reledmac
Typeset scholarly editions with LaTeX[*]

Maïeul Rouquette[†]

based on the original ledmac by

Peter Wilson

Herries Press

which was based on the original edmac, tabmac and edstanza by

John Lavagnino, Dominik Wujastyk, Herbert Breger and Wayne Sullivan.

Abstract

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

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

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

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

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

Contents

[*]This file (`reledmac.dtx`) has version number v2.0.0, last revised 2015/06/14.

[†]`maieul at maieul dot net`

1 Introduction

1.1 Aim of the package

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

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

Eledmac allows the scholar engaged in preparing a critical edition to focus attention wholly on the task of creating the critical text and evaluating the variant readings, text-critical notes and testimonia. LaTeX and Eledmac will take care of the formatting and visual correlation of all the disparate types of information.

Apart from reledmac there are some other LaTeX packages for critical edition typesetting. However, the am of reledmac is to provide a “all in one” and flexible tool in the field of critical edition.

Any suggestion for new features are welcome.

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

We do not suggest that you need to read the source code for this package in order to use it; we provide this code primarily for reference, and many of our comments on it repeat material that is also found in the earlier sections.

But no documentation, however thorough, can cover every question that comes up, and many can be answered quickly by consultation of the code. On a first reading, we suggest that you should read only the general documentation in sections 2, unless you are particularly interested in the innards of reledmac.

1.2 History

1.2.1 edmac

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

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

The present version was developed in the summer of 1990, with the intent of adding necessary features, streamlining and documenting the code, and further generalizing it to make it easily adaptable to the needs of editors in different disciplines. John did most of the general reworking and documentation, with the financial assistance of the Division of the Humanities and Social Sciences, California Institute of Technology. Dominik adapted the code to the conventions of Frank Mittelbach’s doc option, and added some documentation, multiple-column footnotes, cross-references, and crop marks.[1] A description by John and Dominik of this version of edmac was published as ‘An overview of edmac: a Plain TeX format for critical editions’, *TUGboat 11* (1990), pp. 623–643.

From 1991 through 1994, the macros continued to evolve, and were tested at a number of sites. We are very grateful to all the members of the (now defunct) `edmac@mailbase.ac.uk` discussion group who helped us with smoothing out bugs

[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).

and infelicities in the macros. Ron Whitney and our anonymous reviewer at the TUG were both of great help in ironing out last-minute wrinkles, while Ron made some important suggestions which may help to make future versions of edmac even more efficient. Wayne Sullivan, in particular, provided several important fixes and contributions, including adapting the Mittelbach/Schöpf 'New Font Selection Scheme' for use with PLAIN TEX and edmac. Another project Wayne has worked on is a DVI post-processor which works with an edmac that has been slightly modified to output `\specials`. This combination enables you to recover to some extent the text of each line, as ASCII code, facilitating the creation of concordances, an *index verborum*, etc.

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

1.2.2 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

[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.

February 2004. Sidenotes and regular footnotes in numbered text were added in April 2004.

This port was called ledmac (LaTeX edmac).

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

1.2.3 eledmac

Important changes were put in version 1.0, to make ledmac more easily extensible (see 6 p. 20). These changes can trigger small problems with the old customization.

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

To migrate from ledmac to eledmac, please read Appendix Appendix A.2 (p. 245).

1.2.4 reledmac

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

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

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

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

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

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

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

2 How the package works

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

A file may mix *numbered* and *unnumbered* text.

Numbered text is printed with marginal line numbers and can include footnotes and endnotes that are referenced to those line numbers: this is how you will want to print the text that you are editing.

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

3 Options

The package can be loaded with a number of global options which are listed here. It is advised to read the relevant parts of the handbook before reading this section.

draft underlines lemmas in the main text.

eledmac-compat help to migrate from eledmac to reledmac (see Appendix A.9.4 p. 249).

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

noeledsec disables tools for `\eledsection` and related commands (14.2 p. 43).

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

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

noledgroup reledmac allows to use of series of critical notes and new series of normal notes inside minipage and ledgroup environments (see 9 p. 33). 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.

nopbinverse prevents page break inside verses.

noquotation by default, the quotation environment is redefined inside numbered text. You can disable this redefinition with `noquotation` (see 15 p. 44).

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

series reledmac defines six levels of notes: A, B, C, D, E, Z. Using all these levels consumes memory space and processing speed. This is why, if your work does not require all of the A-E, Z 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.

xindy and `xindy+hyperref` are for selecting `xindy` as the index processor (12.3 p. 39).

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

4 Text lines and paragraphs numbering

4.1 Text lines numbering

`\beginnumbering` `\endnumbering`

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

```
\beginnumbering
text
\endnumbering
```

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

If the line numbering of a text is to be continuous from start to end, then the whole text will be typed between one pair of `\beginnumbering` and `\endnumbering` commands. But your text will most often contain chapter or other divisions marking sections that should be independently numbered, and these will be appropriate places to begin new numbered sections.

reledmac has to read and store in memory a certain amount of information about the entire section when it encounters a `\beginnumbering` command, so it speeds up the processing and reduces memory use when a text is divided into a larger number of sections (at the expense of multiplying the number of external files that are generated).

4.2 Paragraphs

4.2.1 Basis

`\pstart` `\pend`

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

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

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

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

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

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

The lines of this paragraph are
not numbered.

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

4.2.2 Producting automatically `\pstart`...`\pend`

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

```
\begingroup
  \beginnumbering
  \autopar

  A paragraph of numbered text.

  Another paragraph of numbered
  text.

  \endnumbering
\endgroup
```

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

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

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

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

brackets, or to add brackets after a `\pend`, just add a `\relax` between `\pstart`/`\pend` and the brackets.

. This feature is also useful when typesetting verses (see 8 p. 30) or eledpar (see 17.3 p. 47).

A `\noindent` is automatically added before this argument.

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

`\AtEveryPstart`
`\AtEveryPend`

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

4.2.5 Numbering paragraphs (`\pstart`)

`\numberpstarttrue`
`\numberpstartfalse`
`\thepstart`

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

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

On each `\beginnumbering` the numbering restarts.

`\sidepstartnumtrue`

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

`\labelpstarttrue`

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

4.2.6 Languages written in Right to Left

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

4.2.7 Memory limits

`\pausenumbering`
`\resumenumbering`

This paragraph is kept for history, but problem described below should not appear with recent version of LaTeX. reledmac stores a lot of information about line numbers and footnotes in memory as it goes through a numbered section. But at the end of such a section, it empties its memory out, so to speak. If your text has a very long numbered section it is possible that your LaTeX may reach its memory limit. There are two solutions to this. The first is to get a larger LaTeX with increased memory.

The second solution is to split your long section into several smaller ones. The trouble with this is that your line numbering will start again at zero with each new section. To avoid this problem, we provide `\pausenumbering` and `\resumenumbering` which are just like `\endnumbering` ... `\beginnumbering`, except that they arrange for your line numbering to continue across the break. Use `\pausenumbering` only between numbered paragraphs:

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

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

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

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

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

4.3 Lineation commands

4.3.1 Disabling lineation

`\numberlinefalse` `\numberlinetrue`

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

4.3.2 Setting lineation start and step

`\firstlinenum` `\linenumincrement`

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

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

`\linenumberlist`

You can define `\linenumberlist` to specify a non-uniform distribution of printed line numbers. For example:
`\def\linenumberlist{1,2,3,5,7,11,13,17,19,23,29}`
to have numbers printed on prime-numbered lines only. There must be no spaces within the definition which consists of comma-separated decimal numbers. The numbers can be in any order but it is easier to read if you put them in numerical order. Either omitting the definition of `\linenumberlist` or following the vacuous definition
`\def\linenumberlist{}`
the standard numbering sequence is applied. The standard sequence is that specified by

the combination of the `firstlinenum`, `linenumincrement`, `firstsublinenum` and `linenumincrement` counter values.

4.3.3 Setting lineation reset

`\lineation`

Lines can be numbered either by page, by pstart or by section; you specify this using the `\lineation{`⟨*arg*⟩`}` macro, where ⟨*arg*⟩ is either `page`, `pstart` or `section`.

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

4.3.4 Setting line number margin

`\linenummargin`

The command `\linenummargin`⟨*location*⟩ specifies the margin where the line (or pstart) numbers will be printed. The permissable value for ⟨*location*⟩ is one out of the list `left`, `right`, `inner`, or `outer`, for example `\linenummargin{inner}`. The package's default setting is
`\linenummargin{left}`
to typeset the numbers in the left hand margin. You can change this whenever you're not in the middle of making a paragraph.

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

4.3.5 Other settings

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

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

4.4 Changing the line numbers

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

4.4.1 Sublineation

`\startsub`
`\endsub`

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

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

4.4.2 Locking lineation

`\startlock` `\endlock`

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

`\lockdisp`

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

4.4.3 Setting and changing line number

`\setline` `\advanceline`

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

`\setlinenum`

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

4.4.4 Line number style

`\linenumberstyle` `\sublinenumberstyle`

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

`Alph`	Uppercase letters (A... Z).
`alph`	Lowercase letters (a... z).
`arabic`	Arabic numerals (1, 2, ...)

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

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

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

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

4.4.5 Skipping and hidding number

`\skipnumbering` When inserted into a numbered line the macro `\skipnumbering` causes the numbering of that particular line to be skipped; that is, the line number is unchanged and no line number will be printed. Note that if you use it in `\stanza`, you must call it at the beginning of the verse.

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

4.4.6 Execute code at each line

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

5 Apparatus commands

5.1 Terminology

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

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

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

So :

- `\Afootnote` is a critical footnote of the series A.
- `\Bendnote` is a critical footnote of the series B.
- `\footnoteC` is a critical footnote of the series C.

5.2 Critical notes

5.2.1 The lemma

`\edtext`

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

`\edtext{`⟨*lemma*⟩`}{`⟨*commands*⟩`}`

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

For example:

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

I am happy : I saw my friend Smith on 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.}
}
```

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

1 Smith] Jones C, D.

1–2 I saw my friend Smith on Tuesday.] The date was July 16, 1954.

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

5.2.2 Footnotes

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

`\Afootnote`
`\Bfootnote`
`\Cfootnote`
`\Dfootnote`
`\Efootnote`
`\Zfootnote`

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

If you need more series of critical notes, please look at 5.5.1 p. 19.

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

- `fulllines` to disable `\Xtwolines` and `\Xmorethantwolines` features for this note (cf. 6.2.2 p. 22).

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

Example: \Afootnote[nonum]{⟨*text*⟩}.

5.2.3 Endnotes

\Aendnote \Bendnote \Cendnote \Dendnote \Eendnote \Zendnote

The package also maintains six separate series of endnotes.

If you do not need the endnotes facility, you should use noend option when loading eledmac.

The mechanism is similar to the one for footnotes: each macro takes one or more optional arguments and one single argument, like:
\Aendnote[⟨*option*⟩]{⟨*text*⟩}.

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

- fulllines to disable \Xendtwolines and \Xendmorethantwolines features for this particular note (cf. 6.2.2 p. 22).
- nosep to disable the lemma separator for this particular note.

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

\doendnotesbysection

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

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

Note that by default inside endnotes no separator is used between the lemma and the content. However you can use the \Xendlemmaseparator macro to define one (6.3.2 p. 25).

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

5.2.4 Paragraph in critical apparatus

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

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

5.2.5 Change lemma and line number

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

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

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

1 Smith] Jones C, D.

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

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

This command does not change the marginal line numbers in any way; it just changes the numbers passed to the footnotes. Its use comes in situations that `\edtext` has trouble dealing with for whatever reason. If you need notes for overlapping passages that aren't nested, for instance, you can use `\lemma` and `\linenum` to generate such notes despite the limitations of `\edtext`. If the ⟨*lemma*⟩ argument to `\edtext` is extremely long, you may run out of memory; here again you can specify a note with an abbreviated lemma using `\lemma` and `\linenum`. The numbers used in `\linenum` need not be entered manually; you can use the '`x-`' symbolic cross-referencing commands below (10 p. 34) to compute them automatically.

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

5.2.6 Changing the names of commands for critical apparatus

The commands for generating the apparatus have been given rather bland names, because editors in different fields have widely divergent notions of what sort of notes are required, where they should be printed, and what they should be called. But this does not mean you have to type `\Afootnote` when you would rather say something you find more meaningful, like `\variant`.

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

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

5.3 Disambiguation of identical words in the apparatus

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

5.3.1 Basic use

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

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

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

5.3.2 Notes about input encoding with UTF-8 processor

If you use UTF-8 processor, like XƎLaTeX 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 YPOGEGRAMMENI (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).

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

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

- If you use XeLaTeX, add this line in your preamble: `\XeTeXinputnormalization 1`.
- If you use LuaLaTeX, use the uninormalize package of Michal Hoftich[15] with the `buffer` option set to true.

With these tools, XeTeX / LuaTeX will dynamicaly normalize unicode input when reading the file. Consequently, you will have no problems with the `\sameword` command.

5.3.3 Use with `\lemma` command

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

For example in the following example:

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

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

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

- In the content of `\lemma`, use `\sameword` with no optional argument.
- In the first argument of `\edtext`, use `\sameword` with the optional argument `[`$\langle X \rangle$`]`. $\langle X \rangle$ is the depth of the `\edtext` where the `\lemma` is used. So if the `\lemma` is called in a `\edtext` inside another `\edtext`, $\langle X \rangle$ is equal to 2. If the `\lemma` is called in a `\edtext` “of first level”, $\langle X \rangle$ is equal to `1`. If the lemma is called in both 1 and 2 `\edtext` depth, $\langle X \rangle$ is `1,2`. If that word is referenced in the lemma of every `\edtext` depth, $\langle X \rangle$ can also be set to `inlemma`.

Note that only words that are actually referenced in a `\lemma` need the optional argument. Therefore, the first `\sameword` in the example above should have “1” as its optional argument, to be referenced correctly in the lemma.

Note also that the $\langle X \rangle$ does not refer to the level where the `\sameword` occurs, but to the level of the `\lemma` that refers to that `\sameword`. For example:

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

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

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

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

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

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

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

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

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

5.3.4 Customizing

`\showwordrank`

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

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

5.4 Familiar notes

5.4.1 Basic use

`\footnoteA`
`\footnoteB`
`\footnoteC`
`\footnoteD`
`\footnoteE`
`\footnoteZ`

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

5.4.2 Customizing mark

\thefootnoteA
\bodyfootmarkA
\footfootmarkA

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

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

The command `\footfootmarkA` controls the appearance of the mark at the start of the footnote text. It is defined as:
`\newcommand*{\footfootmarkA}{\textsuperscript{\@nameuse{@thefnmarkA}}}`

There are similar command triples for the other series.

5.4.3 Separator for multiple footnotes

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

\multfootsep

`\multfootsep` is used as the separator between footnote markers. Its default definition is:
`\providecommand*{\multfootsep}{\textsuperscript{\normalfont,}}`
and can be changed if necessary.

5.5 Changing series

5.5.1 Create a new series

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

5.5.2 Delete series

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

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

5.5.3 Series order

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

seriesatbegin
seriesatend

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

5.6 Position of critical and familiar footnotes

\fnpos
\mpfnpos

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

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

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

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

6 Critical apparatus appearance

Some commands can be used to change the display of the footnotes. All can have an optional argument `[`$\langle s\rangle$`]`, 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 concern all 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 unity[16], it will be relative to fontsize of the footnote, except for commands concerning the place kept by the notes — including blank space.

There is also name convention:

- Names prefixed by `X` are for setting of critical footnotes.
- Names prefixed by `Xend` are for setting of critical endnotes.
- Names suffixed by `X` are for setting of familiar footnotes.

6.1 Notes arrangement in a series

\Xarrangement
\arrangementX

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

Use `\Xarrangement[`$\langle s\rangle$`]{`$\langle a\rangle$`}` to change the arrangement of the $\langle s\rangle$ series of critical footnotes and `\arrangementX[`$\langle s\rangle$`]{`$\langle a\rangle$`}` 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;
- `twocol` formats them as separate paragraphs, but in two columns;
- `threecol`, in three columns.

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

- `normal`, restore normal arrangement.

You should set up the page layout parameters, and in particular the `\baselineskip` of the footnotes, before you call this macro because its action depends on these; too much or too little space will be allotted for the notes on the page if these macros use the wrong values.[17]

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

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

6.2 Control line number printing

6.2.1 Print line number only at first time

`\Xnumberonlyfirstinline`

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

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

`numberonlyfirstinXtwolines`

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 `\XnumberonlyfirstinXtwolines[`⟨*s*⟩`]`, the distinction is made. Use `\XnumberonlyfirstinXtwolines[`⟨*s*⟩`][false`

`\Xsymlinenum`

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

6.2.2 Abbreviate line range

`\Xtwolines`
`\Xmorethantwolines`

If a lemma is printed on two subsequent lines, Eledmac 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:

[17] There is one tiny proviso about using paragraphed notes: you should not force any explicit line-breaks inside such notes: do not use `\par`, `\break`, or `\penalty=-10000`. If you must have a line-break for some obscure reason, just suggest the break very strongly: `\penalty=-9999` will do the trick. XII.6.3 p. 126 explains why this restriction is necessary.

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

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

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

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

It is possible to disable `\Xtwolinesbutnotmore[`⟨*series*⟩`]` with `\Xtwolinesbutnotmore[`⟨*series*⟩`][fa`

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

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

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

You can disable `\Xtwolines` and related for a specific note by using the '[fulllines]' argument in the note macro cf. 5.2.2 p. 14.

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

6.2.3 Disable line number

\Xnonumber You can use `\Xnonumber[`⟨*s*⟩`]` if you do not want to have the line number in a footnote. To cancel it, use `\Xnonumber[`⟨*s*⟩`][false]`.

6.2.4 Printing pstart number

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

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

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

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

6.2.5 Space around number

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

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

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

6.2.6 Space around line symbol

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

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

6.2.7 Space in place of number

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

6.2.8 Boxing line number and line symbol

`\Xboxlinenum` It could be useful to put the line number inside a fixed box: the content of the note will be printed after this box. You can use `\Xboxlinenum[`⟨*s*⟩`]{`⟨*l*⟩`}` to do that. To subsequently disable this feature, use `\Xboxlinenum` with length equal to 0 pt. One use of this feature is to print line number in a column, and the note in an other column:

