceskip=\z@skip \xspaceskip=\z@skip%
3709     \@nameuse{footfmt#1}{#1}{#2}\egroup}
3710
3711 %
```

\twocolfootfmtX   \twocolfootfmtX{⟨*series*⟩}

```
3712 \notbool{parapparatus@}{\newcommand*}{\newcommand}{\twocolfootfmtX}[2]{%
3713     \protected@edef\@currentlabel{%
3714         \@nameuse{@thefnmark#1}%
3715     }%
3716   \normal@pars%
3717   \hangindent=\csuse{hangindentX@#1}%
3718   \everypar{\hangindent=\csuse{hangindentX@#1}}%
3719   \hsize \csuse{hsizetwocolX@#1}%
3720 \nottoggle{parindentX@#1}{\parindent=\z@}{}%
3721   \tolerance=5000\relax%
3722   \par%
3723 \@tempdima=\parindent%
3724 \csuse{colalignX@#1}%
3725 \parindent=\@tempdima%
3726   {\hspace{\parindent}%
3727   \csuse{notenumfontX@#1}\wrapped@footfootmarkX{#1}\strut%
3728     \csuse{wrapcontentX@#1}{#2}%
3729     \strut\par}%
3730 \allowbreak%
3731 }%
3732
3733 %
```

\twocolfootgroupX   \twocolfootgroupX{⟨*series*⟩}
\mptwocolfootgroupX

```
3734 \newcommand*{\twocolfootgroupX}[1]{{\csuse{bhookgroupX@#1}\csuse{
notefontsizeX@#1}
3735   \splittopskip=\ht\strutbox
3736   \expandafter
3737   \rigidbalanceX\csname footins#1\endcsname \tw@ \splittopskip}}
```

```
3738
3739 \newcommand*{\mptwocolfootgroupX}[1]{{%
3740   \vskip\skip\@nameuse{mpfootins#1}
3741   \ifl@dpairing\ifparledgroup%
3742     \leavevmode\marks\parledgroup@{begin}%
3743     \marks\parledgroup@series{#1}%
3744     \marks\parledgroup@type{footnoteX}%
3745   \fi\fi\normalcolor
3746   \ifparledgroup%
3747     \ifl@dpairing%
3748     \else%
3749       \setnoteswidthliketwocolumnsX@{#1}%
3750       \setnotesXpositionliketwocolumns@{#1}%
3751       \print@footnoteXrule{#1}%
3752     \fi%
3753   \else%
3754     \setnoteswidthliketwocolumnsX@{#1}%
3755     \setnotesXpositionliketwocolumns@{#1}%
3756     \print@footnoteXrule{#1}%
3757   \fi%
3758   \csuse{bhookgroupX@#1}%
3759   \splittopskip=\ht\strutbox
3760   \expandafter
3761   \rigidbalanceX\csname mpfootins#1\endcsname \tw@ \splittopskip}}
3762
3763 %
```

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
3764 \newcommand*{\arrangementX@threecol}[1]{%
3765   \csgdef{series@displayX#1}{threecol}
3766   \expandafter\let\csname regvfootnote#1\endcsname=\threecolvfootnoteX
3767   \expandafter\let\csname footfmt#1\endcsname=\threecolfootfmtX
3768   \expandafter\let\csname footgroup#1\endcsname=\threecolfootgroupX
3769   \dimen\csname footins#1\endcsname=\csuse{maxhnotesX@#1}%
3770   \skip\csname footins#1\endcsname=\csuse{beforenotesX@#1}%
3771   \advance\skip\csname footins#1\endcsname by \csuse{afterruleX@#1}\relax%
3772   \threecolfootsetupX{#1}
3773   \ifnoledgroup@\else%
3774     \expandafter\let\csname mpvfootnote#1\endcsname=\mpnormalvfootnoteX
3775     \expandafter\let\csname mpfootgroup#1\endcsname=\mpthreecolfootgroupX
3776     \skip\csname mpfootins#1\endcsname=\csuse{beforenotesX@#1}%
3777     \advance\skip\csname mpfootins#1\endcsname by\csuse{afterruleX@#1}
3778     \mpthreecolfootsetupX{#1}
3779   \fi%
3780 }
```

```
3781
3782 %
```

`\threecolfootsetupX`
`\mpthreecolfootsetupX`

`\threecolfootsetupX{`⟨*series*⟩`}`

```
3783 \newcommand*{\threecolfootsetupX}[1]{%
3784   \count\csname footins#1\endcsname 333
3785   \csxdef{default@footins#1}{333}%Use this to confine the  notes to one
side only
3786   \multiply\dimen\csname footins#1\endcsname by \thr@@}
3787 \newcommand*{\mpthreecolfootsetupX}[1]{%
3788   \count\csname mpfootins#1\endcsname 333
3789   \multiply\dimen\csname mpfootins#1\endcsname by \thr@@}
3790
3791 %
```

`\threecolvfootnoteX`

`\threecolvfootnoteX{`⟨*series*⟩`}{`⟨*text*⟩`}`

```
3792 \notbool{parapparatus@}{\newcommand*}{\newcommand}{\threecolvfootnoteX}[2]{
%
3793   \csuse{beforeinsertingX@#1}%
3794   \insert\csname footins#1\endcsname\bgroup%
3795     \hsize=\expandafter\dimexpr\csuse{widthX@#1}\relax%
3796     \noindent\csuse{bhooknoteX@#1}%
3797     \csuse{notefontsizeX@#1}%
3798     \footsplitskips%
3799     \@nameuse{footfmt#1}{#1}{#2}\egroup}
3800
3801 %
```

`\threecolfootfmtX`

`\threecolfootfmtX{`⟨*series*⟩`}`

```
3802 \notbool{parapparatus@}{\newcommand*}{\newcommand}{\threecolfootfmtX}[2]{%
3803     \protected@edef\@currentlabel{%
3804         \@nameuse{@thefnmark#1}%
3805     }%
3806   \hangindent=\csuse{hangindentX@#1}%
3807   \everypar{\hangindent=\csuse{hangindentX@#1}}%
3808   \normal@pars%
3809   \hsize \csuse{hsizethreecolX@#1}%
3810   \nottoggle{parindentX@#1}{\parindent=\z@}{}%
3811   \tolerance=5000\relax%
3812 \@tempdima=\parindent%
3813   \csuse{colalignX@#1}%
3814   \parindent=\@tempdima%
3815   {\hspace{\parindent}%
3816   \csuse{notenumfontX@#1}\wrapped@footfootmarkX{#1}\strut%
3817     \csuse{wrapcontentX@#1}{#2}%
3818     \strut\par}\allowbreak}
3819
3820 %
```

`\threecolfootgroupX`   `\threecolfootgroupX{`⟨*series*⟩`}`
`\mpthreecolfootgroupX`

```
3821 \newcommand*{\threecolfootgroupX}[1]{{\csuse{bhookgroupX@#1}\csuse{
     notefontsizeX@#1}
3822   \splittopskip=\ht\strutbox
3823   \expandafter
3824   \rigidbalanceX\csname footins#1\endcsname \thr@@ \splittopskip}}
3825
3826 \newcommand*{\mpthreecolfootgroupX}[1]{{%
3827   \vskip\skip\@nameuse{mpfootins#1}
3828   \ifl@dpairing\ifparledgroup
3829     \leavevmode\marks\parledgroup@{begin}%
3830     \marks\parledgroup@series{#1}%
3831     \marks\parledgroup@type{footnoteX}%
3832   \fi\fi\normalcolor
3833   \ifparledgroup%
3834     \ifl@dpairing%
3835     \else%
3836       \setnoteswidthliketwocolumnsX@{#1}%
3837       \setnotesXpositionliketwocolumns@{#1}%
3838       \print@footnoteXrule{#1}%
3839     \fi%
3840   \else%
3841     \setnoteswidthliketwocolumnsX@{#1}%
3842     \setnotesXpositionliketwocolumns@{#1}%
3843     \print@footnoteXrule{#1}%
3844   \fi%
3845   \csuse{bhookgroupX@#1}%
3846   \splittopskip=\ht\strutbox
3847   \expandafter
3848   \rigidbalanceX\csname mpfootins#1\endcsname \thr@@ \splittopskip}}
3849
3850 %
```

XIII.4.5   Paragraphed footnotes

The following macros set footnotes as one paragraph.

`\arrangementX@threecol`   `\footparagraphX{`⟨*series*⟩`}`

```
3851 \newcommand*{\arrangementX@paragraph}[1]{%
3852   \csgdef{series@displayX#1}{paragraph}%
3853   \expandafter\newcount\csname prevpage#1@num\endcsname%
3854   \expandafter\newcount\csname prevpage#1@numR\endcsname%
3855   \expandafter\let\csname footstart#1\endcsname=\parafootstartX
3856   \expandafter\let\csname regvfootnote#1\endcsname=\para@vfootnoteX
3857   \expandafter\let\csname footfmt#1\endcsname=\parafootfmtX
3858   \expandafter\let\csname footgroup#1\endcsname=\para@footgroupX
3859   \expandafter\let\csname footnoterule#1\endcsname=\normalfootnoteruleX
3860   \count\csname footins#1\endcsname=1000
```

```
3861    \csxdef{default@footins#1}{1000}%Use this to confine the  notes to one
     side only
3862    \dimen\csname footins#1\endcsname=\csuse{maxhnotesX@#1}
3863    \skip\csname footins#1\endcsname=\csuse{beforenotesX@#1}%
3864    \advance\skip\csname footins#1\endcsname by\csuse{afterruleX@#1}%
3865    \para@footsetupX{#1}
3866    \ifnoledgroup@\else
3867      \expandafter\let\csname mpvfootnote#1\endcsname=\mppara@vfootnoteX
3868      \expandafter\let\csname mpfootgroup#1\endcsname=\mppara@footgroupX
3869      \count\csname mpfootins#1\endcsname=1000
3870      \dimen\csname mpfootins#1\endcsname=\csuse{maxhnotesX@#1}
3871      \skip\csname mpfootins#1\endcsname=\csuse{beforenotesX@#1}%
3872      \advance\skip\csname mpfootins#1\endcsname by\csuse{afterruleX@#1}%
3873    \fi
3874    }
3875
3876  %
```

\para@footsetupX   \para@footsetupX{⟨*series*⟩}

```
3877  \newcommand*{\para@footsetupX}[1]{{\csuse{bhookgroupX@#1}\csuse{
     notefontsizeX@#1}
3878    \setnoteswidthliketwocolumnsX@{#1}%
3879    \ifcsempty{widthX@#1}%
3880      {}%
3881      {\columnwidth=\expandafter\dimexpr\csuse{widthX@#1}\relax}%
3882    \dimen0=\baselineskip
3883    \multiply\dimen0 by 1024
3884    \divide\dimen0 by \columnwidth \multiply\dimen0 by \footfudgefiddle\relax
     %
3885    \expandafter
3886    \xdef\csname footfudgefactor#1\endcsname{%
3887      \expandafter\strip@pt\dimen0 }}}
3888
3889  %
```

\parafootstartX   \parafootstartX{⟨*series*⟩}

```
3890  \newcommand*{\parafootstartX}[1]{%
3891      \ifdimequal{0pt}{\prenotesX@}{}%
3892        {%
3893        \iftoggle{prenotesX@}{%
3894              \togglefalse{prenotesX@}%
3895              \skip\csname footins#1\endcsname=%
3896                \glueexpr\csuse{prenotesX@}+\csuse{afterruleX@#1}\relax%
3897            }%
3898          {}%
3899        }%
3900    \leftskip=\z@
3901    \rightskip=\z@
```

```
3902      \nottoggle{parindentX@#1}{\parindent=\z@}{}%
3903      \vskip\skip\@nameuse{footins#1}%
3904      \setnoteswidthliketwocolumnsX@{#1}%
3905      \setnotesXpositionliketwocolumns@{#1}%
3906      \print@footnoteXrule{#1}%
3907      }
3908
3909 %
```

\para@vfootnoteX   \para@vfootnoteX{⟨*series*⟩}{⟨*text*⟩}
\mppara@vfootnoteX

```
3910 \newcommand*{\para@vfootnoteX}[2]{%
3911  \csuse{beforeinsertingX@#1}%
3912  \insert\csname footins#1\endcsname%
3913   \bgroup
3914     \csuse{notefontsizeX@#1}
3915     \footsplitskips
3916     \setbox0=\vbox{\hsize=\maxdimen%
3917        \let\bidi@RTL@everypar\@empty%
3918        \noindent\csuse{bhooknoteX@#1}%
3919        \@nameuse{footfmt#1}{#1}{#2}%
3920     \setbox0=\hbox{\unvxhX{0}{#1}}%
3921     \dp0=\z@
3922     \ht0=\csname footfudgefactor#1\endcsname\wd0
3923     \box0
3924     \penalty0
3925   \egroup}
3926 \newcommand*{\mppara@vfootnoteX}[2]{%
3927   \get@thisfootnoteX{#1}%
3928   \get@fnmarkX{#1}{\thisfootnote}%
3929   \edef\this@footnoteX@reading{\the\csname footnote#1@reading\endcsname}%
3930   \global\setbox\@nameuse{mpfootins#1}\vbox{%
3931     \unvbox\@nameuse{mpfootins#1}
3932     \csuse{notefontsizeX@#1}
3933     \footsplitskips
3934     \setbox0=\vbox{\hsize=\maxdimen%
3935        \let\bidi@RTL@everypar\@empty%
3936        \noindent\color@begingroup%
3937        \csuse{bhooknoteX@#1}%
3938        \@nameuse{footfmt#1}{#1}{#2}\color@endgroup}%
3939     \setbox0=\hbox{\unvxhX{0}{#1}}%
3940     \dp0=\z@
3941     \ht0=\csname footfudgefactor#1\endcsname\wd0
3942     \box0
3943     \penalty0}}
3944
3945 %
```

\unvxhX `\newcommand*{\unvxhX}[2]{%` 2th is optional for retro-compatibility

```
3947    \setbox0=\vbox{\unvbox#1%
3948      \global\setbox1=\lastbox}%
3949    \unhbox1
3950    \unskip              % remove \rightskip,
3951    \unskip              % remove \parfillskip,
3952    \unpenalty           % remove \penalty of 10000,
3953    \hskip\csuse{afternoteX@#2}%
3954    \relax}% but add the glue to go between the notes
3955
3956 %
```

\parafootfmtX   \parafootfmtX{⟨*series*⟩}

```
3957 \newcommand*{\parafootfmtX}[2]{%
3958    \protected@edef\@currentlabel{%
3959        \@nameuse{@thefnmark#1}%
3960    }%
3961    \insertparafootsepX{#1}%
3962    \ledsetnormalparstuff@common%
3963    {\csuse{notenumfontX@#1}%
3964    \csuse{notenumfontX@#1}%
3965    \wrapped@footfootmarkX{#1}%
3966    \strut%
3967    \csuse{wrapcontentX@#1}{#2}%
3968    \penalty-10}}
3969
3970 %
```

\para@footgroupX   \para@footgroupX{⟨*series*⟩}
\mppara@footgroupX

```
3971 \newcommand*{\para@footgroupX}[1]{%
3972    \hsize=\expandafter\dimexpr\csuse{widthX@#1}\relax%
3973    \unvbox\csname footins#1\endcsname
3974    \ifcsstring{raggedX@#1}{L}{\RaggedLeft}{}%
3975    \ifcsstring{raggedX@#1}{R}{\RaggedRight}{}%
3976    \makehboxofhboxes
3977    \setbox0=\hbox{\unhbox0 \removehboxes}%
3978    \csuse{bhookgroupX@#1}
3979    \csuse{notefontsizeX@#1}
3980    \unhbox0\par}
3981
3982 \newcommand*{\mppara@footgroupX}[1]{{%
3983    \setnoteswidthliketwocolumnsX@{#1}%
3984    \vskip\skip\@nameuse{mpfootins#1}
3985    \ifl@dpairing\ifparledgroup
3986      \leavevmode%
3987      \leavevmode\marks\parledgroup@{begin}%
3988      \marks\parledgroup@series{#1}%
3989      \marks\parledgroup@type{footnoteX}%
3990    \fi\fi\normalcolor
```

```
3991    \ifparledgroup%
3992      \ifl@dpairing%
3993      \else%
3994        \setnoteswidthliketwocolumnsX@{#1}%
3995        \setnotesXpositionliketwocolumns@{#1}%
3996        \print@footnoteXrule{#1}%
3997      \fi%
3998    \else%
3999        \setnoteswidthliketwocolumnsX@{#1}%
4000        \setnotesXpositionliketwocolumns@{#1}%
4001        \print@footnoteXrule{#1}%
4002    \fi%
4003    \unvbox\csname mpfootins#1\endcsname
4004    \ifcsstring{raggedX@#1}{L}{\RaggedLeft}{}%
4005    \ifcsstring{raggedX@#1}{R}{\RaggedRight}{}%
4006    \makehboxofhboxes
4007    \setbox0=\hbox{\unhbox0 \removehboxes}%
4008    \csuse{bhookgroupX@#1}%
4009    \csuse{notefontsizeX@#1}%
4010    \nottoggle{parindentX@#1}{\parindent=\z@}{}%
4011    \unhbox0\par}}
4012
4013 %
```

Insertion of the footnotes separator   The command \insertparafootsepX{⟨*series*⟩} must be called at the beginning of \parafootftmX.

```
\insertparafootsepX 4014 \newcommand{\insertparafootsepX}[1]{%
4015      \ifledRcol@%
4016        \ifnumequal{\csuse{prevpage#1@numR}}{\page@numR}%
4017          {\csuse{Xparafootsep@#1}}%
4018          {}%
4019        \global\csname prevpage#1@numR\endcsname=\page@numR%
4020      \else%
4021        \ifnumequal{\csuse{prevpage#1@num}}{\page@num}%
4022          {\csuse{Xparafootsep@#1}}%
4023          {}%
4024        \global\csname prevpage#1@num\endcsname=\page@num%
4025      \fi%
4026 }
4027 %
```

XIII.5   Wrapping footnote marks in hyperlink

\wrapped@footfootmarkX   \wrapped@footfootmarkX prints the footnote mark of the footpage, wrapped in hyperref package's commands, if needed.

```
4028 \newcommand{\wrapped@footfootmarkX}[1]{%
```

```
4029    \ifdefined\hypertarget%
4030      \hyperlink%
4031          {@bodyfootmark#1@\this@footnoteX@reading}%
4032          {\@nameuse{footfootmark#1}}%
4033      \Hy@raisedlink{%
4034        \hypertarget%
4035          {@footnotemark#1@\this@footnoteX@reading}%
4036          {}%
4037        }%
4038    \else%
4039      \@nameuse{footfootmark#1}%
4040    \fi%
4041 }%
4042 %
```

\wrapped@bodyfootmarkX   \wrapped@bodyfootmarkX prints the footnote mark of the text body, wrapped in hyperref package's commands, if needed.

```
4043 \newcommand{\wrapped@bodyfootmarkX}[1]{%
4044    \ifdefined\hypertarget%
4045      \hyperlink%
4046          {@footnotemark#1@\expandafter\the\csname footnote#1@reading\
endcsname}%
4047          {\@nameuse{bodyfootmark#1}}%
4048      \Hy@raisedlink{%
4049        \hypertarget%
4050          {@bodyfootmark#1@\expandafter\the\csname footnote#1@reading\
endcsname}%
4051          {}%
4052        }%
4053    \else%
4054      \@nameuse{bodyfootmark#1}%
4055    \fi%
4056 }%
4057 %
```

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 numbered 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.

```
dsetnormalparstuff@common  4058  \newcommand*{\ledsetnormalparstuff@common}{%
      \Xledsetnormalparstuff  4059    \ifluatex%
       \ledsetnormalparstuffX  4060      \textdir\footnote@luatextextdir%
                              4061    \pardir\footnote@luatexpardir%
                              4062    \fi%
                              4063    \csuse{\csuse{footnote@dir}}%
                              4064    \normal@pars%
                              4065    \parfillskip \z@ \@plus 1fil}%
                              4066
                              4067  \newcommand*{\Xledsetnormalparstuff}[1]{%
                              4068    \ledsetnormalparstuff@common%
                              4069    \nottoggle{Xparindent@#1}{\parindent=\z@}{\hspace{\parindent}}%
                              4070  }%
                              4071
                              4072  \newcommand*{\ledsetnormalparstuffX}[1]{%
                              4073    \ledsetnormalparstuff@common%
                              4074    \nottoggle{parindentX@#1}{\parindent=\z@}{\hspace{\parindent}}%
                              4075  }%
                              4076  %
```

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   These two commands are called at the beginning of critical or familiar notes groups.
noteswidthliketwocolumns@   They set, if the option is enabled, the \hsize. They are also called at the on the setup
oteswidthliketwocolumnsX@   for paragraphed notes.

```
                              4077
                              4078  \newdimen\old@hsize%
                              4079  \AtBeginDocument{\old@hsize=\hsize}%
                              4080
                              4081  \newcommand{\setXnoteswidthliketwocolumns@}[1]{%
                              4082    \global\let\hsize@fornote=\hsize%
                              4083    \global\old@hsize=\hsize%
                              4084    \let\old@columwidth=\columnwidth%
                              4085    \iftoggle{Xnoteswidthliketwocolumns@#1}%
                              4086      {%
                              4087      \csuse{setwidthliketwocolumns@\columns@position}%
                              4088      \global\let\hsize@fornote=\hsize%
                              4089      }%
                              4090      {}%
                              4091    \let\hsize=\hsize@fornote%
                              4092    \let\columnwidth=\old@columwidth%
                              4093  }%
                              4094
                              4095  \newcommand{\setnoteswidthliketwocolumnsX@}[1]{%
```

```
4096    \global\let\hsize@fornote=\hsize%
4097    \global\old@hsize=\hsize%
4098    \let\old@columwidth=\columnwidth%
4099    \iftoggle{noteswidthliketwocolumnsX@#1}%
4100      {%
4101      \csuse{setwidthliketwocolumns@\columns@position}%
4102      \global\let\hsize@fornote=\hsize%
4103      }%
4104      {}%
4105    \let\hsize=\hsize@fornote%
4106    \let\columnwidth=\old@columwidth%
4107 }%
4108
4109 %
```

\setXnotespositionliketwocolumns@   These two commands set the position of the critical / familiar footnotes, depending on
\setnotesXpositionliketwocolumns@   the hooks Xnoteswidthliketwocolumns and noteswidthliketwocolumnsX. They
call commands which are defined only in reledpar, because this feature has no sens
without reledpar.

```
4110 \newcommand{\setXnotespositionliketwocolumns@}[1]{%
4111    \iftoggle{Xnoteswidthliketwocolumns@#1}{%
4112      \csuse{setnotespositionliketwocolumns@\columns@position}%
4113    }{}%
4114 }%
4115
4116 \newcommand{\setnotesXpositionliketwocolumns@}[1]{%
4117    \iftoggle{noteswidthliketwocolumnsX@#1}{%
4118      \csuse{setnotespositionliketwocolumns@\columns@position}%
4119    }{}%
4120 }%
4121
4122 %
```

XVI   Footnotes' order

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

```
4123 \def\@fnpos{familiar-critical}
4124 \def\@mpfnpos{critical-familiar}
4125 \newcommand{\fnpos}[1]{\xdef\@fnpos{#1}}
4126 \newcommand{\mpfnpos}[1]{\xdef\@mpfnpos{#1}}
4127 %
```

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@Xfootnoterule₂₈   \newcommand{\print@Xfootnoterule}[1]{%
\print@footnoteXrule₂₉     \vskip-\csuse{Xafterrule@#1}%Because count in \dimen\csuse{#1footins}
               4130         \nointerlineskip%
               4131         \moveleft-\leftskip\vbox{\csuse{#1footnoterule}}%
               4132         \nointerlineskip%
               4133         \vskip\csuse{Xafterrule@#1}%
               4134       }%
               4135
               4136       \newcommand{\print@footnoteXrule}[1]{%
               4137         \vskip-\csuse{afterruleX@#1}%Because count in \dimen\csuse{footins#1}
               4138         \nointerlineskip%
               4139         \moveleft-\leftskip\vbox{\csuse{footnoterule#1}}%
               4140         \nointerlineskip%
               4141         \vskip\csuse{afterruleX@#1}%
               4142       }%
               4143
               4144       %
```

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@Xprenotes 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

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

```
4145  \newtoggle{Xprenotes@}%
4146  \toggletrue{Xprenotes@}%
4147  \newcommand{\Xprenotes@}{0pt}%
4148  \newcommand*{\Xprenotes}[1]{\renewcommand{\Xprenotes@}{#1}}%
4149  \newcommand{\preXnotes}[1]{\led@warning@preXnotes@deprecated\Xprenotes{#1}}
      %For compatibility
4150  %
```

The same, but for familiar footnotes.

```
\Xprenotes  4151  \newtoggle{prenotesX@}
\Xprenotes@ 4152  \toggletrue{prenotesX@}
       4153  \newcommand{\prenotesX@}{0pt}
       4154  \newcommand*{\prenotesX}[1]{\renewcommand{\prenotesX@}{#1}}
       4155  %
```

XVIII.0.3   Internal commands

```
firstXseries@      4156  \gdef\firstXseries@{}
prepare@Xprenotes  4157  \newcommand{\prepare@Xprenotes}[1]{%
            4158    \ifdimequal{0pt}{\Xprenotes@}%
            4159    {}%
            4160    {%
            4161      \IfStrEq{\firstXseries@}{}{%
            4162        \global\skip\csuse{#1footins}=\Xprenotes@%
            4163        \global\advance\skip\csname #1footins\endcsname by\csuse{Xafterrule@
                  #1}%
            4164        \gdef\firstXseries@{#1}%
            4165      }%
            4166      {%
            4167        \ifseriesbefore{#1}{\firstXseries@}%
            4168          {%
            4169          \global\skip\csuse{#1footins}=\csuse{Xbeforenotes@\firstXseries@}%
```

```
4170        \global\advance\skip\csname #1footins\endcsname by\csuse{Xafterrule@
      #1}%
4171        \gdef\firstXseries@{#1}%
4172        }%
4173        {}%
4174      }%
4175    }%
4176 }
4177 %
```

The same thing is required for familiar notes and \prenotesX.

```
firstseriesX@78 \gdef\firstseriesX@{}
prepare@prenotesX79 \newcommand{\prepare@prenotesX}[1]{%
4180    \ifdimequal{0pt}{\prenotesX@}%
4181    {}%
4182    {%
4183      \IfStrEq{\firstseriesX@}{}{%
4184        \global\skip\csuse{footins#1}=\prenotesX@%
4185        \global\advance\skip\csname footins#1\endcsname by\csuse{afterruleX@
      #1}%
4186        \gdef\firstseriesX@{#1}%
4187      }%
4188      {%
4189       \ifseriesbefore{#1}{\firstseriesX@}%
4190        {%
4191        \global\skip\csuse{footins#1}=\csuse{beforenotesX@\firstseriesX@}%
4192        \global\advance\skip\csname footins#1\endcsname by\csuse{afterruleX@
      #1}%
4193        \gdef\firstXseries@{#1}%
4194        }%
4195        {}%
4196      }%
4197    }%
4198 }
4199 %
```

XIX   Endnotes

First, check the noend option.

```
4200 \ifbool{noend@}{}{%Used instead of \ifnoend@ to prevent expansion problem
4201 %
```

XIX.1   Internal commands

\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.

```
4202 \newcommand{\l@dend@open}[1]{%
4203   \global\booltrue{l@dend@#1}%
4204   \expandafter\immediate%
4205     \expandafter\openout%
4206       \csname l@d@#1end\endcsname%
4207       =\l@auxdir\jobname.#1end\relax%
4208 }%
4209 \newcommand{\l@dend@close}[1]{%
4210   \global\boolfalse{l@dend@#1}%
4211   \expandafter\immediate%
4212     \expandafter\closeout\csname l@d@#1end\endcsname%
4213 }%
4214
4215 %
```

\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.

```
4216 \newcommand{\l@dend@stuff}{%
4217  \def\do##1{%
4218    \ifbool{l@dend@##1}{}%
4219      {\l@dend@open{##1}}%
4220    \expandafter\immediate\expandafter\write\csname l@d@##1end\endcsname{\
   string\l@d@section{\the\section@num}}%
4221    }%
4222  \dolistloop{\@series}%
4223 }%
4224
4225 %
```

\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.

- #6 Side (L or R).

- #7 Label for cross-referencing.

```
4226  \global\notbool{parapparatus@}{}{\long}\def\endprint#1#2#3#4#5#6#7{{%
4227    \csuse{Xendbhooknote@#4}%
4228    \csuse{Xendnotefontsize@#4}%
4229    \hangindent=\csuse{Xendhangindent@#4}%
4230    \ifXendinsertsep@%
4231      \hskip\csuse{Xendafternote@#4}\relax%
4232      \csuse{Xendsep@#4}%
4233    \else%
4234      \iftoggle{Xendparagraph@#4}%
4235        {\global\Xendinsertsep@true}%
4236        {}%
4237    \fi%
4238    \xdef\@currentseries{#4}%
4239    \def\do##1{%
4240      \setkeys[mac]{truefootnoteoption}{##1}%
4241    }%
4242    \notblank{#5}{\docsvlist{#5}}{}%
4243    \IfStrEq{#6}{R}{\ledRcol@true}{}%
4244    \def\@this@crossref@start{#7:start}%
4245    \def\@this@crossref@end{#7:end}%
4246    \printlineendnote{#1}{#4}%
4247    \IfStrEq{#6}{R}{\ledRcol@false}{}%
4248    \undef\@this@crossref@start%
4249    \undef\@this@crossref@end%
4250    \nottoggle{Xendlemmadisablefontselection@#4}%
4251      {\select@lemmafont#1|}%
4252      {}%
4253    \bgroup%
4254      \csuse{Xendlemmafont@#4}%
4255      \csuse{Xendwraplemma@#4}{#2}%
4256    \egroup%
4257    \ifboolexpr{%
4258      togl {nosep@}%
4259      or test{\ifcsempty{Xendlemmaseparator@#4}}%
4260      }%
4261      {\hskip\csuse{Xendinplaceoflemmaseparator@#4}\relax}%
4262      {\nobreak%
4263       \hskip\csuse{Xendbeforelemmaseparator@#4}%
4264       \csuse{Xendlemmaseparator@#4}%
4265       \hskip\csuse{Xendafterlemmaseparator@#4}%
4266       \relax%
4267      }%
4268    \csuse{Xendwrapcontent@#4}{#3}%
4269    \nottoggle{Xendparagraph@#4}{\par}{}%
4270    \def\do##1{%
4271      \setkeys[mac]{falsefootnoteoption}{##1}%
```

```
4272    }%
4273    \notblank{#5}{\docsvlist{#5}}{}%
4274 }}%
4275
4276 \let\l@d@section=\@gobble
4277
4278 %
```

\printlineendnote   This macro controls, in endnote, 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.