```
\Xhangindent{1em}
\Xafternumber{0em}
\Xboxlinenum{1em}
```

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

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

L to align left (default value);

R to align right;

C to center.

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

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

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

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

`\Xendboxlinenum`
`\Xendboxlinenumalign`
`\Xendboxstartlinenumalign`
`\Xendboxendlinenumalign`

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

6.3 Separator between the lemma and the note

6.3.1 For footnotes

`\Xlemmaseparator`

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

`\Xbeforelemmaseparator`

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

`\Xafterlemmaseparator`

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

`\noXlemmaseparator`

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

`\Xinplaceoflemmaseparator`

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

6.3.2 For endnotes

`\Xendlemmaseparator`

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

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

`\Xendbeforelemmaseparator`

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

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

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

6.4 Font style

6.4.1 For line number

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

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

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

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

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

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

6.4.2 For the lemma

lemmadisablefontselection 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.

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

6.5 Styles of notes content

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

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

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

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

6.6 Arbitrary code at the beginning of notes

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

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

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

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

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

6.7 Options for footnotes in columns

6.7.1 Alignment

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

`<code>` must be one of the following command:

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

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

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

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

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

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

`\Centering` to have text centered *with hyphenation*.

6.7.2 Size of the columns

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

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

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

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

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

6.8 Options for paragraphed footnotes

6.8.1 Mark separation of notes

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

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

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

6.8.2 Ragging

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

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

6.9 Options for block of notes

6.9.1 Text before notes

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

6.9.2 Spacing

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

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

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

`\prenotesX` You can want the space before the first printed (in a page) series of familiar notes not to be the same as before other series. Default value is `0pt`. You can do it with `\prenotesX{`⟨*l*⟩`}`. You can disable this feature by setting the length to 0 pt.

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

`\prenotesX` You can want the space before the first printed (in a page) series of familiar notes not to be the same as before other series. Default value is `0pt`. You can do it with `\prenotesX{`⟨*l*⟩`}`. You can disable this feature by setting the length to 0 pt.

6.9.3 Rule

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

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

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

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

6.9.4 Maximum height

`\Xmaxhnotes` By default, one series of critical notes can take 80% of the page size, before being broken to the next page. If you want to change the size use `\Xmaxhnotes[`⟨*s*⟩`]{`⟨*l*⟩`}`. Be 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.

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

6.10 Footnotes and the reledpar columns

`Xnoteswidthliketwocolumns`
`noteswidthliketwocolumnsX`

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

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

6.11 Endnotes in one paragraph

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

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

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

7 Fonts

One of the most important features of the appearance of the notes, and indeed of your whole document, will be the fonts used. We will first describe the commands that give you control over the use of fonts in the different structural elements of the document, 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.

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

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

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

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

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

8 Verse

8.1 Basic

`\stanza` `\&`

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

8.2 Define stanza indents

`\stanzaindentbase`

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

`\setstanzaindents`

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

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

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

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

If you want the hanging verse to be flush right, you can use `\sethanginsymbol`: see p. 8.6 p. 32.

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

8.3 Repeating stanza indents

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

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

is like

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

Be careful: the feature is changed in eledmac 1.5.1. See Appendix A.3 p. 245.

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

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

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

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

8.4 Manual stanza indent

`\stanzaindent`
`\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`.

8.5 Stanza breaking

`\setstanzapenalties`

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

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

The first entry "1" is a control value. If it is zero, then no penalties are passed on to TeX, which is the default. Values between 0 and 10000 are penalty values; values between 10001 and 20000 have 10000 subtracted and the result is given as a negative penalty. The mechanism used for indentations and penalties requires unsigned values less than 32768. No penalty is placed after the last line, so the final `,0` in then example above could be omitted. A penalty of 10000 will prevent a page break; such a penalty is included automatically where there is stanza hanging indentation. A penalty of -10000 (corresponding to the entry value 20000 in this context) forces a page break. Values in between act as suggestions as to the desirability of a page break at a given line. There is a subtle interaction between penalties and *glue*, so it may take some adjustment of skips and penalties to achieve the best results.

8.6 Hanging symbol

`\sethangingsymbol`

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

```
\sethangingsymbol{[\,}
```

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

```
\sethangingsymbol{\protect\hfill}
```

8.7 Long verse and page break

If you want to prevent page breaks inside long verses, use the option `nopbinverse` when loading package, or use `\lednopbinversetrue`. Read 16.2 p. 45 for further details.

8.8 Hanging symbol

`\hangingsymbol`

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

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

8.9 Content before/after verses

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

- `\stanza` command can take a optional argument (in brackets). Its content will be printed before the stanza.
- `&` can be replaced by `\newverse` with two optional arguments (in brackets). The first will be printed after the current verse, the second before the next verse.
- `\&` can take a optional argument (in brackets). Its content will be printed after the stanza.

8.10 Various tools

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

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

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

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

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

9 Grouping

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

ledgroup Minipages, of course, are not broken across pages. Footnotes in a `ledgroup` 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}`

10 Cross referencing

The package provides a simple cross-referencing facility that allows you to mark places in the text with labels, and generate page and line number references to those places elsewhere using those labels.

10.1 Basic use

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

`\edpageref` `\edlineref` `\sublineref` `\pstartref` 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.

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 `\edlabel`s in the text.

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

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

10.2 Refer to a critical notes

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

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

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

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

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

10.3 Cross-referencing which return a number in any case

\xpageref
\xlineref
\xsublineref
\xpstartref

Where #1 stands for the reference.

However, there are situations in which you will want reledmac to return a number without displaying any warning messages about undefined labels or the like: if you want to use the reference in a context where LaTeX is looking for a number, such a warning will lead to a complaint that the number is missing. This is the case for references used within the argument to `\linenum`, for example (see 5.2.5 p. 15).

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

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

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

\xxref

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

The `\xxref{`⟨*lab1*⟩`}{`⟨*lab2*⟩`}` command generates a reference to a sequence of lines, for use in the second argument of `\edtext`. It takes two arguments, both of which are labels: e.g., `\xxref{mouse}{elephant}`. It calls `\linenum` (q.v., 5.2.5 p. 15 above) and sets the beginning page, line and subline numbers to those of the place where `\edlabel{mouse}` was placed, and the ending numbers to those where `\edlabel{elephant}` occurs.

10.4 Not automatic cross-referencing

\edmakelabel

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

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

10.5 Normal LaTeX cross-referencing

\label
\ref
\pageref

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

10.6 References to lines commented in the apparatus

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

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

If you use `\linenum` (5.2.5 p. 15) to refer to these labels, reledmac will use your line settings to refer to the passage.

`\appref` `\apprefwithpage` You can also use `\appref{`⟨*label*⟩`}` and `\apprefwithpage{`⟨*label*⟩`}` to refer to these 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.

`\apprefprefixsingle` `\apprefprefixmore` If you redefine `\apprefprefixsingle`, its content will be printed before the line numbers of a `\appref`-reference. If you redefine `\apprefprefixmore`, its content will be printed before the line numbers, if you refer to more than one line.

For example, you may use:

```
\renewcommand{\apprefprefixsingle}{line~}
\renewcommand{\apprefprefixmore}{lines~}
```

Note that if `\apprefprefixmore` is empty, `\apprefprefixsingle` will used in any case.

`\Xtwolinesappref` `\Xmorethantwolinesappref` `\Xtwolinesbutnotmoreappref` `\Xtwolinesonlyinsamepage` If you use `\Xtwolines`, `\Xmorethantwolines`, `\Xtwolinesbutnotmore` and/or `\Xtwolinesonlyinsamepage` (6.2.2 p. 22) *without the optional series argument*, the setting will also be available for `\appref`.

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

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

`\Xendtwolinesapprefwithpage` `\Xendmorethantwolinesapprefwithpage` `\Xendtwolinesbutnotmoreapprefwithpage` `\Xendtwolinesonlyinsamepageapprefwithpage` If you use one of `\Xendtwolines`, `\Xendmorethantwolines`, `\Xendtwolinesbutnotmore`, `\Xendtwolinesonlyinsamepage`) (6.2.2 p. 22) *without the optional series argument*, the setting will also be available for `\apprefwithpage`.

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

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

11 Side notes

11.1 Basis

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

\ledinnernote
\ledouternote

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

\ledleftnote
\ledrightnote
\ledsidenote
\sidenotemargin

\ledsidenote{⟨*text*⟩} will put ⟨*text*⟩ into the margin specified by the current setting of \sidenotemargin{⟨*location*⟩}. The permissible value for ⟨*location*⟩ is one out of the list left, right, inner, or outer, for example \sidenotemargin{outer}. The package's default setting is
\sidenotemargin{right}
to typeset \ledsidenotes in the right hand margin. This is the opposite to the default margin for line numbers. The style for a \ledsidenote follows that for a \ledleftnote or a \ledrightnote depending on the margin it is put in.

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

11.2 Setting

11.2.1 Width

\ledlsnotewidth
\ledrsnotewidth

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

11.2.2 Vertical position

\rightnoteupfalse
\leftnoteupfalse

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

11.2.3 Distance to the main text

\ledlsnotesep
\ledrsnotesep
\ledlsnotefontsetup
\ledrsnotefontsetup

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

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

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

These can of course be changed to suit.

11.2.4 Separator between notes

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

12 Indexing

12.1 Basis

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

Note that the file `.idx` will contain the right reference only after the third run, because of the internal indexing mechanism of eledmac. That means you must first run three times (Xe/Lua)LaTeX, then run `makeindex` and finally run again (Xe/Lua)LaTeXto get an index with the right page numbers.

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

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

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

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

12.2 Separator between page and line numbers

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

`-` is the default separator used by the MakeIndex program.

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

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

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

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

12.3 Using xindy

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

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

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

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

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

1. Use `xindy+hyperref` option when loading the reledmac package. When you run (Xe/Lua)LaTeXwith this option, a `.xdy` configuration file will be generated with all the settings needed to allow internal hyperlinking in each index entry which is created by `\edindex`.
2. Use `hyperindex=false` option when loading hyperref.
3. Uncomment — by removing the semicolons at the beginning of the relevant lines — some lines in the <code>.xdy</code> file provided in the "examples" folder in order to restore internal links in the index to be used by the standard `index` command.[21].

12.4 Advanced setting

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

13 Tabular material

LaTeX's normal `tabular` and `array` environments cannot be used where line numbering is being done; more precisely, they can be used but with odd results, so don't use

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

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

them. However, reledmac provides some simple tabulation environments that can be line numbered. The environments can also be used in normal unnumbered text.

`edarrayl`
`edarrayc`
`edarrayr`
`edtabularl`
`edtabularc`
`edtabularr`

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

```
\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`
`\spreadmath`
`\spreadtext`

The distance between the columns is controlled by the length `\edtabcolsep`.

`\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}
```

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

`\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}
```

$$\begin{array}{rrrrr} \text{1} & \text{2} & \text{3} & \text{4} & \text{5} \\ \text{Q} & & \text{fd} & \text{h} & \text{qwertziohg} \\ \text{v} & \text{wptz} & \text{x} & \text{y} & \text{vb} \\ \text{g} & \text{nnn} & \multicolumn{3}{c}{\upbracefill} \\ \multicolumn{3}{c}{\downbracefill} & \text{pq} & \text{dgh} \\ \text{k} & & \text{l} & \text{co} & \text{ghweropjklmnbvcxys} \\ \text{1} & \text{2} & \text{3} & \multicolumn{2}{c}{\hrulefill} \end{array}$$

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}
```

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

`\edatleft` `\edatright` `\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` `\edaftertab` `\edbeforetab{⟨text⟩}{⟨entry⟩}`, where ⟨*entry*⟩ is an entry in the leftmost column, typesets ⟨*text*⟩ left justified before the ⟨*entry*⟩. Similarly `\edaftertab{⟨entry⟩}{⟨text⟩}`, where ⟨*entry*⟩ is an entry in the rightmost column, typesets ⟨*text*⟩ right justified after the ⟨*entry*⟩.

For example:

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

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

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

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

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

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

14 Sectioning commands

14.1 Sectioning commands without line numbers or critical notes

The standard sectioning commands (`\chapter`, `\section` etc.) can be used inside numbered text. In this case, you must call them as an optional argument of `\pstart` (4.2.3 p. 8):

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

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

14.2 Sectioning commands with line numbering and critical notes

You have to use the following commands:

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

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

```
\pstart
\eledsection*{xxxx\ledsidenote{section}}
```

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

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

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

14.3 Optimization

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

15 Quotation environments

The `quotation` and `quote` environment can be used so that same definition/note appears both inside and outside a numbered section. The typographical consequences will resemble the outside numbered sections, based on the styles of the *book* class. However, if you use a package that redefines these environments, these redefinitions won't be available inside the numbered section. You must open any quotation environments inside a `\start`-`\pend` block, not outside. A quotation environment MUST not be opened immediately after a `\pstart` and MUST not be closed immediately before a `\pend`.

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

16 Page breaks

16.1 Control page breaking

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

- `\ledpb` adds a page break.
- `\lednopb` prevents a page break, by adding one line to the current page if needed.

These commands have effect only at the second run.

These two commands take effect at the beginning of line in which they are called. For example, if you call `\ledpb` at l. 444, the l. 443 will be at the p. n, and the l. 444 at the p. $n + 1$. However you can change the behavior, and decide they will have effect after the end of the line, adding `\ledpbsetting{after}` at the beginning of your file (better: in your preamble). With the previous example, the l. 444 will be at the p. n and the l. 445 will be at the p. $n + 1$.

`\ledpbsetting`

If you are using reledpar to typeset parallel pages you must use `\lednopb` on both sides in the two corresponding lines. This is especially important when you are using stanzas; otherwise the pages will run out of sync.

16.2 Prevent page break in a long verses

`\lednopbinversetrue`

You can also decide to prevent page breaks between two lines of a long verse. To do this, use `nopbinverse` when loading package, or add `\lednopbinversetrue` in the beginning of your file (better: in your preamble).

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

17 Miscellaneous

`\extensionchars`

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

`\ifledfinal`

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

`\showlemma`

The lemma within the text is printed via `\showlemma{lemma}`. Normally, or with the 'final' option, the definition of `\showlemma` is:
`\newcommand*{\showlemma}[1]{#1}`
so it just produces its argument. With the 'draft' option it is defined as
`\newcommand*{\showlemma}[1]{\textit{#1}}`
so that its argument is typeset in an italic font, which may make it easier to check that all lemmas have been treated.

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

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

17.1 Known and suspected limitations

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.

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

`\ballast` LaTeX is a three-pass system, but even after a document has been processed three times, there are some tricky situations in which the page breaks decided by TeX never settle down. At each successive run, reledmac may oscillate between two different sets of page decisions. To stop this happening, should it arise, Wayne Sullivan suggested the inclusion of the quantity `\ballast`. The amount of `\ballast` will be subtracted from the penalties which apply to the page breaks calculated on the *previous* run through TeX, thus reinforcing these breaks. So if you find your page breaks oscillating, say
`\setcounter{ballast}{100}`
or some such figure, and with any luck the page breaks will settle down. Luckily, this problem does not crop up at all often.

The restriction on explicit line-breaking in paragraphed footnotes, mentioned in a footnote 17 p. 21, and described in more detail on XII.6.3 p. 125, really is a nuisance if that is something you need to do. There are some possible solutions, described by Michael Downes, but this area remains unsatisfactory.

`\footfudgefiddle` For paragraphed footnotes TeX has to estimate the amount of space required. If it underestimates this then the notes may get too long and run off the bottom of the text block. `\footfudgefiddle` can be increased from its default 64 (say to 68) to increase the estimate. You have to use `\renewcommand` for this, like:
`\renewcommand{\footfudgefiddle}{68}`

17.2 Use with other packages

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

`\morenoexpands` You can define the macro `\morenoexpands` to modify macros that you call within `\edtext`. Because of the way reledmac numbers the lines the arguments to `\edtext` can be processed more than once and in some cases a macro should only be processed once. One example is the `\colorbox` macro from the color package, which you might use like this:

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

If you actually try this[22] you will find LaTeX whinging '`Missing { inserted`', and then

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

things start to fall apart. The trick in this case is to specify either:

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

or

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

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

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

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

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

It took Peter Wilson a little while to discover all this. If you run into this sort of problem you may have to spend some time experimenting before hitting on a solution.

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

17.3 Parallel typesetting

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

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

I Implementation overview

We present the reledmac code in roughly the order in which it is used during a run of TeX. The order is *exactly* that in which it is read when you load The Eledmac package, because the same file is used to generate this manual and to generate the LaTeX package file.

Most of what follows consists of macro definitions, but there are some commands that are executed immediately—especially at the start of the code. The documentation generally describes the code from the point of view of what happens when the macros are executed, though. As each macro is introduced, its name is printed in the margin.

After package options, we begin with the commands you use to start and stop line numbering in a section of text (Section II). Next comes the machinery for writing and reading the auxiliary file for each section that helps us count lines, and for creating list macros encoding the information from that file (Section V); this auxiliary file will be read at the start of each section, to create those list macros, and a new version of the file will be started to collect information from the body of the section.

Next are commands for marking sections of the text for footnotes (Section VI), followed by the macros that take each paragraph apart, attach the line numbers and insertions, and send the result to the vertical list (Section VII). The footnote commands (Section XII) and output routine (Section **??**) finish the main part of the processing; cross-referencing (Section XXIII) and endnotes (Section XIX) complete the story.

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

II Preliminaries

II.1 Links with original edmac

Generally, these are the modifications to the original. edmac code:

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

II.2 Package declaration

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

```
⟨*code⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{reledmac}[2015/06/14 v2.0.0 typesset critical edition]%
```

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.

```
\RequirePackage{xkeyval}
```

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

```
\newif\ifparledgroup
```

And now, the options of reledmac.

```
\DeclareOptionX{series}[A,B,C,D,E,Z]{\xdef\default@series{#1}}
\ExecuteOptionsX{series}%

\newif\if@noeled@sec%
\DeclareOptionX{noeledsec}{\@noeled@sectrue}

\newif\ifnocritical@%
\DeclareOptionX{nocritical}{\nocritical@true}%

\newif\ifnofamiliar@%
\DeclareOptionX{nofamiliar}{\nofamiliar@true}%

\newif\ifnoledgroup@%
\DeclareOptionX{noledgroup}{\noledgroup@true}%

\newif\ifnoend@%
\DeclareOptionX{noend}{%
 \let\l@dend@open\@gobble%
 \let\l@d@end\relax
 \let\l@dend@close\relax%
  \global\let\l@dend@stuff=\relax%
  \global\chardef\l@d@end=16%
  \noend@true%
}%

\newif\ifnoquotation@
\DeclareOptionX{noquotation}{\noquotation@true}


\newif\ifledfinal
\DeclareOptionX{final}{\ledfinaltrue}
\DeclareOptionX{draft}{\ledfinalfalse}
\ExecuteOptionsX{final}

\newif\ifparapparatus@
\DeclareOptionX{parapparatus}{\parapparatus@true}

\newif\iflednopbinverse
\DeclareOptionX{nopbinverse}{\lednopbinversetrue}
```

```

\newif\ifwidthliketwocolumns%
\DeclareOptionX{widthliketwocolumns}{\widthliketwocolumnstrue}%

\newif\ifxindy@
\DeclareOptionX{xindy}[eledmac-markup-attr.xdy]{%
  \AtBeginDocument{\immediate\openout\eledmac@xindy@out=#1}%
  \newwrite\eledmac@xindy@out%
  \xindy@true%
  \gdef\eledmacmarkuplocrefdepth{:depth 1}%
  \AtEndDocument{\immediate\closeout\eledmac@xindy@out}%
}%

\newif\ifxindyhyperref@
\DeclareOptionX{xindy+hyperref}{%
  \xindyhyperref@true%
}%

\newif\ifeledmaccompat@%
\DeclareOptionX{eledmac-compat}{%
  \eledmaccompat@true%
}%
```

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.

```
\ProcessOptionsX*\relax

```

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

```
\RequirePackage{xargs}
\RequirePackage{etoolbox}
\RequirePackage{etex}
\reserveinserts{32}
\RequirePackage{suffix}
\RequirePackage{xstring}
\RequirePackage{ifluatex}
\RequirePackage{ragged2e}
\RequirePackage{ragged2e}
\RequirePackage{ifxetex}%
```

II.5 Boolean flags

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

```
\newif\ifl@dmemoir
\@ifclassloaded{memoir}{\l@dmemoirtrue}{\l@dmemoirfalse}

```

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

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

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

```
\newif\ifl@indextools%
\@ifpackageloaded{indextools}{%
  \l@indextoolstrue%
  \l@imakeidxtrue%
  \let\imki@wrindexentry\indtl@wrindexentry%
  }{}%
```

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.

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

II.6 Messages

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

`\reledmac@warning` Write a warning message.

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

`\reledmac@error` Write an error message.

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

`\led@err@NumberingStarted`
`d@err@NumberingNotStarted`
`umberingShouldHaveStarted`

```
\newcommand*{\led@err@NumberingStarted}{%
  \reledmac@error{Numbering has already been started}{\@ehc}}
\newcommand*{\led@err@NumberingNotStarted}{%
  \reledmac@error{Numbering was not started}{\@ehc}}
\newcommand*{\led@err@NumberingShouldHaveStarted}{%
  \reledmac@error{Numbering should already have been started}{\@ehc}}
```

`d@err@edtextoutsidepstart`

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

\led@mess@NotesChanged

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

\led@mess@SectionContinued

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

\led@err@LineationInNumbered

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

\led@warn@BadLineation
\led@warn@BadLinenummargin
\led@warn@BadLockdisp
\led@warn@BadSublockdisp

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

\led@warn@NoLineFile

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

\led@warn@LineFileObsolete

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

\led@warn@BadAdvancelineSubline
\led@warn@BadAdvancelineLine

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

\led@warn@BadSetline
\led@warn@BadSetlinenum

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

led@err@PstartNotNumbered
\led@err@PstartInPstart
\led@err@PendNotNumbered
\led@err@PendNoPstart
ed@err@AutoparNotNumbered
rr@NumberingWithoutPstart

```
\newcommand*{\led@err@PstartNotNumbered}{%
  \reledmac@error{\string\pstart\space must be used within a
                numbered section}{\@ehc}}
\newcommand*{\led@err@PstartInPstart}{%
  \reledmac@error{\string\pstart\space encountered while another
                \string\pstart\space was in effect}{\@ehc}}
\newcommand*{\led@err@PendNotNumbered}{%
  \reledmac@error{\string\pend\space must be used within a
                numbered section}{\@ehc}}
\newcommand*{\led@err@PendNoPstart}{%
  \reledmac@error{\string\pend\space must follow a \string\pstart}{\@ehc}}
\newcommand*{\led@err@AutoparNotNumbered}{%
  \reledmac@error{\string\autopar\space must be used within a
                numbered section}{\@ehc}}
\newcommand*{\led@err@NumberingWithoutPstart}{%
  \reledmac@error{\string\beginnumbering...\string\endnumbering\space without \string\pstart}{\@ehc}}%
```

\led@warn@BadAction

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

\led@warn@DuplicateLabel
ed@warn@AppLabelOutEdtext
\led@warn@RefUndefined

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

\led@warn@NoMarginpars

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

ed@warn@BadSidenotemargin

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

\led@warn@NoIndexFile

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

led@warn@SeriesStillExist

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

\led@err@ManySidenotes
\led@err@ManyLeftnotes
\led@err@ManyRightnotes

```
\newcommand{\led@err@ManySidenotes}{%
  \ifledRcol@%
   \reledmac@warning{\itemcount@\space sidenotes on line \the\line@numR\space p. \the\page@
  \else%
   \reledmac@warning{\itemcount@\space sidenotes on line \the\line@num\space p. \the\page@n
  \fi%
}%
\newcommand{\led@err@ManyLeftnotes}{%
  \ifledRcol@%
   \reledmac@warning{\itemcount@\space leftnotes on line \the\line@numR\space p. \the\page@n
  \else%
   \reledmac@warning{\itemcount@\space leftnotes on line \the\line@num\space p. \the\page@nu
  \fi%
}%
\newcommand{\led@err@ManyRightnotes}{%
  \ifledRcol@%
   \reledmac@warning{\itemcount@\space rightnotes on line \the\line@numR\space p. \the\page@
  \else%
   \reledmac@warning{\itemcount@\space rightnotes on line \the\line@num\space p. \the\page@n
  \fi%
}%
```

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

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

\led@err@EdtextWithoutFootnote

```
\newcommand{\led@err@EdtextWithoutFootnote}{%
  \reledmac@error{edtext without Xfootnote. Check syntaxis.}{\@ehd}%
}%
```

\led@err@FootnoteWithoutEdtext

```
\newcommand{\led@err@FootnoteWithoutEdtext}{%
  \reledmac@error{Xfootnote without edtext. Check syntax.}{\@ehd}%
}%
```

\led@error@ImakeidxAfterEledmac

```
\newcommand{\led@error@ImakeidxAfterEledmac}{%
  \reledmac@error{Imakeidx must be loaded before eledmac.}{\@ehd}%
}%
```

or@IndextoolsAfterEledmac

```
\newcommand{\led@error@IndextoolsAfterEledmac}{%
  \reledmac@error{Indextools must be loaded before eledmac.}{\@ehd}%
}%
```

error@fail@patch@@makecol

```
\newcommand{\led@error@fail@patch@@makecol}{%
  \reledmac@error{Fail to patch \string\@makecol\space command.}{\@ehd}%
}%
```

ror@fail@patch@@reinserts

```
\newcommand{\led@error@fail@patch@@reinserts}{%
  \reledmac@error{Fail to patch \string\@reinserts\space command.}{\@ehd}%
}%
```

r@fail@patch@@doclearpage

```
\newcommand{\led@error@fail@patch@@doclearpage}{%
  \reledmac@error{Fail to patch \string\@doclearpage\space command.}{\@ehd}%
}%
```

r@fail@patch@@iiiminipage

```
\newcommand{\led@error@fail@patch@@iiiminipage}{%
  \reledmac@error{Fail to patch \string\@iiiminipage\space command.}{\@ehd}%
}%
```

or@fail@patch@endminipage

```
\newcommand{\led@error@fail@patch@endminipage}{%
  \reledmac@error{Fail to patch \string\endminipage\space command.}{\@ehd}%
}%
```

II.7 Gobbling

Here, we define some commands which gobble their arguments.

\@gobblethree
\@gobblefour
\@gobblefive

```
\providecommand*{\@gobblethree}[3]{}
\providecommand*{\@gobblefour}[4]{}
\providecommand*{\@gobblefive}[5]{}
```

II.8 Miscellaneous commands

\showlemma

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

```
\ifledfinal
  \newcommand*{\showlemma}[1]{#1}
\else
  \newcommand*{\showlemma}[1]{\underline{#1}}
\fi

```

`\linenumberlist`

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

Initialize it as `\empty`.

```
\let\linenumberlist=\empty

```

`\@l@dtempcnta` `\@l@dtempcntb`

In imitation of LaTeX, we create a couple of scratch counters.

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).

```
\newcount\@l@dtempcnta \newcount\@l@dtempcntb
```

II.9 Prepare reledpar

`\ifnumberingR` `\ifl@dpairing` `\ifl@dpaging` `\l@dpagingtrue` `\l@dpagingfalse` `\ifl@dprintingpages` `\l@dprintingpagestrue` `\l@dprintingpagesfalse` `\ifl@dprintingcolumns` `\l@dprintingcolumnstrue` `\l@dprintingcolumnsfalse` `\l@dpairingtrue` `\l@dpairingfalse` `\ifpst@rtedL` `\pst@rtedLtrue` `\pst@rtedLfalse` `\l@dnumpstartsL` `\ifledRcol` `\ifledRcol@`

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

```
\newif\ifl@dpairing
\newif\ifl@dpaging%
\newif\ifl@dprintingpages%
\newif\ifl@dprintingcolumns%
\newif\ifpst@rtedL
\newcount\l@dnumpstartsL
```

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

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

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

```
\newif\ifnumberingR
```

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.

```
\newcount\section@num
\section@num=0
\let\extensionchars=\empty
```

`\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.

```
\newif\ifnumbering
```

`\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.

```
\newcommand*{\beginnumbering}{%
  \ifnumbering
    \led@err@NumberingStarted
    \endnumbering
  \fi
  \global\numberingtrue
  \global\advance\section@num \@ne
  \initnumbering@reg
  \message{Section \the\section@num }%
  \line@list@stuff{\jobname.\extensionchars\the\section@num}%
  \l@dend@stuff
  \setcounter{pstart}{1}
  \ifl@dpairing
    \global\l@dnumpstartsL \z@
    \global\pst@rtedLfalse
```

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.

```
  \else
    \begingroup
    \initnumbering@quote
    \ifwidthliketwocolumns%
       \csuse{setwidthliketwocolumns@\columns@position}%
       \csuse{setpositionliketwocolumns@\columns@position}%
    \fi%
  \fi
  \gdef\eled@sections@@{}%
  \if@noeled@sec\else%
   \makeatletter\InputIfFileExists{\jobname.eledsec\the\section@num}{}{}\makeatother%
   \immediate\openout\eled@sectioning@out=\jobname.eledsec\the\section@num\relax%
  \fi%
}
\newcommand*{\initnumbering@reg}{%
  \global\pst@rtedLfalse
  \global\l@dnumpstartsL \z@
  \global\absline@num \z@
  \gdef\normal@page@break{}
  \gdef\l@prev@pb{}
  \gdef\l@prev@nopb{}
  \global\line@num \z@
  \global\subline@num \z@
  \global\@lock \z@
  \global\sub@lock \z@
  \global\sublines@false
  \global\let\next@page@num=\relax
  \global\let\sub@change=\relax
  \resetprevline@
  \resetprevpage@num
  }

```

`\endnumbering` `\endnumbering` must follow the last text for a numbered section. It takes care of notifying you when changes have been noted in the input that require running the file through again to move everything to the right place.