```
4279 \newcommand{\printlineendnote}[2]{%
4280    \l@dp@rsefootspec#1|%
4281    \iftoggle{Xendnumberonlyfirstintwolines@#2}{%
4282          \edef\lineinfo@{\l@dparsedstartpage - \l@dparsedstartline - \
       l@dparsedstartsub - \l@dparsedendpage - \l@dparsedendline - \
       l@dparsedendsub}%
4283          }%
4284          {%
4285          \edef\lineinfo@{\l@dparsedstartpage - \l@dparsedstartline - \
       l@dparsedstartsub}%
4286          }%
4287    \ifboolexpr{%
4288       togl {nonum@}%
4289       or togl {Xendnonumber@#2}%
4290       }%
4291       {\hspace{\csuse{Xendinplaceofnumber@#2}}}%
4292       {%
4293         \iftoggle{Xendnumberonlyfirstinline@#2}%
4294           {\ifcsdef{prevendline#2}%
4295             {\ifcsequal{prevendline#2}{lineinfo@}%
4296               {%
4297               \csuse{Xendbhookinplaceofnumber@#2}%
4298               \ifcsempty{Xendsymlinenum@#2}%
4299                 {\hspace{\csuse{Xendinplaceofnumber@#2}}}%
4300                 {\printsymlineendnotearea{#2}}%
4301               \csuse{Xendahookinplaceofnumber@2}%
4302               }%
4303               {\printlineendnotearea{#1}{#2}}}%
4304             {\printlineendnotearea{#1}{#2}}%
4305           }%
4306           {\printlineendnotearea{#1}{#2}}%We keep every time line
4307       \csxdef{prevendline#2}{\lineinfo@}%
4308       }%
4309 }%
4310 %
```

\printsymlineendnotearea₁₁ \newcommand{\printsymlineendnotearea}[1]{%

```
4312    \hspace{\csuse{Xendbeforesymlinenum@#1}}%
4313    \csuse{Xendnotenumfont@#1}%
4314    \ifdimequal{\csuse{Xendboxsymlinenum@#1}}{\z@}%
4315      {\csuse{Xendsymlinenum@#1}}%
4316      {\hbox to \csuse{Xendboxsymlinenum@#1}%
4317        {\csuse{Xendsymlinenum@#1}\hfill}%
4318      }%
4319    \hspace{\csuse{Xendaftersymlinenum@#1}}%
4320 }%
4321 %
```

\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.)

```
4322 \newcommand{\printlineendnotearea}[2]{%
4323    \csuse{Xendbhooklinenumber@#2}%
4324    \hspace{\csuse{Xendbeforenumber@#2}}%
4325    \bgroup%
4326      \csuse{Xendnotenumfont@#2}%
4327      \ifdimequal{\csuse{Xendboxlinenum@#2}}{0pt}%
4328        {\printendlines#1||\ifledRcol@\@Rlineflag\fi}%
4329        {\leavevmode%
4330         \hbox to \csuse{Xendboxlinenum@#2}%
4331         {%
4332         \IfSubStr{RC}{\csuse{Xendboxlinenumalign@#2}}{\hfill}{}%
4333         \printendlines#1||\ifledRcol@\@Rlineflag\fi%
4334         \IfSubStr{LC}{\csuse{Xendboxlinenumalign@#2}}{\hfill}{}%
4335        }}%
4336    \egroup%
4337    \hspace{\csuse{Xendafternumber@#2}}%
4338    \csuse{Xendahooklinenumber@#2}%
4339 }%
4340 %
```

XIX.2   User level commands

XIX.2.1   Inserting contents to endnotes

The \Xendnotes commands are defined upper, when defining apparatus commands by series. Here, we define. only \toendnotes command not specific to a series, in order to insert arbitrary code. The regular version writes an unexpanded argument, while the regular version writes a once-expanded argument.

```
\toendnotes*4341 \newcommandx{\toendnotes}[2][1,usedefault]{%
\toendnotes*4342    \ifboolexpr{bool{numbering} or bool{numberingR}}{%
4343       \def\do##1{%
```

```
4344        \reledmac@warning{toto}\expandafter\immediate\expandafter\write\
     csname l@d@##1end\endcsname%
4345          {\unexpanded{#2}\@percentchar}%
4346      }%
4347      \ifstrempty{#1}%
4348        {\dolistloop{\@series}}%
4349        {\docsvlist{#1}}%
4350    }{\led@err@toendnotes@outsidenumbering}%
4351  }%
4352  \WithSuffix\newcommandx\toendnotes*[2][1,usedefault]{%
4353    \ifboolexpr{bool{numbering} or bool{numberingR}}{%
4354      \def\do##1{%
4355        \reledmac@warning{toto}\expandafter\immediate\expandafter\write\
     csname l@d@##1end\endcsname%
4356          {#2\@percentchar}%
4357      }%
4358      \ifstrempty{#1}%
4359        {\dolistloop{\@series}}%
4360        {\docsvlist{#1}}%
4361    }{\led@err@toendnotes@outsidenumbering}%
4362  }%
4363  %
```

XIX.2.2   Printing endnotes

\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.

```
4364  \newif\ifXendinsertsep@%
4365  \newcommand*{\doendnotes}[1]{%
4366    \l@dend@close{#1}%
4367     \begingroup
4368        \makeatletter
4369        \expandafter\let\csname #1end\endcsname=\endprint
4370        \input\l@auxdir\jobname.#1end%
4371        \global\Xendinsertsep@false%
4372     \endgroup}
4373  %
```

\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.

```
4374  \newcommand*{\doendnotesbysection}[1]{%
4375  \l@dend@close{#1}%
4376  \global\expandafter\advance\csname #1end@bysection\endcsname by 1%
```

```
4377  \begingroup%
4378      \makeatletter%
4379      \def\l@d@section##1{%
4380        \ifnumequal{##1}{\csname #1end@bysection\endcsname}%
4381          {\cslet{#1end}{\endprint}}%
4382          {\cslet{#1end}{\@gobblefive}}%
4383      }%
4384      \input\l@auxdir\jobname.#1end%
4385      \global\Xendinsertsep@false%
4386  \endgroup%
4387  }%
4388  %
```

We close now the conditional period, which depends on \ifnoend@, because the fol-
lowing commands can be used by other commands than those speicific to endnotes.

```
4389  }%
4390  %
```

\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.

```
4391  \newcommand*{\setprintendlines}[6]{%
4392    \l@d@pnumfalse \l@d@dashfalse
4393    \ifnum#4=#1 \else
4394      \l@d@pnumtrue
4395      \l@d@dashtrue
4396    \fi
4397  %
```

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.

```
4398    \ifl@d@pnum \l@d@elintrue \else \l@d@elinfalse \fi
4399    \ifnum#2=#5 \else
4400        \l@d@elintrue
4401        \l@d@dashtrue
4402    \fi
4403  %
```

We print the starting sub-line if it is nonzero.

```
4404    \l@d@ssubfalse
4405    \ifnum#3=0 \else
```

```
4406          \l@d@ssubtrue
4407      \fi
4408  %
```

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.

```
4409      \l@d@eslfalse
4410      \ifnum#6=0 \else
4411          \ifnum#6=#3
4412              \ifl@d@elin \l@d@esltrue \else \l@d@eslfalse \fi
4413          \else
4414              \l@d@esltrue
4415              \l@d@dashtrue
4416          \fi
4417      \fi%
4418  %
```

```
4419      \ifl@d@dash%
4420        \ifboolexpr{togl{fulllines@} or test{\ifcsempty{Xendtwolines@\
      @currentseries}}}%
4421          {}%
4422          {%
4423          \setistwofollowinglines{#1}{#2}{#4}{#5}%
4424            \ifboolexpr{%
4425              (%
4426                    togl {Xendtwolinesbutnotmore@\@currentseries}%
4427                    and not%
4428                      (%
4429                       bool {istwofollowinglines@}%
4430                      )%
4431              )%
4432              or%
4433              (%
4434                (not test{\ifnumequal{#1}{#4}})%
4435                  and togl{Xendtwolinesonlyinsamepage@\@currentseries}%
4436              )%
4437            }%
4438            {}%
4439            {%
4440            \l@d@dashfalse%
4441            \l@d@Xtwolinestrue%
4442            \l@d@elinfalse%
4443            \l@d@eslfalse%
4444            \ifcsempty{Xendmorethantwolines@\@currentseries}%
4445              {}%
4446              {\ifistwofollowinglines@\else%
4447                \l@d@Xmorethantwolinestrue%
4448                \fi%
4449              }%
4450            }%
```

```
4451        }%
4452    \fi%
4453 %
```

End of \setprintendlines.

```
4454 }%
4455 %
```

\printendlines   Now we are ready to print it all.

```
4456 \def\printendlines#1|#2|#3|#4|#5|#6|#7|#8|{\begingroup
4457    \setprintendlines{#1}{#2}{#3}{#4}{#5}{#6}%
4458 %
```

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.

```
4459    \ifdimequal{\csuse{Xendboxstartlinenum@\@currentseries}}{0pt}%
4460       {\bgroup}%
4461       {\leavevmode\hbox to \csuse{Xendboxstartlinenum@\@currentseries}\bgroup
    \hfill}%
4462    \wrap@edcrossref{\@this@crossref@start}{\printnpnum{#1}}%
4463    \ifl@d@dash%
4464     \ifl@d@pnum%
4465      \csuse{Xendlineprefixsingle@\@currentseries}%
4466     \else%
4467      \ifcsempty{Xendlineprefixmore@\@currentseries}%
4468        {\csuse{Xendlineprefixsingle@\@currentseries}}
4469        {\csuse{Xendlineprefixmore@\@currentseries}}%
4470     \fi%
4471    \else%
4472     \csuse{Xendlineprefixsingle@\@currentseries}%
4473    \fi%
4474    \wrap@edcrossref{\@this@crossref@start}{\linenumrep{#2}}%
4475    \iftoggle{Xendlineflag@\@currentseries}{\ifledRcol@\@Rlineflag\fi}{}%
4476    \ifl@d@ssub%
4477       \csuse{Xendsublinesep@\@currentseries}%
4478       \wrap@edcrossref{\@this@crossref@start}{\sublinenumrep{#3}}%
4479    \fi%
4480    \egroup%
4481 %
```

And now, print the dash + the end line number, or the line number range symbol.

```
4482    \ifdimequal{\csuse{Xendboxendlinenum@\@currentseries}}{0pt}%
4483       {\bgroup}%
4484       {\hbox to \csuse{Xendboxendlinenum@\@currentseries}\bgroup}%
4485    \ifl@d@Xtwolines%
```

```
4486        \ifl@d@Xmorethantwolines%
4487          \csuse{Xendmorethantwolines@\@currentseries}%
4488        \else%
4489          \csuse{Xendtwolines@\@currentseries}%
4490        \fi%
4491      \else%
4492        \ifl@d@dash%
4493          \ifdefined\linerangesep@%
4494            \linerangesep@%
4495          \else%
4496            \csuse{Xendlinerangeseparator@\@currentseries}%
4497          \fi%
4498        \fi%
4499        \ifl@d@pnum%
4500         \wrap@edcrossref{\@this@crossref@end}\printnpnum{#4}%
4501        \fi%
4502        \ifl@d@elin%
4503          \ifl@d@pnum\csuse{Xendlineprefixsingle@\@currentseries}\fi%
4504          \wrap@edcrossref{\@this@crossref@end}{\linenumrep{#5}}%
4505          \iftoggle{Xendlineflag@\@currentseries}{\ifledRcol@\@Rlineflag\fi}{}%
4506        \fi%
4507        \ifl@d@esl%
4508          \ifl@d@elin%
4509            \csuse{Xendsublinesep@\@currentseries}%
4510          \fi%
4511          \wrap@edcrossref{\@this@crossref@end}{\sublinenumrep{#6}}%
4512        \fi%
4513      \fi%
4514      \ifdimequal{\csuse{Xendboxendlinenum@\@currentseries}}{0pt}%
4515        {}%
4516        {\hfill}%Prevent underfull hbox
4517      \egroup%
4518      \endgroup%
4519    }%
4520
4521  %
```

\printnpnum   A macro to print a page number in an endnote. Should not be override anymore

```
4522  \newcommand*{\printnpnum}[1]{\csuse{Xendbeforepagenumber@\@currentseries
      }#1\csuse{Xendafterpagenumber@\@currentseries}}
4523
4524  %
```

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@.

```
4525 \newcommand{\newseries}[1]{%
4526     \def\do##1{\newseries@{##1}}%
4527     \docsvlist{#1}
4528 }
4529 %
```

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

```
4530 \newcommand{\@series}{}
4531 %
```

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

\newseries@₃₂ \newcommand{\newseries@}[1]{

```
4533 %
```

XX.1   Test if series is still existing

```
4534     \xifinlist{#1}{\@series}{\led@warn@SeriesStillExist{#1}}%
4535     {%
4536 %
```

XX.2   Init specific to reledpar

When calling \newseries@ after having loaded reledpar, we need to load specific
setting.

```
4537         \ifdefined\newseries@par%
4538           \newseries@par{#1}%
4539         \fi%
4540 %
```

XX.3   For critical footnotes

Critical footnotes are those which start with letters. We look for the \nocritical
option of reledmac.

```
4541     \unless\ifnocritical@
4542 %
```

XX.3.1   Options

```
4543        \newtoggle{Xlineflag@#1}
4544        \newtoggle{Xparindent@#1}
4545        \newtoggle{Xlemmadisablefontselection@#1}
4546        \csgdef{Xwrapcontent@#1}{}%
4547        \csgdef{Xbeforeinserting@#1}{}%
4548        \csgdef{Xhangindent@#1}{0pt}%
4549        \csgdef{Xragged@#1}{}%
4550        \csgdef{Xhsizetwocol@#1}{0.45 \hsize}%
4551        \csgdef{Xhsizethreecol@#1}{.3 \hsize}%
4552        \csgdef{Xcolalign@#1}{\raggedright}%
4553        \csgdef{Xnotenumfont@#1}{\normalfont}%
4554        \csgdef{Xnotefontsize@#1}{\footnotesize}%
4555        \csgdef{Xbhooknote@#1}{}%
4556        \csgdef{Xbhookgroup@#1}{}%

4557
4558        \csgdef{Xboxlinenum@#1}{0pt}%
4559        \csgdef{Xboxlinenumalign@#1}{L}%

4560
4561        \csgdef{Xboxstartlinenum@#1}{0pt}%
4562        \csgdef{Xboxendlinenum@#1}{0pt}%

4563
4564        \csgdef{Xboxsymlinenum@#1}{0pt}%
4565        \newtoggle{Xnumberonlyfirstinline@#1}%
4566        \newtoggle{Xnumberonlyfirstintwolines@#1}%
4567        \csgdef{Xtwolines@#1}{}%
4568        \csgdef{Xmorethantwolines@#1}{}%
4569        \csgdef{Xsublinesep@#1}{\fullstop}%
4570        \newtoggle{Xtwolinesbutnotmore@#1}%
4571        \newtoggle{Xtwolinesonlyinsamepage@#1}%
4572        \newtoggle{Xonlypstart@#1}%
4573        \newtoggle{Xpstarteverytime@#1}%
4574        \newtoggle{Xpstart@#1}%
4575        \newtoggle{Xstanza@#1}%
4576        \csgdef{Xstanzaseparator@#1}{}%
4577        \csgdef{Xsymlinenum@#1}{}%
4578        \newtoggle{Xnonumber@#1}%
4579        \csgdef{Xbeforenumber@#1}{0pt}%
4580        \csgdef{Xtxtbeforenumber@#1}{}%
4581        \csgdef{Xafternumber@#1}{0.5em}%
4582        \newtoggle{Xnonbreakableafternumber@#1}%
4583        \csgdef{Xbeforesymlinenum@#1}{\csuse{Xbeforenumber@#1}}%
4584        \csgdef{Xaftersymlinenum@#1}{\csuse{Xafternumber@#1}}%
4585        \csgdef{Xinplaceofnumber@#1}{1em}%
4586        \global\cslet{Xlemmaseparator@#1}{\rbracket}%
4587        \csgdef{Xbeforelemmaseparator@#1}{0em}%
4588        \csgdef{Xafterlemmaseparator@#1}{0.5em}%
4589        \csgdef{Xinplaceoflemmaseparator@#1}{1em}%
4590        \csgdef{Xbeforenotes@#1}{1.2em \@plus .6em \@minus .6em}
```

```
4591        \csgdef{Xafterrule@#1}{0pt}

4592
4593        \csgdef{Xtxtbeforenotes@#1}{}
4594        \newtoggle{Xtxtbeforesnotes@#1@typeset}%Not directly used by user,
     but internal

4595
4596        \csgdef{Xmaxhnotes@#1}{0.8\vsize}
4597        \newtoggle{Xnoteswidthliketwocolumns@#1}%
4598        \csgdef{Xparafootsep@#1}{}%
4599        \csgdef{Xafternote@#1}{1em plus.4em minus.4em}
4600        \csgdef{Xlinerangeseparator@#1}{\endashchar}%

4601
4602        \csgdef{Xlemmafont@#1}{}%
4603        \csgdef{Xwraplemma@#1}{}
4604        \csgdef{Xwidth@#1}{\hsize}%
4605 %
```

XX.3.2   Create inserts, needed to add notes in foot

As regards inserts, see chapter 15 of *The TeXbook* by D. Knuth.

```
4606        \expandafter\newinsert\csname #1footins\endcsname%
4607        \unless\ifnoledgroup@%
4608          \expandafter\newinsert\csname mp#1footins\endcsname%
4609        \fi%
4610 %
```

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.

```
4611        \global\notbool{parapparatus@}{\expandafter\newcommand\expandafter
     *}{\expandafter\newcommand}\csname #1footnote\endcsname[2][]{%
4612          \if@edtext@secondarg@%
4613            \ifledRcol%
4614              \ifcsstring{Xonlyside@#1}{L}{\
     led@error@note@called@onrightside{#1footnote}}{}%
4615            \else%
4616              \ifcsstring{Xonlyside@#1}{R}{\
     led@error@note@called@onleftside{#1footnote}}{}%
4617            \fi%
4618            \begingroup%
4619            \newcommand{\content}{##2}%
4620            \ifnumberedpar@%
4621              \ifledRcol%
4622                \ifluatex%
4623                  \footnotelang@lua[R]%
4624                \fi%
4625                \@ifundefined{xpg@main@language}%if polyglossia
4626                  {}%
```

```
4627                              {\footnotelang@poly[R]}%
4628                      \footnoteoptions@{R}{##1}{true}%
4629                      \xright@appenditem{%
4630                      \ifbool{indtl@innote}%
4631                        {\unexpanded{\let\index\nindex}}%
4632                        {}%
4633                      \ifbool{indtl@notenumber}%
4634                        {\unexpanded{\let\index\nindex}}%There is no note
…number so
4635                        {}%
4636                      \noexpand\Xnote@true%
4637                      \noexpand\prepare@Xprenotes{#1}%
4638                      \noexpand\prepare@edindex@fornote{\l@d@nums}%
4639                      \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.
4640                      \noexpand\setcounter{stanzaR}{\the\c@stanzaR}%Save
stanzaR counter for footnote
4641                      \unexpanded{\def\@this@crossref@start}{\theedtext:
start}%
4642                      \unexpanded{\def\@this@crossref@end}{\theedtext:end}%
4643                      \noexpand\csuse{v#1footnote}{#1}%
4644                       {{\l@d@nums}{\expandonce\@tag}{\expandonce\content}}
%
4645                      \noexpand\Xnote@false%
4646                      \unexpanded{\undef\@this@crossref@start}%
4647                      \unexpanded{\undef\@this@crossref@end}%
4648                      \ifbool{indtl@innote}%
4649                        {\unexpanded{\let\index\orig@@index}}%
4650                        {}%
4651                      \ifbool{indtl@notenumber}%
4652                        {\unexpanded{\let\index\orig@@index}}%
4653                        {}%
4654                    }\to\inserts@listR
4655                    \footnoteoptions@{R}{##1}{false}%
4656                    \global\advance\insert@countR \@ne%
4657                \else%
4658                    \ifluatex%
4659                      \footnotelang@lua%
4660                    \fi%
4661                    \@ifundefined{xpg@main@language}%if polyglossia
4662                        {}%
4663                        {\footnotelang@poly}%
4664                    \footnoteoptions@{L}{##1}{true}%
4665                    \xright@appenditem{%
4666                    \ifbool{indtl@innote}%
4667                        {\unexpanded{\let\index\nindex}}%
4668                        {}%
4669                    \ifbool{indtl@notenumber}%
```

```
4670                          {\unexpanded{\let\index\nindex}}%There is no note
     …number so
4671                          {}%
4672                        \noexpand\Xnote@true%
4673                        \noexpand\prepare@Xprenotes{#1}%
4674                        \noexpand\prepare@edindex@fornote{\l@d@nums}%
4675                        \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.
4676                        \ifl@dpairing%
4677                          \noexpand\setcounter{stanzaL}{\the\c@stanzaL}%Save
     stanzaR counter for footnote
4678                        \fi%
4679                        \unexpanded{\def\@this@crossref@start}{\theedtext:
     start}%
4680                        \unexpanded{\def\@this@crossref@end}{\theedtext:end}%
4681                        \noexpand\csuse{v#1footnote}%
4682                          {#1}%
4683                          {{\l@d@nums}{\expandonce\@tag}{\expandonce\content
     }}%
4684                        \unexpanded{\undef\@this@crossref@start}%
4685                        \unexpanded{\undef\@this@crossref@end}%
4686                        \noexpand\Xnote@false%
4687                        \ifbool{indtl@innote}%
4688                          {\unexpanded{\let\index\orig@@index}}%
4689                          {}%
4690                        \ifbool{indtl@notenumber}%
4691                          {\unexpanded{\let\index\orig@@index}}%
4692                          {}%
4693                       }\to\inserts@list
4694                      \global\advance\insert@count \@ne%
4695                      \footnoteoptions@{L}{##1}{false}%
4696                  \fi
4697               \else
4698                  \csuse{v#1footnote}{#1}{{0|0|0|0|0|0|0|0}{}{##1}}%
4699               \fi%
4700               \endgroup%
4701           \else%
4702               \led@err@FootnoteNotInSecondArgEdtext{#1}%
4703           \fi%
4704       \ignorespaces%
4705                }
4706  %
```

We need to be able to modify `reledmac`'s footnote macros and restore their

```
4707       \global\csletcs{#1@@footnote}{#1footnote}
4708  %
```

XX.3.4   Set standard display

```
4709        \Xarrangement@normal{#1}%
4710 %
```

End of for critical footnotes.

```
4711      \fi
4712 %
```

XX.4   For familiar footnotes

Familiar footnotes are those which end with letters. We look for the `nofamiliar` option
of `reledmac`.

```
4713      \unless\ifnofamiliar@
4714 %
```

XX.4.1   Options

```
4715        \newtoggle{parindentX@#1}
4716        \csgdef{wrapcontentX@#1}{}%
4717        \csgdef{hangindentX@#1}{0pt}%
4718        \csgdef{beforeinsertingX@#1}{}%
4719        \csgdef{raggedX@#1}{}%
4720        \csgdef{hsizetwocolX@#1}{0.45 \hsize}%
4721        \csgdef{hsizethreecolX@#1}{.3 \hsize}%
4722        \csgdef{colalignX@#1}{\raggedright}%
4723        \csgdef{notenumfontX@#1}{\normalfont}%
4724        \csgdef{notefontsizeX@#1}{\footnotesize}%
4725        \csgdef{bhooknoteX@#1}{}%
4726        \csgdef{bhookgroupX@#1}{}%
4727        \csgdef{afterruleX@#1}{0pt}
4728        \csgdef{beforenotesX@#1}{1.2em \@plus .6em \@minus .6em}
4729        \csgdef{maxhnotesX@#1}{0.8\vsize}%
4730        \newtoggle{noteswidthliketwocolumnsX@#1}%
4731        \csgdef{parafootsepX@#1}{}%
4732        \csgdef{afternoteX@#1}{1em plus.4em minus.4em}
4733        \csgdef{widthX@#1}{\hsize}%
4734 % End of for familiar footnotes.
4735 % \subsubsection{Create inserts, needed to add notes in foot}
4736 % As regards inserts, see chapter 15 of the TeXBook by D. Knuth.
4737 %      \begin{macrocode}
4738        \expandafter\newinsert\csname footins#1\endcsname%
4739        \unless\ifnoledgroup@%
4740          \expandafter\newinsert\csname mpfootins#1\endcsname%
4741        \fi%
4742 %
```

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.

```
4743
4744        \global\expandafter\newcommand\csname footnote#1\endcsname[1]{%
4745            \begingroup%
4746                \prepare@prenotesX{#1}%
4747                \newcommand{\content}{##1}%
4748 %
```

If we are preparing parallel typesetting, we cannot just increase the footnote counter. Read `reledpar`'s handbook about that (V.2.1 p. 46).

```
4749                \global\expandafter\advance\csname footnote#1@reading\
    endcsname by \@ne%
4750                \ifboolexpr{bool{l@dpairing} or bool{l@dprintingpages} or
    bool{l@dprintingcolumns}}{%
4751                    \ifcsdef{footnote#1reading\the\csname footnote#1@reading\
    endcsname=typeset}%
4752                        {\setcounter{footnote#1}{\csuse{footnote#1reading\the\
    csname footnote#1@reading\endcsname=typeset}}}%
4753                        {\setcounter{footnote#1}{\the\csname footnote#1@reading
    \endcsname}}%
4754                }{%
4755                    \stepcounter{footnote#1}%
4756                }%
4757 %
```

We also have to check consistency with `\onlysideX` setting.

```
4758                \ifledRcol%
4759                    \ifcsstring{onlysideX@#1}{L}{\
    led@error@note@called@onrightside{footnote#1}}{}%
4760                \else%
4761                    \ifcsstring{onlysideX@#1}{R}{\
    led@error@note@called@onleftside{footnote#1}}{}%
4762                \fi%
4763 %
```

And now, the feature not depending of wether we are preparing parallel typesetting

```
4764                \protected@csxdef{@thefnmark#1}{\csuse{thefootnote#1}}%
4765                \nottoggle{nomk@}%Nomk is set to true when using \
    footnoteXnomk with \parpackage
4766                    {\csuse{@footnotemark#1}}%
4767                    {}%
4768                \ifluatex%
4769                    \xdef\footnote@luatextextdir{\the\textdir}%
4770                    \xdef\footnote@luatexpardir{\the\pardir}%
4771                \fi%
4772                \if@ledgroup%
```

```
4773                    \led@set@index@fornote{#1}%
4774                \fi%
4775                \csuse{vfootnote#1}{#1}{\expandonce\content}\m@mmf@prepare%
4776                \ifbool{indtl@innote}%
4777                    {\let\index\orig@@index}%
4778                    {}%
4779                \ifbool{indtl@notenumber}%
4780                    {\let\index\orig@@index}%
4781                    {}%
4782            \endgroup%
4783        }
4784 %
```

Then define the counters. The LATEX counter footnoteX is the only one manipulated by the user. This is this the one which is printed. The TEX counter \footnoteX@reading is increased at each footnote. It is used for hyperlinks, for using hyperlink package, and for getting the correct footnote number when using parallel typesetting (V.2.1 p. 46).

```
4785        \newcounter{footnote#1}
4786            \global\expandafter\renewcommand\csname thefootnote#1\endcsname{\
      arabic{footnote#1}}
4787        \expandafter\newcount\csname footnote#1@reading\endcsname%
4788 %
```

Do not forget to initialize series

```
4789        \arrangementX@normal{#1}%
4790    \fi
4791 %
```

XX.5   The endnotes

Endnotes are commands like \Xendnote, where X is a series letter. First, we check for the noend options.

```
4792    \unless\ifnoend@
4793 %
```

XX.5.1   The auxiliary file

\l@d@Xend
\ifl@dend@X
\l@dend@Xtrue
\l@dend@Xfalse Endnotes of all varieties are saved up in a file, one by series, typically named ⟨*jobname*⟩.Xend. \l@d@end is the output stream number for this file, and \ifl@dend@X is a flag that is true when the file is open.

```
4794        \expandafter\newwrite\csname l@d@#1end\endcsname%
4795        \expandafter\newif\csname ifl@dend@#1\endcsname%
4796 %
```

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.

```
4797
4798        \global\expandafter\newcommandx\csname #1endnote\endcsname[2][1,
      usedefault]{%
4799            \bgroup%
4800            \newlinechar='40%
4801            \global\@noneed@Footnotetrue%
4802            \newcommand{\content}{##2}%
4803            \stepcounter{labidx}%
4804            \expandafter\immediate\expandafter\write\csname l@d@#1end\
      endcsname{%
4805                \unexpanded{\def\sw@list@inedtext}{\expandafter\unexpanded\
      expandafter\sw@inthisedtext}} %Volountary space, to add linebreak in the
      output file
4806            \expandafter\string\csname #1end\endcsname%
4807            {\ifnumberedpar@\l@d@nums\fi}%
4808            {\ifnumberedpar@\expandonce\@tag\fi}%
4809            {\expandonce\content}%
4810            {#1}%
4811            {\unexpanded{##1}}%
4812            {\ifledRcol R\else L\fi}%
4813            {\theedtext}%
4814            \@percentchar%
4815            }%
4816            \egroup%
4817            \ignorespaces%
4818        }%
4819 %
```

XX.5.3 Tools

The \Xtoendnotes command inserts any arbitrary content into the endnote file. It is an alias of the more generalist \addtoennotes

```
4820
4821        \global\expandafter\newcommand\csname #1toendnotes\endcsname[1]{%
4822          \toendnotes[#1]{##1}%
4823        }%
4824
4825        \expandafter\WithSuffix\expandafter\newcommand\csname #1toendnotes\
      endcsname*[1]{%
4826          \toendnotes*[#1]{##1}%
4827        }%
4828
```

4829 %

XX.5.4   Internal commands

\Xendnote commands called \Xend commands on to the endnote file; these are anal-
ogous 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.

4830
4831   \global\cslet{#1end}{\@gobblefive}
4832 %

We need to store the number of times \doendnotesbysection is called for one series.

4833   \global\expandafter\newcount\csname #1end@bysection\endcsname%
4834 %

XX.5.5   The options

4835   \csgdef{Xendwraplemma@#1}{}
4836   \csgdef{Xendwrapcontent@#1}{}%
4837   \csgdef{Xendtwolines@#1}{}%
4838   \csgdef{Xendmorethantwolines@#1}{}%
4839   \newtoggle{Xendtwolinesbutnotmore@#1}{}%
4840   \newtoggle{Xendtwolinesonlyinsamepage@#1}{}%
4841   \newtoggle{Xendlemmadisablefontselection@#1}%
4842   \csgdef{Xendnotenumfont@#1}{\normalfont}%
4843   \csgdef{Xendnotefontsize@#1}{\footnotesize}%
4844   \csgdef{Xendbhooknote@#1}{}%
4845
4846   \csgdef{Xendsublinesep@#1}{\fullstop}%
4847
4848   \csgdef{Xendbeforenumber@#1}{0pt}
4849   \csgdef{Xendafternumber@#1}{0.5em}
4850
4851   \csgdef{Xendboxlinenum@#1}{0pt}%
4852   \csgdef{Xendboxlinenumalign@#1}{L}%
4853
4854   \csgdef{Xendboxstartlinenum@#1}{0pt}%
4855   \csgdef{Xendboxendlinenum@#1}{0pt}%
4856
4857   \csgdef{Xendlemmaseparator@#1}{}%
4858   \csgdef{Xendbeforelemmaseparator@#1}{0em}%
4859   \csgdef{Xendafterlemmaseparator@#1}{0.5em}%
4860   \csgdef{Xendinplaceoflemmaseparator@#1}{0.5em}%
4861
4862   \newtoggle{Xendparagraph@#1}%

```
4863        \csgdef{Xendafternote@#1}{1em plus.4em minus.4em}%
4864        \csgdef{Xendsep@#1}{}%
4865
4866        \csgdef{Xendinplaceofnumber@#1}{0pt}%
4867        \newtoggle{Xendnonumber@#1}%
4868
4869        \csgdef{Xendhangindent@#1}{0pt}%
4870        \newtoggle{Xendnumberonlyfirstinline@#1}%
4871        \newtoggle{Xendnumberonlyfirstintwolines@#1}%
4872
4873        \csgdef{Xendbeforesymlinenum@#1}{\csuse{Xendbeforenumber@#1}}%
4874        \csgdef{Xendaftersymlinenum@#1}{\csuse{Xendafternumber@#1}}%
4875        \csgdef{Xendsymlinenum@#1}{}%
4876        \csgdef{Xendboxsymlinenum@#1}{0pt}%
4877
4878        \csgdef{Xendbhooklinenumber@#1}{}%
4879        \csgdef{Xendehooklinenumber@#1}{}%
4880        \csgdef{Xendbhookinplaceofnumber@#1}{}%
4881        \csgdef{Xendehookinplaceofnumber@#1}{}%
4882
4883        \csgdef{Xendlinerangeseparator@#1}{\endashchar}%
4884
4885        \csgdef{Xendbeforepagenumber@#1}{p.}%
4886        \csgdef{Xendafterpagenumber@#1}{) }%
4887        \csgdef{Xendlineprefixsingle@#1}{}%
4888        \csgdef{Xendlineprefixmore@#1}{}%
4889
4890        \newtoggle{Xendlineflag@#1}
4891
4892        \csgdef{Xendlemmafont@#1}{}%
4893 %
```

End of endnotes declaration

```
4894    \fi%
4895 %
```

Dump series in \@series

```
4896        \listxadd{\@series}{#1}
4897    }
4898 }% End of \newseries
4899 %
```

XX.6   Init standards series (A,B,C,D,E)

```
4900 \expandafter\newseries\expandafter{\default@series}
4901 %
```

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.

```
4902  \newcommand{\seriesatbegin}[1]{%
4903      \StrDel{\@series}{#1}[\@series]%
4904      \edef\@new{}%
4905      \listeadd{\@new}{#1}%
4906      \listeadd{\@new}{\@series}%
4907      \xdef\@series{\@new}%
4908  }
4909  %
```

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

```
4910  \newcommand{\seriesatend}[1]{%
4911      \StrDel{\@series}{#1}[\@series]%
4912      \edef\@new{}%
4913      \listeadd{\@new}{\@series}%
4914      \listeadd{\@new}{#1}%
4915      \xdef\@series{\@new}%
4916  }
4917  %
```

XXI.2   Test series order

\ifseriesbefore   \ifseriesbefore{⟨*seriesA*⟩}{⟨*seriesB*⟩}{⟨*true*⟩}{⟨*false*⟩} expands ⟨*true*⟩ if ⟨*seriesA*⟩ is printed before ⟨*seriesB*⟩, expands ⟨*false*⟩ otherwise.

```
4918  \newcommand{\ifseriesbefore}[4]{%
4919   \StrPosition{\@series}{#1}[\@first]%
4920   \StrPosition{\@series}{#2}[\@second]%
4921   \ifnumgreater{\@second}{\@first}{#3}{#4}%
4922  }
4923  %
```

XXI.2.1   Get the first series

In some specific case, we need to know the first series of the list of series.

\@getfirstseries   \newcommand{\@getfirstseries}{%
```
4925   \ifdefempty{\@series}%
4926      {\xdef\@firstseries{}}%
4927      {\StrChar{\@series}{1}[\@firstseries]}%
4928  }%
4929  %
```

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.

```
4930  \newcommandx{\settoggle@series}[5][4,5,usedefault]{%
4931      \def\do##1{%
4932        \global\settoggle{#2@##1}{#3}%
4933        \ifstrequal{#4}{critical}{
4934            \csuse{Xarrangement@\csuse{series@display##1}}{##1}%
4935        }{}
4936        \ifstrequal{#4}{familiar}{
4937            \csuse{arrangementX@\csuse{series@displayX##1}}{##1}%
4938        }{}
4939      }%
4940      \ifstrempty{#1}{%
```

```
4941                  \dolistloop{\@series}%
4942                  \ifstrempty{#5}{}{%
4943                     \docsvlist{#5}%
4944                  }
4945              }%
4946              {%
4947                  \docsvlist{#1}%
4948              }%
4949  }
4950  %
```

\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.

```
4951  \newcommandx\setcommand@series[5][4,5,usedefault]{%
4952      \def\do##1{
4953          \csgdef{#2@##1}{#3}
4954          \ifstrequal{#4}{critical}{%
4955              \csuse{Xarrangement@\csuse{series@display##1}}{##1}%
4956          }{}
4957          \ifstrequal{#4}{familiar}{%
4958              \csuse{arrangementX@\csuse{series@displayX##1}}{##1}%
4959          }{}%
4960      }%
4961      \ifstrempty{#1}{%
4962              \dolistloop{\@series}%
4963              \ifstrempty{#5}{}{%
4964                  \docsvlist{#5}
4965              }
4966      }%
4967      {%
4968              \docsvlist{#1}%
4969      }%
4970  }%
4971  %
```

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`.

```
4972 \newcommandx{\newhookcommand@series}[2][2,usedefault]{%
4973   \global\expandafter\newcommand\expandafter*\csname #1\endcsname[2][]{%
4974     \setcommand@series{##1}{#1}{##2}[][#2]%
4975   }%
4976   \ifstrempty{#2}{}{%
4977     \def\do##1{%
4978       \global\expandafter\newcommand\expandafter*\csname #1##1\endcsname
      [1]{%
4979         \csuse{#1}[##1]{####1}%
4980       }%
4981     }%
4982     \docsvlist{#2}%
4983   }%
4984 }
4985 %
```

`\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`.

```
4986 \newcommandx{\newhooktoggle@series}[2][2,usedefault]{%
4987   \global\expandafter\newcommandx\expandafter*\csname #1\endcsname[2][1,2={
      true},usedefault]{%
4988     \settoggle@series{##1}{#1}{##2}[][#2]%
4989     }%
4990   \ifstrempty{#2}{}{%
4991     \def\do##1{%
4992       \global\expandafter\newcommand\expandafter*\csname #1##1\endcsname{%
4993         \csuse{#1}[##1]%
4994       }%
4995     }%
4996     \docsvlist{#2}%
4997   }%
4998 }
4999 %
```

`whooktoggle@series@reload` `\newhookcommand@toggle@reload` does the same thing as `\newhooktoggle@series` but the commands created by this macro also reload the series arrangement, depending of type os notes

```
5000 \newcommand{\newhooktoggle@series@reload}[2]{%
```

```
5001    \global\expandafter\newcommandx\expandafter*\csname #1\endcsname[2][1,2={
true},usedefault]{%
5002        \settoggle@series{##1}{#1}{##2}[#2]%
5003        }%
5004    }%
5005    %
```

`\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.

```
5006    \newcommand{\newhookcommand@series@reload}[2]{%
5007      \global\expandafter\newcommand\expandafter*\csname #1\endcsname[2][]{%
5008        \setcommand@series{##1}{#1}{##2}[#2]%
5009      }%
5010    }
5011    %
```

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.

```
5012    \unless\ifnocritical@
5013      \newhookcommand@series{Xwrapcontent}%
5014      \newhookcommand@series{Xbeforeinserting}%
5015      \newhookcommand@series{Xlemmafont}%
5016      \newhookcommand@series{Xwraplemma}%
5017      \newhooktoggle@series{Xparindent}
5018      \newhookcommand@series{Xhangindent}
5019      \newhookcommand@series{Xragged}
5020      \newhookcommand@series{Xhsizetwocol}
5021      \newhookcommand@series{Xhsizethreecol}
5022      \newhookcommand@series{Xcolalign}%
5023      \newhookcommand@series{Xnotenumfont}
5024      \newhookcommand@series{Xbhooknote}
5025      \newhookcommand@series@reload{Xbhookgroup}{critical}
5026      \newhookcommand@series{Xboxsymlinenum}%
5027      \newhookcommand@series{Xsymlinenum}
5028      \newhookcommand@series{Xbeforenumber}
5029      \newhookcommand@series{Xtxtbeforenumber}
5030      \newhookcommand@series{Xafternumber}
5031      \newhookcommand@series{Xbeforesymlinenum}
5032      \newhookcommand@series{Xaftersymlinenum}
5033      \newhookcommand@series{Xinplaceofnumber}
5034      \newhookcommand@series{Xlemmaseparator}
5035      \newhookcommand@series{Xbeforelemmaseparator}
5036      \newhookcommand@series{Xafterlemmaseparator}
5037      \newhookcommand@series{Xinplaceoflemmaseparator}
5038      \newhookcommand@series{Xtxtbeforenotes}
```

```
5039  \newhookcommand@series@reload{Xafterrule}{critical}
5040  \newhooktoggle@series{Xnumberonlyfirstinline}
5041  \newhooktoggle@series{Xnumberonlyfirstintwolines}
5042  \newhooktoggle@series{Xnonumber}
5043  \newhooktoggle@series{Xpstart}
5044  \newhooktoggle@series{Xpstarteverytime}%
5045
5046  \newhooktoggle@series{Xstanza}%
5047  \newhookcommand@series{Xstanzaseparator}%
5048
5049  \newhooktoggle@series{Xonlypstart}
5050  \newhooktoggle@series{Xnonbreakableafternumber}
5051  \newhooktoggle@series{Xlemmadisablefontselection}
5052  \newhookcommand@series@reload{Xmaxhnotes}{critical}
5053  \newhookcommand@series@reload{Xbeforenotes}{critical}
5054  \newhooktoggle@series@reload{Xnoteswidthliketwocolumns}{critical}%
5055  \newhookcommand@series@reload{Xnotefontsize}{critical}
5056
5057  \newhookcommand@series{Xboxlinenum}%
5058  \newhookcommand@series{Xboxlinenumalign}%
5059
5060  \newhookcommand@series{Xboxstartlinenum}%
5061  \newhookcommand@series{Xboxendlinenum}%
5062
5063  \newhookcommand@series{Xafternote}%
5064  \newhookcommand@series{Xparafootsep}
5065
5066  \newhookcommand@series@reload{Xwidth}{critical}%
5067
5068  \ifundef{\Xhsize}%
5069    {%
5070      \newcommandx{\Xhsize}[2][1,usedefault]{%
5071        \led@warning@Xhsize@deprecated%
5072        \Xwidth[#1]{#2}%
5073      }%
5074    }%
5075    {}%
5076  \fi
5077  \newhooktoggle@series{Xlineflag}[appref,SEref]
5078  \newhookcommand@series{Xtwolines}[appref,SEref]
5079  \newhookcommand@series{Xmorethantwolines}[appref,SEref]
5080  \newhookcommand@series{Xsublinesep}[appref,SEref,side]
5081  \newhooktoggle@series{Xtwolinesbutnotmore}[appref,SEref]
5082  \newhooktoggle@series{Xtwolinesonlyinsamepage}[appref,SEref]
5083  \newhookcommand@series{Xlinerangeseparator}[appref,SEref]
5084  %
```

XXI.3.5   Options for familiar notes

Before generating the optional commands for familiar notes, we check the `\nofamiliar` option.

```
5085 \unless\ifnofamiliar@
5086   \newhookcommand@series{wrapcontentX}%
5087   \newhookcommand@series{beforeinsertingX}%
5088   \newhooktoggle@series{parindentX}
5089   \newhookcommand@series{hangindentX}
5090   \newhookcommand@series{raggedX}
5091   \newhookcommand@series{hsizetwocolX}
5092   \newhookcommand@series{hsizethreecolX}
5093   \newhookcommand@series{colalignX}%
5094   \newhookcommand@series{notenumfontX}
5095   \newhookcommand@series{bhooknoteX}
5096   \newhookcommand@series@reload{bhookgroupX}{familiar}
5097   \newhookcommand@series@reload{beforenotesX}{familiar}
5098   \newhookcommand@series@reload{maxhnotesX}{familiar}
5099   \newhooktoggle@series@reload{noteswidthliketwocolumnsX}{familiar}%
5100   \newhookcommand@series@reload{afterruleX}{familiar}
5101   \newhookcommand@series@reload{notefontsizeX}{familiar}
5102   \newhookcommand@series{afternoteX}
5103   \newhookcommand@series{parafootsepX}
5104   \newhookcommand@series@reload{widthX}{familiar}%
5105   \ifundef{\hsizeX}%
5106     {%
5107       \newcommandx{\hsizeX}[2][1,usedefault]{%
5108         \led@warning@hsizeX@deprecated%
5109         \widthX[#1]{#2}%
5110       }%
5111     }%
5112     {}%
5113 \fi
5114 %
```

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.

```
5115 \unless\ifnoend@
5116   \newhookcommand@series{Xendwraplemma}
5117   \newhookcommand@series{Xendwrapcontent}
5118   \newhookcommand@series{Xendnotenumfont}
5119   \newhookcommand@series{Xendlemmafont}%
5120   \newhookcommand@series{Xendbhooknote}
5121
5122   \newhookcommand@series{Xendboxlinenum}%
5123   \newhookcommand@series{Xendboxlinenumalign}%
```

```
5124
5125    \newhookcommand@series{Xendboxstartlinenum}%
5126    \newhookcommand@series{Xendboxendlinenum}%
5127
5128    \newhookcommand@series{Xendnotefontsize}
5129    \newhooktoggle@series{Xendlemmadisablefontselection}
5130    \newhookcommand@series{Xendlemmaseparator}
5131    \newhookcommand@series{Xendbeforelemmaseparator}
5132    \newhookcommand@series{Xendafterlemmaseparator}
5133    \newhookcommand@series{Xendinplaceoflemmaseparator}
5134
5135    \newhookcommand@series{Xendbeforenumber}%
5136    \newhookcommand@series{Xendafternumber}%
5137
5138    \newhooktoggle@series{Xendparagraph}
5139    \newhookcommand@series{Xendafternote}
5140    \newhookcommand@series{Xendsep}
5141
5142    \newhookcommand@series{Xendinplaceofnumber}%
5143    \newhooktoggle@series{Xendnonumber}%
5144
5145    \newhooktoggle@series{Xendnumberonlyfirstinline}%
5146    \newhooktoggle@series{Xendnumberonlyfirstintwolines}%
5147
5148    \newhookcommand@series{Xendsymlinenum}%
5149    \newhookcommand@series{Xendbeforesymlinenum}%
5150    \newhookcommand@series{Xendaftersymlinenum}%
5151    \newhookcommand@series{Xendboxsymlinenum}%
5152
5153    \newhookcommand@series{Xendbhooklinenumber}%
5154    \newhookcommand@series{Xendahooklinenumber}%
5155    \newhookcommand@series{Xendbhookinplaceofnumber}%
5156    \newhookcommand@series{Xendahookinplaceofnumber}%
5157
5158    \newhookcommand@series{Xendhangindent}%
5159
5160
5161 \fi
5162 \newhooktoggle@series{Xendlineflag}[apprefwithpage,SErefwithpage]
5163 \newhookcommand@series{Xendtwolines}[apprefwithpage,SErefwithpage]
5164 \newhookcommand@series{Xendmorethantwolines}[apprefwithpage,SErefwithpage]
5165 \newhooktoggle@series{Xendtwolinesbutnotmore}[apprefwithpage,SErefwithpage]
5166 \newhooktoggle@series{Xendtwolinesonlyinsamepage}[apprefwithpage,
     SErefwithpage]
5167 \newhookcommand@series{Xendlinerangeseparator}[apprefwithpage,SErefwithpage
     ]
5168 \newhookcommand@series{Xendbeforepagenumber}[apprefwithpage,SErefwithpage,
     SErefonlypage]
5169 \newhookcommand@series{Xendafterpagenumber}[apprefwithpage,SErefwithpagem]
5170 \newhookcommand@series{Xendlineprefixsingle}[apprefwithpage,SErefwithpage]
```

```
5171 \newhookcommand@series{Xendlineprefixmore}[apprefwithpage,SErefwithpage]
5172 \newhookcommand@series{Xendsublinesep}[apprefwithpage,SErefwithpage]
5173
5174 %
```

XXI.4   Hooks for a particular footnote

\newhooktoggle@specific   \newhooktoggle@specific is a generic command to create boolean hook specific to
a note.

```
5175 \newcommand{\newhooktoggle@specific}[1]{%
5176   \newtoggle{#1@}%
5177   \define@key[mac]{truefootnoteoption}{#1}[]{\global\settoggle{#1@}{true}}%
When enabling footnote option
5178   \define@key[mac]{falsefootnoteoption}{#1}[]{\global\settoggle{#1@}{false
}}
5179 }
5180 %
```

\newhookarg@specific   \newhookarg@specific is a generic command to create argumen hook specific to a
note.

```
5181 \newcommand{\newhookarg@specific}[1]{%
5182   \define@key[mac]{truefootnoteoption}{#1}{\global\def\linerangesep@{##1}}%
When enabling footnote option
5183   \define@key[mac]{falsefootnoteoption}{#1}{\global\undef\linerangesep@}%
When
5184 }
5185 %
```

And now, we define some hooks specific to a note.

```
5186 \newhooktoggle@specific{fulllines}%
5187 \newhooktoggle@specific{nonum}
5188 \newhooktoggle@specific{nosep}
5189 \newhookarg@specific{linerangesep}
5190 %
```

\linerangesep@   \linerangesep@ is defined by the option linerangesep of critical notes to change
temporarily the line range separator for a specific line. As we have to define it before
typesetting the line and undefine it after, we use the family of xkeyval package's key.

```
5191 %
```

\nomk@   \nomk@ toggle is used by reledpar to remove the footnote mark in the text when using
\footnoteXmk. Read reledpar handbook.

```
5192 \newtoggle{nomk@}%
5193 %
```

XXI.5   Alias

\Xnolemmaseparator   \Xnolemmaseparator[⟨*series*⟩] is just an alias for \Xlemmaseparator[⟨*series*⟩]{}.

```
5194 \newcommandx*{\Xnolemmaseparator}[1][1]{\Xlemmaseparator[#1]{}}
5195 %
```

XXII   Output routine

Now we begin the output routine and associated things.

XXII.1   Page number management

\pageno   \pageno is a page number, starting at 1, and \advancepageno increments the number.
\advancepageno
```
5196 \countdef\pageno=0  \pageno=1
5197 \newcommand*{\advancepageno}{\ifnum\pageno<\z@ \global\advance\pageno\m@ne
5198   \else\global\advance\pageno\@ne\fi}
5199
5200 %
```

XXII.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 commands

\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.

```
5201 \newcommand*{\l@ddoxtrafeet}{%
5202   \IfStrEq{familiar-critical}{\@fnpos}
5203     {\do@feetX\Xdo@feet}%
5204     {%
5205     \IfStrEq{critical-familiar}{\@fnpos}%
5206         {\Xdo@feet\do@feetX}%
5207         {\do@feetX\Xdo@feet}%
5208     }%
5209 }%
5210
5211 %
```

\Xdo@feet   \Xdo@feet is the code extending \@makecol to cater for the extra critical feet.

```
5212 \newcommand*{\Xdo@feet}{%
5213   \setbox\@outputbox \vbox{%
5214     \unvbox\@outputbox
5215     \@opXfeet}}
5216 %
```

\@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.

```
5217 \newcommand\print@Xnotes[1]{%
5218   \xdef\@currentseries{#1}%
5219   \csuse{#1footstart}{#1}%
5220   \csuse{#1footgroup}{#1}%%
5221 }%
5222 %
```

We print all series of notes by looping on them. We check before printing them that
they are not voided.

```
5223 \newcommand*{\@opXfeet}{%
5224   \unless\ifnocritical@%
5225     \gdef\firstXseries@{}%
5226     \def\do##1{%
5227       \ifvoid\csuse{##1footins}\else%
5228         \global\skip\csuse{##1footins}=\csuse{Xbeforenotes@##1}%
5229         \global\advance\skip\csuse{##1footins} by\csuse{Xafterrule@##1}%
5230         \print@Xnotes{##1}%
5231       \fi%
5232     }%
5233     \dolistloop{\@series}%
5234   \fi%
5235 }%
5236 %
```

\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.

```
5237 \newcommand*{\l@ddodoreinxtrafeet}{%
5238   \IfStrEq{familiar-critical}{\@fnpos}
5239     {\@doreinfeetX\X@doreinfeet}%
5240     {%
5241     \IfStrEq{critical-familiar}{\@fnpos}%
5242       {\X@doreinfeet\@doreinfeetX}%
5243       {\@doreinfeetX\X@doreinfeet}%
5244     }%
5245 }
5246
5247 %
```

\X@doreinfeet   \X@doreinfeet is the code for catering for the extra critical footnotes within \@reinserts.

```
5248 \newcommand*{\X@doreinfeet}{%
5249   \unless\ifnocritical@%
5250     \def\do##1{%
5251       \ifvoid\csuse{##1footins}\else%
5252         \insert\csuse{##1footins}{\unvbox\csuse{##1footins}}%
5253       \fi}%
5254     \dolistloop{\@series}
5255   \fi%
5256 }
5257
5258 %
```

\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

```
5259 \newcommand\print@notesX[1]{%
5260   \xdef\@currentseries{#1}%
5261   \csuse{footstart#1}{#1}%
5262   \csuse{footgroup#1}{#1}%
5263 }%
5264 %
```

We print all the series of notes by looping on them. We check before printing them that
they are not voided.

```
5265 \newcommand*{\do@feetX}{%
5266   \unless\ifnofamiliar@%
5267     \gdef\firstseriesX@{}%
5268     \setbox\@outputbox \vbox{%
5269       \unvbox\@outputbox%
5270       \def\do##1{%
5271       \ifvoid\csuse{footins##1}\else%
5272         \global\skip\csuse{footins##1}=\csuse{beforenotesX@##1}%
5273         \global\advance\skip\csuse{footins##1} by\csuse{afterruleX@##1}%
5274         \print@notesX{##1}%
5275       \fi%
5276       }%
5277       \dolistloop{\@series}}%
5278   \fi%
5279 }%
5280
5281 \newcommand{\@doreinfeetX}{%
5282   \unless\ifnofamiliar@%
5283     \def\do##1{%
5284       \ifvoid\csuse{footins##1}\else
5285         \insert%
5286           \csuse{footins##1}
5287           {\unvbox\csuse{footins##1}}%
5288       \fi%
5289     }%
5290     \dolistloop{\@series}%
```

```
5291    \fi%
5292 }%
5293
5294 %
```

XXII.3   Patching standard output of commands

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.)

```
5295 \@ifclassloaded{memoir}{%
5296 %
```

memoir is loaded so we use memoir's built in hooks.

```
5297    \g@addto@macro{\m@mdoextrafeet}{\l@ddoxtrafeet}%
5298    \g@addto@macro{\m@mdodoreinextrafeet}{\l@ddodoreinxtrafeet}%
5299    }{%
5300 %
```

memoir has not been loaded, so patch \@makecol and \@reinserts.

```
5301    \@ifpackageloaded{fancyhdr}{%
5302      \patchcmd%
5303        {\latex@makecol}%
5304        {\xdef\@freelist{\@freelist\@midlist}}%
5305        {\xdef\@freelist{\@freelist\@midlist}\l@ddoxtrafeet}%
5306        {}%
5307        {\led@error@fail@patch@@makecol}%
5308      }{%
5309      \patchcmd%
5310        {\@makecol}%
5311        {\xdef\@freelist{\@freelist\@midlist}}%
5312        {\xdef\@freelist{\@freelist\@midlist}\l@ddoxtrafeet}%
5313        {}%
5314        {\led@error@fail@patch@@makecol}%
5315      }%
5316
5317    \patchcmd%
5318      {\@reinserts}%
5319      {\ifvbox}%
5320      {\l@ddodoreinxtrafeet\ifvbox}%
5321      {}%
5322      {\led@error@fail@patch@@reinserts}%
5323 }
5324
5325 %
```

It turns out that \@doclearpage also needs modifying.

`\if@led@nofoot`   We have to check if there are any leftover feet.

```
5326 \newif\if@led@nofoot
5327
5328 %


5329 \@ifclassloaded{memoir}{%
5330 %
```

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

`\@mem@extranofeet`
```
5331 \g@addto@macro{\@mem@extranofeet}{%%
5332   \def\do#1{%
5333     \unless\ifnocritical@%
5334       \ifvoid\csuse{#1footins}\else\@mem@nofootfalse\fi%
5335     \fi%
5336     \unless\ifnofamiliar@%
5337       \ifvoid\csuse{footins#1}\else\@mem@nofootfalse\fi%
5338     \fi%
5339     }
5340   \dolistloop{\@series}%
5341   }%
5342 }{%
5343 %
```

As memoir is not loaded we have patch `\@doclearpage`.

`\@led@testifnofoot`
`\@doclearpage`
```
5344 \newcommand*{\@led@testifnofoot}{%
5345   \@led@nofoottrue%
5346   \ifvoid\footins\else%
5347     \@led@nofootfalse%
5348   \fi%
5349   \def\do##1{%
5350     \unless\ifnocritical@%
5351       \ifvoid\csuse{##1footins}\else%
5352         \@led@nofootfalse%
5353       \fi%
5354     \fi%
5355     \unless\ifnofamiliar@%
5356       \ifvoid\csuse{footins##1}\else%
5357         \@led@nofootfalse%
5358       \fi%
5359     \fi%
5360   }%
5361   \dolistloop{\@series}%
5362 }%
5363
5364 \pretocmd%
5365   {\@doclearpage}%
```

```
5366  {\@led@testifnofoot}%
5367  {}%
5368  {\led@error@fail@patch@@doclearpage}%
5369
5370  \patchcmd%
5371  {\@doclearpage}%
5372  {\ifvoid\footins}%
5373  {\if@led@nofoot}%
5374  {}%
5375  {\led@error@fail@patch@@doclearpage}%
5376
5377  }
5378
5379  %
```

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 typing \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.

```
5380  \list@create{\labelref@list}
5381  %
```

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

```
5382  \newcommand*{\zz@@@}{000|000} % set two counters to zero in one go
5383
5384  %
```

\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.[32]

---

[32]The remaining macros in this section were kindly revised by Wayne Sullivan, who substantially improved their efficiency and flexibility.

This version of the original edmac \label uses \@bsphack and \@esphack to eliminate extra space problems and also use the LaTeX write methods for the .aux file.

Jesse Billett[33] found that the original code could be off by several pages. This version, hopefully cures that, and also allows for non-arabic page numbering.

```
5385  \newcommand*{\edlabel}[1]{%
5386    \ifl@dpairing\ifautopar%
5387      \strut%
5388    \fi\fi%
5389    \@bsphack%
5390    \ifboolexpr{bool{ledRcol} or bool{ledRcol@}}{%
5391      \ifXnote@%
5392        \protected@write\@auxout{}%
5393        {\string\l@dmake@labelsR\space\thepage|\l@dparsedstartline|\
       l@dparsedstartsub|\the\c@pstartR|{#1}}%
5394        \ifdef{\hypertarget}%
5395          {\Hy@raisedlink{\hypertarget{#1}{}}}%
5396          {}%
5397      \else%
5398        \write\linenum@outR{\string\@lab}%
5399        \ifx\labelref@listR\empty%
5400          \xdef\label@refs{\zz@@@}%
5401        \else%
5402          \gl@p\labelref@listR\to\label@refs%
5403        \fi%
5404        \ifvmode%
5405            \advancelabel@refs%
5406        \fi%
5407  %
```

Use code from the kernel \label command to write the correct page number. Also define an hypertarget if hyperref package is loaded.

```
5408        \protected@write\@auxout{}%
5409          {\string\l@dmake@labelsR\space\thepage|\label@refs|\the\c@pstartR
       |{#1}}%
5410        \ifdef{\hypertarget}%
5411          {\Hy@raisedlink{\hypertarget{#1}{}}}%
5412          {}%
5413      \fi%
5414    }{%
5415      \ifXnote@%
5416        \protected@write\@auxout{}%
5417        {\string\l@dmake@labels\space\thepage|\l@dparsedstartline|\
       l@dparsedstartsub|\the\c@pstart|{#1}}%
5418        \ifdef{\hypertarget}%
5419          {\Hy@raisedlink{\hypertarget{#1}{}}}%
5420          {}%
5421      \else%
```

---

[33] (jdb43@cam.ac.uk) via the ctt thread 'ledmac cross referencing', 25 August 2003.

```
5422        \write\linenum@out{\string\@lab}%
5423        \ifx\labelref@list\empty%
5424          \xdef\label@refs{\zz@@@}%
5425        \else%
5426          \gl@p\labelref@list\to\label@refs%
5427        \fi%
5428        \ifvmode%
5429            \advancelabel@refs%
5430        \fi%
5431        \protected@write\@auxout{}%
5432          {\string\l@dmake@labels\space\thepage|\label@refs|\the\c@pstart
       |{#1}}%
5433          \ifdef{\hypertarget}%
5434            {\Hy@raisedlink{\hypertarget{#1}{}}}%
5435            {}%
5436      \fi%
5437    }%
5438 \@esphack}%
5439
5440 %
```

\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
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.

```
5441 \newcounter{line}%
5442 \newcounter{subline}%
5443 \newcommand{\advancelabel@refs}{%
5444    \setcounter{line}{\expandafter\labelrefsparseline\label@refs}%
5445    \stepcounter{line}%
5446    \ifsublines@%
5447        \setcounter{subline}{\expandafter\labelrefsparsesubline\label@refs}
       %
5448        \stepcounter{subline}{1}%
5449        \def\label@refs{\theline|\thesubline}%
5450    \else%
5451        \def\label@refs{\theline|0}%
5452    \fi%
5453 }
5454 \def\labelrefsparseline#1|#2{#1}
5455 \def\labelrefsparsesubline#1|#2{#2}
5456 %
```

\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.

```
5457  \newcommand*{\l@dmake@labels}{}
5458  \def\l@dmake@labels#1|#2|#3|#4|#5{%
5459    \expandafter\ifx\csname the@label\csuse{XR@prefix}#5\endcsname \relax\
      else
5460      \led@warn@DuplicateLabel{\csuse{XR@prefix}#5}%
5461    \fi
5462    \expandafter\gdef\csname the@label\csuse{XR@prefix}#5\endcsname
      {#1|#2|#3|#4|\relax}%
5463    \ignorespaces}
5464
5465  %
```

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.