```
\def\endnumbering{%
  \ifnumbering
    \global\numberingfalse
    \normal@pars
    \ifnum\l@dnumpstartsL=0%
      \led@err@NumberingWithoutPstart%
    \fi%
    \ifl@dpairing
      \global\pst@rtedLfalse
    \else
```

```
      \ifx\insertlines@list\empty\else
        \global\noteschanged@true
      \fi
      \ifx\line@list\empty\else
        \global\noteschanged@true
      \fi
    \fi
    \ifnoteschanged@
      \led@mess@NotesChanged
    \fi
  \else
    \led@err@NumberingNotStarted
  \fi
  \autoparfalse
  \if@noeled@sec\else%
    \immediate\closeout\eled@sectioning@out%
  \fi%
  \ifl@dpairing\else
    \global\l@dnumpstartsL=\z@%
    \endgroup
  \fi
}
```

\pausenumbering
\resumenumbering

The \pausenumbering macro is just the same as \endnumbering, but with the \ifnumbering flag set to true, to show that numbering continues across the gap.[23]

```
\newcommand{\pausenumbering}{%
  \ifautopar\global\autopar@pausetrue\fi%
  \endnumbering\global\numberingtrue}
```

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

```
\newcommand*{\resumenumbering}{%
  \ifnumbering
     \ifautopar@pause\autopar\fi
     \global\pst@rtedLtrue
     \global\advance\section@num \@ne
     \led@mess@SectionContinued{\the\section@num}%
     \line@list@stuff{\jobname.\extensionchars\the\section@num}%
     \l@dend@stuff
     \ifl@dpairing\else%
        \begingroup%
        \initnumbering@quote%
        \ifwidthliketwocolumns%
          \csuse{setwidthliketwocolumns@\columns@position}%
          \csuse{setpositionliketwocolumns@\columns@position}%
        \fi%
```

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

```
    \fi%
  \else
    \led@err@NumberingShouldHaveStarted
    \endnumbering
    \beginnumbering
  \fi}


```

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.

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

`\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 .

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

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

```
\newtoks\led@toksa \newtoks\led@toksb
\global\led@toksa={\\}
\long\def\xright@appenditem#1\to#2{%
  \global\led@toksb=\expandafter{#2}%
  \xdef#2{\the\led@toksb\the\led@toksa\expandafter{#1}}%
  \global\led@toksb={}}
```

\xleft@appenditem

`\xleft@appenditem` expands an item and appends it to the left end of a list macro; it is otherwise identical to `\xright@appenditem`.

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

\gl@p

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

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

```

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@
\bypstart@true
\bypstart@false
\ifbypage@
\bypage@true
\bypage@false

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

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

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

```
\newif\ifbypage@
\newif\ifbypstart@
```

\lineation

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

```
\newcommand*{\lineation}[1]{{
```

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

```
  \ifnumbering
    \led@err@LineationInNumbered
  \else
%   \end{macrocode}
% If the argument is \verb+page+.
%   \begin{macrocode}
    \def\@tempa{#1}\def\@tempb{page}%
    \ifx\@tempa\@tempb
        \global\bypage@true
        \global\bypstart@false
        \unless\ifnocritical@%
          \Xpstart[][false]%
        \fi%
%   \end{macrocode}
% If the argument is \verb+pstart+.
%   \begin{macrocode}
    \else
       \def\@tempb{pstart}%
       \ifx\@tempa\@tempb
           \global\bypage@false
           \global\bypstart@true
           \unless\ifnocritical@%
             \Xpstart%
           \fi%
%   \end{macrocode}
% And finally, if the argument is \verb+section+ (default).
%   \begin{macrocode}
       \else
          \def\@tempb{section}
          \ifx\@tempa\@tempb
           \global\bypage@false
           \global\bypstart@false
           \unless\ifnocritical@%
             \Xpstart[][false]%
           \fi%
```

In other case, it is an error.

```
   \else
            \led@warn@BadLineation
        \fi
       \fi
    \fi
  \fi}}
```

V.2 Line number margin

\linenummargin
\line@margin
\l@dgetline@margin

`\linenummargin{`⟨*word*⟩`}` specify which margin line numbers are in; it takes one argument, a string, which value can be `left` ; `right`; `inner` or `outer`.

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

```
\newcount\line@margin

\newcommand*{\linenummargin}[1]{{%
  \l@dgetline@margin{#1}%
  \ifnum\@l@dtempcntb>\m@ne
    \ifledRcol
      \global\line@marginR=\@l@dtempcntb
    \else
      \global\line@margin=\@l@dtempcntb
    \fi
  \fi}}
```

```
\newcommand*{\l@dgetline@margin}[1]{%
  \def\@tempa{#1}\def\@tempb{left}%
  \ifx\@tempa\@tempb
    \@l@dtempcntb \z@
  \else
     \def\@tempb{right}%
     \ifx\@tempa\@tempb
       \@l@dtempcntb \@ne
     \else
       \def\@tempb{outer}%
       \ifx\@tempa\@tempb
         \@l@dtempcntb \tw@
       \else
         \def\@tempb{inner}%
         \ifx\@tempa\@tempb
           \@l@dtempcntb \thr@@
         \else
           \led@warn@BadLinenummargin
           \@l@dtempcntb \m@ne
         \fi
       \fi
     \fi
  \fi}

```

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.

```
\newcounter{firstlinenum}
  \setcounter{firstlinenum}{5}
\newcounter{linenumincrement}
  \setcounter{linenumincrement}{5}
```

\c@firstsublinenum
\c@sublinenumincrement

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

```
\newcounter{firstsublinenum}
  \setcounter{firstsublinenum}{5}
\newcounter{sublinenumincrement}
  \setcounter{sublinenumincrement}{5}

```

\firstlinenum
\linenumincrement
\firstsublinenum
\sublinenumincrement

These macros can be used to set the corresponding counters.

```

```

```
\newcommand*{\firstlinenum}[1]{%
  \ifledRcol%
    \setcounter{firstlinenumR}{#1}%
  \else%
    \setcounter{firstlinenum}{#1}%
  \fi%
}
\newcommand*{\linenumincrement}[1]{%
  \ifledRcol%
    \setcounter{linenumincrementR}{#1}%
  \else%
    \setcounter{linenumincrement}{#1}%
  \fi%
}
\newcommand*{\firstsublinenum}[1]{%
  \ifledRcol%
    \setcounter{firstsublinenumR}{#1}%
  \else%
    \setcounter{firstsublinenum}{#1}%
  \fi%
}
\newcommand*{\sublinenumincrement}[1]{%
  \ifledRcol%
    \setcounter{sublinenumincrementR}{#1}%
  \else%
    \setcounter{sublinenumincrement}{#1}%
  \fi%
}

```

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.

```
\newcount\lock@disp
\newcommand{\lockdisp}[1]{{%
  \l@dgetlock@disp{#1}%
  \ifnum\@l@dtempcntb>\m@ne
    \global\lock@disp=\@l@dtempcntb
  \else
    \led@warn@BadLockdisp
  \fi}}
\newcommand*{\l@dgetlock@disp}[1]{
  \def\@tempa{#1}\def\@tempb{first}%
  \ifx\@tempa\@tempb
    \@l@dtempcntb \z@
  \else
     \def\@tempb{last}%
     \ifx\@tempa\@tempb
       \@l@dtempcntb \@ne
     \else
       \def\@tempb{all}%
       \ifx\@tempa\@tempb
         \@l@dtempcntb \tw@
       \else
         \@l@dtempcntb \m@ne
       \fi
     \fi
  \fi}

```

\sublockdisp
\sublock@disp

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

```
\newcount\sublock@disp
\newcommand{\sublockdisp}[1]{{%
  \l@dgetlock@disp{#1}%
  \ifnum\@l@dtempcntb>\m@ne
    \global\sublock@disp=\@l@dtempcntb
  \else
    \led@warn@BadSublockdisp
  \fi}}

```

V.5 Line number style

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

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

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

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

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

Initialise the number styles to arabic.

```
\linenumberstyle{arabic}
  \let\linenumr@p\linenumrep
\sublinenumberstyle{arabic}
  \let\sublinenumr@p\sublinenumrep

```

V.6 Line number printing

\leftlinenum
\rightlinenum
\linenumsep
\numlabfont
\ledlinenum

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

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

\ledlinenum typesets the line (and subline) number.

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

```
\newlength{\linenumsep}
  \setlength{\linenumsep}{1pc}
\newcommand*{\numlabfont}{\normalfont\scriptsize}
\newcommand*{\ledlinenum}{%
  \bgroup%
  \ifluatex%
    \luatextextdir TLT%
  \fi%
  \numlabfont\linenumrep{\line@num}%
  \ifsublines@
    \ifnum\subline@num>0\relax
      \unskip\fullstop\sublinenumrep{\subline@num}%
    \fi
  \fi%
  \egroup%
}%

\newcommand*{\leftlinenum}{%
  \ledlinenum
  \kern\linenumsep}
\newcommand*{\rightlinenum}{%
  \kern\linenumsep
  \ledlinenum}

```

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`.

```
565 \newcount\line@num
```

`\subline@num` The count `\subline@num` stores a sub-line number that qualifies `\line@num`. For example, line 10 might have sub-line numbers 1, 2 and 3, which might be printed as lines 10.1, 10.2, 10.3.

```
566 \newcount\subline@num
```

`\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.

```
567 \newif\ifsublines@
```

`\absline@num` The count `\absline@num` stores the absolute number of lines since the start of the section: that is, the number we 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.

```
568 \newcount\absline@num
```

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

V.8 Line number locking counter

`\@lock` `\sub@lock` The counts `\@lock` and `\sub@lock` tell us the state of line-number and sub-line-number 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.

```
\newcount\@lock
\newcount\sub@lock
```

V.9 Line number associated to lemma

\line@list
\insertlines@list
\actionlines@list
\actions@list

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

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

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

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

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

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

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

- `\actions@list`: action codes corresponding to the line numbers in `\actionlines@list`. These codes tell reledmac what action it is supposed to take at each of these lines. One action, the page-start action, is generated behind the scenes by reledmac itself; the others, for specifying sub-lineation, line-number locking, and line-number alteration, are generated only by explicit commands in your input file. The page-start and line-number-alteration actions require arguments, to specify the new values for the page or line numbers; instead of storing those arguments in another list, we have chosen the action-code values so that they can encode both the action and the argument in these cases. Action codes greater than -1000 are page-start actions, and the code value is the page number; action codes less than

-5000 specify line numbers, and the code value is a transformed version of the line number; action codes between these two values specify other actions which require no argument.

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

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

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

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

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

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

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

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

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

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

Here are the commands to create these lists:

```
    \list@create{\line@list}
    \list@create{\insertlines@list}
    \list@create{\actionlines@list}
    \list@create{\actions@list}

```

\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.

```
\newcount\page@num
\newcount\endpage@num
\newcount\endline@num
\newcount\endsubline@num
```

\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.

```
\newif\ifnoteschanged@
```

\resetprevline@

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

\resetprevline@

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

\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@

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

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.

```
\newread\@inputcheck
\newcommand*{\read@linelist}[1]{%
  \ifledRcol%
    \list@clearing@regR%
  \else%
    \list@clearing@reg%
  \fi%
```

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

```
  \list@clear{\maxlinesinpar@list}
```

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.

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

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.

```
  \ifledRcol
    \global\page@numR=\m@ne
    \ifx\actionlines@listR\empty
      \gdef\next@actionlineR{1000000}%
    \else
      \gl@p\actionlines@listR\to\next@actionlineR
      \gl@p\actions@listR\to\next@actionR
    \fi
  \else
    \global\page@num=\m@ne
    \ifx\actionlines@list\empty
      \gdef\next@actionline{1000000}%
    \else
      \gl@p\actionlines@list\to\next@actionline
      \gl@p\actions@list\to\next@action
    \fi
  \fi
}
```

\list@clearing@reg

Clears the lists for \read@linelist

```
\newcommand*{\list@clearing@reg}{%
  \list@clear{\line@list}%
  \list@clear{\insertlines@list}%
  \list@clear{\actionlines@list}%
  \list@clear{\actions@list}%
  \list@clear{\linesinpar@listL}%
  \list@clear{\linesonpage@listL}%
  }%
```

\get@linelistfile

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

```
\newcommand*{\get@linelistfile}[1]{%
  \InputIfFileExists{#1}{%
    \global\noteschanged@false
    \begingroup
      \catcode`\[=1 \catcode`\]=2
      \makeatletter \catcode`\^^M=9}{%
    \led@warn@NoLineFile{#1}%
    \global\noteschanged@true
    \begingroup}%
}

```

This version of \read@linelist creates list macros containing data for the entire section, so they could get rather large. It would be no more difficult to read the line-list file incrementally rather than all at once: we could read, at the start of each paragraph, only the commands relating to that paragraph. But this would require that we have two line-lists open at once, one for reading, one for writing, and on systems without version numbers we would have to do some file renaming outside of LaTeX for that to work. We have retained this slower approach to avoid that sort of hacking about, but have provided the \pausenumbering and \resumenumbering macros to help you if you run into macro memory limitations (see 4.2.7 p. 9 above).

V.11 Commands within the line-list file

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

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

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

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

```
\newcommand{\line@list@version}[1]{%
  \IfStrEq{#1}{\this@line@list@version}%
    {}%
    {\ifledRcol%
       \led@warn@Obsolete{\jobname.\extensionchars\the\section@num}%
     \else%
       \led@warn@Obsolete{\jobname.\extensionchars\the\section@num}%
     \fi%
     \endinput%
    }%
}%
```

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

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.

```
\newcommand{\@nl}[2]{%
  \fix@page{#1}%
  \@nl@reg}
\newcommand*{\@nl@reg}{%
  \ifx\l@dchset@num\relax \else
    \advance\absline@num \@ne
    \set@line@action
    \let\l@dchset@num=\relax
    \advance\absline@num \m@ne
    \advance\line@num \m@ne
  \fi
```

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

```
  \advance\absline@num \@ne
        \ifx\next@page@num\relax \else
            \page@action
            \let\next@page@num=\relax
        \fi
        \ifx\sub@change\relax \else
           \ifnum\sub@change>\z@
              \sublines@true
           \else
              \sublines@false
           \fi
           \sub@action
           \let\sub@change=\relax
        \fi
```

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

```
        \ifcase\@lock
           \or
              \@lock \tw@
           \or \or
              \@lock \z@
        \fi
        \ifcase\sub@lock
           \or
              \sub@lock \tw@
           \or \or
              \sub@lock \z@
        \fi
```

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

```
        \ifsublines@
             \ifnum\sub@lock<\tw@
               \advance\subline@num \@ne
             \fi
        \else
             \ifnum\@lock<\tw@
               \advance\line@num \@ne \subline@num \z@
             \fi
        \fi}

```

\last@page@num
\fix@page

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

```
\newcount\last@page@num
  \last@page@num=-10000
\newcommand*{\fix@page}[1]{%
  \ifnum #1=\last@page@num
  \else
    \ifbypage@
      \csxdef{lastlinenumberon@\the\last@page@num}{\the\line@num}%
      \line@num=\z@ \subline@num=\z@
    \fi
    \page@num=#1\relax
    \last@page@num=#1\relax
    \def\next@page@num{#1}%
    \listxadd{\normal@page@break}{\the\absline@num}
  \fi}

```

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

```
\newcommand*{\@pend}[1]{}
\newcommand*{\@pendR}[1]{}
\newcommand*{\@lopL}[1]{}
\newcommand*{\@lopR}[1]{}

```

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

```
\newcommand*{\sub@on}{\ifsublines@
     \let\sub@change=\relax
  \else
     \def\sub@change{1}%
  \fi}
\newcommand*{\sub@off}{\ifsublines@
     \def\sub@change{-1}%
  \else
     \let\sub@change=\relax
  \fi}

```

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

```
\newcommand*{\@adv}[1]{\ifsublines@
       \advance\subline@num by #1\relax
       \ifnum\subline@num<\z@
         \led@warn@BadAdvancelineSubline
          \subline@num \z@
       \fi
  \else
       \advance\line@num by #1\relax
       \ifnum\line@num<\z@
         \led@warn@BadAdvancelineLine
          \line@num \z@
```

```
      \fi
  \fi
  \set@line@action}

```

\@set

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

```
\newcommand*{\@set}[1]{\ifsublines@
    \subline@num=#1\relax
  \else
    \line@num=#1\relax
  \fi
  \set@line@action}

```

\l@d@set
\l@dchset@num

The `\l@d@set{⟨num⟩}` macro sets the line number for the next `\pstart` to the value specified as its argument. This is used to implement `\setlinenum`.

`\l@dchset@num` is a flag to the `\@nl`? macro. If it is not `\relax` then a linenumber change is to be done.

```
\newcommand*{\l@d@set}[1]{%
  \line@num=#1\relax
  \advance\line@num \@ne
  \def\l@dchset@num{#1}}
\let\l@dchset@num\relax

```

\page@action

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

```
\newcommand*{\page@action}{%
  \xright@appenditem{\the\absline@num}\to\actionlines@list
  \xright@appenditem{\next@page@num}\to\actions@list}
```

\set@line@action

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

```
\newcommand*{\set@line@action}{%
  \xright@appenditem{\the\absline@num}\to\actionlines@list
  \ifsublines@
       \@l@dtempcnta=-\subline@num
  \else
       \@l@dtempcnta=-\line@num
  \fi
  \advance\@l@dtempcnta by -5000
  \xright@appenditem{\the\@l@dtempcnta}\to\actions@list}
```

\sub@action

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

```
\newcommand*{\sub@action}{%
  \xright@appenditem{\the\absline@num}\to\actionlines@list
  \ifsublines@
     \xright@appenditem{-1001}\to\actions@list
  \else
      \xright@appenditem{-1002}\to\actions@list
  \fi}
```

\lock@on
\do@lockon
\do@lockonL

\lock@on adds an entry to the action-code list to turn line number locking on. The current setting of the sub-lineation flag tells us whether this applies to line numbers or 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.

```
\newcommand*{\lock@on}{\futurelet\next\do@lockon}
\newcommand*{\do@lockon}{%
  \ifx\next\lock@off
    \global\let\lock@off=\skip@lockoff
  \else
    \do@lockonL
  \fi}
\newcommand*{\do@lockonL}{%
  \xright@appenditem{\the\absline@num}\to\actionlines@list
  \ifsublines@
    \xright@appenditem{-1005}\to\actions@list
    \ifnum\sub@lock=\z@
      \sub@lock \@ne
    \else
      \ifnum\sub@lock=\thr@@
        \sub@lock \@ne
      \fi
    \fi
  \else
    \xright@appenditem{-1003}\to\actions@list
    \ifnum\@lock=\z@
      \@lock \@ne
    \else
      \ifnum\@lock=\thr@@
        \@lock \@ne
      \fi
    \fi
  \fi}

```

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

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

```
\newcommand*{\do@lockoffL}{%
  \xright@appenditem{\the\absline@num}\to\actionlines@list
  \ifsublines@
    \xright@appenditem{-1006}\to\actions@list
    \ifnum\sub@lock=\tw@
      \sub@lock \thr@@
    \else
```

```
      \sub@lock \z@
    \fi
  \else
    \xright@appenditem{-1004}\to\actions@list
    \ifnum\@lock=\tw@
      \@lock \thr@@
    \else
      \@lock \z@
    \fi
  \fi}
\newcommand*{\do@lockoff}{\do@lockoffL}
\newcommand*{\skip@lockoff}{\global\let\lock@off=\do@lockoff}
\global\let\lock@off=\do@lockoff

```

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

```
\newcommand*{\n@num}{%
  \ifledRcol%
    \xright@appenditem{\the\absline@numR}\to\actionlines@listR
    \xright@appenditem{-1007}\to\actions@listR
  \else%
    \xright@appenditem{\the\absline@num}\to\actionlines@list%
    \xright@appenditem{-1007}\to\actions@list%
  \fi%
}%

```

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

```
\newcommand*{\n@num@stanza}{%
  \ifledRcol%
    \xright@appenditem{\the\absline@numR}\to\actionlines@listR%
    \xright@appenditem{-1008}\to\actions@listR%
  \else%
    \xright@appenditem{\the\absline@num}\to\actionlines@list%%
    \xright@appenditem{-1008}\to\actions@list%
  \fi%
}
```

\ifl@dhidenumber \hidenumbering \h@num \hidenumbering hides number in margin. It uses action code 1009.

```
\newif\ifl@dhidenumber
\newcommand*{\hidenumbering}{
  \ifledRcol%
    \write\linenum@outR{\string\hide@num}%
  \else%
    \write\linenum@out{\string\hide@num}%
  \fi%
}%
\newcommand*{\hide@num}{%
  \ifledRcol%
    \xright@appenditem{\the\absline@numR}\to\actionlines@listR%
    \xright@appenditem{-1009}\to\actions@listR%
  \else%
    \xright@appenditem{\the\absline@num}\to\actionlines@list%%
    \xright@appenditem{-1009}\to\actions@list%
  \fi%
}
```

\@ref
\insert@count

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

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

```
    \newcount\insert@count
```

- #2, a sequence of other line-list-file commands, executed to determine the ending line-number. (This may also include other \@ref commands, corresponding to uses of \edtext within the first argument of another instance of \edtext.)

\dummy@ref

When nesting of \@ref commands does occur, it is necessary to temporarily redefine \@ref within \@ref, so that we are only doing one of these at a time.

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

\@ref@reg

The first thing \@ref (i.e. \@ref@reg) itself does is to add the specified number of items to the \insertlines@list list.

```
\newcommand*{\@ref}[2]{%
  \@ref@reg{#1}{#2}}
\newcommand*{\@ref@reg}[2]{%
  \global\insert@count=#1\relax
  \global\advance\@edtext@level by 1%
  \loop\ifnum\insert@count>\z@
    \xright@appenditem{\the\absline@num}\to\insertlines@list
    \global\advance\insert@count \m@ne
  \repeat
```

Next, process the second argument to determine the page and line numbers for the end of this lemma. We temporarily equate \@ref to a different macro that just executes its argument, so that nested \@ref commands are just skipped this time. Some other macros need to be temporarily redefined to suppress their action.

```
  \begingroup
    \let\@ref=\dummy@ref
    \let\@lopL\@gobble
    \let\page@action=\relax
    \let\sub@action=\relax
    \let\set@line@action=\relax
    \let\@lab=\relax
    \let\@lemma=\relax%
    \let\@sw\@gobblethree%
    #2
    \global\endpage@num=\page@num
    \global\endline@num=\line@num
    \global\endsubline@num=\subline@num
  \endgroup
```

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

```
    \xright@appenditem%
      {\the\page@num|\the\line@num|%
       \ifsublines@ \the\subline@num \else 0\fi|%
       \the\endpage@num|\the\endline@num|%
       \ifsublines@ \the\endsubline@num \else 0\fi}\to\line@list
```

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.

```
   \expandafter\list@create\expandafter{\csname sw@list@edtext@tmp@\the\@edtext@level\endc
     \providebool{lemmacommand@\the\@edtext@level}%
     \boolfalse{lemmacommand@\the\@edtext@level}%
```

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

```
  #2%
```

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.

```
  \ifnum\@edtext@level>0%
   \def\create@this@edtext@level{\expandafter\list@create\expandafter{\csname sw@list@edte
   \ifcsundef{sw@list@edtext@\the\@edtext@level}{\create@this@edtext@level}{}%
    \letcs{\@tmp}{sw@list@edtext@\the\@edtext@level}%
    \letcs{\@tmpp}{sw@list@edtext@tmp@\the\@edtext@level}
    \xright@appenditem{\expandonce\@tmpp}\to\@tmp%
    \global\cslet{sw@list@edtext@\the\@edtext@level}{\@tmp}%
  \fi%
```

Decrease edtext level counter.

```
  \global\advance\@edtext@level by -1%
}

```

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`.

```
\newwrite\linenum@out
```

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

Once any file is opened on this stream, we keep it open forever, or else switch to another file that we keep open. The reason is that we want the output routine to write the page number for every page to this file; otherwise we 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 inelegant 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`.

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

\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.

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

\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.

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

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

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

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

```
  \iffirst@linenum@out@
     \immediate\closeout\linenum@out%
     \global\first@linenum@out@false%
     \immediate\openout\linenum@out=#1\relax%
   \immediate\write\linenum@out{\string\line@list@version{\this@line@list@version}}%
  \else
```

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.

```
     \if@minipage%
       \leavevmode%
     \fi%
     \closeout\linenum@out%
     \openout\linenum@out=#1\relax%
  \fi}

```

`\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.

```
\newcommand*{\new@line}{%
  \IfStrEq{\led@pb@setting}{after}%
    {\xifinlist{\the\absline@num}{\l@prev@nopb}%
      {\xifinlist{\the\absline@num}{\normal@page@break}%
        {\numgdef{\@next@page}{\thepage+1}%
        \write\linenum@out{\string\@nl[\@next@page][\@next@page]}%
      }%
      {\write\linenum@out{\string\@nl[\the\c@page][\thepage]}}%
    }%
    {\write\linenum@out{\string\@nl[\the\c@page][\thepage]}}}%
  {}%
  \IfStrEq{\led@pb@setting}{before}%
    {\numdef{\next@absline}{\the\absline@num+1}%
    \xifinlist{\next@absline}{\l@prev@nopb}%
      {\xifinlist{\the\absline@num}{\normal@page@break}%
        {\numgdef{\nc@page}{\c@page+1}%
        \write\linenum@out{\string\@nl[\nc@page][\nc@page]}%
        }%
        {\write\linenum@out{\string\n@l[\the\c@page][\thepage]}}%
      }%
      {\write\linenum@out{\string\@nl[\the\c@page][\thepage]}}%
    }%
    {}%
 \IfStrEqCase{\led@pb@setting}{{before}{\relax}{after}{\relax}}[\write\linenum@out{\strin
}

```

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

```
\newif\if@noneed@Footnote%

\newcommand*{\flag@start}{%
  \ifledRcol%
    \edef\next{\write\linenum@outR{%
                  \string\@ref[\the\insert@countR][]}}%
    \next%
    \ifnum\insert@countR<1%
      \if@noneed@Footnote\else%
        \led@err@EdtextWithoutFootnote%
      \fi%
    \fi%
  \else%
    \edef\next{\write\linenum@out{%
```

```
                   \string\@ref[\the\insert@count][]}%
  \next%
  \ifnum\insert@count<1%
    \if@noneed@Footnote\else%
      \led@err@EdtextWithoutFootnote%
    \fi%
  \fi%
\fi}%

```

\startsub
\endsub

\startsub and \endsub turn sub-lineation on and off, by writing appropriate instructions to the line-list file. When sub-lineation is in effect, the line number counter is frozen and the sub-line counter advances instead. If one of these commands appears in the middle of a line, it 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 attached to the last word preceding the change, not the first word that follows the change. This is because sub-lineation will often turn on and off in mid-line—stage directions, for example, often are mixed with dialogue in that way—and when a line is mixed we want to label it using the system that was in effect at its start. But when sub-lineation begins at the very start of a line we have a problem, if we don't put in this code.

```
\newcommand*{\startsub}{\dimen0\lastskip
  \ifdim\dimen0>0pt \unskip \fi
  \write\linenum@out{\string\sub@on}%
  \ifdim\dimen0>0pt \hskip\dimen0 \fi}
\def\endsub{\dimen0\lastskip
  \ifdim\dimen0>0pt \unskip \fi
  \write\linenum@out{\string\sub@off}%
  \ifdim\dimen0>0pt \hskip\dimen0 \fi}

```

\advanceline

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

```
\newcommand*{\advanceline}[1]{\write\linenum@out{\string\@adv[#1]}}
```

\setline

You can use \setline{⟨*num*⟩} in running text (i.e., within \pstart...\pend) to set the current visible line-number to a specified positive value.

```
\newcommand*{\setline}[1]{%
  \ifnum#1<\z@
    \led@warn@BadSetline
  \else
    \write\linenum@out{\string\@set[#1]}%
  \fi}

```

\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.

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

```

\startlock
\endlock

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

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

```

\ifl@dskipnumber
\ifl@dskipversenumber
\l@dskipnumbertrue
\l@dskipnumberfalse
\skipnumbering

In numbered text `\skipnumbering` will suspend the numbering for that particular line.

```
\newif\ifl@dskipnumber
\newif\ifl@dskipversenumber%
\newcommand*{\skipnumbering}{%
  \leavevmode%
  \ifledRcol%
    \ifinstanza%
       \write\linenum@outR{\string\n@num@stanza}%
    \else%
      \write\linenum@outR{\string\n@num}%
    \fi%
    \advanceline{-1}%
  \else%
    \ifinstanza%
       \write\linenum@out{\string\n@num@stanza}%
    \else%
      \write\linenum@out{\string\n@num}%
    \fi%
    \advanceline{-1}%
  \fi%
}%

```

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. 103). 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.

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

`\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.

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

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

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

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` `\morenoexpands` We need to turn off macro expansion for certain sorts of macros we are likely to see within the lemma and within the notes.

The first class is font-changing macros. We suppress expansion for them by letting them become equal to zero.[24] This is done because we want to pass into our notes the generic commands to change to roman or whatever, and not their expansions that will ask for a particular style at a specified size. The notes may well be in a smaller font, so the command should be expanded later, when the note's environment is in effect.

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

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

We also need to eliminate all reledmac macros like `\edlabel` and `\setline` that write things to auxiliary files: that writing should be done only once. And we make

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

`\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.)

```
\newcommand*{\no@expands}{%
  \let\select@@lemmafont=0%
  \let\startsub=\relax  \let\endsub=\relax
  \let\startlock=\relax \let\endlock=\relax
  \let\edlabel=\@gobble
  \let\setline=\@gobble \let\advanceline=\@gobble
  \let\sameword\sameword@inedtext%
  \let\edtext=\dummy@edtext
  \l@dtabnoexpands
  \morenoexpands}
\let\morenoexpands=\relax

```

`\@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.

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

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

```
\newcount\@edtext@level%
\@edtext@level=0%
```

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

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

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

```
  \ifnumberedpar@%
```

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

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

By default, we do not use `\lemma`

```
      \global\@lemmacommand@false%
```

```
      \begingroup%
```

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`.

```
      \ifledRcol%
        \ifcsundef{sw@list@edtextR@\the\@edtext@level}%
              {\global\let\sw@inthisedtext\empty}%
              {\ifcsempty{sw@list@edtextR@\the\@edtext@level}%
{\global\let\sw@inthisedtext\empty}%
{\expandafter\gl@p\csname sw@list@edtextR@\the\@edtext@level\endcsname\to\sw@inthisedtext}
      }%
      \else%
        \ifcsundef{sw@list@edtext@\the\@edtext@level}%
              {\global\let\sw@inthisedtext\empty}%
              {\ifcsempty{sw@list@edtext@\the\@edtext@level}%
{\global\let\sw@inthisedtext\empty}%
{\expandafter\gl@p\csname sw@list@edtext@\the\@edtext@level\endcsname\to\sw@inthisedtext}%
      }%
      \fi%
```

`\@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.

```
        \global\renewcommand{\@tag}{%
          \no@expands #1%
          }%
```

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

```
        \set@line%
```

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

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

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

```
      \ignorespaces #2\relax%
```

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

```
      \@ifundefined{xpg@main@language}{%if not polyglossia
         \flag@start}%
         {\if@RTL\flag@end\else\flag@start\fi%
         }%
```

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

```
     \if@lemmacommand@%
       \ifledRcol%
         \write\linenum@outR{\string\@lemma}%
       \else%
         \write\linenum@out{\string\@lemma}%
       \fi%
     \fi%
```

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.

```
    \endgroup%
    \showlemma{#1}%
```

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

```
    \ifx\end@lemmas\empty \else%
      \gl@p\end@lemmas\to\x@lemma%
      \x@lemma%
      \global\let\x@lemma=\relax%
    \fi%
    \@ifundefined{xpg@main@language}{%if not polyglossia
         \flag@end}%
     {\if@RTL\flag@start\else\flag@end\fi% With polyglossia, you must track whether the language reads
         }%
```

We switch to false some flags.

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

```
      \global\@noneed@Footnotefalse%
      \global\advance\@edtext@level by -1%
      \global\@lemmacommand@false%
```

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

```
  \else%
 \showlemma{#1} (\textbf{\textsc{Edtext outside numbered paragraph}})\led@err@edtextoutside
  \fi%
}%

\newcommand*{\flag@end}{%
  \ifledRcol%
    \write\linenum@outR{[]}%
  \else%
    \write\linenum@out{[]}%
  \fi}%

```

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

```
\newif\ifnumberline
\numberlinetrue
```

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

One instance of `\edtext` may generate several notes, or it may generate none — it is legitimate for argument #2 to `\edtext` to be empty. But `\flag@start` and `\flag@end` induce the generation of a single entry in `\line@list` during the next run, and it is vital to also remove one and only one `\line@list` entry here.