```
5466  \AtBeginDocument{%
5467    \def\l@dmake@labels#1|#2|#3|#4|#5{}%
5468  }
5469
5470  %
```

`\@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`.

```
5471
5472  \newcommand*{\@lab}{%
5473    \ifledRcol
5474      \xright@appenditem{\linenumr@p{\line@numR}|%
5475        \ifsublines@ \sublinenumr@p{\subline@numR}\else 0\fi}%
5476        \to\labelref@listR
5477    \else
5478      \xright@appenditem{\linenumr@p{\line@num}|%
5479        \ifsublines@ \sublinenumr@p{\subline@num}\else 0\fi}%
5480        \to\labelref@list
5481    \fi}
5482  %
```

`\applabel` `\applabel`, if called in `\edtext` will insert automatically both a start and an end label for the current edtext lines.

```
5483  \newcommand*{\applabel}[1]{%
5484    \if@edtext@secondarg@%
5485  %
```

Label should not be already defined.

```
5486        \ifcsundef{the@label#1}{%
5487          \csdef{the@label#1}{applabel}%
5488        }%
5489        {%
5490          \led@warn@DuplicateLabel{#1 (applabel)}%
5491        }%
5492  %
```

Parse the \edtext line numbers.

```
5493        \expandafter\l@dp@rsefootspec\l@d@nums|%
5494  %
```

Use the LATEX standard hack for label.

```
5495        \@bsphack%
5496  %
```

And now, write the data in the auxiliary file.

```
5497        \ifledRcol%
5498          \protected@write\@auxout{}%
5499            {\string\l@dmake@labelsR\space\l@dparsedstartpage|\
              l@dparsedstartline|\l@dparsedstartsub|\the\c@pstartR|{#1:start}}%
5500          \ifdef{\hypertarget}%
5501            {\Hy@raisedlink{\hypertarget{#1:start}{}}}%
5502            {}%
5503          \protected@write\@auxout{}%
5504            {\string\l@dmake@labelsR\space\l@dparsedendpage|\l@dparsedendline
              |\l@dparsedendsub|\the\c@pstartR|{#1:end}}%
5505        \else%
5506          \protected@write\@auxout{}%
5507            {\string\l@dmake@labels\space\l@dparsedstartpage|\
              l@dparsedstartline|\l@dparsedstartsub|\the\c@pstart|{#1:start}}%
5508          \ifdef{\hypertarget}%
5509            {\Hy@raisedlink{\hypertarget{#1:start}{}}}%
5510            {}%
5511          \protected@write\@auxout{}%
5512            {\string\l@dmake@labels\space\l@dparsedendpage|\l@dparsedendline
              |\l@dparsedendsub|\the\c@pstart|{#1:end}}%
5513        \fi%
5514  %
```

Use the LATEX standard hack for label.

```
5515        \@esphack%
5516  %
```

Warning if \applabel is called outside of \edtext.

```
5517    \else%
5518      \led@warn@AppLabelOutSecondArgEdtext{#1}%
5519    \fi%
5520 %
```

End of \applabel

```
5521 }%
5522 %
```

\edlabelS   \edlabelS and \edlabelE are just used to mark the beginning and the end of a passage.
\edlabelE
\edlabelSE
```
5523 \newcommand{\edlabelS}[1]{%
5524    \edlabel{#1:start}%
5525 }
5526 \newcommand{\edlabelE}[1]{%
5527    \edlabel{#1:end}%
5528 }
5529 \newcommand{\edlabelSE}[1]{%
5530    \edlabelS{#1}%
5531    \edlabelE{#1}%
5532 }
5533 %
```

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

```
5534 \newrobustcmd{\wrap@edcrossref}[2]{%
5535    \ifdef{\hyperlink}%
5536      {\hyperlink{#1}{#2}}%
5537      {#2}%
5538 }
5539 %
```

\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.

```
5540 \newcommand*{\edpageref}[1]{\l@dref@undefined{#1}\wrap@edcrossref{#1}{\
    l@dgetref@num{1}{#1}}}
5541 \newcommand*{\xpageref}[1]{\l@dgetref@num{1}{#1}}
5542
5543 %
```

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

```
5544 \newcommand*{\edlineref}[1]{\l@dref@undefined{#1}\wrap@edcrossref{#1}{\
     l@dgetref@num{2}{#1}\xflagref{#1}}}%
5545 \newcommand*{\xlineref}[1]{\l@dgetref@num{2}{#1}}%

5546
5547 %
```

\sublineref   If the specified label exists, \sublineref gives its sub-line number.
\xsublineref
```
5548 \newcommand*{\sublineref}[1]{\l@dref@undefined{#1}\wrap@edcrossref{#1}{\
     l@dgetref@num{3}{#1}}}
5549 \newcommand*{\xsublineref}[1]{\l@dgetref@num{3}{#1}}

5550
5551 %
```

\pstarteref   If the specified label exists, \pstartref gives its pstart number.
\xpstartref
```
5552 \newcommand*{\pstartref}[1]{\l@dref@undefined{#1}\wrap@edcrossref{#1}{\
     l@dgetref@num{4}{#1}}}
5553 \newcommand*{\xpstartref}[1]{\l@dgetref@num{4}{#1}}

5554
5555 %
```

\xflagref   \xflagref finds the side flag of any ref defined with \edlabel.
```
5556 \newcommand*{\xflagref}[1]{\l@dgetref@num{5}{#1}}
5557 %
```

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.
```
5558 \newcommand*{\l@dref@undefined}[1]{%
5559   \expandafter\ifx\csname the@label#1\endcsname\relax
5560     \led@warn@RefUndefined{#1}%
5561   \fi}

5562
5563 %
```

\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), (4) pstart number or (5) side flag. (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.

```
5564 \newcommand*{\l@dgetref@num}[2]{%
5565   \expandafter
5566   \ifx\csname the@label#2\endcsname \relax
5567     000%
5568   \else
5569     \expandafter\expandafter\expandafter
5570     \l@dlabel@parse\csname the@label#2\endcsname|#1%
5571   \fi}
5572
5573 %
```

\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 fiver numbers to pick out. (It was earlier given as the first argument of \l@dgetref@num.)

```
5574 \newcommand*{\l@dlabel@parse}{}
5575 \def\l@dlabel@parse#1|#2|#3|#4|#5|#6{%
5576   \ifcase #6%
5577   \or #1%
5578   \or #2%
5579   \or #3%
5580   \or #4%
5581   \or #5%
5582   \fi}
5583 %
```

\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 cannot 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.

```
5584 \newcommand*{\xxref}[2]{%
5585   {%
5586     \expandafter\ifx\csname the@label#1\endcsname \relax%
5587       \expandafter\let\csname the@@label#1\endcsname\zz@@@%
5588     \else%
5589       \expandafter\def\csname the@@label#1\endcsname{\l@dgetref@num
5590   {1}{#1}|\l@dgetref@num{2}{#1}|\l@dgetref@num{3}{#1}}%
5590     \fi%
5591     \expandafter\ifx\csname the@label#2\endcsname \relax%
5592       \expandafter\let\csname the@@label#2\endcsname\zz@@@%
```

```
5593      \else%
5594          \expandafter\def\csname the@@label#2\endcsname{\l@dgetref@num
     {1}{#2}|\l@dgetref@num{2}{#2}|\l@dgetref@num{3}{#2}}%
5595      \fi%
5596      \letcs{\@tempa}{the@@label#1}%
5597      \letcs{\@tempb}{the@@label#2}%
5598      \linenum{\@tempa|%
5599       \@tempb}}}%

5600
5601  %
```

<div style="margin-left:2em">

\appref \appref, \SEref, \apprefwithpage, \SErefwithpage and \SEonlypage print cross-
\apprefwithpage ref to some start / end lines defined by specific commands. It prints the lines as they
\SEref should be printed in the apparatus (critical notes for not suffixed versions, endnotes for
\SErefwithpage suffixed versions).
\SErefwithpage

Here we define hooks similar to some those related to critical footnotes or endnotes.

So, first declare the default value of the hooks for the pseudo-series. Also declare
the internal toggle which are switch by reledmac.

</div>

```
5602  \def\Xtwolines@appref{}%
5603  \def\Xtwolines@SEref{}%

5604
5605  \def\Xmorethantwolines@appref{}%
5606  \def\Xmorethantwolines@SEref{}%

5607
5608  \def\Xlinerangeseparator@appref{\endashchar}%
5609  \def\Xlinerangeseparator@SEref{\endashchar}%

5610
5611  \def\Xsublinesep@appref{\fullstop}%
5612  \def\Xsublinesep@SEref{\fullstop}%

5613
5614  \newtoggle{Xtwolinesbutnotmore@appref}%
5615  \newtoggle{Xtwolinesbutnotmore@SEref}%

5616
5617  \newtoggle{Xtwolinesonlyinsamepage@appref}%

5618
5619  \newtoggle{Xtwolinesonlyinsamepage@SEref}%

5620
5621  \newtoggle{Xlineflag@appref}%
5622  \toggletrue{Xlineflag@appref}%Here exception
5623  \newtoggle{Xlineflag@SEref}%
5624  \toggletrue{Xlineflag@SEref}%%Here exception

5625
5626  \def\Xendtwolines@apprefwithpage{}%
5627  \def\Xendtwolines@SErefwithpage{}%

5628
5629  \def\Xendmorethantwolines@apprefwithpage{}%
5630  \def\Xendmorethantwolines@SErefwithpage{}%

5631
```

```
5632 \def\Xendlinerangeseparator@apprefwithpage{\endashchar}
5633 \def\Xendlinerangeseparator@SErefwithpage{\endashchar}
5634 \def\Xendlinerangeseparator@SErefonlypage{\endashchar}
5635
5636 \def\Xendbeforepagenumber@apprefwithpage{p.}%
5637 \def\Xendbeforepagenumber@SErefwithpage{p.}%
5638 \def\Xendbeforepagenumber@SEonlypage{p.}%
5639
5640 \def\Xendafterpagenumber@apprefwithpage() }%
5641 \def\Xendafterpagenumber@SErefwithpage() }%
5642
5643
5644 \def\Xendlineprefixsingle@apprefwithpage{}%
5645 \def\Xendlineprefixsingle@SErefwithpage{}%
5646
5647 \def\Xendlineprefixmore@apprefwithpage{}%
5648 \def\Xendlineprefixmore@SErefwithpage{}%
5649
5650 \newtoggle{Xendtwolinesbutnotmore@apprefwithpage}%
5651 \newtoggle{Xendtwolinesbutnotmore@SErefwithpage}%
5652
5653 \def\Xendsublinesep@apprefwithpage{\fullstop}%
5654 \def\Xendsublinesep@SErefwithpage{\fullstop}%
5655
5656 \newtoggle{Xendtwolinesonlyinsamepage@apprefwithpage}%
5657 \newtoggle{Xendtwolinesonlyinsamepage@SErefwithpage}%
5658
5659 \newtoggle{Xendlineflag@apprefwithpage}
5660 \toggletrue{Xendlineflag@apprefwithpage}%Here, exception
5661 \newtoggle{Xendlineflag@SErefwithpage}
5662 \toggletrue{Xendlineflag@SErefwithpage}%Here, exception
5663
5664 %
```

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.

```
5665
5666 \xdef\Xboxstartlinenum@appref{0pt}
5667 \xdef\Xboxstartlinenum@SEref{0pt}
5668
5669 \xdef\Xboxendlinenum@appref{0pt}
5670 \xdef\Xboxendlinenum@SEref{0pt}
5671
5672 \xdef\Xendboxstartlinenum@apprefwithpage{0pt}
5673 \xdef\Xendboxstartlinenum@SErefwithpage{0pt}
5674
5675 \xdef\Xendboxendlinenum@apprefwithpage{0pt}
5676 \xdef\Xendboxendlinenum@SErefwithpage{0pt}
5677
```

```
5678 %
```

Now, declare the default values of \@apprefprefixsingle and \@apprefprefixmore, \@SErefprefix, \@SErefprefixmore and the commands which defines them.

```
5679 \newcommand\@apprefprefixsingle{}%
5680 \newcommand\@SErefprefixsingle{}%
5681
5682 \newcommand\@apprefprefixmore{}%
5683 \newcommand\@SErefprefixmore{}%
5684
5685 \newcommand{\setapprefprefixsingle}[1]{%
5686   \gdef\@apprefprefixsingle{#1}%
5687 }
5688 \newcommand{\setSErefprefixsingle}[1]{%
5689   \gdef\@SErefprefixsingle{#1}%
5690 }
5691
5692 \newcommand{\setapprefprefixmore}[1]{%
5693   \gdef\@apprefprefixmore{#1}%
5694 }
5695 \newcommand{\setSErefprefixmore}[1]{%
5696   \gdef\@SErefprefixmore{#1}%
5697 }
5698
5699 %
```

And not \setSErefonlypageprefixsingle and \setSErefonlypageprefixmore.

```
5700 \let\setSErefonlypageprefixsingle\XendbeforepagenumberSErefonlypage%
5701 \newcommand{\setSErefonlypageprefixmore}[1]{%
5702   \gdef\SErefonlypage@prefixmore{#1}%
5703 }%
5704 %
```

And now, the main commands: \appref, \apprefwithpage, \SEref and \SErefwithpage. These commands call \refformated@ and \refformatedwithpage, which calls \printlines and \printendlines . That is why we have previously declared all hooks values tested inside these last commands.

```
5705
5706 \newcommandx{\appref}[2][1,usedefault]{\refformated@{#1}{#2}{appref}}
5707 \newcommandx{\SEref}[2][1,usedefault]{\refformated@{#1}{#2}{SEref}}
5708
5709 \newcommandx{\apprefwithpage}[2][1,usedefault]{\refformatedwithpage@
     {#1}{#2}{appref}}
5710 \newcommandx{\SErefwithpage}[2][1,usedefault]{\refformatedwithpage@
     {#1}{#2}{SEref}}
5711 \newcommandx{\SErefonlypage}[2][1,usedefault]{\refformatedonlypage@
     {#1}{#2}{SEref}}
5712
5713
```

```
5714 \newcommand{\refformated@}[3]{%
5715   \def\do##1{%
5716     \setkeys[mac]{truefootnoteoption}{##1}%
5717   }%
5718 \notblank{#1}{\docsvlist{#1}}{}%
5719 \xdef\@currentseries{#3}%
5720 \ifcsempty{@#3prefixmore}%
5721   {\@apprefprefixsingle}%
5722   {%
5723     \IfEq{\xlineref{#2:start}}{\xlineref{#2:end}}%
5724       {\csuse{@#3prefixsingle}}%
5725       {\csuse{@#3prefixmore}}%
5726   }%
5727 \ifboolexpr{%
5728   test{\ifcsundef{the@label#2:start}}%
5729   or test{\ifcsundef{the@label#2:end}}%
5730 }%
5731   {\led@warn@pairRefUndefined{#2}\nfss@text{\reset@font\bfseries ??}}%
5732   {%
5733     \def\@this@crossref@start{#2:start}%
5734     \def\@this@crossref@end{#2:end}%
5735     \printlines\xpageref{#2:start}|\xlineref{#2:start}|\xsublineref{#2:
   start}|\xpageref{#2:end}|\xlineref{#2:end}|\xsublineref{#2:end}|\relax|\
   xflagref{#2:start}|%
5736     \undef\@this@crossref@end%
5737     \undef\@this@crossref@start%
5738   }%
5739   \def\do##1{%
5740     \setkeys[mac]{falsefootnoteoption}{##1}%
5741   }%
5742 \notblank{#1}{\docsvlist{#1}}{}%
5743 }%
5744
5745 \newcommand{\refformatedwithpage@}[3]{%
5746   \def\do##1{%
5747     \setkeys[mac]{truefootnoteoption}{##1}%
5748   }%
5749 \notblank{#1}{\docsvlist{#1}}{}%
5750 \xdef\@currentseries{#3withpage}%
5751 \ifboolexpr{%
5752   test{\ifcsundef{the@label#2:start}}%
5753   or test{\ifcsundef{the@label#2:end}}%
5754 }%
5755   {\led@warn@pairRefUndefined{#2}\nfss@text{\reset@font\bfseries ??}}%
5756   {%
5757     \def\@this@crossref@start{#2:start}%
5758     \def\@this@crossref@end{#2:end}%
5759     \printendlines\xpageref{#2:start}|\xlineref{#2:start}|\xsublineref{#2:
   start}|\xpageref{#2:end}|\xlineref{#2:end}|\xsublineref{#2:end}|\relax|\
   xflagref{#2:start}|%
```

```
5760      \undef\@this@crossref@end%
5761      \undef\@this@crossref@start%
5762    }%
5763    \def\do##1{%
5764      \setkeys[mac]{falsefootnoteoption}{##1}%
5765    }%
5766    \notblank{#1}{\docsvlist{#1}}{}%
5767 }%
5768
5769 \newcommand{\refformatedonlypage@}[3]{%
5770    \def\do##1{%
5771      \setkeys[mac]{truefootnoteoption}{##1}%
5772    }%
5773    \notblank{#1}{\docsvlist{#1}}{}%
5774    \xdef\@currentseries{#3onlypage}%
5775    \ifboolexpr{%
5776      test{\ifcsundef{the@label#2:start}}%
5777      or test{\ifcsundef{the@label#2:end}}%
5778    }%
5779      {\led@warn@pairRefUndefined{#2}\nfss@text{\reset@font\bfseries ??}}%
5780      {\ifnumequal{\xpageref{#2:end}}{\xpageref{#2:start}}%
5781          {%
5782          \printnpnum{%
5783            \wrap@edcrossref{#2:start}{\xpageref{#2:start}}%
5784            }%
5785          }%
5786        {%
5787          \ifcsvoid{#3onlypage@prefixmore}%
5788            {}%
5789            {\csletcs{Xendbeforepagenumber@#3onlypage}{#3onlypage@prefixmore}}%
5790          \ifdefined\linerangesep@%
5791            \printnpnum{%
5792              \wrap@edcrossref{#2:start}{\xpageref{#2:start}}%
5793              \linerangesep@%
5794              \wrap@edcrossref{#2:end}{\xpageref{#2:end}}%
5795            }%
5796          \else%
5797            \printnpnum{%
5798              \wrap@edcrossref{#2:start}{\xpageref{#2:start}}%
5799              \csuse{Xendlinerangeseparator@\@currentseries}%
5800              \wrap@edcrossref{#2:end}{\xpageref{#2:end}}%
5801            }%
5802          \fi%
5803        }%
5804    }%
5805    \def\do##1{%
5806      \setkeys[mac]{falsefootnoteoption}{##1}%
5807    }%
5808    \notblank{#1}{\docsvlist{#1}}{}%
5809 }%
```

```
5810  %
```

\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 insert \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.

```
5811  \newcommand*{\edmakelabel}[2]{\expandafter\xdef\csname the@label#1\
      endcsname{#2}}
5812
5813  %
```

(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. 123 and V.9 p. 92), since \xxref makes
a call to \linenum in order to do its work.)

XXIII.1   Compatibility with `xref`

Here, we provide compatibility with the xref to enable reledmac's cross-referencing
to external documents. We assume that the user loads xref *before* reledmac, but uses
\externaldocument *after* loading reledmac.

\XR@test   First, we patch the xr macro \XR@test, which is called on every line of the external
.aux file, in order to also call macros specific to reledmac.

```
5814  \pretocmd{\XR@test}%
5815    {\XR@test@mac+++#1#2#3#4+++}%
5816    {}%
5817    {}%
5818  %
```

\XR@test@mac   The \XR@test@mac takes the full content of a line of the external .aux files, with the
three final dots added by xr.

```
5819  \long\def\XR@test@mac+++#1+++{\XR@test@mac@test#1}
5820  %
```

\XR@test@mac@test   And finally, \XR@test@mac@test does the job. This code is based on the \XR@test
macro of the xr package. However, not that the \XR@prefix is not called here, but it is
integrated directly in \l@dmake@labels and \l@dmake@labelsR.

```
5821  \long\def\XR@test@mac@test#1#2...{%The triple dots (NOT \ldots) are because
      of the line 22 of xr.sty  v5.02 1994/05/28
5822    \ifx#1\l@dmake@labels%
5823      \l@dmake@labels#2%
5824    \else
```

```
5825      \ifx#1\l@dmake@labelsR%
5826        \l@dmake@labelsR #2%
5827      \fi%
5828    \fi%
5829 }%
5830 %
```

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.

```
5831 \pretocmd{\@xympar}%
5832    {\ifnumberedpar@
5833       \led@warn@NoMarginpars
5834       \@esphack
5835    \else}%
5836    {}%
5837    {}%
5838
5839 \apptocmd{\@xympar}%
5840    {\fi}%
5841    {}
5842    {}
5843
5844 %
```

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

```
5845 \newcount\sidenote@margin
5846 \newcommand*{\sidenotemargin}[1]{{%
5847    \l@dgetsidenote@margin{#1}%
5848    \ifnum\@l@dtempcntb>\m@ne
```

```
5849      \ifledRcol
5850        \global\sidenote@marginR=\@l@dtempcntb
5851      \else
5852        \global\sidenote@margin=\@l@dtempcntb
5853      \fi
5854    \fi}}
5855  \newcommand*{\l@dgetsidenote@margin}[1]{%
5856    \def\@tempa{#1}\def\@tempb{left}%
5857    \ifx\@tempa\@tempb
5858      \@l@dtempcntb \z@
5859    \else
5860      \def\@tempb{right}%
5861      \ifx\@tempa\@tempb
5862        \@l@dtempcntb \@ne
5863      \else
5864        \def\@tempb{outer}%
5865        \ifx\@tempa\@tempb
5866          \@l@dtempcntb \tw@
5867        \else
5868          \def\@tempb{inner}%
5869          \ifx\@tempa\@tempb
5870            \@l@dtempcntb \thr@@
5871          \else
5872            \led@warn@BadSidenotemargin
5873            \@l@dtempcntb \m@ne
5874          \fi
5875        \fi
5876      \fi
5877    \fi}
5878  \sidenotemargin{right}
5879
5880  %
```

`\l@dlp@rbox`
`\l@drp@rbox` We need two boxes to store sidenote texts.

```
5881  \newbox\l@dlp@rbox
5882  \newbox\l@drp@rbox
5883
5884  %
```

`\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`

```
5885  \newdimen\ledlsnotewidth \ledlsnotewidth=\marginparwidth
5886  \newdimen\ledrsnotewidth \ledrsnotewidth=\marginparwidth
5887  \newdimen\ledlsnotesep \ledlsnotesep=\linenumsep
5888  \newdimen\ledrsnotesep \ledrsnotesep=\linenumsep
5889  \newcommand*{\ledlsnotefontsetup}{\raggedleft\footnotesize}
5890  \newcommand*{\ledrsnotefontsetup}{\raggedright\footnotesize}
5891
```

5892 %

\ledleftnote
\ledrightnote
\ledinnernote
\ledouterote
\ledsidenote

\ledleftnote, \ledrightnote, \ledinnernote, \ledouternote are the user commands for left, right, inner and outer sidenotes. The two last one are just alias for the two first one, depending of the page number. \ledsidenote{⟨*text*⟩} is the command for a moveable sidenote.

```
5893 \newcommand*{\ledleftnote}[1]{\edtext{}{\l@dlsnote{#1}}}
5894 \newcommand*{\ledrightnote}[1]{\edtext{}{\l@drsnote{#1}}}
5895 \newcommand*{\ledsidenote}[1]{\edtext{}{\l@dcsnote{#1}}}%
5896 \newcommand*{\ledinnernote}[1]{\edtext{}{\l@disnote{#1}}}%
5897 \newcommand*{\ledouternote}[1]{\edtext{}{\l@dosnote{#1}}}%
5898 %
```

\l@dlsnote
\l@drsnote
\l@dcsnote
\l@desnote
\l@disnote

. The 'footnotes' for left, right, and moveable sidenotes. The whole scheme is reminiscent of the critical footnotes code.

```
5899 \newif\ifrightnoteup
5900   \rightnoteuptrue
5901
5902 \newcommand*{\l@dlsnote}[1]{%
5903   \begingroup%
5904   \newcommand{\content}{#1}%
5905   \ifnumberedpar@
5906     \ifledRcol%
5907       \xright@appenditem{\noexpand\vl@dlsnote{\expandonce\content}}%
5908                           \to\inserts@listR
5909       \global\advance\insert@countR \@ne%
5910     \else%
5911       \xright@appenditem{\noexpand\vl@dlsnote{\expandonce\content}}%
5912                           \to\inserts@list
5913       \global\advance\insert@count \@ne%
5914     \fi
5915   \fi%
5916   \ignorespaces%
5917   \endgroup%
5918 }%
5919
5920 \newcommand*{\l@drsnote}[1]{%
5921   \begingroup%
5922   \newcommand{\content}{#1}%
5923   \ifnumberedpar@
5924     \ifledRcol%
5925       \xright@appenditem{\noexpand\vl@drsnote{\expandonce\content}}%
5926                           \to\inserts@listR
5927       \global\advance\insert@countR \@ne%
5928     \else%
5929       \xright@appenditem{\noexpand\vl@drsnote{\expandonce\content}}%
5930                           \to\inserts@list
5931       \global\advance\insert@count \@ne%
```

```
5932      \fi
5933    \fi\ignorespaces%
5934    \endgroup%
5935  }%
5936
5937  \newcommand*{\l@dcsnote}[1]{%
5938    \begingroup%
5939    \newcommand{\content}{#1}%
5940    \ifnumberedpar@
5941      \ifledRcol%
5942        \xright@appenditem{\noexpand\vl@dcsnote{\expandonce\content}}%
5943                          \to\inserts@listR
5944        \global\advance\insert@countR \@ne%
5945      \else%
5946        \xright@appenditem{\noexpand\vl@dcsnote{\expandonce\content}}%
5947                          \to\inserts@list
5948        \global\advance\insert@count \@ne%
5949      \fi
5950    \fi\ignorespaces%
5951    \endgroup%
5952  }%
5953
5954  \newcommand*{\l@disnote}[1]{%
5955    \begingroup%
5956    \newcommand{\content}{#1}%
5957    \ifnumberedpar@%
5958      \ifledRcol%
5959        \xright@appenditem{\noexpand\vl@disnote{\expandonce\content}}%
5960                          \to\inserts@listR%
5961        \global\advance\insert@countR \@ne%
5962      \else%
5963        \xright@appenditem{\noexpand\vl@disnote{\expandonce\content}}%
5964                          \to\inserts@list%
5965        \global\advance\insert@count \@ne%
5966      \fi%
5967    \fi\ignorespaces%
5968    \endgroup%
5969  }%
5970
5971  \newcommand*{\l@dosnote}[1]{%
5972    \begingroup%
5973    \newcommand{\content}{#1}%
5974    \ifnumberedpar@%
5975      \ifledRcol%
5976        \xright@appenditem{\noexpand\vl@dosnote{\expandonce\content}}%
5977                          \to\inserts@listR%
5978        \global\advance\insert@countR \@ne%
5979      \else%
5980        \xright@appenditem{\noexpand\vl@dosnote{\expandonce\content}}%
5981                          \to\inserts@list%
```

```
5982        \global\advance\insert@count \@ne%
5983      \fi%
5984    \fi\ignorespaces%
5985    \endgroup%
5986 }%
5987
5988 %
```

\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
\vl@disnote   them, in case many sidenote co-exist for the same line. That is there some special test
\vl@dosnote   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.

\vl@disnote and \vl@dosnote just call \vl@dlsnote or \vl@drsnote, depending
of the page.

```
5989 \newcommand*{\vl@dlsnote}[1]{%
5990    \ifledRcol@%
5991      \@l@dtempcntb=\sidenote@marginR%
5992      \ifnum\@l@dtempcntb>\@ne%
5993        \advance\@l@dtempcntb by\page@numR%
5994      \fi%
5995    \else%
5996      \@l@dtempcntb=\sidenote@margin%
5997      \ifnum\@l@dtempcntb>\@ne%
5998        \advance\@l@dtempcntb by\page@num%
5999      \fi%
6000    \fi%
6001    \ifodd\@l@dtempcntb%
6002      \listgadd{\l@dcsnotetext@l}{#1}%
6003    \else%
6004      \listgadd{\l@dcsnotetext}{#1}%
6005    \fi
6006 }
6007 \newcommand*{\vl@drsnote}[1]{%
6008    \ifledRcol@%
6009      \@l@dtempcntb=\sidenote@marginR%
```

```
6010        \ifnum\@l@dtempcntb>\@ne%
6011          \advance\@l@dtempcntb by\page@numR%
6012        \fi%
6013      \else%
6014        \@l@dtempcntb=\sidenote@margin%
6015        \ifnum\@l@dtempcntb>\@ne%
6016          \advance\@l@dtempcntb by\page@num%
6017        \fi%
6018      \fi%
6019      \ifodd\@l@dtempcntb%
6020        \listgadd{\l@dcsnotetext}{#1}%
6021      \else%
6022        \listgadd{\l@dcsnotetext@r}{#1}%
6023      \fi%
6024    }
6025    \newcommand*{\vl@dcsnote}[1]{\listgadd{\l@dcsnotetext}{#1}}
6026
6027    \newcommand{\vl@disnote}[1]{%
6028      \ifledRcol@%
6029        \@tempcnta=\page@numR%
6030      \else%
6031        \@tempcnta=\page@num%
6032      \fi%
6033      \ifodd\@tempcnta% ODD => right page => inner side =  left side
6034        \vl@dlsnote{#1}%
6035      \else%
6036        \vl@drsnote{#1}%
6037      \fi%
6038    }%
6039
6040    \newcommand{\vl@dosnote}[1]{%
6041      \ifledRcol@%
6042        \@tempcnta=\page@numR%
6043      \else%
6044        \@tempcnta=\page@num%
6045      \fi%
6046      \ifodd\@tempcnta% ODD => right page => outer side = right side
6047        \vl@drsnote{#1}%
6048      \else%
6049        \vl@dlsnote{#1}%
6050      \fi%
6051    }%
6052
6053    %
```

\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.