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

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

```
  \ifx\line@list\empty
    \global\noteschanged@true
    \xdef\l@d@nums{000|000|000|000|000|000|\edfont@info}%
 \else
   \gl@p\line@list\to\@tempb
   \xdef\l@d@nums{\@tempb|\edfont@info}%
   \global\let\@tempb=\undefined
 \fi}

```

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

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

```

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. 89).

```
\unless\ifnocritical@
\newcommand*{\lemma}[1]{%
  \global\@lemmacommand@true%
  \global\renewcommand{\@tag}{%
    \no@expands #1%
  }%
  \ignorespaces%
}%
```

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

```
\newcommand{\@lemma}{%
  \booltrue{lemmacommand@\the\@edtext@level}%
}%
\fi
```

\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.

```
\newif\if@lemmacommand@%
```

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. 69): 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.

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

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

```
\def\line@set#1|#2\\#3|#4\\{%
  \gdef\@tempb{#1}%
  \ifx\@tempb\empty
        \l@d@add{#3}%
  \else
        \l@d@add{#1}%
  \fi
  \gdef\@tempb{#4}%
  \ifx\@tempb\empty\else
      \l@d@add{|}\line@set#2\\#4\\%
  \fi}
```

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

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

```

VI.4 Lemma disambiguation

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

The general mechanism is the following:

- **At the first run**, each `\sameword` command increments an etoolbox counter the name of which contains the argument of the `\sameword` commands.
- Then this counter, associated with the argument of `\sameword` is stored, with the `\@sw` command, in the auxiliary file of the current eledmac section (the `.1`, `.2`... file).
- **When this auxiliary file is read at the second run**, different operations are achieved:
    1. Get the rank of each `\sameword` in a line (relative rank) from the rank of each `\sameword` in all the numbered section (absolute rank):
        - For each paired `\sameword` argument and absolute line number, a counter is defined. Its value corresponds to the number of times `\sameword{`⟨*argument*⟩`}+` is called from the beginning of the lineation to the end of the current line. We also store the same data for the preceding absolute line number, if it does not have `\sameword{`⟨*argument*⟩`}`.
        - For each `\sameword` having the same argument, we subtract from its absolute rank the number stored for the paired `\sameword` argument and previous absolute line number. Consequently, we obtain the relative rank.
        - See the following example which explain how for same `\sameword` absolute ranks are transformed to relative rank.

          ```
          At line 1:
          ```

```
absolute rank 1 becomes relative rank 1-0 = 1
1 is stored for this \sameword and the line 1
At line 2:
absolute rank 2 becomes relative rank 2-1 = 1
absolute rank 3 becomes relative rank 3-2 = 2
3 is stored for this \sameword and the line 2
At line 3:
no \sameword for this line.
3 is stored for this \sameword and the line 3
At line 4:
absolute rank 4 becomes relative rank 4-3 = 1
3 is stored for this \sameword and the line 4
```

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

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

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

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

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

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

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

- At the second run, when a critical notes is called, we associate it to the next item of the list associated to is `\edtext` level. So, in the previous example:
  - Critical notes (A) and (B) are associated with $\{1,2,2,3\}$.
  - Critical note (C) is associated with $\{1,2,2,3,4,3\}$.
  - Critical note (D) is associated with $\{5\}$.
  - Critical note (E) is associated with $\{5,4\}$.
- At the second run, when a critical note is printed:
  - The `\sameword` command is let `\sameword@inedtext`.
  - At each call of this `\sameword@inedtext`, we step to the next element of the list associated to the note. Let it be r.

- For the word marked by `\sameword`, we calculate how many time it is called in its line. To do it:
  * We get the absolute line number of the current `\sameword`. This absolute line number was stored with list of relative rank for the current `\edtext`. That means, in the previous example, that, if the absolute line number of `\edtext` was 1, that critical notes (A) and (B) were not associated with $\{1, 2, 2, 3\}$ but with $\{(1, 1), (2, 1), (2, 1), (3, 1)\}$. Such method to know the absolute line number associated to a `\sameword` is required because a `\edtext` can be overlap many lines, but `\sameword` can't get it.
  * We get the value associated, when reading the auxiliary file, to the pair compose by the current marked word and the current absolute line number. Let this value be n.
- If $n > 1$, that mean the current word appears more than once time in its line. In this case, we call `\showwordrank` with the word as first argument and r as second argument. If the word is called only once, we just print it.

After theory, implementation.

`\get@sw@txt` As the argument of `\sameword` can contain active character if we use inputenc with `utf8` option instead of native UTF-8 engine, we store its detokenized content in a macro in order to allow dynamic name of macro with `\csname`.[25]

Because there is a bug with `\detokenize` and XƎTEX when using non BMP characters[26], we detokenize only for not XƎTEXengines. In any case, in XƎTEX, a `\csname` construction can contain UTF-8 characters without a problem, as UTF-8 characters are not managed with category code, but instead read directly as UTF-8 characters.

```
\newcommand{\get@sw@txt}[1]{%
  \ifxetex%
    \xdef\sw@txt{#1}%
  \else%
    \expandafter\xdef\expandafter\sw@txt\expandafter{\detokenize{#1}}%
  \fi%
}%
```

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

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

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

```
  \unless\ifledRcol%
    \csnumgdef{sw@\sw@txt}{\csuse{sw@\sw@txt}+1}%
```

Then, write its value to the numbered file.

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

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

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

```
  \else%
     \csnumgdef{sw@\sw@txt@R}{\csuse{sw@\sw@txt@R}+1}%
    \protected@write\linenum@outR{}{\string\@sw{\sw@txt}{\csuse{sw@\sw@txt@R}}{#1}}%
  \fi%
```

And print the word.

```
  #2%
}%
```

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

`\if@addsw`

```
\newif\if@addsw%
```

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

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

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

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

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

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

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

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

And do the same thing for the right side.

```
  \else%
    \csxdef{sw@\sw@txt @\the\absline@numR @\the\section@numR @R}{#2}%
    \numdef{\prev@line}{\the\absline@numR-1}%
    \ifcsundef{sw@\sw@txt @\prev@line @\the\section@numR @R}{%
      \csnumgdef{sw@\sw@txt @\prev@line @\the\section@numR @R}{#2-1}%
     }{}%
 \numdef{\the@sw}{#2-\csuse{sw@\sw@txt @\prev@line @\the\section@numR @R}}%
  \fi%
```

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

```
  \@tempcnta=\@edtext@level
  \@whilenum{\@tempcnta>0}\do{%
    \ifcsdef{sw@list@edtext@tmp@\the\@tempcnta}%
      {%
        \@addswfalse%
        \notbool{lemmacommand@\the\@tempcnta}%
          {\@addswtrue}%
          {\IfStrEq{#3}{inlemma}%
            {\@addswtrue}%
            {%
             \def\do##1{%
                \ifnumequal{##1}{\the\@tempcnta}%
                  {\@addswtrue\listbreak}%
                  {}%
             }%
             \docsvlist{#3}%
            }%
          }%
          \if@addsw%
            \letcs{\@tmp}{sw@list@edtext@tmp@\the\@tempcnta}%
            \ifledRcol%
              \xright@appenditem{{\the@sw}{\the\absline@numR}}\to\@tmp%
            \else%
              \xright@appenditem{{\the@sw}{\the\absline@num}}\to\@tmp%
            \fi%
            \cslet{sw@list@edtext@tmp@\the\@tempcnta}{\@tmp}%
          \fi%
      }%
      {}%
    \advance\@tempcnta by -1%
  }%
}%
```

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

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

Just a precaution.

```
      \ifx\sw@list@inedtext\empty%
        \def\the@sw{999}%
        \def\this@absline{-99}%
      \else%
```

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

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

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

```
      \ifcsdef{sw@\sw@txt @\this@absline @\the\section@num}{%
        \numdef{\prev@line}{\this@absline-1}%
     \numdef{\sw@atthisline}{\csuse{sw@\sw@txt @\this@absline @\the\section@num}-\csuse{sw
        }%
        {\numdef{\sw@atthisline}{0}}%
```

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

```
   \ifnumgreater{\sw@atthisline}{1}%
     {\showwordrank{#2}{\the@sw}}%
     {#2}%
```

And the same for right side.

```
 \else%
   \ifx\sw@list@inedtext\empty%
    \def\the@sw{999}%
    \def\this@absline{-99}%
   \else%
    \gl@p\sw@list@inedtext\to\@tmp%
    \edef\the@sw{\expandafter\@firstoftwo\@tmp}%
    \edef\this@absline{\expandafter\@secondoftwo\@tmp}%
   \fi%
   \ifcsdef{sw@\sw@txt @\this@absline @\the\section@numR @R}{%
    \numdef{\prev@line}{\this@absline-1}%
   \numdef{\sw@atthisline}{\csuse{sw@\sw@txt @\this@absline @\the\section@numR @R}-\csuse{sw@\sw@txt
    }%
    {\numdef{\sw@atthisline}{0}}%
    \ifnumgreater{\sw@atthisline}{1}%
     {\showwordrank{#2}{\the@sw}}%
     {#2}%
 \fi%
}%
```

\showwordrank

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

VII Paragraph decomposition and reassembly

In order to be able to count the lines of text and affix line numbers, we add an extra stage of processing for each paragraph. We send the paragraph into a box register, rather than straight onto the vertical list, and when the paragraph ends we slice the paragraph into its component lines; to each line we add any notes or line numbers, add a command to write to the line-list, and then at last send the line to the vertical list. This section contains all the code for this processing.

VII.1 Boxes, counters, \pstart and \pend

\raw@text
\ifnumberedpar@
\numberedpar@true
\numberedpar@false
\num@lines
\one@line
\par@line

Here are numbers and flags that are used internally in the course of the paragraph decomposition.

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

```
\newbox\raw@text
\newif\ifnumberedpar@
\newcount\num@lines
\newbox\one@line
\newcount\par@line
```

`\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.

```

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

\newcounter{pstart}
\renewcommand{\thepstart}{{\bfseries\@arabic\c@pstart}. }
\newif\ifnumberpstart
\numberpstartfalse
\newif\iflabelpstart
\labelpstartfalse
\newcommandx*{\pstart}[1][1]{%
  \normal@pars%
  \ifstrempty{#1}{\at@every@pstart}{\noindent#1}%
  \ifautopar%
    \autopar%
  \fi%
  \ifluatex%
    \edef\l@luatextextdir@L{\the\luatextextdir}%
  \fi%
  \if@nobreak%
    \let\@oldnobreak\@nobreaktrue%
  \else%
    \let\@oldnobreak\@nobreakfalse%
  \fi%
  \@nobreaktrue%
\ifnumbering \else%
  \led@err@PstartNotNumbered%
  \beginnumbering%
\fi%
\ifnumberedpar@%
  \led@err@PstartInPstart%
  \pend%
\fi%
\list@clear{\inserts@list}%
\global\let\next@insert=\empty%
\begingroup\normal@pars%
\global\advance \l@dnumpstartsL\@ne
\global\setbox\raw@text=\vbox\bgroup%
  \ifautopar\else%
  \ifnumberpstart%
    \ifinstanza\else%
     \ifsidepstartnum\else%
       \thepstart%
      \fi%
     \fi%
    \fi%
   \fi%
\numberedpar@true%
\iflabelpstart\protected@edef\@currentlabel%
     {\p@pstart\thepstart}
\fi%
\l@dzeropenalties%
}
```

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

```
\newcommandx*{\pend}[1][1]{\ifnumbering \else%
  \led@err@PendNotNumbered%
\fi%
\global\l@dskipversenumberfalse%
\ifnumberedpar@ \else%
  \led@err@PendNoPstart%
\fi%
```

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.

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

We check if lineation is by pstart: in this case, we reset line number, but only in the second line of the pstart. 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.

```
  \csnumdef{pstartline}{0}%
  \loop\ifvbox\raw@text%
    \csnumdef{pstartline}{\pstartline+1}%
    \do@line%
    \ifbypstart@%
     \ifnumequal{\pstartline}{1}{\setline{1}\resetprevline@}{}%
    \fi%
  \repeat%
```

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

```
  \flush@notes%
  \endgroup%
  \ignorespaces%
```

Increase pstart counter.

```
  \ifnumberpstart%
     \pstartnumtrue%
  \fi%
  \addtocounter{pstart}{1}%
```

Restore paragraph, nobreak setting and autopar setting.

```
  \normal@pars%
  \@oldnobreak%
  \ifautopar%
    \autopar%
  \fi%
```

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

```
  \ifstrempty{#1}{\at@every@pend}{\noindent#1}%
}

```

Here, two macros to insert content after every \pend, between numbered line. \AtEveryPend is the user macro, \at@every@pend is macro set by it.

\AtEveryPend
\at@every@pend

```

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

```

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

```
\newcommand*{\l@dzeropenalties}{%
  \brokenpenalty \z@ \clubpenalty \z@
  \displaywidowpenalty \z@ \interlinepenalty \z@ \predisplaypenalty \z@
  \postdisplaypenalty \z@ \widowpenalty \z@}

```

`\autopar` In most cases it is only an annoyance to have to label the paragraphs to be numbered with `\pstart` and `\pend`. `\autopar` will do that automatically, allowing you to start a paragraph with its first word and no other preliminaries, and to end it with a blank line or a `\par` command. The command should be issued within a group, after `\beginnumbering` has been used to start the numbering; all paragraphs within the group will be affected.

A few situations can cause problems. One is a paragraph that begins with a begin-group character or command: `\pstart` will not get invoked until after such a group beginning is processed; as a result the character that ends the group will be mistaken for the end of the `\vbox` that `\pstart` creates, and the rest of the paragraph will not be numbered. Such paragraphs need to be started explicitly using `\indent`, `\noindent`, or `\leavevmode` — or `\pstart`, since you can still include your own `\pstart` and `\pend` commands even with `\autopar` on.

Prematurely ending the group within which `\autopar` is in effect will cause a similar problem. You must either leave a blank line or use `\par` to end the last paragraph before you end the group.

The functioning of this macro is more tricky than the usual `\everypar`: we do not want anything to go onto the vertical list at all, so we have to end the paragraph, erase any evidence that it ever existed, and start it again using `\pstart`. We remove the paragraph-indentation box using `\lastbox` and save the width, and then skip backwards over the `\parskip` that has been added for this paragraph. Then we start again with `\pstart`, restoring the indentation that we saved, and locally change `\par` so that it will do our `\pend` for us.

```
\newif\ifautopar
\autoparfalse
\newcommand*{\autopar}{
  \ifledRcol
    \ifnumberingR \else
    \led@err@AutoparNotNumbered
    \beginnumberingR
    \fi
    \else
    \ifnumbering \else
    \led@err@AutoparNotNumbered
    \beginnumbering
    \fi
  \fi
  \autopartrue
  \everypar{\setbox0=\lastbox
    \endgraf \vskip-\parskip
   \pstart \noindent \kern\wd0 \ifnumberpstart\ifinstanza\else\thepstart\fi\fi
    \let\par=\pend}%
  \ignorespaces}
```

`\normal@pars` We also define a macro which we can rely on to turn off the `\autopar` definitions at various important places, if they are in force. We will want to do this within a footnotes, for example.

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

```

\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.

```
\newif\ifautopar@pause
```

VII.2 Processing one line

VII.2.1 General process

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

```
\newcommand*{\l@dunhbox@line}[1]{\unhbox #1}
\newcommand*{\do@line}{%
 {\vbadness=10000
  \splittopskip=\z@
  \do@linehook
\l@demptyd@ta
  \global\setbox\one@line=\vsplit\raw@text to\baselineskip}%
 \unvbox\one@line \global\setbox\one@line=\lastbox
 \getline@num
 \IfStrEq{\led@pb@setting}{before}{\led@check@pb\led@check@nopb}{}
 \ifnum\@lock>\@ne
   \inserthangingsymboltrue
 \else
   \inserthangingsymbolfalse
 \fi
 \check@pb@in@verse
 \ifl@dhidenumber%
   \global\l@dhidenumberfalse%
   \f@x@l@cks%
 \else%
   \affixline@num%
 \fi%
```

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

```
 \xifinlist{\the\l@dnumpstartsL}{\eled@sections@@}%
    {\print@eledsection}%
    {\print@line}%
 \IfStrEq{\led@pb@setting}{after}{\led@check@pb\led@check@nopb}{}
 }%
```

VII.2.2 Process for “normal” line

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

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

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

```
    \affixpstart@num%
```

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

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

User hook.