```
6054    \newcommand*{\setl@dlp@rbox}[1]{%
```

```
6055     \begingroup%
6056       \parindent\z@\hsize=\ledlsnotewidth%
6057       \ledlsnotefontsetup%We kept it outside of the vbox, because can affect
    the ragging
6058       \global\setbox\l@dlp@rbox%
6059       \ifleftnoteup%
6060         =\vbox to\z@{{\ledlsnotefontsetup\vss #1}}%We put \
    ledlsnotefontsetup inside footnote because required for color command. Note
    the {} to keep setting local.
6061       \else%
6062         =\vbox to 0.70\baselineskip{{\ledlsnotefontsetup\strut#1\vss}}%
6063       \fi%
6064     \endgroup%
6065 }
6066
6067 \newcommand*{\setl@drp@rbox}[1]{%
6068   \begingroup%
6069       \parindent\z@\hsize=\ledrsnotewidth%
6070       \ledrsnotefontsetup%We kept it outside of the vbox, because can affect
    the ragging
6071       \global\setbox\l@drp@rbox%
6072       \ifrightnoteup%
6073         =\vbox to\z@{{\ledrsnotefontsetup\vss#1}}%We put \ledrsnotefontsetup
    inside footnote because required for color command. Note the {} to keep
    setting local.
6074       \else%
6075         =\vbox to0.7\baselineskip{{\ledrsnotefontsetup\strut#1\vss}}%
6076       \fi%
6077   \endgroup%
6078 }%
6079 \newif\ifleftnoteup
6080   \leftnoteuptrue
6081 %
```

\@sidenotesep   This macro is used to separate sidenotes of the same line.

```
6082 \newcommand{\setsidenotesep}[1]{\gdef\@sidenotesep{#1}}
6083 \newcommand{\@sidenotesep}{, }
6084 %
```

\affixside@note   This macro puts any moveable sidenote text into the left or right sidenote box, depend-
ing 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.

```
6085 \newcommand*{\affixside@note}{%
6086     \def\sidenotecontent@{}%
6087     \numgdef{\itemcount@}{0}%
6088     \def\do##1{%
```

```
6089        \ifnumequal{\itemcount@}{0}%
6090            {%
6091            \appto\sidenotecontent@{##1}}% Not print not separator before
    the 1st note
6092            {\appto\sidenotecontent@{\@sidenotesep ##1}%
6093            }%
6094            \numgdef{\itemcount@}{\itemcount@+\@ne}%
6095        }%
6096    \dolistloop{\l@dcsnotetext}%
6097    \ifnumgreater{\itemcount@}{1}{\led@err@ManySidenotes}{}%
6098 %
```

And we do the same for left and right notes (not movable).

```
6099    \gdef\@templ@d{}%
6100    \gdef\@templ@n{\l@dcsnotetext\l@dcsnotetext@l\l@dcsnotetext@r}%
6101    \ifx\@templ@d\@templ@n \else%
6102        \if@twocolumn%
6103            \if@firstcolumn%
6104                \setl@dlp@rbox{##1}{\sidenotecontent@}%
6105            \else%
6106                \setl@drp@rbox{\sidenotecontent@}%
6107            \fi%
6108        \else%
6109            \@l@dtempcntb=\sidenote@margin%
6110            \ifnum\@l@dtempcntb>\@ne%
6111                \advance\@l@dtempcntb by\page@num%
6112            \fi%
6113            \ifodd\@l@dtempcntb%
6114                \setl@drp@rbox{\sidenotecontent@}%
6115                \gdef\sidenotecontent@{}%
6116                \numgdef{\itemcount@}{0}%
6117                \dolistloop{\l@dcsnotetext@l}%
6118                \ifnumgreater{\itemcount@}{1}{\led@err@ManyLeftnotes}{}%
6119                \setl@dlp@rbox{\sidenotecontent@}%
6120            \else%
6121                \setl@dlp@rbox{\sidenotecontent@}%
6122                \gdef\sidenotecontent@{}%
6123                \numgdef{\itemcount@}{0}%
6124                \dolistloop{\l@dcsnotetext@r}%
6125                \ifnumgreater{\itemcount@}{1}{\led@err@ManyRightnotes}{}%
6126                \setl@drp@rbox{\sidenotecontent@}%
6127            \fi%
6128        \fi%
6129    \fi%
6130 }
6131 %
```

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.

```
6132 \ifnoledgroup@\else%
6133 \newcommand*{\l@dfeetbeginmini}{\@ledgrouptrue\l@dedbeginmini\
     l@dfambeginmini}
6134 \newcommand*{\l@dfeetendmini}{%
6135     \IfStrEq{critical-familiar}{\@mpfnpos}%
6136         {\l@dedendmini\l@dfamendmini}%
6137         {%
6138             \IfStrEq{familiar-critical}{\@mpfnpos}%
6139                 {\l@dfamendmini\l@dedendmini}%
6140                 {\l@dedendmini\l@dfamendmini}%
6141         }%
6142     }%
6143 %
```

`\l@dedbeginmini` These handle the initiation and closure of critical footnotes in a minipage environment.
`\l@dedendmini`
```
6144 \newcommand*{\l@dedbeginmini}{%
6145   \unless\ifnocritical@%
6146     \def\do##1{\csletcs{v##1footnote}{mpv##1footnote}}%
6147     \dolistloop{\@series}%
6148   \fi%
6149   }
6150 \newcommand*{\l@dedendmini}{%
6151   \unless\ifnocritical@%
6152     \ifl@dpairing%
6153       \ifledRcol%
6154         \flush@notesR%
6155       \else%
6156         \flush@notes%
6157       \fi%
6158     \fi
6159     \def\do##1{%
6160       \ifvoid\csuse{mp##1footins}\else%
6161         \ifl@dpairing\ifparledgroup%
6162           \ifledRcol%
6163             \dimgdef{\parledgroup@beforenotesR}{\parledgroup@beforenotesR+\
     skip\@nameuse{mp##1footins}}%
6164           \else%
6165             \dimgdef{\parledgroup@beforenotesL}{\parledgroup@beforenotesL
     +\skip\@nameuse{mp##1footins}}%
```

```
6166          \fi%
6167          \fi\fi%
6168          \csuse{mp##1footgroup}{##1}%
6169      \fi}%
6170      \dolistloop{\@series}%
6171    \fi%
6172 }%
6173
6174 %
```

\l@dfambeginmini These handle the initiation and closure of familiar footnotes in a minipage environment.
\l@dfamendmini

```
6175 \newcommand*{\l@dfambeginmini}{%
6176  \unless\ifnofamiliar@%
6177    \def\do##1{\csletcs{vfootnote##1}{mpvfootnote##1}}%
6178      \dolistloop{\@series}%
6179  \fi%
6180 }%
6181
6182 \newcommand*{\l@dfamendmini}{%
6183  \unless\ifnofamiliar@%
6184    \def\do##1{%
6185      \ifvoid\csuse{mpfootins##1}\else%
6186      \csuse{mpfootgroup##1}{##1}%
6187    \fi}%
6188    \dolistloop{\@series}%
6189  \fi%
6190 }%
6191 %
```

\@iiiminipage This is our extended form of the kernel \@iiiminipage defined in ltboxes.dtx.

```
6192 \patchcmd%
6193    {\@iiiminipage}%
6194    {\let\@footnotetext\@mpfootnotetext}%
6195    {\let\@footnotetext\@mpfootnotetext\l@dfeetbeginmini}%
6196    {}%
6197    {\led@error@fail@patch@@iiiminipage}%
6198 %
```

\endminipage This is our extended form of the kernel \endminipage defined in ltboxes.dtx.

```
6199 \patchcmd%
6200    {\endminipage}%
6201    {\footnoterule}%
6202    {\footnoterule\l@advance@parledgroup@beforenormalnotes}%
6203    {}%
6204    {\led@error@fail@patch@endminipage}
6205
6206 \patchcmd%
```

```
6207    {\endminipage}%
6208    {\@minipagefalse}%
6209    {\l@dfeetendmini\@minipagefalse}%
6210    {}%
6211    {\led@error@fail@patch@endminipage}
6212
6213  %
```

`\l@dunboxmpfoot`
`\l@advance@parledgroup@beforenormalnotes`

\@ldunboxmpfoot insert normal footnotes for ledgroup.

```
6214  \newcommand*{\l@dunboxmpfoot}{%
6215      \vskip\skip\@mpfootins
6216      \normalcolor
6217      \footnoterule
6218      \l@advance@parledgroup@beforenormalnotes
6219      \unvbox\@mpfootins%
6220  }
6221  %
```

When using parallel ledgroup, we need to store the vertical space added before footnote, in order to compensate them between left and right pages.

```
6222  \newcommand{\l@advance@parledgroup@beforenormalnotes}{%
6223    \ifparledgroup
6224      \ifl@dpairing
6225        \ifledRcol
6226            \dimgdef{\parledgroup@beforenotesR}{\parledgroup@beforenotesR+\
      skip\@mpfootins}
6227        \else
6228            \dimgdef{\parledgroup@beforenotesL}{\parledgroup@beforenotesL+\
      skip\@mpfootins}
6229        \fi
6230      \fi
6231    \fi
6232  }
6233  %
```

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.

```
6234
6235  \newenvironment{ledgroup}{%
6236    \resetprevpage@num%
6237    \def\@mpfn{mpfootnote}\def\thempfn{\thempfootnote}\c@mpfootnote\z@%
6238    \let\@footnotetext\@mpfootnotetext
6239    \l@dfeetbeginmini%
6240  }{%
6241    \par
6242    \unskip
```

```
6243    \ifvoid\@mpfootins\else
6244      \l@dunboxmpfoot
6245    \fi
6246    \l@dfeetendmini%
6247    \@ledgroupfalse%
6248  }
6249

6250
6251  %
```

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.

```
6252  \newenvironment{ledgroupsized}[2][l]{%
6253  %
```

Set the various text measures.

```
6254    \hsize #2\relax
6255  %
```

Initialize fills for centering.

```
6256    \let\ledllfill\hfil
6257    \let\ledrlfill\hfil
6258    \def\@tempa{#1}\def\@tempb{l}%
6259  %
```

Left adjusted numbered lines

```
6260        \ifx\@tempa\@tempb
6261      \let\ledllfill\relax
6262    \else
6263      \def\@tempb{r}%
6264      \ifx\@tempa\@tempb
6265  %
```

Right adjusted numbered lines

```
6266        \let\ledrlfill\relax
6267      \fi
6268    \fi
6269  %
```

Set up the footnoting.

```
6270    \def\@mpfn{mpfootnote}\def\thempfn{\thempfootnote}\c@mpfootnote\z@
6271    \let\@footnotetext\@mpfootnotetext
6272    \l@dfeetbeginmini%
6273  }{%
6274    \par
6275    \unskip
6276    \ifvoid\@mpfootins\else
```

```
6277        \l@dunboxmpfoot
6278    \fi
6279    \l@dfeetendmini%
6280 }
6281
6282 %
```

Close the \ifnoledgroup@\else.

```
6283 \fi%
6284 %
```

\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`.

```
6285 \newif\ifledgroupnotesL@
6286 \newif\ifledgroupnotesR@
6287 %
```

XXVI   Indexing

Here is some code for indexing using page and line numbers.

XXVI.1   Looking on package order

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

```
6288 \AtBeginDocument{%
6289    \unless\ifl@imakeidx%
6290       \@ifpackageloaded{imakeidx}{\led@error@ImakeidxAfterEledmac}{}%
6291    \fi%
6292    \unless\ifl@indextools%
6293       \@ifpackageloaded{indextools}{\led@error@indextoolsAfterEledmac}{}%
6294    \fi%
6295 }
6296 %
```

XXVI.2   Auxiliary macros for \edindex

\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
```
6297 \newcommand{\pagelinesep}{-}
6298 \newcommand{\edindexlab}{$&}
6299 \newcounter{labidx}
6300 \setcounter{labidx}{0}
6301
6302 %
```

`\doedindexlabel`   This macro sets an `\edlabel`.

```
6303 \newcommand{\doedindexlabel}{%
6304   \stepcounter{labidx}%
6305   \edlabel{\edindexlab\thelabidx}%
6306 }
6307
6308 %
```

`\thepageline`   This macro makes up the page/line number combo from the label/ref. The associated counter is never directly used, but it is required in order to not have any error message with `\edgls`.

```
6309 \newcounter{pageline}%
6310 \renewcommand{\thepageline}{%
6311   \thepage%
6312   \pagelinesep%
6313   \xlineref{\edindexlab\thelabidx}%
6314 }
6315 %
```

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

```
6316 \newcommand{\thestartpageline}{%
6317   \l@dparsedstartpage%
6318   \pagelinesep%
6319   \l@dparsedstartline%
6320 }
6321 \newcommand{\theendpageline}{%
6322   \l@dparsedendpage%
6323   \pagelinesep%
6324   \l@dparsedendline%
6325 }
6326 %
```

XXVI.3   Code specific to `\edindex` in critical footnotes

`\if@edindex@fornote@true`   This boolean test is switching at the beginning of each critical note, to allow index re-
ferring to this note.

```
6327 \newif\if@edindex@fornote@
6328 %
```

`\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 |.

```
6329 \newcommand{\prepare@edindex@fornote}[1]{%
6330     \l@dp@rsefootspec#1|%
6331     \@edindex@fornote@true%
6332 }
6333 %
```

\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.

```
6334 \newcommand{\get@edindex@ledinnote@command}{%
6335   \ifxindy@%
6336     \gdef\@ledinnote@command{%
6337       ledinnote\thelabidx%
6338     }%
6339     \ifxindyhyperref@%
6340       \immediate\write\eledmac@xindy@out{%
6341         (define-attributes ("ledinnote\thelabidx"))^^J
6342         \space\space(markup-locref^^J
6343           \eledmacmarkuplocrefdepth^^J
6344           :open "\string\ledinnote[\edindexlab\thelabidx]{\@index@command
    }{"^^J
6345           :close "}"^^J
6346           :attr "ledinnote\thelabidx"^^J
6347           )
6348         }%
6349     \else%
6350       \immediate\write\eledmac@xindy@out{%
6351         (define-attributes ("ledinnote\thelabidx"))^^J
6352         \space\space(markup-locref^^J
6353           \eledmacmarkuplocrefdepth^^J
6354           :open "\string\ledinnote{\@index@command}{"^^J
6355           :close "}"^^J
6356           :attr "ledinnote\thelabidx"^^J
6357           )
6358         }%
6359     \fi%
6360 %
```

If we do not use xindy option, \@ledinnote@command will produce something like
ledinnote{formattingcommand}.

```
6361   \else%
6362     \gdef\@ledinnote@command{%
6363       ledinnote[\edindexlab\thelabidx]{\@index@command}%
6364     }%
6365   \fi%
6366 }
6367 %
```

XXVI.4   Analysis of command in indexed text

`\get@index@command` This macro is used to analyze if a text to be indexed has a command after a |.

```
6368 \def\get@index@command#1|#2+{%
6369   \gdef\@index@txt{#1}%
6370   \gdef\@index@command{#2}%
6371   \xdef\@index@parenthesis{}%
6372   \IfBeginWith{\@index@command}{(}{%
6373     \StrGobbleLeft{\@index@command}{1}[\@index@command@]%
6374     \global\let\@index@command\@index@command@%
6375     \xdef\@index@parenthesis{(}%
6376     }{}%
6377   \IfBeginWith{\@index@command}{)}{%
6378     \StrGobbleLeft{\@index@command}{1}[\@index@command@]%
6379     \global\let\@index@command\@index@command@%
6380     \xdef\@index@parenthesis{)}%
6381     }{}%
6382 }
6383 %
```

XXVI.5   Code for the formatted index

`\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.

```
6384 \newcommandx\ledinnote[3][1,usedefault]{%
6385   \ifboolexpr{%
6386     test{\ifdefequal{\iftrue}{\ifHy@hyperindex}}%
6387     or%
6388     bool {xindyhyperref@}%
6389     }%
6390     {%
6391     \csuse{#2}{\hyperlink{#1}{\ledinnotemark{#3}}}%
6392     }%
6393     {%
6394     \csuse{#2}{\ledinnotemark{#3}}%
6395     }%
6396 }%
6397 \newcommand{\ledinnotehyperpage}[2]{\csuse{#1}{\ledinnotemark{\hyperpage
     {#2}}}}%
6398 \newcommand{\ledinnotemark}[1]{#1\emph{n}}%
6399 %
```

XXVI.6   Main code

`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

```
6400 \newcommandx{\@wredindex}[2][1=\expandonce\jobname,usedefault]{%#1 = the
      index name, #2 = the text
6401   \ifl@imakeidx%
6402     \if@edindex@fornote@%
6403       \IfSubStr[1]{#2}{|}{\get@index@command#2+}{\get@index@command#2|+}%
6404       \get@edindex@ledinnote@command%
6405       \expandafter\imki@wrindexentry{#1}{\@index@txt|(\@ledinnote@command
      }{\thestartpageline}%
6406       \expandafter\imki@wrindexentry{#1}{\@index@txt|)\@ledinnote@command
      }{\theendpageline}%
6407     \else%
6408       \get@edindex@hyperref{#2}%
6409       \imki@wrindexentry{#1}{\@index@txt\@edindex@hyperref}{\thepageline}%
6410     \fi%
6411   \else%
6412     \if@edindex@fornote@%
6413       \IfSubStr[1]{#2}{|}{\get@index@command#2+}{\get@index@command#2|+}%
6414       \get@edindex@ledinnote@command%
6415       \expandafter\protected@write\@indexfile{}%
6416   {\string\indexentry{\@index@txt|(\@ledinnote@command}{\thestartpageline}
6417   }%
6418       \expandafter\protected@write\@indexfile{}%
6419   {\string\indexentry{\@index@txt|)\@ledinnote@command}{\theendpageline}
6420       }%
6421     \else%
6422       \protected@write\@indexfile{}%
6423   {\string\indexentry{#2}{\thepageline}
6424   }%
6425     \fi%
6426   \fi%
6427   \endgroup
6428   \@esphack%
6429 }
6430 %
```

Need to add the definition of \edindex to \makeindex, and initialise \edindex to do nothing.

```
6431 \pretocmd{\makeindex}{%
6432   \def\edindex{\@bsphack
6433   \doedindexlabel
6434   \begingroup
```

```
6435    \@sanitize
6436    \@wredindex}}{}{}
6437 \newcommand{\edindex}[1]{\@bsphack\@esphack}
6438 %
```

XXVI.7   Hyperlink

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

```
6439 \newcommand{\hyperlinkformat}[3]{%
6440    \ifstrempty{#1}%
6441       {\hyperlink{#2}{#3}}%
6442       {\csuse{#1}{\hyperlink{#2}{#3}}%
6443    }}
6444 %
```

\hyperlinkR   \hyperlinkR command is to be used to create a internal hyperlink and \ledRflag, when indexing.

```
6445 \newcommand{\hyperlinkR}[2]{%
6446  \hyperlink{#1}{#2\@Rlineflag}%
6447 }%
6448
6449 %
```

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

```
6450 \newcommand{\hyperlinkformatR}[3]{%
6451    \hyperlinkformat{#1}{#2}{#3\@Rlineflag}%
6452 }%
6453
6454 %
```

\get@edindex@hyperref   \get@edindex@hyperref is to be used to define the \@edindex@hyperref macro,
\@edindex@hyperref   which, in index, links to the point where the index was called (with hyperref.

```
6455 \newcommand{\get@edindex@hyperref}[1]{%
6456 %
```

We have to disable temporary spaces to work through a xstring bug (or feature?)

```
6457    \edef\temp@{%
6458    \catcode`\ =9 %space need for catcode
6459    \detokenize{#1}%For active character in unicode
6460    \catcode`\ =10 % space need for catcode
6461    }%
6462 %
```

Now, we define \@edindex@hyperref if the hyperindex of hyperref is enabled.

```
6463    \ifdefequal{\iftrue}{\ifHy@hyperindex}{%
6464      \IfSubStr{\temp@}{|}%
6465        {\get@index@command#1+%
6466        \ifledRcol%
6467          \gdef\@edindex@hyperref{|\@index@parenthesis %space kept
6468          hyperlinkformatR{\@index@command}%
6469          {\edindexlab\thelabidx}}%
6470        \else%
6471          \gdef\@edindex@hyperref{|\@index@parenthesis %space kept
6472          hyperlinkformat{\@index@command}%
6473          {\edindexlab\thelabidx}}%
6474        \fi%
6475        }%
6476        {\get@index@command#1|+%
6477         \ifledRcol%
6478         \gdef\@edindex@hyperref{|hyperlinkR{\edindexlab\thelabidx}}%
6479         \else%
6480         \gdef\@edindex@hyperref{|hyperlink{\edindexlab\thelabidx}}%
6481         \fi%
6482        }%
6483      }%
6484 %
```

```
6485 % If we use both xindy and hyperref, first get the \protect\cs{
     index@command} command.
6486 % Then define \protect\cs{@edindex@hyperref} in the form \verb+eledmacXXX+
6487 %     \begin{macrocode}
6488      {\ifxindyhyperref@%
6489        \IfSubStr{\temp@}{|}%
6490          {\get@index@command#1+}%
6491          {\get@index@command#1|+}%
6492        \gdef\@edindex@hyperref{|eledmac\thelabidx}%
6493 %
```

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.

```
6494        \IfStrEq{\@index@parenthesis}{(}%
6495          {%
6496          \csxdef{xindyparenthesis@\@index@txt}{\thelabidx}%
6497          \gdef\@edindex@hyperref{|(eledmac\thelabidx}%
6498          }%
6499          {}%
6500 %
```

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.

```
6501        \IfStrEq{\@index@parenthesis}{)}%
```

```
6502          {%
6503          \xdef\@edindex@hyperref{|)eledmac\csuse{xindyparenthesis@\
      @index@txt}}%
6504          \global\csundef{xindyparenthesis@\@index@txt}%
6505          }%
6506 %
```

Write in the .xdy file the attributes of the location.

```
6507          {%
6508          \immediate\write\eledmac@xindy@out{%
6509            (define-attributes ("eledmac\thelabidx"))^^J
6510            \space\space(markup-locref^^J
6511              \eledmacmarkuplocrefdepth^^J
6512            :open "\string\hyperlink%
6513                  \ifledRcol R\fi%
6514                  {\edindexlab\thelabidx}%
6515                  {\ifdefempty{\@index@command}%
6516                    {}%
6517                    {\@backslashchar\@index@command}%
6518                  {"^^J
6519            :close "}}"^^J
6520            :attr "eledmac\thelabidx"^^J
6521            )
6522          }%
6523          }%
6524 %
```

And now, in any other case.

```
6525      \else%
6526        \gdef\@index@txt{#1}%
6527        \gdef\@edindex@hyperref{}%
6528      \fi%
6529      }%
6530 }
6531 %
```

XXVI.8 'innote' and 'notenumber' option of `indextols` package

`\led@set@index@fornote` The `\led@set@index@fornote` is called when a familiar footnote is inserted — and not when it is read — and changes the `\index` command depending of the option of the `indextools` package. Its only argument is the note series.

```
6532 \newcommand{\led@set@index@fornote}[1]{%
6533    \ifbool{indtl@innote}%
6534      {\let\index\nindex}%
6535      {}%
6536    \ifbool{indtl@notenumber}%
6537      {%
```

```
6538        \renewcommand{\index}[2][\indtl@jobname]{%
6539          \orig@@index[##1]{%
6540            ##2|innotenumber{\csuse{thefootnote#1}}}%
6541            }%
6542          }%
6543        }%
6544        {}%
6545 }%
6546 %
```

`\led@reinit@index@fornote`   The `\led@reinit@index@fornote` just reset the default value of `\index`.

```
6547 \newcommand{\led@reinit@index@fornote}{%
6548   \ifbool{indtl@innote}%
6549     {\let\index\orig@@index}%
6550     {}%
6551  \ifbool{indtl@notenumber}%
6552   {\let\index\orig@@index}%
6553   {}%
6554 }%
6555 %
```

XXVII   Glossaries

Here, we define the `\gls`-like commands prefixed by ed, only if the package glossaries is loaded.

```
6556 \AtBeginDocument{%
6557   \@ifpackageloaded{glossaries}{%
6558 %
```

First those which arguments are [⟨*options*⟩]{⟨*label*⟩}[⟨*insert*⟩].

```
6559      \renewcommand{\do}[1]{%
6560        \expandafter\DeclareRobustCommandx\csname ed#1\endcsname[3][1,3,
usedefault]{%
6561          \doedindexlabel%
6562          \csname#1\endcsname[counter=pageline,##1]{##2}[##3]%
6563        }%
6564        \expandafter\WithSuffix\expandafter\DeclareRobustCommandx\csname ed
#1\endcsname*[3][1,3,usedefault]{%
6565          \doedindexlabel%
6566          \csname#1\endcsname*[counter=pageline,##1]{##2}[##3]%
6567        }%
6568      }%
6569      \docsvlist{%
6570        gls,%
6571        Gls,%
6572        GLS,%
6573        glspl,%
```

```
6574        Glspl,%
6575        GLSpl,%
6576        glstext,%
6577        Glstext,%
6578        GLStext,%
6579        Glsfirst,%
6580        GLSfirst,%
6581        glsplural%
6582        Glsplural,%
6583        GLSplural,%
6584        glsfirstplural,%
6585        Glsfirstplural,%
6586        GLSfirstplural,%
6587        glsname,%
6588        Glsname,%
6589        GLSname,%
6590        glssymbol,%
6591        Glssymbol,%
6592        GLSsymbol,%
6593        glsdesc,%
6594        Glsdesc,%
6595        GLSdesc,%
6596        glsuseri,%
6597        Glsuseri,%
6598        GLSuseri,%
6599        glsuserii,%
6600        Glsuserii,%
6601        GLSuserii,%
6602        glsuseriii,%
6603        Glsuseriii,%
6604        GLSuseriii,%
6605        glsuseriv,%
6606        Glsuseriv,%
6607        GLSuseriv,%
6608        glsuserv,%
6609        Glsuserv,%
6610        GLSuserv,%
6611        glsuservi,%
6612        Glsuservi,%
6613        GLSuservi%
6614        }%
6615 %
```

First those which arguments are [⟨*options*⟩]{⟨*label*⟩}{⟨*link text*⟩}.

```
6616      \renewcommand{\do}[1]{%
6617        \expandafter\DeclareRobustCommandx\csname ed#1\endcsname[3][1,
      usedefault]{%
6618          \doedindexlabel%
6619          \csname#1\endcsname[counter=pageline,##1]{##2}{##3}%
6620        }%
```

```
6621        \expandafter\WithSuffix\expandafter\DeclareRobustCommandx\csname ed
     #1\endcsname*[3][1,usedefault]{%
6622           \doedindexlabel%
6623           \csname#1\endcsname*[counter=pageline,##1]{##2}{##3}%
6624        }%
6625      }%
6626      \docsvlist{glsdisp,glslink}%
6627    }{}%
6628 }%
6629 %
```

XXVIII   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`.

XXVIII.1   Hanging symbol management

`\@hangingsymbol`
`\sethangingsymbol`
`\ifinstanza`

The macro `\@hangingsymbol` is used to insert a symbol on each hanging of verses. It is set by user level macro `\sethangingsymbol`.

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.

```
6630 \def\@hangingsymbol{}
6631 \newcommand*{\sethangingsymbol}[1]{%
6632   \gdef\@hangingsymbol{#1}%
6633 }%
6634 \newif\ifinstanza
6635 %
```

`\inserthangingymbol`
`\ifinserthangingsymbol`

The boolean `\ifinserthangingsymbol` is set to TRUE when `\@lock` is greater than 1, i.e. when we are not in the first line of a verse. The switch of `\ifinserthangingsymbol` is made in `\do@line` before the printing of line but after the line number calculation.

```
6636 \newif\ifinserthangingsymbol
6637 \newcommand{\inserthangingsymbol}{%
6638 \ifinserthangingsymbol%
6639    \ifinstanza%
6640       \@hangingsymbol%
6641    \fi%
6642 \fi%
6643 }
6644 %
```

XXVIII.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.

```
6645  \newcommand*{\ampersand}{\char`\&}
6646
6647  %
```

XXVIII.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.

```
6648     \chardef\body=\catcode`\@
6649     \catcode`\@=11
6650     \chardef\next=\catcode`\&
6651     \catcode`\&=\active
6652
6653  %
```

XXVIII.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 indentations are multiples of this value. The default value of \stanzaindentbase is 20pt.

```
6654     \newcount\stanza@count
6655     \newlength{\stanzaindentbase}
6656     \setlength{\stanzaindentbase}{20pt}
6657
6658  %
```

\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 *penalties*, which will be discussed below.

```
6659  \def\strip@szacnt#1,#2|{\def\@tempb{#1}\def\@tempa{#2|}}
6660  \newcommand*{\setstanzavalues}[2]{\def\@tempa{#2,,|}%
6661          \stanza@count\z@
6662        \def\next{\expandafter\strip@szacnt\@tempa
6663              \ifx\@tempb\empty\let\next\relax\else
6664              \expandafter\mathchardef\csname #1@\number\stanza@count
6665              @\endcsname\@tempb\relax
6666              \advance\stanza@count\@ne\fi\next}%
```

```
6667              \next}
6668
6669 %
```

\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).

```
6670 \newcommand*{\setstanzaindents}[1]{\setstanzavalues{sza}{#1}}
6671 \newcommand*{\setstanzapenalties}[1]{\setstanzavalues{szp}{#1}}
6672 %
6673 %
```

\managestanza@modulo Since version 0.13, the stanzaindentsrepetition counter can be used when the in-
                      dentation 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.

```
6674 \newcounter{stanzaindentsrepetition}
6675 \newcount\stanza@modulo
6676
6677 \newcommand*{\managestanza@modulo}[0]{%
6678     \advance\stanza@modulo\@ne%
6679     \ifnum\stanza@modulo>\value{stanzaindentsrepetition}%
6680      \stanza@modulo\@ne%
6681     \fi%
6682 }
6683 %
```

\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.

```
6684 \newcommand{\stanzaindent}[1]{%
6685   \hspace{\dimexpr#1\stanzaindentbase-\parindent\relax}%
6686   \ignorespaces%
6687 }%
6688 \WithSuffix\newcommand\stanzaindent*[1]{%
6689   \stanzaindent{#1}%
6690   \global\advance\stanza@modulo-\@ne%
6691   \ifnum\stanza@modulo=0%
6692     \global\stanza@modulo=\value{stanzaindentsrepetition}%
6693   \fi%
6694   \ignorespaces%
6695 }%
6696 %
```

XXVIII.5   Numbering stanza

Here, macro for numbering stanza. First, the stanza counter.

`\thestanza` 6697 `\newcounter{stanza}`
6698 `\renewcommand{\thestanza}{%`
6699 `  \textbf{\arabic{stanza}}%`
6700 `}`
6701 `%`

`\ifnumberstanza`   Then, macro to activate automatically numbering of stanza.

6702 `\newif\ifnumberstanza%`
6703 `%`

`\@insertstanzanumber`   Now, macro called at the first line of of verse of a stanza.

6704 `\newcommand{\@insertstanzanumber}[0]{%`
6705 `  \ifnumberstanza%`
6706 `    \ifl@dpairing%`
6707 `      \ifledRcol%`
6708 `      \stanzanumwrapper{\thestanzaR}%`
6709 `      \else%`
6710 `      \stanzanumwrapper{\thestanzaL}%`
6711 `      \fi%`
6712 `    \else%`
6713 `      \stanzanumwrapper{\thestanza}%`
6714 `    \fi%`
6715 `    \setline{1}%`
6716 `  \fi%`
6717 `}%`
6718 `%`

`\@advancestanzanumber`   Also a command to advance the counter of stanza.

6719 `\newcommand{\@advancestanzanumber}[0]{%`
6720 `  \ifnumberstanza%`
6721 `    \ifl@dpairing%`
6722 `      \ifledRcol%`
6723 `        \addtocounter{stanzaR}{1}%`
6724 `      \else%`
6725 `        \addtocounter{stanzaL}{1}%`
6726 `      \fi%`
6727 `    \else%`
6728 `      \addtocounter{stanza}{1}%`
6729 `    \fi%`
6730 `  \fi%`
6731 `}%`
6732 `%`

\stanzanumwrapper   And finally, the wrapper for stanza number

```
6733 \newcommand{\stanzanumwrapper}[1]{%
6734    \flagstanza{#1}%
6735 }%
6736 %
```

XXVIII.6   Stanza number in note

Here, the command called when printing stanza number in notes.

```
6737 \newcommand{\printstanza}[0]{%
6738     \ifboolexpr{bool{l@dpairing} or bool{l@dprintingpages} or bool{
     l@dprintingcolumns}}{%
6739         \ifledRcol@%
6740             \thestanzaR%
6741         \else%
6742             \thestanzaL%
6743         \fi%
6744     }{%
6745         \thestanza%
6746     }%
6747 }
6748 %
```

XXVIII.7   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.

```
6749 \newcommandx{\stanza@line}[1][1]{
6750     \ifnum\value{stanzaindentsrepetition}=0
6751         \parindent=\csname sza@\number\stanza@count
6752                 @\endcsname\stanzaindentbase
6753     \else
6754         \parindent=\csname sza@\number\stanza@modulo
6755                 @\endcsname\stanzaindentbase
6756         \managestanza@modulo
6757     \fi
6758     \pstart[#1]\stanza@hang\ignorespaces}
6759 \xdef\stanza@hang{\noexpand\leavevmode\noexpand\startlock
6760             \hangindent\expandafter
6761             \noexpand\csname sza@0@\endcsname\stanzaindentbase
```

```
6762            \hangafter\@ne}
6763 \def\sza@penalty{\count@\csname szp@\number\stanza@count @\endcsname
6764            \ifnum\count@>\@M\advance\count@-\@M\penalty-\else
6765            \penalty\fi\count@}
6766 %
```

\@startstanza Now we have the components of the \stanza macro, which appears at the start of a
\stanza group of lines. This macro initializes the count and checks to see if hanging indentation
\@stopstanza and penalties are to be included. Hanging indentation suspends the line count, so that
\AtEveryStopStanza the enumeration is by verse line rather than by print line. If the print line count is
\AtEveryStanza desired, invoke \let\startlock\relax and do the same for \endlock. Here and
\newverse 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 \&.

```
6767 \xdef\@startstanza[#1]{%
6768     \noexpand\instanzatrue\expandafter
6769     \begingroup%
6770     \catcode`\noexpand\&\active%
6771     \global\stanza@count\@ne\stanza@modulo\@ne
6772     \noexpand\ifnum\expandafter\noexpand
6773     \csname sza@00@\endcsname=\z@\let\noexpand\stanza@hang\relax
6774     \let\noexpand\endlock\relax\noexpand\else\interlinepenalty
6775     \@M\rightskip\z@ plus 1fil\relax\noexpand\fi\noexpand\ifnum
6776     \expandafter\noexpand\csname szp@00@\endcsname=\z@
6777     \let\noexpand\sza@penalty\relax\noexpand\fi%
6778     \def\noexpand&{%
6779            \noexpand\newverse[][]}%
6780     \def\noexpand\&{\noexpand\@stopstanza}%
6781     \noexpand\@advancestanzanumber%
6782     \noexpand\stanza@line[#1]%
6783     \noexpand\@insertstanzanumber%
6784     \let\par\relax\ignorespaces%No paragraph in verses
6785 }
6786
6787 \newcommandx{\stanza}[1][1,usedefault]{%
6788   \ifboolexpr{not test{\ifdefvoid{\at@every@stanza}} and test{\ifstrempty
      {#1}}}%
6789       {\@startstanza[\at@every@stanza]}%
6790       {\@startstanza[#1]}%
6791 }%
6792
6793 \newcommandx{\@stopstanza}[1][1,usedefault]{%
6794   \unskip%
6795   \endlock%
6796   \ifboolexpr{not test{\ifdefvoid{\at@every@stop@stanza}} and test{\
      ifstrempty{#1}}}%
6797       {\pend[\at@every@stop@stanza]}%
6798       {\pend[#1]}%
```

```
6799    \endgroup%
6800    \instanzafalse%
6801 }
6802
6803 \newcommand{\AtEveryStopStanza}[1]{%
6804    \ifstrempty{#1}%
6805       {\xdef\at@every@stop@stanza{}}%
6806       {\gdef\at@every@stop@stanza{#1}}%
6807 }%
6808 \def\at@every@stop@stanza{}%
6809
6810 \newcommand{\AtEveryStanza}[1]{%
6811    \ifstrempty{#1}%
6812       {\xdef\at@every@stanza{}}%
6813       {\gdef\at@every@stanza{#1}}%
6814 }%
6815 \def\at@every@stanza{}%
6816
6817
6818 \newcommandx*{\newverse}[2][1,2,usedefault]{%
6819    \unskip%
6820    \endlock\pend[#1]\sza@penalty\global%
6821    \advance\stanza@count\@ne\stanza@line[#2]%
6822    }
6823
6824 %
```

\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.

```
6825 \newcommand*{\flagstanza}[2][\stanzaindentbase]{%
6826    \hskip -#1\llap{#2}\hskip #1\ignorespaces}
6827
6828 %
```

XXVIII.8   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.

```
6829    \catcode`\&=\next
6830    \catcode`\@=\body
6831    \setstanzavalues{szp}{0}
6832
6833 %
```

XXIX   Apparatus of Manuscripts

XXIX.1   User level macro

\msdata   The user level \msdata command only writes the manuscripts data in numbered auxiliary file.

```
6834 \newcommand{\msdata}[1]{%
6835   \leavevmode%
6836   \unless\ifstopmsdata@inserted@%
6837     \stopmsdata%
6838     \led@warning@msdatawithoutstop%
6839   \fi%
6840   \global\stopmsdata@inserted@false%
6841   \unless\ifledRcol%
6842     \protected@write\linenum@out{}{%
6843       \string\@msd{#1}%
6844     }%
6845   \else%
6846     \protected@write\linenum@outR{}{%
6847       \string\@msd{#1}%
6848     }%
6849   \fi%
6850 }%
6851 %
```

\stopmsdata   The user level \stopmsdata command only writes information about the end of manuscripts data in numbered auxiliary file.

```
6852 \newcommand{\stopmsdata}[0]{%
6853   \leavevmode%
6854   \unless\ifledRcol%
6855     \protected@write\linenum@out{}{%
6856       \string\@stopmsd%
6857     }%
6858   \else%
6859     \protected@write\linenum@outR{}{%
6860       \string\@stopmsd%
6861     }%
6862   \fi%
6863   \global\stopmsdata@inserted@true%
6864 }%
6865 %
```

\ifstopmsdata@inserted@   The \ifstopmsdata@inserted@ boolean is set to TRUE at every \stopmsdata and reset to FALSE at all \msdata. It also set to TRUE at every \beginnumbering. It is used to automatically insert \stopmsdata if forgotten before \msdata

```
6866 \newif\ifstopmsdata@inserted@%
6867 %
```

XXIX.2   Setting macro

Setting macros for the manuscripts apparatus tools is very easy: they just save their argument in an internal macro.

\setmsdataseries   In which series of notes will be printed the apparatus of manuscripts?

```
6868 \newcommand{\setmsdataseries}[1]{%
6869   \gdef\@msdata@series{#1}%
6870 }%
6871 \def\@msdata@series{A}%
6872 %
```

\setmsdatalabel   The label for the manuscripts data.

```
6873 \def\ms@data@label{Ms.}%
6874 \newcommand{\setmsdatalabel}[1]{%
6875   \gdef\ms@data@label{#1}%
6876 }%
6877 %
```

XXIX.3   Counters and lists

\@msd@c   \@msd@c is a counter incremented at each \@msd read in auxiliary file.

```
6878 \numdef{\@msd@c}{0}
6879 \numdef{\@msd@cR}{0}
6880 %
```

\@msd@c   \add@msd@ is a counter incremented at each \add@msddata, that is at each time we prepare the insertion of manuscripts data footnote.

```
6881 \numdef{\add@msd@c}{0}%
6882 \numdef{\add@msd@cR}{0}%
6883 %
```

\@msdata@list   The \@msdata@list will contain, for each line, the lists of command to be executed to insert the manuscripts apparatus. It will be filled on \add@msdata and looped on \insert@msdata, then emptied.

```
6884 \def\@msdata@list{}%
6885 %
```

XXIX.4   Auxiliary file macros

\@msd   The \@msd macro is written in the auxiliary file. It just defines three macros by \msdata macro, which allow us to know the manuscripts data, the line number and the absolute line number where it was called

  It also stores the action code 1010 in the list of actions by line.

```
6886 \newcommand{\@msd}[1]{%
6887   \unless\ifledRcol%
6888     \numdef{\@msd@c}{\@msd@c+\@ne}%
6889     \csgdef{@msdata@\@msd@c @data}{#1}%
6890     \csxdef{@msdata@\@msd@c @linenumber}{\the\line@num}%
6891     \csxdef{@msdata@\@msd@c @abslinenumber}{\the\absline@num}%
6892     \xright@appenditem{\the\absline@num}\to\actionlines@list%
6893     \xright@appenditem{-1010}\to\actions@list%
6894   \else%
6895     \numdef{\@msd@cR}{\@msd@cR+\@ne}%
6896     \csgdef{@msdata@\@msd@cR @dataR}{#1}%
6897     \csxdef{@msdata@\@msd@cR @linenumberR}{\the\line@numR}%
6898     \csxdef{@msdata@\@msd@cR @abslinenumberR}{\the\absline@numR}%
6899     \xright@appenditem{\the\absline@numR}\to\actionlines@listR%
6900     \xright@appenditem{-1010}\to\actions@listR%
6901   \fi%
6902 }%
6903 %
```

\@endmsd Inserted in the auxiliary file by \stopmsd, the \@stopmsd macro will store in two com-
mands the line number and the absolute line number on which it is called.

```
6904 \newcommand{\@stopmsd}[0]{%
6905   \unless\ifledRcol%
6906     \ifcsundef{@msdata@\@msd@c @stoplinenumber}{%
6907       \csxdef{@msdata@\@msd@c @stopabslinenumber}{\the\absline@num}%
6908       \csxdef{@msdata@\@msd@c @stoplinenumber}{\the\line@num}%
6909     }{}%
6910   \else%
6911     \ifcsundef{@msdata@\@msd@cR @stoplinenumberR}{%
6912       \csxdef{@msdata@\@msd@cR @stopabslinenumberR}{\the\absline@numR}%
6913       \csxdef{@msdata@\@msd@cR @stoplinenumberR}{\the\line@numR}%
6914     }%
6915     {}%
6916   \fi%
6917 }%
6918 %
```

XXIX.5 Action macro

\add@msdata \add@msdata is executed on each line when action code 1010 is seen. It will not insert
immediately the manuscript data footnote, as action code are executed before the line be
typeset, and, consequently, could be on the previous page. So it just store the manuscript
data footnote to \@msdata@list.

```
6919 \newcommand{\add@msdata}{%
6920   \bgroup%
6921   \normalfont%
6922   \unless\ifledRcol@%
```

```
6923        \numgdef{\add@msd@c}{\add@msd@c+\@ne}%
6924          \ifcsdef{@msdata@\add@msd@c @data}{%
6925            \letcs{\@data}{@msdata@\add@msd@c @data}%
6926            \edef\l@d@nums{%
6927              000|% Start page = we don't print it
6928              \csuse{@msdata@\add@msd@c @linenumber}|% Start line number
6929              000|% Start subline number, for now, not used
6930              000|% End page number, we don't print it
6931              \ifnumless{\csuse{@msdata@\add@msd@c @stopabslinenumber}}{\csuse{
    @lastabsline@forpage@\the\page@num}}%
6932                {\csuse{@msdata@\add@msd@c @stoplinenumber}}%End line number if
     in the same page
6933                {\csuse{@lastline@forpage@\the\page@num}}%Otherwiser, last
    number of the page
6934              |%
6935              000|% End sub line number, for now, not used
6936              \edfont@info%Font
6937              }%
6938            \@msd@options@fullpagefalse%
6939            \if@firstlineofpage%Try if the data are for the full page. If yes
    , will add options to the list.
6940              \unless\if@msdata@insertedfrompreviouspage%
6941                \ifnumless{\csuse{@lastabsline@forpage@\the\page@num}}{\csuse
    {@msdata@\add@msd@c @stopabslinenumber}+\@ne}%
6942                {%
6943                \numdef{\@tmp}{\add@msd@c+\@ne}%
6944                \ifcsdef{@msdata@\@tmp @abslinenumber}%
6945                  {\ifnumequal{\csuse{@msdata@\@tmp @abslinenumber}}{\csuse{
    @lastabsline@forpage@\the\page@num}}%
6946                    {}%
6947                    {\@msd@options@fullpagetrue}%
6948                  }%
6949                  {\@msd@options@fullpagetrue}%
6950                }%
6951                {}%
6952              \fi%
6953            \fi%
6954            \listxadd{\@msdata@list}{%
6955              \@msd@options@iffullpage%
6956              \noexpand\csuse{v\@msdata@series footnote}{\@msdata@series}{{\
    expandonce\l@d@nums}{\ms@data@label}{\expandonce\@data}}%
6957              \reset@msd@options@iffullpage%
6958            }%
6959          }%
6960          {}%
6961    \else%
6962      \numgdef{\add@msd@cR}{\add@msd@cR+\@ne}%
6963        \ifcsdef{@msdata@\add@msd@cR @dataR}{%
6964          \letcs{\@data}{@msdata@\add@msd@cR @dataR}%
6965          \edef\l@d@nums{%
```

```
6966          000|% Start page = we don't print it
6967          \csuse{@msdata@\add@msd@cR @linenumberR}|% Start line number
6968          000|% Start subline number, for now, not used
6969          000|% End page number, we don't print it
6970          \ifnumless{\csuse{@msdata@\add@msd@cR @stopabslinenumberR}}{\
     csuse{@lastline@forpageR@\the\page@numR}}%
6971              {\csuse{@msdata@\add@msd@cR @stoplinenumberR}}%End line number
     if in the same page
6972              {\csuse{@lastline@forpageR@\the\page@numR}}%Otherwiser, last
     number of the page
6973          |%
6974          000|% End sub line number, for now, not used
6975          \edfont@info%Font
6976          }%
6977          \@msd@options@fullpagefalse%
6978          \if@firstlineofpageR%
6979            \unless\if@msdata@insertedfrompreviouspage%
6980              \ifnumless{\csuse{@lastabsline@forpageR@\the\page@numR}}{\
     csuse{@msdata@\add@msd@c @stopabslinenumberR}+\@ne}%
6981                {%
6982                \numdef{\@tmp}{\add@msd@cR+\@ne}%
6983                \ifcsdef{@msdata@\@tmp @abslinenumberR}%
6984                  {\ifnumequal{\csuse{@msdata@\@tmp @abslinenumberR}}{\csuse{
     @lastabsline@forpageR@\the\page@numR}}%
6985                    {}%
6986                    {\@msd@options@fullpagetrue}%
6987                  }%
6988                  {\@msd@options@fullpagetrue}%
6989                }%
6990                {}%
6991              \fi%
6992          \fi%
6993          \listxadd{\@msdata@list}{%
6994            \@msd@options@iffullpage%
6995            \noexpand\csuse{v\@msdata@series footnote}{\@msdata@series}{{\
     expandonce\l@d@nums}{\ms@data@label}{\expandonce\@data}}%
6996            \reset@msd@options@iffullpage%
6997          }%
6998        }%
6999        {}%
7000    \fi%
7001    \egroup%
7002 }%
7003 %
```

@insertedfrompreviouspage   The \if@msdata@insertedfrompreviouspage boolean is set to TRUE if reledmac
automatically inserts data from previous page in the first line of a page.

```
7004 \newif\if@msdata@insertedfrompreviouspage%
7005 %
```

\add@msdata@firstlineofpage   \add@msdata@firstlineofpage is called at the first line of every page. It inserts man-
uscript data which start on one of the previous pages and continue on this page.

```
7006 \newcommand{\add@msdata@firstlineofpage}{%
7007   \bgroup%
7008   \normalfont%
7009   \unless\ifledRcol@%
7010     \ifcsdef{@msdata@\add@msd@c @data}{%
7011     \ifnumless{\the\absline@num-\@ne}{\csuse{@msdata@\add@msd@c
    @stopabslinenumber}}%
7012       {%
7013       \global\@msdata@insertedfrompreviouspagetrue%
7014       \letcs{\@data}{@msdata@\add@msd@c @data}%
7015       \edef\l@d@nums{%
7016           000|% Start page = we don't print it
7017           \numexpr\the\line@num+\@ne\relax|% Start line number = first line
    of the page. As \add@msdata@firstlineofpage is called before line number
    has been incremented, we increment it for printing
7018           000|% Start subline number, for now, not used
7019           000|% End page number, we don't print it
7020           \ifnumless{\csuse{@msdata@\add@msd@c @stopabslinenumber}}{\csuse{
    @lastabsline@forpage@\the\page@num}}%
7021               {\csuse{@msdata@\add@msd@c @stoplinenumber}}%End line number if
    in the same page
7022               {\csuse{@lastline@forpage@\the\page@num}}%Otherwise, last
    number of the page
7023           |%
7024           000|% End sub line number, for now, not used
7025           \edfont@info%Font
7026           }%
7027           \@msd@options@fullpagefalse%
7028           \ifnumless{\csuse{@lastabsline@forpage@\the\page@num}}{\csuse{
    @msdata@\add@msd@c @stopabslinenumber}+\@ne}%We will test if the ms data is
    for the full page
7029           {%
7030           \numdef{\@tmp}{\add@msd@c+\@ne}%
7031           \ifcsdef{@msdata@\@tmp @abslinenumber}%
7032             {\ifnumequal{\csuse{@msdata@\@tmp @abslinenumber}}{\csuse{
    @lastabsline@forpage@\the\page@num}}%
7033               {}%
7034               {\@msd@options@fullpagetrue}%
7035             }%
7036             {\@msd@options@fullpagetrue}%
7037           }%
7038           {}%
7039           \listxadd{\@msdata@list}{%
7040           \@msd@options@iffullpage%
7041           \noexpand\csuse{v\@msdata@series footnote}{\@msdata@series}{{\
    expandonce\l@d@nums}{\ms@data@label}{\expandonce\@data}}%
7042           \reset@msd@options@iffullpage%
```

```
7043            }%
7044          }%
7045          {\global\@msdata@insertedfrompreviouspagefalse}%
7046    }{}%
7047    \else%
7048      \ifcsdef{@msdata@\add@msd@cR @dataR}{%
7049      \ifnumless{\the\absline@numR-\@ne}{\csuse{@msdata@\add@msd@cR
    @stopabslinenumberR}}%
7050            {%
7051            \global\@msdata@insertedfrompreviouspagetrue%
7052            \letcs{\@data}{@msdata@\add@msd@cR @dataR}%
7053            \edef\l@d@nums{%
7054            000|% Start page = we don't print it
7055            \numexpr\the\line@numR+\@ne\relax|% Start line number = first
    line of the page. As \add@msdata@firstlineofpage is called before line
    number has been incremented, we increment it for printing
7056            000|% Start subline number, for now, not used
7057            000|% End page number, we don't print it
7058            \ifnumless{\csuse{@msdata@\add@msd@cR @stopabslinenumberR}}{\
    csuse{@lastline@forpageR@\the\page@numR}}%
7059              {\csuse{@msdata@\add@msd@cR @stoplinenumberR}}%End line number
    if in the same page
7060              {\csuse{@lastline@forpageR@\the\page@numR}}%Otherwise, last
    number of the page
7061            |%
7062            000|% End sub line number, for now, not used
7063            \edfont@info%Font
7064            }%
7065            \@msd@options@fullpagefalse%
7066            \ifnumless{\csuse{@lastabsline@forpageR@\the\page@numR}}{\csuse{
    @msdata@\add@msd@cR @stopabslinenumberR}+\@ne}%
7067            {%
7068            \numdef{\@tmp}{\add@msd@cR+\@ne}%
7069            \ifcsdef{@msdata@\@tmp @abslinenumberR}%
7070              {\ifnumequal{\csuse{@msdata@\@tmp @abslinenumberR}}{\csuse{
    @lastabsline@forpageR@\the\page@numR}}%
7071                {}%
7072                {\@msd@options@fullpagetrue}%
7073              }%
7074              {\@msd@options@fullpagetrue}%
7075            }%
7076            {}%
7077            \listxadd{\@msdata@list}{%
7078              \@msd@options@iffullpage%
7079              \noexpand\csuse{v\@msdata@series footnote}{\@msdata@series}{{\
    expandonce\l@d@nums}{\ms@data@label}{\expandonce\@data}}%
7080              \reset@msd@options@iffullpage%
7081            }%
7082        }%
7083        {\global\@msdata@insertedfrompreviouspagefalse}%
```

```
7084    }{}%
7085    \fi%
7086    \egroup%
7087 }%
7088 %
```

XXIX.6   Inserting footnote

Just before inserting standard insert (familiar and critical footnotes, sidenotes), we call
\insert@msdata to insert manuscripts data's footnotes.

```
\insert@msdata 89 \newcommand{\insert@msdata}{%
7090    \def\do##1{##1}%
7091    \dolistloop{\@msdata@list}%
7092    \global\let\@msdata@list\relax%
7093 }%
7094 %
```

XXIX.7   Other

```
\@msd@options@iffullpage        \@msd@options@iffullpage sets some options if the manuscripts data are for all the
\reset@msd@options@iffullpage   page. \reset@msd@options@iffullpage resets them after the footnote. \if@msd@options@fullpage
                                is switch to true in add@msdata@firstlineofpage if these option must be inserted.
```

```
7095 \newif\if@msd@options@fullpage%
7096 \newcommand{\@msd@options@iffullpage}[0]{%
7097    \if@msd@options@fullpage%
7098       \noexpand\toggletrue{nonum@}%
7099       \ifdefvoid{\ms@data@label}%
7100          {\noexpand\toggletrue{nosep@}}%
7101          {}%
7102    \fi%
7103 }%
7104 \newcommand{\reset@msd@options@iffullpage}[0]{%
7105    \noexpand\togglefalse{nonum@}%
7106    \noexpand\togglefalse{nosep@}%
7107 }%
7108 %
```

XXX   Arrays and tables

XXX.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.

```
7109 \newtoks\@emptytoks
7110
7111 %
```

The rest is from `amsmath`.

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

```
7112 \newtoks\l@denvbody
7113
7114 %
```

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

```
7115 \newcommand{\addtol@denvbody}[1]{%
7116   \global\l@denvbody\expandafter{\the\l@denvbody#1}}
7117
7118 %
```

\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.

```
7119 \newcommand{\l@dcollect@body}[1]{%
7120   \l@denvbody{\expandafter#1\expandafter{\the\l@denvbody}}%
7121   \edef\processl@denvbody{\the\l@denvbody\noexpand\end{\@currenvir}}%
7122   \l@denvbody\@emptytoks \def\l@dbegin@stack{b}%
7123   \begingroup
7124     \expandafter\let\csname\@currenvir\endcsname\l@dcollect@@body
7125     \edef\processl@denvbody{\expandafter\noexpand\csname\@currenvir\
    endcsname}%
7126     \processl@denvbody%
7127     }%
7128
7129 %
```

\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.

```
7130 \def\l@dpush@begins#1\begin#2{%
7131   \ifx\end#2\else b\expandafter\l@dpush@begins\fi}
7132
7133 %
```

`\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.

```
7134  \def\l@dcollect@@body#1\end#2{%
7135    \edef\l@dbegin@stack{\l@dpush@begins#1\begin\end
7136                        \expandafter\@gobble\l@dbegin@stack}%
7137    \ifx\@empty\l@dbegin@stack
7138      \endgroup
7139      \@checkend{#2}%
7140      \addtol@denvbody{#1}%
7141    \else
7142      \addtol@denvbody{#1\end{#2}}%
7143    \fi
7144    \processl@denvbody % A little tricky! Note the grouping
7145  }
7146
7147  %
```

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>
```

XXX.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`.

XXX.2.1   Disabling and restoring commands

`\l@dtabnoexpands`   More no expansion for critical and familiar footnotes in tabular environment.

```
7148 \newcommand*{\l@dtabnoexpands}{%
7149   \let\rtab=0%
7150   \let\ctab=0%
7151   \let\ltab=0%
7152   \let\rtabtext=0%
7153   \let\ltabtext=0%
7154   \let\ctabtext=0%
7155   \let\edbeforetab=0%
7156   \let\edaftertab=0%
7157   \let\edatleft=0%
7158   \let\edatright=0%
7159   \let\edvertline=0%
7160   \let\edvertdots=0%
7161   \let\edrowfill=0%
7162 }
7163
7164 %
```

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

```
7165 \newcommand{\disable@familiarnotes}{%
7166   \unless\ifnofamiliar@%
7167     \def\do##1{%
7168         \csletcs{footnote@@##1}{footnote##1}%
7169         \expandafter\renewcommand \csname footnote##1\endcsname[1]{%
7170           \protected@csxdef{@thefnmark##1}{\csuse{thefootnote##1}}%
7171           \csuse{@footnotemark##1}%
7172         }%
7173     }%
7174     \dolistloop{\@series}%
7175   \fi%
7176 }%
7177 \newcommand{\restore@familiarnotes}{%
7178   \unless\ifnofamiliar@%
7179     \def\do##1{%
7180         \csletcs{footnote##1}{footnote@@##1}%
7181     }%
7182     \dolistloop{\@series}%
7183   \fi%
7184 }%
7185
7186 %
```

`\disable@sidenotes`   The sames, for side notes.
`\restore@sidenotes`

```
7187 \newcommand{\disable@sidenotes}{%
7188   \let\@@ledrightnote\ledrightnote%
7189   \let\@@ledleftnote\ledleftnote%
7190   \let\@@ledsidenote\ledsidenote%
7191   \let\ledrightnote\@gobble%
7192   \let\ledleftnote\@gobble%
7193   \let\ledsidenote\@gobble%
7194 }%
7195 \newcommand{\restore@sidenotes}{%
7196   \let\ledrightnote\@@ledrightnote%
7197   \let\ledleftnote\@@ledleftnote%
7198   \let\ledsidenote\@@ledsidenote%
7199 }%
7200 %
```

`\disable@notes`   Disable/restore side and familiar notes.
`\restore@notes`

```
7201 \newcommand{\disable@notes}{%
7202   \disable@sidenotes%
7203   \disable@familiarnotes%
7204 }%
7205 \newcommand{\restore@notes}{%
```

```
7206    \restore@sidenotes%
7207    \restore@familiarnotes%
7208 }%
7209 %
```

\EDTEXT   We need to be able to modify the \edtext macros and also restore their original defi-
\xedtext   nitions.

```
7210 \let\EDTEXT=\edtext
7211 \newcommand{\xedtext}[2]{\EDTEXT{#1}{#2}}
7212 %
```

\EDLABEL   We need to be able to modify and restore the \edlabel macro.
\xedlabel
```
7213 \let\EDLABEL=\edlabel
7214 \newcommand*{\xedlabel}[1]{\EDLABEL{#1}}
7215 %
```

\EDINDEX   Macros supporting modification and restoration of \edindex.
\xedindex
\nulledindex
```
7216 \let\EDINDEX=\edindex
7217 \newcommand{\xedindex}{\@bsphack%
7218    \@ifnextchar [{\l@d@index}{\l@d@index[\jobname]}}
7219 \newcommand{\nulledindex}[2][\jobname]{\@bsphack\@esphack}
7220
7221 %
```

\@line@@num   Macro supporting restoration of \linenum.

```
7222 \let\@line@@num=\linenum
7223 %
```

\l@dgobblearg   \l@dgobbleoptarg[⟨*arg*⟩]{⟨*arg*⟩} replaces these two arguments (first is optional) by
\relax.

```
7224 \newcommand*{\l@dgobbleoptarg}[2][]{\relax}%
7225
7226 %
```

\Relax   \let\Relax=\relax
\NEXT   \let\NEXT=\next
```
7229
7230 %
```

\l@dmodforedtext   Modify and restore various macros for when \edtext is used.
\l@drestoreforedtext

```
7231 \newcommand{\l@dmodforedtext}{%
7232   \let\edtext\relax
7233   \def\do##1{\global\csletcs{##1footnote}{l@dgobbleoptarg}}%
7234   \dolistloop{\@series}%
7235   \let\edindex\nulledindex
7236   \let\linenum\@gobble}
7237 \newcommand{\l@drestoreforedtext}{%
7238   \def\do##1{\global\csletcs{##1footnote}{##1@@footnote}}
7239   \dolistloop{\@series}%
7240   \let\edindex\xedindex}
7241 %
```

\l@dnullfills   Nullify and restore some column fillers, etc.
\l@drestorefills

```
7242 \newcommand{\l@dnullfills}{%
7243   \def\edlabel##1{}%
7244   \def\edrowfill##1##2##3{}%
7245 }
7246 \newcommand{\l@drestorefills}{%
7247   \def\edrowfill##1##2##3{\@EDROWFILL@{##1}{##2}{##3}}%
7248 }
7249
7250 %
```

\letsforverteilen   Gathers some lets and other code that is common to the *verteilen* macros.

```
7251 \newcommand{\letsforverteilen}{%
7252   \let\edtext\xedtext
7253   \let\edindex\xedindex
7254   \def\do##1{\global\csletcs{##1footnote}{##1@@footnote}}
7255   \dolistloop{\@series}%
7256   \let\linenum\@line@@num
7257   \hilfsskip=\l@dcolwidth%
7258   \advance\hilfsskip by -\wd\hilfsbox
7259   \def\edlabel##1{\xedlabel{##1}}}
7260
7261 %
```

\disablel@dtabfeet   Declarations for using or using \edtext inside tabulars. The default at this point is for
\enablel@dtabfeet   \edtext.

```
7262 \newcommand\disablel@dtabfeet{\l@dmodforedtext}%
7263 \newcommand\enablel@dtabfeet{\l@drestoreforedtext}%
7264 %
```

XXX.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.

```
7265 \newcount\l@dampcount
7266   \l@dampcount=1\relax
7267 \newcount\l@dcolcount
7268   \l@dcolcount=0\relax
7269
7270 %
```

\hilfsbox
\hilfsskip   Some (temporary) helper items.
\Hilfsbox
\hilfscount

```
7271 \newbox\hilfsbox
7272 \newskip\hilfsskip
7273 \newbox\Hilfsbox
7274 \newcount\hilfscount
7275
7276 %
```