```
    \do@insidelinehook%
```

Left line number

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

Restore marginal and footnotes.

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

Print left notes.

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

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

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

If we use LuaLaTeX then restore the direction.

```
        \ifluatex%
          \luatextextdir\l@luatextextdir@L%
        \fi%
```

Insert, if needed, the hanging symbol.

```
        \inserthangingsymbol %Space keept for backward compatibility
```

And so, print the line.

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

Right line number

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

Print right notes.

```
        \l@drsn@te
      }}%
```

And reinsert penalties (for page breaking)…

```
    \add@penalties%
}
```

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.

```
\def\print@eledsection{%
    \add@inserts\affixside@note%
    \numdef{\temp@}{\l@dnumpstartsL-1}%
    \xifinlist{\temp@}{\eled@sections@@}{\@nobreaktrue}{\@nobreakfalse}%
    \@eled@sectioningtrue%
    \csuse{eled@sectioning@\the\l@dnumpstartsL}%
    \@eled@sectioningfalse%
    \global\csundef{eled@sectioning@\the\l@dnumpstartsL}%
    \if@RTL%
      \hspace{-3\paperwidth}%
    {\hbox{\l@dunhbox@line{\one@line}} \new@line}%
    \else%
      \hspace{3\paperwidth}%
    {\new@line \hbox{\l@dunhbox@line{\one@line}}}%
    \fi%
    \vskip-\baselineskip%
}
```

VII.2.4 Hooks

`\do@linehook`
`\do@insidelinehook`

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

```
\newcommand*{\do@linehook}{}
\newcommand*{\do@insidelinehook}{}
```

`\dolinehook`
`\doinsidelinehook`

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

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

```

VII.2.5 Sidenotes and marginal line number initialization

`\l@demptyd@ta`
`\l@dld@ta`
`\l@drd@ta`
`\l@dcsnotetext`
`\l@dcsnotetext@l`
`\l@dcsnotetext@r`

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

```
\newcommand*{\l@demptyd@ta}{%
  \gdef\l@dld@ta{}%
  \gdef\l@drd@ta{}%
  \gdef\l@dcsnotetext@l{}%
  \gdef\l@dcsnotetext@r{}%
  \gdef\l@dcsnotetext{}}

```

\l@dlsn@te \l@drsn@te

Zero width boxes of the left and right side notes, together with their kerns.

```
\newcommand{\l@dlsn@te}{%
  \hb@xt@ \z@{\hss\box\l@dlp@rbox\kern\ledlsnotesep}}
\newcommand{\l@drsn@te}{%
   \hb@xt@ \z@{\kern\ledrsnotesep\box\l@drp@rbox\hss}}

```

\ledllfill \ledrlfill

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

```
\newcommand*{\ledllfill}{\hfil}
\newcommand*{\ledrlfill}{}

```

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.

```
\newcommand*{\getline@num}{%
  \global\advance\absline@num \@ne%
  \do@actions
  \do@ballast
  \ifnumberline
      \ifsublines@
         \ifnum\sub@lock<\tw@
           \global\advance\subline@num \@ne
         \fi
      \else
        \ifnum\@lock<\tw@
           \global\advance\line@num \@ne
           \global\subline@num \z@
        \fi
      \fi
  \fi
}
```

\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. 115).

\ballast@count \c@ballast

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

```
\newcount\ballast@count
\newcounter{ballast}
  \setcounter{ballast}{0}
```

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

```
\newcommand*{\do@ballast}{\global\ballast@count \z@
  \begingroup
    \advance\absline@num \@ne
    \ifnum\next@actionline=\absline@num
      \ifnum\next@action>-1001\relax
        \global\advance\ballast@count by -\c@ballast
       \fi
     \fi
  \endgroup}
```

`\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`.

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

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.

```
    \ifnum\next@action>-1001
       \global\page@num=\next@action
       \ifbypage@
         \global\line@num=\z@ \global\subline@num=\z@
         \resetprevline@
       \fi
```

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

```
    \else
       \ifnum\next@action<-4999
          \@l@dtempcnta=-\next@action
          \advance\@l@dtempcnta by -5001
          \ifsublines@
             \global\subline@num=\@l@dtempcnta
          \else
             \global\line@num=\@l@dtempcnta
          \fi
```

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

```
       \else
```

```
          \@l@dtempcnta=-\next@action
          \advance\@l@dtempcnta by -1000
          \do@actions@fixedcode
        \fi
    \fi
```

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.

```
    \ifx\actionlines@list\empty
         \gdef\next@actionline{1000000}%
    \else
         \gl@p\actionlines@list\to\next@actionline
         \gl@p\actions@list\to\next@action
         \global\let\do@actions@next=\do@actions
    \fi
  \fi
```

Make the recursive call, if necessary.

```
\do@actions@next}

```

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

```
\newcommand*{\do@actions@fixedcode}{%
  \ifcase\@l@dtempcnta
  \or%                    % 1001
    \global\sublines@true
  \or%                    % 1002
    \global\sublines@false
  \or%                    % 1003
      \global\@lock=\@ne
  \or%                    % 1004
    \ifnum\@lock=\tw@
      \global\@lock=\thr@@
    \else
      \global\@lock=\z@
    \fi
  \or%                    % 1005
     \global\sub@lock=\@ne
  \or%                    % 1006
    \ifnum\sub@lock=\tw@
      \global\sub@lock=\thr@@
    \else
      \global\sub@lock=\z@
    \fi
  \or%                    % 1007
    \l@dskipnumbertrue
  \or%                    % 1008
    \l@dskipversenumbertrue%
  \or%    % 1009
    \l@dhidenumbertrue
  \else
    \led@warn@BadAction
  \fi}


```

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:

$$n = int((linenum - firstlinenum)/linenumincrement)$$
$$m = firstlinenum + (n \times linenumincrement)$$

(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.

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

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

```
\ifledgroupnotesL@\else
  \ifnumberline
    \ifl@dskipnumber
      \global\l@dskipnumberfalse
    \else
      \ifsublines@
         \@l@dtempcntb=\subline@num
         \ifnum\subline@num>\c@firstsublinenum
             \@l@dtempcnta=\subline@num
             \advance\@l@dtempcnta by-\c@firstsublinenum
             \divide\@l@dtempcnta by\c@sublinenumincrement
             \multiply\@l@dtempcnta by\c@sublinenumincrement
             \advance\@l@dtempcnta by\c@firstsublinenum
         \else
             \@l@dtempcnta=\c@firstsublinenum
         \fi
```

That takes care of computing the values for comparison, but if line number locking is in effect we have to make a further check. If this check fails, then we disable the line-number display by setting the counters to arbitrary but unequal values.

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

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

```
         \else
            \@l@dtempcntb=\line@num
```

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

```
            \ifx\linenumberlist\empty
               \ifnum\line@num>\c@firstlinenum
                  \@l@dtempcnta=\line@num
                  \advance\@l@dtempcnta by-\c@firstlinenum
                  \divide\@l@dtempcnta by\c@linenumincrement
                  \multiply\@l@dtempcnta by\c@linenumincrement
                  \advance\@l@dtempcnta by\c@firstlinenum
               \else
                  \@l@dtempcnta=\c@firstlinenum
               \fi
            \else
```

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

```
               \@l@dtempcnta=\line@num
               \edef\rem@inder{,\linenumberlist,\number\line@num,}%
               \edef\sc@n@list{\def\noexpand\sc@n@list
            ####1,\number\@l@dtempcnta,####2|{\def\noexpand\rem@inder{####2}}}%
               \sc@n@list\expandafter\sc@n@list\rem@inder|%
               \ifx\rem@inder\empty%
                 \advance\@l@dtempcnta\@ne
               \fi
            \fi
```

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

```
            \ch@ck@l@ck
         \fi
```

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

```
         \ifnum\@l@dtempcnta=\@l@dtempcntb
            \ifl@dskipversenumber\else
```

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

```
              \if@twocolumn
                 \if@firstcolumn
                    \gdef\l@dld@ta{\llap{{\leftlinenum}}}%
                 \else
                    \gdef\l@drd@ta{\rlap{{\rightlinenum}}}%
                 \fi
              \else
                 \@l@dtempcntb=\line@margin
                 \ifnum\@l@dtempcntb>\@ne
                    \advance\@l@dtempcntb \page@num
                 \fi
                 \ifodd\@l@dtempcntb
                    \gdef\l@drd@ta{\rlap{{\rightlinenum}}}%
                  \else
                    \gdef\l@dld@ta{\llap{{\leftlinenum}}}%
                 \fi
              \fi
           \fi
        \fi
```

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

```
        \f@x@l@cks
      \fi
    \fi
  \fi
}

```

\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.

```
\newcommand*{\ch@cksub@l@ck}{%
  \ifcase\sub@lock
    \or
      \ifnum\sublock@disp=\@ne
         \@l@dtempcntb=\z@ \@l@dtempcnta=\@ne
      \fi
    \or
      \ifnum\sublock@disp=\tw@ \else
         \@l@dtempcntb=\z@ \@l@dtempcnta=\@ne
      \fi
    \or
      \ifnum\sublock@disp=\z@
         \@l@dtempcntb=\z@ \@l@dtempcnta=\@ne
      \fi
  \fi}
```

Similarly for line numbers.

```
\newcommand*{\ch@ck@l@ck}{%
    \ifcase\@lock
       \or
         \ifnum\lock@disp=\@ne
            \@l@dtempcntb=\z@ \@l@dtempcnta=\@ne
         \fi
       \or
         \ifnum\lock@disp=\tw@ \else
            \@l@dtempcntb=\z@ \@l@dtempcnta=\@ne
         \fi
       \or
         \ifnum\lock@disp=\z@
            \@l@dtempcntb=\z@ \@l@dtempcnta=\@ne
         \fi
    \fi}
```

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

```
\newcommand*{\f@x@l@cks}{%
  \ifcase\@lock
  \or
    \global\@lock=\tw@
  \or \or
    \global\@lock=\z@
  \fi
  \ifcase\sub@lock
  \or
    \global\sub@lock=\tw@
  \or \or
    \global\sub@lock=\z@
  \fi}

```

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 \rightstartnum commands. After the try, it is set to FALSE.

\leftpstartnum
\rightstartnum
\ifsidepstartnum

```

\newif\ifsidepstartnum
\newcommand*{\affixpstart@num}{%
    \ifsidepstartnum
        \if@twocolumn
            \if@firstcolumn
                  \gdef\l@dld@ta{\llap{{\leftpstartnum}}}%
            \else
                  \gdef\l@drd@ta{\rlap{{\rightpstartnum}}}%
            \fi
        \else
             \@l@dtempcntb=\line@margin
            \ifnum\@l@dtempcntb>\@ne
                  \advance\@l@dtempcntb \page@num
            \fi
            \ifodd\@l@dtempcntb
                  \gdef\l@drd@ta{\rlap{{\rightpstartnum}}}%
            \else
                  \gdef\l@dld@ta{\llap{{\leftpstartnum}}}%
            \fi
        \fi
    \fi

}
%

\newif\ifpstartnum
\pstartnumtrue
\newcommand*{\leftpstartnum}{
    \ifpstartnum\thepstart
    \kern\linenumsep\fi
    \global\pstartnumfalse
}
\newcommand*{\rightpstartnum}{
    \ifpstartnum
    \kern\linenumsep
    \thepstart
    \fi
    \global\pstartnumfalse
}
```

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.

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

`\add@inserts`
`\add@inserts@next`

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

It may call itself recursively, and to do this efficiently (using 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`.

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

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

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

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.

```
  \ifx\next@insert\empty
    \ifx\insertlines@list\empty
      \global\noteschanged@true
      \gdef\next@insert{100000}%
    \else
      \gl@p\insertlines@list\to\next@insert
    \fi
  \fi
```

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

```
  \ifnum\next@insert=\absline@num
    \gl@p\inserts@list\to\@insert
    \@insert
    \global\let\@insert=\undefined
    \global\let\next@insert=\empty
    \global\let\add@inserts@next=\add@inserts
  \fi
\fi
```

Make the recursive call, if necessary.

```
\add@inserts@next}

```

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. 106). Finally, the penalty is checked to see that it does not go below -10000.

```
\newcommand*{\add@penalties}{\@l@dtempcnta=\ballast@count
  \ifnum\num@lines>\@ne
    \global\advance\par@line \@ne
    \ifnum\par@line=\@ne
      \advance\@l@dtempcnta \clubpenalty
    \fi
    \@l@dtempcntb=\par@line \advance\@l@dtempcntb \@ne
    \ifnum\@l@dtempcntb=\num@lines
      \advance\@l@dtempcnta \widowpenalty
    \fi
    \ifnum\par@line<\num@lines
      \advance\@l@dtempcnta \interlinepenalty
    \fi
  \fi
    \ifnum\@l@dtempcnta=\z@
      \relax
    \else
      \ifnum\@l@dtempcnta>-10000
        \penalty\@l@dtempcnta
      \else
        \penalty -10000
      \fi
    \fi}

```

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.

```
\newcommand*{\flush@notes}{%
\@xloop
  \ifx\inserts@list\empty \else
    \gl@p\inserts@list\to\@insert
    \@insert
    \global\let\@insert=\undefined
\repeat}

```

\@xloop

`\@xloop` is a variant of the Plain TeX `\loop` macro, useful when it's hard to construct a positive test using the TeX `\if` commands—as in `\flush@notes` above. One says `\@xloop ... \if ... \else ... \repeat`, and the action following `\else` is repeated as long as the `\if` test fails. (This macro will work wherever the Plain TeX `\loop` is used, too, so we could just call it `\loop`; but it seems preferable not to change the definitions of any of the standard macros.)

This variant of `\loop` was introduced by Alois Kabelschacht in *TUGboat* **8** (1987), pp. 184–5.

```
\def\@xloop#1\repeat{%
  \def\body{#1\expandafter\body\fi}%
  \body}

```

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

`\select@lemmafont` is provided to set the right font for the lemma in a note. This macro extracts the font specifier from the line and page number cluster, and issues the associated font-changing command, so that the lemma is printed in its original font.

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

```

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.

```
\newcommandx*{\footnoteoptions@}[3][1=L,usedefault]{%
   \def\do##1{%
      \ifstrequal{#1}{L}{% In Leftside
      \xright@appenditem{\global\noexpand\settoggle{##1@}{#3}}\to\inserts@list% Switch too
      \global\advance\insert@count \@ne% Increment the left insert counter.
      }%
      {%
       \xright@appenditem{\global\noexpand\settoggle{##1@}{#3}}\to\inserts@listR% Switch to
       \global\advance\insert@countR \@ne% Increment the right insert counter insert.
      }%
   }%
   \notblank{#2}{\docsvlist{#2}}{}% Parsing all options
}
```

XII.3 Notes language

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

```
\newcommandx*{\footnotelang@lua}[1][1=L,usedefault]{%
   \ifstrequal{#1}{L}{%
  \xright@appenditem{{\csxdef{footnote@luatextextdir}{\the\luatextextdir}}}\to\inserts@lis
    \global\advance\insert@count \@ne%
  \xright@appenditem{{\csxdef{footnote@luatexpardir}{\the\luatexpardir}}}\to\inserts@list
    \global\advance\insert@count \@ne%
   }%
   {%
  \xright@appenditem{{\csxdef{footnote@luatextextdir}{\the\luatextextdir}}}\to\inserts@lis
    \global\advance\insert@countR \@ne%
  \xright@appenditem{{\csxdef{footnote@luatexpardir}{\the\luatexpardir}}}\to\inserts@list
    \global\advance\insert@countR \@ne%
   }%
}
```

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

```
\newcommandx*{\footnotelang@poly}[1][1=L,usedefault]{%
   \ifstrequal{#1}{L}{%
    \if@RTL%
      \xright@appenditem{{\csxdef{footnote@dir}{@RTLtrue}}}\to\inserts@list%Know the langua
         \global\advance\insert@count \@ne%
    \else
      \xright@appenditem{{\csxdef{footnote@dir}{@RTLfalse}}}\to\inserts@list%Know the lang
          \global\advance\insert@count \@ne%
    \fi%
   \xright@appenditem{{\csxdef{footnote@lang}{\expandonce\languagename}}}\to\inserts@list%
```

```
    \global\advance\insert@count \@ne%
  }%
  {%
   \if@RTL
     \xright@appenditem{{\csxdef{footnote@dir}{@RTLtrue}}}\to\inserts@listR%Know the language of lemma
        \global\advance\insert@countR \@ne%
   \else
      \xright@appenditem{{\csxdef{footnote@dir}{@RTLfalse}}}\to\inserts@listR%Know the language of lem
         \global\advance\insert@countR \@ne%
   \fi
  \xright@appenditem{{\csxdef{footnote@lang}{\expandonce\languagename}}}\to\inserts@listR%Know the la
   \global\advance\insert@countR \@ne%
  }%
}
```

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 formatting needed for that individual note. Within the output routine, the two other macros, `\footstart` and `\footgroup`, are called; the first prints extra vertical space and a footnote rule, if desired; the second does any reformatting of the whole set of the footnotes in this series for this page—such as paragraphing or division into columns—and then sends them to the page.

These four macros, and the other macros and parameters shown here, are distinguished by the 'series letter' that indicates which set of the footnotes we 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`.