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).

```
7277 \newdimen\dcoli
7278 \newdimen\dcolii
7279 \newdimen\dcoliii
7280 \newdimen\dcoliv
7281 \newdimen\dcolv
7282 \newdimen\dcolvi
7283 \newdimen\dcolvii
7284 \newdimen\dcolviii
7285 \newdimen\dcolix
7286 \newdimen\dcolx
7287 \newdimen\dcolxi
7288 \newdimen\dcolxii
7289 \newdimen\dcolxiii
7290 \newdimen\dcolxiv
7291 \newdimen\dcolxv
7292 \newdimen\dcolxvi
7293 \newdimen\dcolxvii
7294 \newdimen\dcolxviii
7295 \newdimen\dcolxix
7296 \newdimen\dcolxx
7297 \newdimen\dcolxxi
7298 \newdimen\dcolxxii
7299 \newdimen\dcolxxiii
7300 \newdimen\dcolxxiv
7301 \newdimen\dcolxxv
7302 \newdimen\dcolxxvi
7303 \newdimen\dcolxxvii
7304 \newdimen\dcolxxviii
7305 \newdimen\dcolxxix
7306 \newdimen\dcolxxx
7307 \newdimen\dcolerr    % added for error handling
```

```
7308
7309 %
```

\l@dcolwidth   This is a cunning way of storing the columnwidths indexed by the column number \l@dcolcount, like an array. (was \Dimenzuordnung)

```
7310 \newcommand{\l@dcolwidth}{\ifcase \the\l@dcolcount \dcoli %???
7311   \or \dcoli \or \dcolii \or \dcoliii
7312   \or \dcoliv \or \dcolv \or \dcolvi
7313   \or \dcolvii \or \dcolviii \or \dcolix \or \dcolx
7314   \or \dcolxi \or \dcolxii \or \dcolxiii
7315   \or \dcolxiv \or \dcolxv \or \dcolxvi
7316   \or \dcolxvii \or \dcolxviii \or \dcolxix \or \dcolxx
7317   \or \dcolxxi \or \dcolxxii \or \dcolxxiii
7318   \or \dcolxxiv \or \dcolxxv \or \dcolxxvi
7319   \or \dcolxxvii \or \dcolxxviii \or \dcolxxix \or \dcolxxx
7320   \else \dcolerr \fi}
7321
7322 %
```

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

```
7323 \newcommand*{\stepl@dcolcount}{\advance\l@dcolcount\@ne
7324   \ifnum\l@dcolcount>30\relax
7325     \led@err@TooManyColumns
7326   \fi}
7327
7328 %
```

\l@dsetmaxcolwidth   Sets the column width to the maximum value seen so far.

```
7329 \newcommand{\l@dsetmaxcolwidth}{%
7330   \ifdim\l@dcolwidth < \wd\hilfsbox
7331     \l@dcolwidth = \wd\hilfsbox
7332   \else \relax \fi}
7333
7334 %
```

\measuremcell   Measure (recursively) the width required for a math cell.

```
7335 \def\measuremcell #1&{%
7336     \ifx #1\\ \ifnum\l@dcolcount=0\let\NEXT\relax%
7337               \else\l@dcheckcols%
7338                     \l@dcolcount=0%
7339                     \let\NEXT\measuremcell%
7340               \fi%
7341     \else\setbox\hilfsbox=\hbox{$\displaystyle{#1}$}%
7342       \stepl@dcolcount%
7343       \l@dsetmaxcolwidth%
7344       \let\NEXT\measuremcell%
```

```
7345        \fi\NEXT}
7346
7347  %
```

\measuretcell   Measure (recursively) the width required for a text cell.

```
7348  \def\measuretcell #1&{%
7349      \ifx #1\\ \ifnum\l@dcolcount=0\let\NEXT\relax%
7350              \else\l@dcheckcols%
7351                      \l@dcolcount=0%
7352                      \let\NEXT\measuretcell%
7353              \fi%
7354      \else\setbox\hilfsbox=\hbox{#1}%
7355        \stepl@dcolcount%
7356        \l@dsetmaxcolwidth%
7357        \let\NEXT\measuretcell%
7358      \fi\NEXT}
7359
7360  %
```

\measuremrow   Measure (recursively) the width required for a math row.

```
7361  \def\measuremrow #1\\{%
7362      \ifx #1&\let\NEXT\relax%
7363      \else\measuremcell #1&\\&\\&%
7364          \let\NEXT\measuremrow%
7365      \fi\NEXT}
7366  %
```

\measuretrow   Measure (recursively) the width required for a text row.

```
7367  \def\measuretrow #1\\{%
7368      \ifx #1&\let\NEXT\relax%
7369      \else\measuretcell #1&\\&\\&%
7370          \let\NEXT\measuretrow%
7371      \fi\NEXT}
7372
7373  %
```

\edtabcolsep   The length \edtabcolsep controls the distance between columns.

```
7374  \newskip\edtabcolsep
7375  \global\edtabcolsep=10pt
7376
7377  %
```

\variab
```
7378  \newcommand{\variab}{\relax}
7379
7380  %
```

\l@dcheckcols Check that the number of columns is consistent.

```
7381 \newcommand*{\l@dcheckcols}{%
7382   \ifnum\l@dcolcount=1\relax
7383   \else
7384     \ifnum\l@dampcount=1\relax
7385     \else
7386       \ifnum\l@dcolcount=\l@dampcount\relax
7387       \else
7388         \l@d@err@UnequalColumns
7389       \fi
7390     \fi
7391     \l@dampcount=\l@dcolcount
7392   \fi}
7393
7394 %
```

\edfilldimen A length.

```
7395 \newdimen\edfilldimen
7396 \edfilldimen=0pt
7397
7398 %
```

\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. We do not use the \roman LaTeX command, because
some packages, like babel can override it in some specific cases (Greek, for example).

```
7399 \newcounter{addcolcount}
7400   \renewcommand{\theaddcolcount}{\romannumeral \c@addcolcount}
7401 %
```

XXX.2.3 Tabular typesetting

\setmcellright Typeset (recursively) cells of display math right justified.

```
7402 \def\setmcellright #1&{\def\edlabel##1{}%
7403       \let\edindex\nulledindex
7404       \ifx #1\\ \ifnum\l@dcolcount=0%\removelastskip
7405                     \let\Next\relax%
7406                 \else\l@dcolcount=0%
7407                     \let\Next=\setmcellright%
7408               \fi%
7409       \else%
7410           \disablel@dtabfeet%
7411           \stepl@dcolcount%
7412           \disable@notes%
7413           \setbox\hilfsbox=\hbox{$\displaystyle{#1}$}%
7414           \restore@notes%
7415           \letsforverteilen%
```

```
7416                \hskip\hilfsskip$\displaystyle{#1}$%
7417                \hskip\edtabcolsep%
7418                \let\Next=\setmcellright%
7419           \fi\Next}
7420
7421  %
```

\settcellright   Typeset (recursively) cells of text right justified.

```
7422  \def\settcellright #1&{\def\edlabel##1{}%
7423           \let\edindex\nulledindex
7424           \ifx #1\\ \ifnum\l@dcolcount=0%\removelastskip
7425                      \let\Next\relax%
7426                  \else\l@dcolcount=0%
7427                        \let\Next=\settcellright%
7428                  \fi%
7429           \else%
7430              \disablel@dtabfeet%
7431              \stepl@dcolcount%
7432              \disable@notes%
7433              \setbox\hilfsbox=\hbox{#1}%
7434              \restore@notes%
7435              \letsforverteilen%
7436              \hskip\hilfsskip#1%
7437              \hskip\edtabcolsep%
7438              \let\Next=\settcellright%
7439           \fi\Next}
7440  %
```

\setmcellleft   Typeset (recursively) cells of display math left justified.

```
7441  \def\setmcellleft #1&{\def\edlabel##1{}%
7442      \let\edindex\nulledindex
7443      \ifx #1\\ \ifnum\l@dcolcount=0 \let\Next\relax%
7444              \else\l@dcolcount=0%
7445                    \let\Next=\setmcellleft%
7446              \fi%
7447      \else   \disablel@dtabfeet%
7448              \stepl@dcolcount%
7449              \disable@notes%
7450              \setbox\hilfsbox=\hbox{$\displaystyle{#1}$}%
7451              \restore@notes%
7452              \letsforverteilen%
7453              $\displaystyle{#1}$\hskip\hilfsskip\hskip\edtabcolsep%
7454              \let\Next=\setmcellleft%
7455      \fi\Next}
7456
7457  %
```

\settcellleft   Typeset (recursively) cells of text left justified.

```
7458 \def\settcellleft #1&{\def\edlabel##1{}%
7459     \let\edindex\nulledindex
7460     \ifx #1\\ \ifnum\l@dcolcount=0 \let\Next\relax%
7461             \else\l@dcolcount=0%
7462                 \let\Next=\settcellleft%
7463             \fi%
7464     \else   \disablel@dtabfeet%
7465             \stepl@dcolcount%
7466             \disable@notes%
7467             \setbox\hilfsbox=\hbox{#1}%
7468             \restore@notes%
7469             \letsforverteilen%
7470             #1\hskip\hilfsskip\hskip\edtabcolsep%
7471             \let\Next=\settcellleft%
7472     \fi\Next}
7473 %
```

\setmcellcenter   Typeset (recursively) cells of display math centered.

```
7474 \def\setmcellcenter #1&{\def\edlabel##1{}%
7475     \let\edindex\nulledindex
7476     \ifx #1\\ \ifnum\l@dcolcount=0\let\Next\relax%
7477             \else\l@dcolcount=0%
7478                 \let\Next=\setmcellcenter%
7479             \fi%
7480     \else   \disablel@dtabfeet%
7481             \stepl@dcolcount%
7482             \disable@notes%
7483             \setbox\hilfsbox=\hbox{$\displaystyle{#1}$}%
7484             \restore@notes%
7485             \letsforverteilen%
7486             \hskip 0.5\hilfsskip$\displaystyle{#1}$\hskip0.5\hilfsskip%
7487             \hskip\edtabcolsep%
7488             \let\Next=\setmcellcenter%
7489     \fi\Next}
7490
7491 %
```

\settcellcenter   Typeset (recursively) cells of text centered.

```
7492 \def\settcellcenter #1&{\def\edlabel##1{}%
7493     \let\edindex\nulledindex
7494     \ifx #1\\ \ifnum\l@dcolcount=0 \let\Next\relax%
7495             \else\l@dcolcount=0%
7496                 \let\Next=\settcellcenter%
7497             \fi%
7498     \else   \disablel@dtabfeet%
7499             \stepl@dcolcount%
7500             \disable@notes%
7501             \setbox\hilfsbox=\hbox{#1}%
```

```
7502            \restore@notes%
7503            \letsforverteilen%
7504            \hskip 0.5\hilfsskip #1\hskip 0.5\hilfsskip%
7505            \hskip\edtabcolsep%
7506            \let\Next=\settcellcenter%
7507        \fi\Next}
7508
7509 %
```

`\NEXT`
```
7510 \let\NEXT=\relax
7511
7512 %
```

`\setmrowright` Typeset (recursively) rows of right justified math.

```
7513 \def\setmrowright #1\\{%
7514     \ifx #1& \let\NEXT\relax
7515     \else \centerline{\setmcellright #1&\\&\\&}
7516            \let\NEXT=\setmrowright
7517     \fi\NEXT}
7518 %
```

`\settrowright` Typeset (recursively) rows of right justified text.

```
7519 \def\settrowright #1\\{%
7520     \ifx #1& \let\NEXT\relax
7521     \else \centerline{\settcellright #1&\\&\\&}
7522            \let\NEXT=\settrowright
7523     \fi\NEXT}
7524
7525 %
```

`\setmrowleft` Typeset (recursively) rows of left justified math.

```
7526 \def\setmrowleft #1\\{%
7527     \ifx #1&\let\NEXT\relax
7528     \else \centerline{\setmcellleft #1&\\&\\&}
7529            \let\NEXT=\setmrowleft
7530     \fi\NEXT}
7531 %
```

`\settrowleft` Typeset (recursively) rows of left justified text.

```
7532 \def\settrowleft #1\\{%
7533     \ifx #1& \let\NEXT\relax
7534     \else \centerline{\settcellleft #1&\\&\\&}
7535            \let\NEXT=\settrowleft
7536     \fi\NEXT}
7537
7538 %
```

`\setmrowcenter` Typeset (recursively) rows of centered math.

```
7539 \def\setmrowcenter #1\\{%
7540    \ifx #1& \let\NEXT\relax%
7541    \else \centerline{\setmcellcenter #1&\\&\\&}
7542         \let\NEXT=\setmrowcenter
7543     \fi\NEXT}
7544 %
```

`\settrowcenter` Typeset (recursively) rows of centered text.

```
7545 \def\settrowcenter #1\\{%
7546    \ifx #1& \let\NEXT\relax
7547    \else \centerline{\settcellcenter #1&\\&\\&}
7548         \let\NEXT=\settrowcenter
7549    \fi\NEXT}
7550
7551 %
```

`\nullsetzen` `\newcommand{\nullsetzen}{%`

```
7553    \stepl@dcolcount%
7554    \l@dcolwidth=0pt%
7555    \ifnum\l@dcolcount=30\let\NEXT\relax%
7556         \l@dcolcount=0\relax
7557    \else\let\NEXT\nullsetzen%
7558    \fi\NEXT}
7559
7560 %
```

`\edatleft` `\edatleft[`⟨*math*⟩`]{`⟨*symbol*⟩`}{`⟨*len*⟩`}`. Left ⟨*symbol*⟩, 2⟨*len*⟩ high with prepended ⟨*math*⟩ vertically centered.

```
7561 \newcommand{\edatleft}[3][\@empty]{%
7562   \ifx#1\@empty
7563     \vbox to 10pt{\vss\hbox{$\left#2\vrule width0pt height #3
7564                       depth 0pt \right. $}\hss}\vfil}
7565   \else
7566     \vbox to 4pt{\vss\hbox{$#1\left#2\vrule width0pt height #3
7567                depth 0pt \right. $}\vfil}
7568   \fi}
7569 %
```

`\edatright` `\edatright[`⟨*math*⟩`]{`⟨*symbol*⟩`}{`⟨*len*⟩`}`. Right ⟨*symbol*⟩, 2⟨*len*⟩ high with appended ⟨*math*⟩ vertically centered.

```
7570 \newcommand{\edatright}[3][\@empty]{%
7571   \ifx#1\@empty
7572     \vbox to 10pt{\vss\hbox{$\left.\vrule width0pt height #3
7573                depth 0pt \right#2 $}\hss}\vfil}
```

```
7574    \else
7575      \vbox to 4pt{\vss\hbox{$\left.\vrule width0pt height #3
7576                 depth 0pt \right#2 #1 $}\vfil}
7577    \fi}
7578
7579 %
```

\edvertline   \edvertline{⟨*len*⟩} vertical line ⟨*len*⟩ high.

```
7580 \newcommand{\edvertline}[1]{\vbox to 8pt{\vss\hbox{\vrule height #1}\vfil}}
7581
7582 %
```

\edvertdots   \edvertdots{⟨*len*⟩} vertical dotted line ⟨*len*⟩ high.

```
7583 \newcommand{\edvertdots}[1]{\vbox to 1pt{\vss\vbox to #1%
7584         {\cleaders\hbox{$\m@th\hbox{.}\vbox to 0.5em{ }$}\vfil}}}
7585
7586 %
```

\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@.

```
7587 \newcommand{\l@dtabaddcols}[2]{%
7588   \l@dcheckstartend{#1}{#2}%
7589   \ifl@dstartendok
7590   \setcounter{addcolcount}{#1}%
7591   \@whilenum \value{addcolcount}<#2\relax \do
7592   {\advance\edfilldimen by \the \csname dcol\theaddcolcount\endcsname
7593    \advance\edfilldimen by \edtabcolsep
7594    \stepcounter{addcolcount}}%
7595   \advance\edfilldimen by \the \csname dcol\theaddcolcount\endcsname
7596   \fi
7597 }
7598
7599 %
```

\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.

```
7600 \newif\ifl@dstartendok
7601 \newcommand{\l@dcheckstartend}[2]{%
7602   \l@dstartendoktrue
7603   \ifnum #1<\@ne
7604     \l@dstartendokfalse
7605     \led@err@LowStartColumn
7606   \fi
7607   \ifnum #2>30\relax
7608     \l@dstartendokfalse
```

```
7609        \led@err@HighEndColumn
7610      \fi
7611      \ifnum #1>#2\relax
7612        \l@dstartendokfalse
7613        \led@err@ReverseColumns
7614      \fi
7615    }
7616
7617    %
```

\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.

```
7618    \newcommand*{\edrowfill}[3]{%
7619      \l@dtabaddcols{#1}{#2}%
7620      \hb@xt@ \the\l@dcolwidth{\hb@xt@ \the\edfilldimen{#3}\hss}}
7621    \let\@edrowfill@=\edrowfill
7622    \def\@EDROWFILL@#1#2#3{\@edrowfill@{#1}{#2}{#3}}
7623
7624    %
```

\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.

```
7625    \newcommand{\leftltab}[1]{%
7626      \hb@xt@\z@{\vbox{\edtabindent%
7627      \moveleft\Hilfsskip\hbox{\ #1}}\hss}}
7628
7629    %
```

\leftrtab \leftrtab{⟨*text*⟩}{⟨*math*⟩} for \edbeforetab in \rtab.

```
7630    \newcommand{\leftrtab}[2]{%
7631      #2\hb@xt@\z@{\vbox{\edtabindent%
7632        \advance\Hilfsskip by\dcoli%
7633        \moveleft\Hilfsskip\hbox{\ #1}}\hss}}
7634
7635    %
```

\leftctab \leftctab{⟨*text*⟩}{⟨*math*⟩} for \edbeforetab in \ctab.

```
7636    \newcommand{\leftctab}[2]{%
7637            \hb@xt@\z@{\vbox{\edtabindent\l@dcolcount=\l@dampcount%
```

```
7638        \advance\Hilfsskip by 0.5\dcoli%
7639        \setbox\hilfsbox=\hbox{\def\edlabel##1{}%
7640        \disablel@dtabfeet$\displaystyle{#2}$%
7641        \advance\Hilfsskip by -0.5\wd\hilfsbox%
7642        \moveleft\Hilfsskip\hbox{\ #1}}\hss}%
7643        #2}
7644
7645 %
```

\rightctab   \rightctab{⟨*math*⟩}{⟨*text*⟩} for \edaftertab in \ctab.

```
7646 \newcommand{\rightctab}[2]{%
7647        \setbox\hilfsbox=\hbox{\def\edlabel##1{}%
7648        \disablel@dtabfeet#2}\l@dampcount=\l@dcolcount%
7649        #1\hb@xt@\z@{\vbox{\edtabindent\l@dcolcount=\l@dampcount%
7650        \advance\Hilfsskip by 0.5\l@dcolwidth%
7651        \advance\Hilfsskip by -\wd\hilfsbox%
7652        \setbox\hilfsbox=\hbox{\def\edlabel##1{}%
7653        \disablel@dtabfeet$\displaystyle{#1}$%
7654        \advance\Hilfsskip by -0.5\wd\hilfsbox%
7655        \advance\Hilfsskip by \edtabcolsep%
7656        \moveright\Hilfsskip\hbox{ #2}}\hss}%
7657        }
7658
7659 %
```

\rightltab   \rightltab{⟨*math*⟩}{⟨*text*⟩} for \edaftertab in \ltab.

```
7660 \newcommand{\rightltab}[2]{%
7661        \setbox\hilfsbox=\hbox{\def\edlabel##1{}%
7662        \disablel@dtabfeet#2}\l@dampcount=\l@dcolcount%
7663        #1\hb@xt@\z@{\vbox{\edtabindent\l@dcolcount=\l@dampcount%
7664        \advance\Hilfsskip by\l@dcolwidth%
7665        \advance\Hilfsskip by-\wd\hilfsbox%
7666        \setbox\hilfsbox=\hbox{\def\edlabel##1{}%
7667        \disablel@dtabfeet$\displaystyle{#1}$%
7668        \advance\Hilfsskip by-\wd\hilfsbox%
7669        \advance\Hilfsskip by\edtabcolsep%
7670        \moveright\Hilfsskip\hbox{ #2}}\hss}%
7671        }
7672
7673 %
```

\rightrtab   \rightrtab{⟨*math*⟩}{⟨*text*⟩} for \edaftertab in \rtab.

```
7674 \newcommand{\rightrtab}[2]{%
7675        \setbox\hilfsbox=\hbox{\def\edlabel##1{}%
7676        \disablel@dtabfeet#2}%
7677        #1\hb@xt@\z@{\vbox{\edtabindent%
7678        \advance\Hilfsskip by-\wd\hilfsbox%
```

```
7679            \advance\Hilfsskip by\edtabcolsep%
7680            \moveright\Hilfsskip\hbox{ #2}}\hss}%
7681            }
7682
7683  %
```

\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.

```
7684  \newcommand{\rtab}[1]{%
7685    \l@dnullfills
7686      \def\edbeforetab##1##2{\leftrtab{##1}{##2}}%
7687      \def\edaftertab##1##2{\rightrtab{##1}{##2}}%
7688      \measurembody{#1}%
7689    \l@drestorefills
7690      \variab
7691      \setmrowright #1\\&\\%
7692      \enablel@dtabfeet}
7693
7694  %
```

\measurembody \measurembody{⟨*body*⟩} measures the array ⟨*body*⟩.

```
7695  \newcommand{\measurembody}[1]{%
7696    \disablel@dtabfeet%
7697    \l@dcolcount=0%
7698    \nullsetzen%
7699    \l@dcolcount=0
7700    \measuremrow #1\\&\\%
7701    \global\l@dampcount=1}
7702
7703  %
```

\rtabtext   \rtabtext{⟨*body*⟩} typesets ⟨*body*⟩ as a tabular with the entries right justified.

```
7704  \newcommand{\rtabtext}[1]{%
7705    \l@dnullfills
7706      \measuretbody{#1}%
7707    \l@drestorefills
7708      \variab
7709      \settrowright #1\\&\\%
7710      \enablel@dtabfeet}
7711
7712  %
```

\measuretbody \measuretbody{⟨*body*⟩} measures the tabular ⟨*body*⟩.

```
7713 \newcommand{\measuretbody}[1]{%
7714   \disable@notes%
7715   \disablel@dtabfeet%
7716   \l@dcolcount=0%
7717   \nullsetzen%
7718   \l@dcolcount=0
7719   \measuretrow #1\\&\\%
7720   \restore@notes%
7721   \global\l@dampcount=1}
7722
7723 %
```

\ltab   Array with entries left justified.

\edbeforetab
\edaftertab
```
7724 \newcommand{\ltab}[1]{%
7725   \l@dnullfills
7726     \def\edbeforetab##1##2{\leftltab{##1}{##2}}%
7727     \def\edaftertab##1##2{\rightltab{##1}{##2}}%
7728     \measurembody{#1}%
7729   \l@drestorefills
7730     \variab
7731     \setmrowleft #1\\&\\%
7732     \enablel@dtabfeet}
7733
7734 %
```

\ltabtext   Tabular with entries left justified.

```
7735 \newcommand{\ltabtext}[1]{%
7736   \l@dnullfills
7737     \measuretbody{#1}%
7738   \l@drestorefills
7739     \variab
7740     \settrowleft #1\\&\\%
7741     \enablel@dtabfeet}
7742
7743 %
```

\ctab   Array with centered entries.

\edbeforetab
\edaftertab
```
7744 \newcommand{\ctab}[1]{%
7745   \l@dnullfills
7746     \def\edbeforetab##1##2{\leftctab{##1}{##2}}%
7747     \def\edaftertab##1##2{\rightctab{##1}{##2}}%
7748     \measurembody{#1}%
7749   \l@drestorefills
7750     \variab
7751     \setmrowcenter #1\\&\\%
7752     \enablel@dtabfeet}
7753
7754 %
```

`\ctabtext`   Tabular with entries centered.

```
7755 \newcommand{\ctabtext}[1]{%
7756   \l@dnullfills
7757     \measuretbody{#1}%
7758   \l@drestorefills
7759     \variab
7760     \settrowcenter #1\\&\\%
7761     \enablel@dtabfeet}
7762
7763 %
```

`\spreadtext` `\newcommand{\spreadtext}[1]{%\l@dcolcount=\l@dampcount%`

```
7765     \hb@xt@ \the\l@dcolwidth{\hbox{#1}\hss}}
7766 %
```

`\spreadmath` `\newcommand{\spreadmath}[1]{%`

```
7768     \hb@xt@ \the\l@dcolwidth{\hbox{$\displaystyle{#1}$}\hss}}
7769
7770 %
```

`\HILFSskip`   More helpers.
`\Hilfsskip`
```
7771 \newskip\HILFSskip
7772 \newskip\Hilfsskip
7773
7774 %
```

`\EDTABINDENT` `\newcommand{\EDTABINDENT}{%`

```
7776     \ifnum\l@dcolcount=30\let\NEXT\relax\l@dcolcount=0%
7777     \else\stepl@dcolcount%
7778         \advance\Hilfsskip by\l@dcolwidth%
7779         \ifdim\l@dcolwidth=0pt\advance\hilfscount\@ne
7780         \else\advance\Hilfsskip by \the\hilfscount\edtabcolsep%
7781         \hilfscount=1\fi%
7782         \let\NEXT=\EDTABINDENT%
7783     \fi\NEXT}%
7784 %
```

`\edtabindent` (was `\tabindent`)

```
7785 \newcommand{\edtabindent}{%
7786     \l@dcolcount=0\relax
7787     \Hilfsskip=0pt%
7788     \hilfscount=1\relax
7789     \EDTABINDENT%
7790     \hilfsskip=\hsize%
```

```
7791        \advance\hilfsskip -\Hilfsskip%
7792        \Hilfsskip=0.5\hilfsskip%
7793        }%
7794
7795 %
```

\EDTAB   (was \TAB)

```
7796 \def\EDTAB #1|#2|{%
7797        \setbox\tabhilfbox=\hbox{$\displaystyle{#1}$}%
7798        \setbox\tabHilfbox=\hbox{$\displaystyle{#2}$}%
7799        \advance\tabelskip -\wd\tabhilfbox%
7800        \advance\tabelskip -\wd\tabHilfbox%
7801        \unhbox\tabhilfbox\hskip\tabelskip%
7802        \unhbox\tabHilfbox}%
7803
7804 %
```

\EDTABtext   (was \TABtext)

```
7805 \def\EDTABtext #1|#2|{%
7806        \setbox\tabhilfbox=\hbox{#1}%
7807        \setbox\tabHilfbox=\hbox{#2}%
7808        \advance\tabelskip -\wd\tabhilfbox%
7809        \advance\tabelskip -\wd\tabHilfbox%
7810        \unhbox\tabhilfbox\hskip\tabelskip%
7811        \unhbox\tabHilfbox}%
7812 %
```

\tabhilfbox   Further helpers.
\tabHilfbox
```
7813 \newbox\tabhilfbox
7814 \newbox\tabHilfbox
7815
7816 %
```

XXX.2.4   Environments

edarrayl edarrayc edarrayr The 'environment' forms for \ltab, \ctab and \rtab.

```
7817 \newenvironment{edarrayl}{\l@dcollect@body\ltab}{}
7818 \newenvironment{edarrayc}{\l@dcollect@body\ctab}{}
7819 \newenvironment{edarrayr}{\l@dcollect@body\rtab}{}
7820
7821 %
```

edtabularl edtabularc edtabularr The 'environment' forms for \ltabtext, \ctabtext and \rtabtext.

315

```
7822 \newenvironment{edtabularl}{\l@dcollect@body\ltabtext}{}
7823 \newenvironment{edtabularc}{\l@dcollect@body\ctabtext}{}
7824 \newenvironment{edtabularr}{\l@dcollect@body\rtabtext}{}
7825
7826 %
```

XXXI   Quotation's commands

\initnumbering@quote This macro, called at the beginning of any numbered section, locally redefines the quotation and quote environments, in order to allow their use inside of numbered sections.

\quotation \initnumbering@quote defines quotation environment.
\endquotation
\quote
\endquote

```
7827 \newcommand{\initnumbering@quote}{
7828     \ifnoquotation@\else
7829     \renewcommand{\quotation}{\par\leavevmode%
7830                                 \parindent=1.5em%
7831                                 \skipnumbering%
7832                                 \ifautopar%
7833                                     \vskip-\parskip%
7834                                 \else%
7835                                     \vskip\topsep%
7836                                 \fi%
7837                                 \global\leftskip=\leftmargin%
7838                                 \global\rightskip=\leftmargin%
7839                             }
7840     \renewcommand{\endquotation}{\par%
7841                                 \global\leftskip=0pt%
7842                                 \global\rightskip=0pt%
7843                                 \leavevmode%
7844                                 \skipnumbering%
7845                                 \ifautopar%
7846                                     \vskip-\parskip%
7847                                 \else%
7848                                     \vskip\topsep%
7849                                 \fi%
7850                             }
7851     \renewcommand{\quote}{\par\leavevmode%
7852                                 \parindent=0pt%
7853                                 \skipnumbering%
7854                                 \ifautopar%
7855                                     \vskip-\parskip%
7856                                 \else%
7857                                     \vskip\topsep%
7858                                 \fi%
7859                                 \global\leftskip=\leftmargin%
7860                                 \global\rightskip=\leftmargin%
7861     }
```

```
7862    \renewcommand{\endquote}{\par%
7863                           \global\leftskip=0pt%
7864                           \global\rightskip=0pt%
7865                           \leavevmode%
7866                           \skipnumbering%
7867                           \ifautopar%
7868                               \vskip-\parskip%
7869                           \else%
7870                               \vskip\topsep%
7871                           \fi%
7872                           }
7873    \fi
7874 }
7875 %
```

XXXII Section's title commands

XXXII.1 Commands to disable some feature

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

```
7876 \newcommand{\ledsectnotoc}{\let\addcontentsline\@gobblethree}
7877 %
```

\ledsectnomark   The \ledsectnomark only disables the \chaptermark, \sectionmark and \subsectionmark macros.

```
7878 \newcommand{\ledsectnomark}{%
7879   \let\chaptermark\@gobble%
7880   \let\sectionmark\@gobble%
7881   \let\subsectionmark\@gobble%
7882 }
7883 %
```

XXXII.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.

XXXII.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).

```
7884 \catcode`\#=12
7885 \notbool{@noeled@sec}{%
7886 %
```

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

```
7887 \ifl@dmemoir
7888   \newcommand\beforeeledchapter{%
7889   \clearforchapter%
7890 }
7891 \else
7892   \newcommand\beforeeledchapter{%
7893   \if@openright%
7894     \cleardoublepage%
7895   \else%
7896     \clearpage%
7897   \fi%
7898 }
7899 \fi
7900 %
```

XXXII.4 Auxiliary commands

\print@leftmargin@eledsection \print@leftmargin@eledsection and \print@rightmargin@eledsection are added
\print@rightmargin@eledsection by reledmac inside the code of sectioning command, in order to affix lines numbers. They include tests for RTL languages.

```
7901 \def\print@rightmargin@eledsection{%
7902   \if@eled@sectioning%
7903     \begingroup%
```

```
7904      \if@RTL%
7905          \let\llap\rlap%
7906          \let\leftlinenum\rightlinenum%
7907          \let\leftlinenumR\rightlinenumR%
7908          \let\l@drd@ta\l@dld@ta%
7909          \let\l@drsn@te\l@dlsn@te%
7910      \fi%
7911      \hfill\l@drd@ta \csuse{LR}{\l@drsn@te}%
7912      \endgroup%
7913    \fi%
7914 }%
7915
7916 \def\print@leftmargin@eledsection{%
7917    \if@eled@sectioning%
7918      \leavevmode%
7919      \begingroup%
7920      \if@RTL%
7921          \let\rlap\llap%
7922          \let\rightlinenum\leftlinenum%
7923          \let\rightlinenumR\leftlinenumR%
7924          \let\l@dld@ta\l@drd@ta%
7925          \let\l@dlsn@te\l@drsn@te%
7926      \fi%
7927      \l@dld@ta\csuse{LR}{\l@dlsn@te}%
7928      \endgroup%
7929    \fi%
7930 }%
7931
7932 %
```

XXXII.5   Patching standard commands

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.

```
7933 \AtBeginDocument{%
7934
7935
7936 \pretocmd{\M@sect}
7937    {\let\old@edtext=\edtext%
7938    \let\edtext=\dummy@edtext@showlemma%
7939    }
```

```
7940    {}
7941    {}
7942
7943  \apptocmd{\M@sect}
7944    {\let\edtext=\old@edtext}
7945    {}
7946    {}
7947
7948  \patchcmd{\M@sect}
7949    { #9}
7950    { #9%
7951  \print@rightmargin@eledsection%
7952    }
7953    {}
7954    {}
7955
7956  \patchcmd{\M@sect}
7957    {\hskip #3\relax}
7958    {\hskip #3\relax%
7959  \print@leftmargin@eledsection%
7960    }
7961    {}
7962    {}
7963
7964  \patchcmd{\@mem@old@ssect}
7965    {#5}
7966    {#5%
7967  \print@leftmargin@eledsection%
7968    }
7969    {}
7970    {}
7971
7972  \patchcmd{\@mem@old@ssect}
7973    {\hskip #1}
7974    {\hskip #1%
7975  \print@rightmargin@eledsection%
7976    }
7977    {}
7978    {}
7979
7980
7981
7982  \patchcmd{\scr@startchapter}{\if@openright\cleardoublepage\else\clearpage\
      fi}{%
7983    \if@eled@sectioning\else%
7984      \ifl@dprintingpages\else%
7985        \if@openright\cleardoublepage\else\clearpage\fi%No clearpage inside a
      \Pages: will keep critical notes from printing on the title page. Here for
      scrbook.
7986      \fi%
```

```
7987  \fi%
7988  }
7989  {}
7990  {}

7992  \patchcmd{\@makechapterhead}
7993    {#1}
7994    {\print@leftmargin@eledsection%
7995      #1%
7996    \print@rightmargin@eledsection%
7997    }
7998    {}
7999    {}

8001  \patchcmd{\@makechapterhead}% For BIDI
8002    {\if@RTL\raggedleft\else\raggedright\fi}%
8003    {\if@eled@sectioning\else%
8004      \if@RTL\raggedleft\else\raggedright\fi%
8005    \fi%
8006    }%
8007    {}%
8008    {}%

8010  \patchcmd{\@makeschapterhead}
8011    {#1}
8012    {\print@leftmargin@eledsection%
8013      #1%
8014     \print@rightmargin@eledsection%
8015    }
8016    {}
8017    {}

8019  \pretocmd{\@sect}
8020    {\let\old@edtext=\edtext
8021    \let\edtext=\dummy@edtext@showlemma%
8022    }
8023    {}
8024    {}

8026  \apptocmd{\@sect}
8027    {\let\edtext=\old@edtext}
8028    {}
8029    {}

8031  \pretocmd{\@ssect}
8032    {\let\old@edtext=\edtext%
8033    \let\edtext=\dummy@edtext@showlemma%
8034    }
8035    {}
8036    {}
```

```
8037
8038  \apptocmd{\@ssect}
8039    {\let\edtext=\old@edtext}
8040    {}
8041    {}
8042
8043  %
```

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.

```
8044  \@ifpackageloaded{nameref}{
8045
8046    \patchcmd{\NR@sect}
8047      {#8}
8048      {#8%
8049      \print@rightmargin@eledsection%
8050      }
8051      {}
8052      {}
8053
8054    \patchcmd{\NR@sect}
8055      {\hskip #3\relax}
8056      {\hskip #3\relax%
8057      \print@leftmargin@eledsection%
8058      }
8059      {}
8060      {}
8061
8062    \patchcmd{\NR@ssect}
8063      {#5}
8064      {#5%
8065      \print@rightmargin@eledsection%
8066      }
8067      {}
8068      {}
8069
8070    \patchcmd{\NR@ssect}
8071      {\hskip #1}
8072      {\hskip #1%
8073      \print@leftmargin@eledsection%
8074      }
8075      {}
8076      {}
8077  }%
8078    {
8079    \patchcmd{\@sect}
8080      {#8}
8081      {#8%
8082      \print@rightmargin@eledsection%
8083      }
```

```
8084      {}
8085      {}
8086
8087   \patchcmd{\@sect}
8088      {\hskip #3\relax}
8089      {\hskip #3\relax%
8090      \print@leftmargin@eledsection%
8091      }
8092      {}
8093      {}
8094
8095   \patchcmd{\@ssect}
8096      {#5}
8097      {#5%
8098      \print@rightmargin@eledsection%
8099      }
8100      {}
8101      {}
8102
8103   \patchcmd{\@ssect}
8104      {\hskip #1}
8105      {\hskip #1%
8106      \print@leftmargin@eledsection%
8107      }
8108      {}
8109      {}
8110   }%
8111 }%
8112 %
```

Close the \notbool{@noeled@sec} statement. Also, we have finished patching the commands, using # with a catcode equal to 12, so we are restoring the normal catcode for #.

```
8113 {}}%
8114 \protect\catcode`\#=6 %Space NEEDS by \catcode
8115 %
```

\chapter   We patch the \chapter command even if the noeledsec option is called, because we can use \chapter in the optional argument of a \pstart in parallel typesetting.

```
8116 \AtBeginDocument{%
8117 \patchcmd{\chapter}{\clearforchapter}{%
8118   \if@eled@sectioning\else%
8119     \ifl@dprintingpages\else%
8120       \clearforchapter%
8121     \fi%
8122   \fi%
8123   }%
8124   {}%
8125   {}%
```

```
8126
8127  \patchcmd{\chapter}{\if@openright\cleardoublepage\else\clearpage\fi}{%
8128    \if@eled@sectioning\else%
8129      \ifl@dprintingpages%
8130        \endgraf%
8131      \else%
8132        \if@openright\cleardoublepage\else\clearpage\fi%No clearpage inside a
      \Pages: will keep critical notes from printing on the title page. Here for
      classical  classes
8133      \fi%
8134    \fi%
8135    }%
8136    {}%
8137    {}%
8138  }%
8139  %
```

\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.

```
8140  \newif\if@eled@sectioning%
8141  %
```

We reopen a new \notbool{@noeled@sec} statement, as we will define the \elesection
commands.

```
8142  \notbool{@noeled@sec}{%
8143  %
```

XXXII.6   Main code of \eledxxx commands

\eled@sectioning@out   \eled@sectioning@out is the output file, to dump the pstarts where a sectioning com-
mand is used.

```
8144  \newwrite\eled@sectioning@out
8145  %
```

\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*

```
8146  \newcommand{\eledchapter}[2][]{%
8147    #2%
8148    \ifledRcol%
8149      \immediate\write\eled@sectioningR@out{%
8150        \string\eled@chapter{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR}{}{R}
8151        }%
8152    \else%
8153      \immediate\write\eled@sectioning@out{%
```

```
8154        \string\eled@chapter{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL}{}{}
8155        }%
8156    \fi%
8157 }
8158
8159 \newcommand{\eledsection}[2][]{%
8160    #2%
8161    \ifledRcol%
8162        \immediate\write\eled@sectioningR@out{%
8163        \string\eled@section{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR}{}{R}
8164        }%
8165    \else%
8166        \immediate\write\eled@sectioning@out{%
8167        \string\eled@section{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL}{}{}
8168        }%
8169    \fi%
8170 }
8171
8172 \newcommand{\eledsubsection}[2][]{%
8173    #2%
8174    \ifledRcol%
8175        \immediate\write\eled@sectioningR@out{%
8176        \string\eled@subsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR}{}{R
}
8177        }%
8178    \else%
8179        \immediate\write\eled@sectioning@out{%
8180        \string\eled@subsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL}{}{}
8181        }%
8182    \fi%
8183 }
8184 \newcommand{\eledsubsubsection}[2][]{%
8185    #2%
8186    \ifledRcol%
8187        \immediate\write\eled@sectioningR@out{%
8188        \string\eled@subsubsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR
}{}{R}
8189        }%
8190    \else%
8191        \immediate\write\eled@sectioning@out{%
8192        \string\eled@subsubsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL
}{}{}
8193        }%
8194    \fi%
8195 }
8196
8197
8198 \WithSuffix\newcommand\eledchapter*[2][]{%
8199    #2%
8200    \ifledRcol%
```

```
8201    \immediate\write\eled@sectioningR@out{%
8202      \string\eled@chapter{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR}{*}{R}
8203      }%
8204  \else%
8205    \immediate\write\eled@sectioning@out{%
8206      \string\eled@chapter{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL}{*}{}
8207      }%
8208  \fi%
8209 }
8210
8211 \WithSuffix\newcommand\eledsection*[2][]{%
8212   #2%
8213   \ifledRcol%
8214     \immediate\write\eled@sectioningR@out{%
8215       \string\eled@section{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR}{*}{R}
8216       }%
8217   \else%
8218     \immediate\write\eled@sectioning@out{%
8219       \string\eled@section{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL}{*}{}
8220       }%
8221   \fi%
8222 }
8223
8224 \WithSuffix\newcommand\eledsubsection*[2][]{%
8225   #2%
8226   \ifledRcol%
8227     \immediate\write\eled@sectioningR@out{%
8228       \string\eled@subsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR}{*}{
8229 R}
8230       }%
8230   \else%
8231     \immediate\write\eled@sectioning@out{%
8232       \string\eled@subsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL
8232 }{*}{}
8233       }%
8234   \fi%
8235 }
8236
8237 \WithSuffix\newcommand\eledsubsubsection*[2][]{%
8238   #2%
8239   \ifledRcol%
8240     \immediate\write\eled@sectioningR@out{%
8241       \string\eled@subsubsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR
8241 }{*}{R}
8242       }%
8243   \else%
8244     \immediate\write\eled@sectioning@out{%
8245       \string\eled@subsubsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL
8245 }{*}{}
8246       }%
```

```
8247    \fi%
8248 }
8249 %
```

XXXII.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.

```
8250 \def\eled@chapter#1#2#3#4#5{%
8251     \ifstrempty{#4}%
8252        {%
8253        \ifstrempty{#1}%
8254           {%
8255           \csgdef{eled@sectioning@#3#5}{\let\edtext=\dummy@edtext@showlemma\
     chapter{#2}}%
8256           \csgdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext{}\chaptermark
     {#2}}%
8257           }%Need for \pairs, because of using parbox.
8258           {%
8259           \csgdef{eled@sectioning@#3#5}{\let\edtext=\dummy@edtext@showlemma\
     chapter[#1]{#2}}%
8260           \csgdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext{}\chaptermark
     {#2}}%Need for \pairs, because of using parbox.
8261           }%
8262        }%
8263        {%
8264        \ifstrempty{#1}%
8265           {\csgdef{eled@sectioning@#3#5}{\let\edtext=\dummy@edtext@showlemma\
     chapter*{#2}}}%
8266           {\csgdef{eled@sectioning@#3#5}{\let\edtext=\dummy@edtext@showlemma\
     chapter*[#1]{#2}}}%Bug in LaTeX!
8267        }%
8268     \listcsgadd{eled@sections#5@@}{#3}%
8269        }
8270 \def\eled@section#1#2#3#4#5{%
8271     \ifstrempty{#4}%
8272        {\ifstrempty{#1}%
8273           {%
8274           \csgdef{eled@sectioning@#3#5}{\section{#2}}%
```

```
8275        \csgdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext{}\sectionmark
     {#2}}%Need for \pairs, because of using parbox.
8276          }%
8277          {%
8278          \csgdef{eled@sectioning@#3#5}{\section[#1]{#2}}%
8279          \csgdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext{}\sectionmark
     {#1}}%Need for \pairs, because of using parbox.
8280          }%
8281        }%
8282        {\ifstrempty{#1}%
8283          {\csgdef{eled@sectioning@#3#5}{\section*{#2}}}%
8284          {\csgdef{eled@sectioning@#3#5}{\section*[#1]{#2}}}%Bug in LaTeX!
8285        }
8286    \listcsgadd{eled@sections#5@@}{#3}%
8287      }
8288 \def\eled@subsection#1#2#3#4#5{%
8289      \ifstrempty{#4}%
8290        {\ifstrempty{#1}%
8291          {%
8292          \csgdef{eled@sectioning@#3#5}{\subsection{#2}}%
8293          \csgdef{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)
8294          }%
8295          {%
8296          \csgdef{eled@sectioning@#3#5}{\subsection[#1]{#2}}%
8297          \csgdef{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)
8298          }%
8299        }%
8300        {\ifstrempty{#1}%
8301          {\csgdef{eled@sectioning@#3#5}{\subsection*{#2}}}%
8302          {\csgdef{eled@sectioning@#3#5}{\subsection*[#1]{#2}}}%Bug in LaTeX!
8303        }
8304    \listcsgadd{eled@sections#5@@}{#3}%
8305      }
8306 \def\eled@subsubsection#1#2#3#4#5{%
8307      \ifstrempty{#4}%
8308        {\ifstrempty{#1}%
8309          {\csgdef{eled@sectioning@#3#5}{\subsubsection{#2}}}%
8310          {\csgdef{eled@sectioning@#3#5}{\subsubsection[#1]{#2}}}%
8311        }%
8312        {\ifstrempty{#1}%
8313          {\csgdef{eled@sectioning@#3#5}{\subsubsection*{#2}}}%
8314          {\csgdef{eled@sectioning@#3#5}{\subsubsection*[#1]{#2}}}%Bug in
     LaTeX!
8315        }
8316    \listcsgadd{eled@sections#5@@}{#3}%
8317      }
```

8318
8319 %

End of the conditional test about `noeledsec` option.

8320 `}{}`
8321 %

XXXIII   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.

8322 `\def\normal@page@break{}`
8323 %

\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.

8324 `\def\l@prev@pb{}`
8325 `\def\l@prev@nopb{}`
8326 %

\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.

8327 `\newcommand{\ledpb}{\write\linenum@out{\string\led@pb}}`
8328 `\newcommand{\ledpbnum}[1]{\write\linenum@out{\string\led@pbnum{#1}}}`
8329 `\newcommand{\lednopb}{\write\linenum@out{\string\led@nopb}}`
8330 `\newcommand{\lednopbnum}[1]{\write\linenum@out{\string\led@nopbnum{#1}}}`
8331 %

\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

```
8332 \newcommand{\led@pb}{\listxadd{\l@prev@pb}{\the\absline@num}}
8333 \newcommand{\led@pbnum}[1]{\listxadd{\l@prev@pb}{#1}}
8334 \newcommand{\led@nopb}{\listxadd{\l@prev@nopb}{\the\absline@num}}
8335 \newcommand{\led@nopbnum}[1]{\listxadd{\l@prev@nopb}{#1}}
8336 %
```

\ledpbsetting   The \ledpbsetting macro only changes the value of \led@pb@macro, for which the
\led@pb@setting default value is before.

```
8337 \def\led@pb@setting{before}
8338 \newcommand{\ledpbsetting}[1]{\gdef\led@pb@setting{#1}}
8339 %
```

\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.

```
8340 \newcommand{\led@check@pb}{\xifinlist{\the\absline@num}{\l@prev@pb}{\
     pagebreak[4]}{}}
8341 \newcommand{\led@check@nopb}{%
8342    \IfStrEq{\led@pb@setting}{before}{%
8343      \xifinlist{\the\absline@num}{\l@prev@nopb}%
8344        {\numdef{\abs@prevline}{\the\absline@num-1}%
8345        \xifinlist{\abs@prevline}{\normal@page@break}%
8346          {\nopagebreak[4]\enlargethispage{\baselineskip}}%
8347          {}}%
8348        {}}%
8349      {}%
8350    {}%
8351    \IfStrEq{\led@pb@setting}{after}{%
8352      \xifinlist{\the\absline@num}{\l@prev@nopb}{%
8353        \xifinlist{\the\absline@num}{\normal@page@break}%
8354          {\nopagebreak[4]\enlargethispage{\baselineskip}}%
8355          {}%
8356 }%
8357        {}}%
8358      {}%
8359    {}%
8360 }
8361 %
```

XXXIV 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.

```
8362 \newcommand{\check@pb@in@verse}{%
8363     \ifinstanza\iflednopbinverse\ifinserthangingsymbol% Using stanzas and
enabling page breaks in verse control, while on a hanging verse.
8364         \ifnum\page@num=\last@page@num\else%If we have change page
8365             \IfStrEq{\led@pb@setting}{before}{%
8366                 \numgdef{\abs@line@verse}{\the\absline@num-1}%
8367                 \ledpbnum{\abs@line@verse}%
8368             }{}%
8369             \IfStrEq{\led@pb@setting}{after}{%
8370                 \numgdef{\abs@line@verse}{\the\absline@num-1}%
8371                 \lednopbnum{\abs@line@verse}%
8372             }{}%
8373         \fi%
8374     \fi\fi\fi%
8375 }
8376 %
```

XXXV   Tools for `hyperref` package

\Hy@raisedlink@left   The hyperref package provides a \Hy@raisedlink command, to be used to add an anchor to the top of a line and not to the bottom of it.[34]

However, this command disrupts the line breaking mechanism when it is called before any word. This is why reledmac defines \Hy@raisedlink@left that is called to the left of words, at the beginning of \edtext or inside the \edlabel commands.[35]

```
8377 \def\Hy@raisedlink@left#1{%
8378     \ifvmode
8379         #1%
8380     \else
8381         \Hy@SaveSpaceFactor
8382         \llap{\smash{%
8383         \begingroup
8384             \let\HyperRaiseLinkLength\@tempdima
8385             \setlength\HyperRaiseLinkLength\HyperRaiseLinkDefault
8386             \HyperRaiseLinkHook
8387         \expandafter\endgroup
```

---

[34]http://tex.stackexchange.com/a/17138/7712.

[35]The code is inspired by an answer given by @unbonpetit. Thanks to him. http://texnique.fr:80/osqa/questions/781/hyraisedlink-perturbe-la-maniere-dont-se-fait-la-coupure-de-ligne/801.

```
8388        \expandafter\raise\the\HyperRaiseLinkLength\hbox{%
8389            \Hy@RestoreSpaceFactor
8390            #1%
8391            \Hy@SaveSpaceFactor
8392        }%
8393        }}%
8394        \Hy@RestoreSpaceFactor
8395        \penalty\@M\hskip\z@\relax
8396    \fi
8397 }
8398 %
```

XXXVI Compatibility with `eledmac`

Here, we define some commands for the `eledmac-compat` option.

```
8399 \ifeledmaccompat@%
8400
8401  \newcommand{\footnormalX}[1]{\arrangementX[#1]{normal}}%
8402  \newcommand{\footparagraphX}[1]{\arrangementX[#1]{paragraph}}%
8403  \newcommand{\foottwocolX}[1]{\arrangementX[#1]{twocol}}%
8404  \newcommand{\footthreecolX}[1]{\XarrangementX[#1]{threecol}}%
8405
8406  \unless\ifnocritical@
8407    \newcommand{\footnormal}[1]{\Xarrangement[#1]{normal}}%
8408    \newcommand{\footparagraph}[1]{\Xarrangement[#1]{paragraph}}%
8409    \newcommand{\foottwocol}[1]{\Xarrangement[#1]{twocol}}%
8410    \newcommand{\footthreecol}[1]{\Xarrangement[#1]{threecol}}%
8411    \let\hsizetwocol\Xhsizetwocol
8412    \let\hsizethreecol\Xhsizethreecol
8413    \let\bhookXnote\Xbhooknote
8414    \let\boxsymlinenum\Xboxsymlinenum
8415    \let\symlinenum\Xsymlinenum
8416    \let\beforenumberinfootnote\Xbeforenumber
8417    \let\afternumberinfootnote\Xafternumber
8418    \let\beforeXsymlinenum\Xbeforesymlinenum
8419    \let\afterXsymlinenum\Xaftersymlinenum
8420    \let\inplaceofnumber\Xinplaceofnumber
8421    \let\Xlemmaseparator\lemmaseparator
8422    \let\afterlemmaseparator\Xafterlemmaseparator
8423    \let\beforelemmaseparator\Xbeforelemmaseparator
8424    \let\inplaceoflemmaseparator\Xinplaceoflemmaseparator
8425    \let\txtbeforeXnotes\Xtxtbeforenotes
8426    \let\afterXrule\Xafterrule
8427    \let\numberonlyfirstinline\Xnumberonlyfirstinline
8428    \let\numberonlyfirstintwolines\Xnumberonlyfirstintwolines
8429    \let\nonumberinfootnote\Xnonumberinfootnote
8430    \let\pstartinfootnote\Xpstart
8431    \let\pstartinfootnoteeverytime\Xpstarteverytime
```

```
8432        \let\onlyXpstart\Xonlypstart
8433        \let\Xnonumberinfootnote\Xnonumber
8434        \let\nonbreakableafternumber\Xnonbreakableafternumber
8435        \let\maxhXnotes\Xmaxhnotes
8436        \let\beforeXnotes\Xbeforenotes
8437        \let\boxlinenum\Xboxlinenum
8438        \let\boxlinenumalign\Xboxlinenumaligm
8439        \let\boxstartlinenum\Xboxstartlinenum
8440        \let\boxendlinenum\Xboxendlinenum
8441        \let\twolines\Xtwolines
8442        \let\morethantwolines\Xmorethantwolines
8443        \let\twolinesbutnotmore\Xtwolinesbutnotmore
8444        \let\twolinesonlyinsamepage\Xtwolinesonlyinsamepage
8445      \fi
8446
8447      \unless\ifnofamiliar@
8448        \let\notesXwidthliketwocolumns\noteswidthliketwocolumnsX
8449      \fi
8450      \newcommandx{\parafootsep}[2][1,usedefault]{%
8451          \Xparafootsep[#1]{#2}%
8452          \parafootsepX[#1]{#2}
8453      }%
8454
8455      \newcommandx{\afternote}[2][1,usedefault]{%
8456          \Xafternote[#1]{#2}%
8457          \afternoteX[#1]{#2}%
8458      }%
8459
8460      \unless\ifnoend@
8461        \let\XendXtwolines\Xendtwolines
8462        \let\XendXmorethantwolines\Xendmorethantwolines
8463        \let\bhookXendnote\Xendbhooknote
8464        \let\boxXendlinenum\Xendboxlinenum%
8465        \let\boxXendlinenumalign\Xendboxlinenumalign%
8466        \let\boxXendstartlinenum\Xendboxstartlinenum%
8467        \let\boxXendendlinenum\Xendboxendlinenum%
8468        \let\XendXlemmaseparator\Xendlemmaseparator
8469        \let\XendXbeforelemmaseparator\Xendbeforelemmaseparator
8470        \let\XendXafterlemmaseparator\Xendafterlemmaseparator
8471        \let\XendXinplaceoflemmaseparator\Xendinplaceoflemmaseparator
8472      \fi
8473
8474      \AtBeginDocument{%
8475        \ifdef\lineref{}{\let\lineref\edlineref}%
8476      }%
8477
8478
8479  \fi%
8480  %
```

```
</code>
```

Appendix A   Things to do when changing versions

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[36] 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:

`\catcode`\<=\active`

---

[36] 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.

```
\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. 115 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 6.6.1 p. 32), and remove any `\Xfootnote` command.

If you have customized the `\XXXXXXfmt` command, please check whether you can achieve the same by the commands documented for display options (7 p. 33) or `\Xfootnote` options (6.2.2 p. 24). Otherwise please add a new ticket on Github to request a new function for doing this.[37]

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 `\XXXXXXfmt` command, but you must replace:

```
\renewcommand*{XXXXfmt}[3]
```

with

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

---

[37]`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. 9.3 p. 46). 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 5.2.3 p. 18). 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 16.2 p. 61.

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 19.1.3 p. 64 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`	`\Xmaxhnotes` and `\maxhnotesX`
`\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	\Xbhooknote
\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.

Appendix A.10   Migration to `reledmac` 2.1.0

Reledmac 2.1.0 fix some bugs when using \Xbhooknote and \bhooknoteX not in order to execute code at the beginning of each notes, but to insert content of at the beginning of each notes.

   People who use these commands to do it, which is not the original idea, must change the following:

1. Horizontal space is no longer automatically added after the content of the \Xbhooknote/\bhooknoteX argument. You must include it manually. So instead of \Xbhooknote{content}, use \Xbhooknote{content }.

2. Indent is no longer automatically added before the content of the \Xbhooknote/\bhooknoteX argument. If you want to keep it, add \indent in the argument of \Xbhooknote/\bhooknoteX.

Appendix A.11   Migration to `reledmac` 2.1.3

Reledmac 2.1.3 fix an historical bug, (style in `ledmac` 0.7!) which doubled the space before the rules of paragraphed familiar footnotes. Consequently, if you use paragraphed familiar footnotes, you should maybe adapt it, playing with \beforenotesX.

Appendix A.12   Migration to `reledmac` 2.3.0

Before `reledmac` 2.3.0, for typesetting verse, any empty line was considered a paragraph inside verses. Counting empty lines this created breaking verse, hanging verses, and also added spurious vertical spaces. Version 2.3.0 disables paragraph in stanza. If you want vertical space, use optional argument of \stanza or \endverse.

Appendix A.13   Migration to `reledmac` 2.4.0

It is not mandatory, but strongly recommended, to change any \renewcommand{\endashchar}{⟨...⟩}
to the use of \Xlinerangeseparator or / and \Xendlinerangeseparator (7.2.3
p. 34).

Appendix A.14   Migration to `reledmac` 2.5.0

It is strongly recommended to stop redefining \printnpnum and to use the hooks doc-
umented in 7.3 p. 38.

\xlineref does not print anymore the side flag (R for right side), because it is in-
compatible with numerical test. Use \xflagref to obtain it.

The \printlines and \printendlines commands take now an eighth argument,
which is the side flag. It is strongly recommended to NEVER redefine these two com-
mands and to use the setting commands instead (or to ask for new setting commands if
the actual does not answer to your needs). However, if you have done it, just change
your redefinition to have a new argument.

It is strongly recommended to stop redefining \fullstop and to use \Xsublinesep
instead.

Appendix A.15   Migration to `reledmac` 2.7.0

\SErefonlypage (introduced in `reledmac` 2.5.0) added an parenthesis after the page
number. This was just an error, linked to a bad imitation of \SErefwithpage. That has
been deleted. And so, the \XendafterpagenumberSErefonlypage to set it was also
deleted.

\rigidbalance is split to two new commands: \Xrigidbalance for critical foot-
notes and \rigidbalanceX for familiar footnotes. If you have redefined it — but why
should you have ?—, you should split your single redefinition in two redefinitions.

Appendix A.16   Migration to `reledmac` 2.7.2

\Xhsize is already defined in the `floatrow` package. It becomes \Xwidth, and, conse-
quently, \hsizeX becomes \widthX.

The ancient names are temporarily maintained as aliases.

Appendix A.17   Migration to `reledmac` 2.8.0

Reledmac 2.8.0 fix spurious indents for paragraphed critical and familiar footnotes
in `ledgroup` and minipage. You can re-establish the indent with \Xparinden and
\parindentX.

Appendix A.18   Migration to `reledmac` 2.13.1

Reledmac 2.5.0 added a bug, which makes the right flag be printed in right side critical
footnotes, even if not explicitly asked by using \Xlineflag.

The version 2.13.1 solves this issue. Please use `\Xlineflag` if you want to add the right flag.

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

H

I

J

K

L

M

Change History

v1.12.0.

v1.24.0.

General: We can reinitialize \AtEveryPstart and \AtEveryPend providing to it an
empty argument. 1

v1.24.1.

General: \lemma is disabled when using 'nocritical' option. 1

v1.24.2.

General: Fix incompatibility between 'nofamiliar' option and 'memoir' package. 1

v1.24.3.

General: Restore marginal numbers and notes with sectioning command (bug
introduced in v1.21.0) 1

v1.24.4.

General: Fix spurious space with \edindex when using xindy+hyperref option. 1

v1.24.5.

General: Fix bug of indent, when a added in 1.1.0, when a \beginnumbering
immediately follow a sectioning command. 1

v2.0.0.

\@iiiminipage: Patch \@iiiminipage instead of redefining it. 265

\@xympar: Patching \@xympar instead of redefining it 256

\endminipage: Patch \endminipage instead of redefining it. 265

\initnumbering@quote: \initnumbering@sectcmd becomes
\initnumbering@quote 315

\l@advance@parledgroup@beforenormalnotes: Some conde of \l@dumboxmpfoot
moved to \l@advance@parledegroup@beforenormalnotes 266

\newseries@: One endnotes file by series. 224

General: \@makecol, \@reinserts and \@doclearpage are patched instead of begin
redefined 240

\doxtrafeeti becomes \do@feetX; \doxtrafeetii becomes \Xdo@feet;
\@opxtrafeeti becomes \@opfeetX; \doreinxtrafeetii becomes
\X@doreinfeet; \doreinxtrafeeti becomes \@doreinfeetX. 240

Add \Xendinplaceofnumber hook. 1

Add \Xendnonumber hook. 1

Add nonum option for endnotes. 1

Fix bug when printing only one series of endnotes, but wanted to keep endnotes for
other series. 1

In order to have a more consistent name's convention, many names has been changed. 1

Many LaTeX's output macros are now patched and not override. 1

Package's name becomes reledmac. 1

Patch \@footnotemark instead of redefine it 184

Suppress indexing command specific to memoir. 271

v2.0.1.

General: Fix bug in eledmac-compat option 1

Fix incompatibility between optional argument of \pstart and \numberpstarttrue 1

v2.1.0.

General: Fix bug with \advanceline at the beginning of a \pstart. 1

Fix bug with \chapter in optional argument of \pstart in parallel typesetting with
scrbook. 1

Fix bug with \eledchapter in parallel typesetting with scrbook. 1

Fix bug with \setline at the beginning of a \pstart. 1

Fix spacing bug with \Xbhooknote and \bhooknoteX when using them to insert text
and not to execute code. 1