```
\newcommand*{\footsplitskips}{%
  \interlinepenalty=\interfootnotelinepenalty
  \unless\ifl@dprintingpages%
    \floatingpenalty=\@MM%
  \fi%
  \splittopskip=\ht\strutbox \splitmaxdepth=\dp\strutbox
  \leftskip=\z@skip \rightskip=\z@skip}

```

\normalfootnoterule

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

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

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.

```
\newcommandx{\Xarrangement}[2][1,usedefault]{%
  \def\do##1{%
     \csname Xarrangement@#2\endcsname{##1}%
  }%
  \ifstrempty{#1}%
    {%
    \dolistloop{\@series}%
    }%
    {
    \docsvlist{#1}%
    }%
}%
```

XII.6.2 Normal footnote

\Xarrangement@normal

We can now define all the parameters for the series of footnotes; initially they use the "normal" footnote formatting.

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

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

(Read *The TeXbook* in order to understand what are the counter, skip and dimen associated to an insertion.)

Instead of repeating ourselves, we define a `\Xarrangement@normal` macro that makes all these assignments for us, for any given series letter. This command is called when people use `\Xarrangement[`⟨*series*⟩`]{normal}`

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

```
\newcommand*{\Xarrangement@normal}[1]{%
  \csgdef{series@display#1}{normal}
  \expandafter\let\csname #1footstart\endcsname=\normalfootstart
  \expandafter\let\csname v#1footnote\endcsname=\normalvfootnote
  \expandafter\let\csname #1footfmt\endcsname=\normalfootfmt
  \expandafter\let\csname #1footgroup\endcsname=\normalfootgroup
  \expandafter\let\csname #1footnoterule\endcsname=%
                                               \normalfootnoterule
  \count\csname #1footins\endcsname=1000
  \dimen\csname #1footins\endcsname=\csuse{Xmaxhnotes@#1}
  \skip\csname #1footins\endcsname=\csuse{Xbeforenotes@#1}%
  \advance\skip\csname #1footins\endcsname by\csuse{Xafterrule@#1}%
```

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.

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

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).

```
  \ifnoledgroup@\else%
    \expandafter\let\csname mpv#1footnote\endcsname=\mpnormalvfootnote
    \expandafter\let\csname mp#1footgroup\endcsname=\mpnormalfootgroup
    \count\csname mp#1footins\endcsname=1000
    \dimen\csname mp#1footins\endcsname=\csuse{Xmaxhnotes@#1}
    \skip\csname mp#1footins\endcsname=\csuse{Xbeforenotes@#1}%
    \advance\skip\csname mp#1footins\endcsname by\csuse{Xafterrule@#1}%
  \fi
}

```

`\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.

```
\notbool{parapparatus@}{\newcommand*}{\newcommand}{\normalvfootnote}[2]{%
  \insert\csname #1footins\endcsname\bgroup
  \csuse{Xbhooknote@#1}
  \csuse{Xnotefontsize@#1}
  \footsplitskips
  \ifl@dpairing\ifl@dpaging\else%
    \setXnoteswidthliketwocolumns@{#1}%
  \fi\fi%
  \setXnotespositionliketwocolumns@{#1}%
  \spaceskip=\z@skip \xspaceskip=\z@skip
  \csname #1footfmt\endcsname #2[#1]\egroup}
```

\mpnormalvfootnote

And a somewhat different version for minipages.

```
\notbool{parapparatus@}{\newcommand*}{\newcommand}{\mpnormalvfootnote}[2]{%
  \global\setbox\@nameuse{mp#1footins}\vbox{%
    \unvbox\@nameuse{mp#1footins}
    \csuse{Xbhooknote@#1}
    \csuse{Xnotefontsize@#1}
    \hsize\columnwidth
    \@parboxrestore
    \color@begingroup
    \csname #1footfmt\endcsname #2[#1]\color@endgroup}}

```

\normalfootfmt

\normalfootfmt is a 'normal' macro to take the footnote line and page number information (see V.9 p. 69), 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.

```


\notbool{parapparatus@}{\newcommandx*}{\newcommandx}{\normalfootfmt}[4][4=Z]{% 4th arg is c
  \Xledsetnormalparstuff{#4}%
  \hangindent=\csuse{Xhangindent@#4}
  \strut{\printlinefootnote{#1}{#4}}%
 {\nottoggle{Xlemmadisablefontselection@#4}{\select@lemmafont#1|#2}{#2}}%
 \iftoggle{nosep@}{\hskip\csuse{Xinplaceoflemmaseparator@#4}}{\ifcsempty{Xlemmaseparator@
    {\hskip\csuse{Xinplaceoflemmaseparator@#4}}%
  {\nobreak\hskip\csuse{Xbeforelemmaseparator@#4}\csuse{Xlemmaseparator@#4}\hskip\csuse{X
 }}%
  #3\strut\par}
```

\normalfootstart

\normalfootstart is a standard footnote-starting macro, called in the output routine whenever there are footnotes of this series to be printed: it skips a bit and then draws a rule.

Any \footstart macro must put onto the page something that takes up space exactly equal to the \skip\Xfootins value for the associated series of notes. TEX makes page computations based on that \skip value, and the output pages will suffer from spacing problems if what you add takes up a different amount of space.

But if the skip \preXnotes@ is greater than 0 pt, it is used instead of \skip\footins for the first printed series in one page.

The \leftskip and \rightskip values are both zeroed here. Similarly, these skips are cancelled in the \vfootnote macros for the various types of notes. Strictly speaking, this is necessary only if you are using paragraphed footnotes, but we have put it here and in the other \vfootnote macros too so that the behavior of 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`.

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

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

```
  \ifdimequal{0pt}{\preXnotes@}{}%
      {%
      \iftoggle{preXnotes@}{%
            \togglefalse{preXnotes@}%
            \skip\csname #1footins\endcsname=%
              \dimexpr\csuse{preXnotes@}+\csuse{Xafterrule@#1}\relax%
            }%
          {}%
      }%
  \vskip\skip\csname #1footins\endcsname%
```

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. 156 for the general description of the problem.

```
  \leftskip0pt \rightskip0pt
  \ifl@dpairing\else%
     \hsize=\old@hsize%
  \fi%
  \setXnoteswidthliketwocolumns@{#1}%
  \setXnotespositionliketwocolumns@{#1}%
%    \end{macrocode}
% And now, print the footnote's rule to finish the footnote's introduction.
%    \begin{macrocode}
  \print@Xfootnoterule{#1}%
  \noindent\leavevmode}
```

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

```
\newcommand*{\normalfootgroup}[1]{%
  {\csuse{Xnotefontsize@#1}\noindent\csuse{Xtxtbeforenotes@#1}}%
  \unvbox\csname #1footins\endcsname%
  \hsize=\old@hsize%
  }%

```

`\mpnormalfootgroup` A somewhat different version for minipages. Notes that, in this case, we don not make distinction between `\Xfootgroup` and `\Xfootstarts` macro.

```
\unless\ifnoledgroup@
\newcommand*{\mpnormalfootgroup}[1]{{
  \vskip\skip\@nameuse{mp#1footins}
  \ifl@dpairing\ifparledgroup%
    \leavevmode\marks\parledgroup@{begin}%
    \marks\parledgroup@series{#1}%
    \marks\parledgroup@type{Xfootnote}%
  \fi\fi\normalcolor%
  \ifparledgroup%
    \ifl@dpairing%
    \else%
      \setXnoteswidthliketwocolumns@{#1}%
      \setXnotespositionliketwocolumns@{#1}%
      \print@Xfootnoterule{#1}%%
    \fi%
  \else%
    \setXnoteswidthliketwocolumns@{#1}%
    \setXnotespositionliketwocolumns@{#1}%
    \print@Xfootnoterule{#1}%%
  \fi%
  \setlength{\parindent}{0pt}
 {\csuse{Xnotefontsize@#1}\csuse{Xtxtbeforenotes@#1}}
 \unvbox\csname mp#1footins\endcsname}}
\fi
```

XII.6.3 Paragraphed footnotes

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

\Xarrangement@paragraph

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

The argument of \Xarrangement@footparagraph is the letter denoting the series of notes to be paragraphed.

```
\newcommand*{\Xarrangement@paragraph}[1]{%
  \csgdef{series@display#1}{paragraph}
  \expandafter\newcount\csname #1prevpage@num\endcsname
  \expandafter\let\csname #1footstart\endcsname=\parafootstart
  \expandafter\let\csname v#1footnote\endcsname=\paravfootnote
  \expandafter\let\csname #1footfmt\endcsname=\parafootfmt
  \expandafter\let\csname #1footgroup\endcsname=\parafootgroup
  \count\csname #1footins\endcsname=1000
 \csxdef{default@#1footins}{1000}%Use this to confine the  notes to one side only
  \dimen\csname #1footins\endcsname=\csuse{Xmaxhnotes@#1}
  \skip\csname #1footins\endcsname=\csuse{Xbeforenotes@#1}%
  \advance\skip\csname #1footins\endcsname by\csuse{Xafterrule@#1}%
  \para@footsetup{#1}
```

And the extra setup for minipages.

```
  \ifnoledgroup@\else
    \expandafter\let\csname mpv#1footnote\endcsname=\mpparavfootnote
    \expandafter\let\csname mp#1footgroup\endcsname=\mpparafootgroup
    \count\csname mp#1footins\endcsname=1000
    \dimen\csname mp#1footins\endcsname=\csuse{Xmaxhnotes@#1}
    \skip\csname mp#1footins\endcsname=\csuse{Xbeforenotes@#1}%
    \advance\skip\csname mp#1footins\endcsname by\csuse{Xafterrule@#1}%
  \fi
}
```

`\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.

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

`\para@footsetup`

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

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

```

`\strip@pt` strip the characters pt from a dimen value.

`\parafootstart`

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

```
\newcommand*{\parafootstart}[1]{%
  \rightskip=0pt \leftskip=0pt \parindent=0pt
    \ifdimequal{0pt}{\preXnotes@}{}%
      {%
      \iftoggle{preXnotes@}{%
            \togglefalse{preXnotes@}%
            \skip\csname #1footins\endcsname=%
            \dimexpr\csuse{preXnotes@}+\csuse{Xafterrule@#1}\relax%
          }%
        {}%
    }%
  \vskip\skip\csname #1footins\endcsname%
  \setXnoteswidthliketwocolumns@{#1}%
  \setXnotespositionliketwocolumns@{#1}%
  \print@Xfootnoterule{#1}%%
  \noindent\leavevmode}
```

\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 `\discretionary`s. If you later unbox these hboxes and stick them together, as the *TeXbook* macros do to make these footnotes, you lose the ability to hyphenate after an explicit hyphen. This can lead to overfull `\hbox`es when you would not expect to find them, and to the uninitiated it might be very hard to see why the problem had arisen.[27]

Wayne Sullivan pointed out to us another subtle problem that arises from the same cause: TEX also leaves the `\language` whatsit nodes out of the horizontal list.[28] So changes from one language to another will not invoke the proper hyphenation rules in such footnotes. Since critical editions often do deal with several languages, especially in a footnotes, we really ought to get this bit of code right.

To get around these problems, Wayne suggested emendations to the *TeXbook* versions of these macros which are broadly the same as those described by Michael: the central idea (also suggested by Donald Knuth in a letter to Michael) is to avoid collecting the text in an `\hbox` in the first place, but instead to collect it in a `\vbox` whose width is (virtually) infinite. The text is therefore typeset in unrestricted horizontal mode, as a paragraph consisting of a single long line. Later, there is an extra level of unboxing to be done: we have to unpack the `\vbox`, as well as the hboxes inside it, but that is not too hard. For details, we refer you to Michael's article, where the issues are clearly explained.[29] Michael's unboxing macro is called `\Xunvxh`: unvbox, extract the last line, and unhbox it.

Doing things this way has an important consequence: as Michael pointed out, you really can't put an explicit line-break into a note built in a `\vbox` the way we are doing.[30]

[27] Michael Downes, 'Line Breaking in `\unhbox`ed Text', *TUGboat* **11** (1990), pp. 605–612.

[28] See *The TeXbook*, p. 455 (editions after January 1990).

[29] Wayne supplied his own macros to do this, but since they were almost identical to Michael's, Peter Wilson have used the latter's `\Xunvxh` macro since it is publicly documented.

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

In other words, be very careful not to say `\break`, or `\penalty-10000`, or any equivalent inside your para-footnote. If you do, most of the note will probably disappear. You *are* allowed to make strong suggestions; in fact `\penalty-9999` will be quite okay. Just do not make the break mandatory. We have not applied any of Michael's solutions here, since we feel that the problem is exiguous, and reledmac is quite baroque enough already. If you think you are having this problem, look up Michael's solutions.

One more thing; we set `\leftskip` and `\rightskip` to zero. This has the effect of neutralizing any such skips which may apply to the main text (cf. XII.6.2 p. 121 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.

```
\newcommand*{\paravfootnote}[2]{%
  \insert\csname #1footins\endcsname
  \bgroup
    \csuse{Xbhooknote@#1}
    \csuse{Xnotefontsize@#1}
    \footsplitskips
    \setbox0=\vbox{\hsize=\maxdimen
      \noindent\csname #1footfmt\endcsname#2[#1]}%
    \setbox0=\hbox{\Xunvxh0[#1]}%
    \dp0=0pt
    \ht0=\csname #1footfudgefactor\endcsname\wd0
```

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

```
    \if@RTL\noindent \leavevmode\fi\box0%
    \penalty0
  \egroup}

```

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.

```
\newcommand*{\mpparavfootnote}[2]{%
  \global\setbox\@nameuse{mp#1footins}\vbox{%
    \unvbox\@nameuse{mp#1footins}%
    \csuse{Xbhooknote@#1}
    \csuse{Xnotefontsize@#1}
    \footsplitskips
    \setbox0=\vbox{\hsize=\maxdimen
    \noindent\color@begingroup\csname #1footfmt\endcsname #2[#1]\color@endgroup}%
    \setbox0=\hbox{\Xunvxh0[#1]}%
    \dp0=\z@
    \ht0=\csname #1footfudgefactor\endcsname\wd0
    \box0
    \penalty0
}}

```

`\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`.

```
\newcommandx*{\Xunvxh}[2][2=Z]{% 2th is optional for retro-compatibility
  \setbox0=\vbox{\unvbox#1%
    \global\setbox1=\lastbox}%
  \unhbox1
  \unskip             % remove \rightskip,
  \unskip             % remove \parfillskip,
  \unpenalty          % remove \penalty of 10000,
  \hskip\csuse{Xafternote@#2}}   % but add the glue to go between the notes

```

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

```
\newcommandx*{\parafootfmt}[4][4=Z]{%
  \Xinsertparafootsep{#4}%
  \Xledsetnormalparstuff{#4}%
  \printlinefootnote{#1}{#4}%
 {\nottoggle{Xlemmadisablefontselection@#4}{\select@lemmafont#1|#2}{#2}}%
 \iftoggle{nosep@}{\hskip\csuse{Xinplaceoflemmaseparator@#4}}{\ifcsempty{Xlemmaseparator@
    {\hskip\csuse{Xinplaceoflemmaseparator@#4}}%
  {\nobreak\hskip\csuse{Xbeforelemmaseparator@#4}\csuse{Xlemmaseparator@#4}\hskip\csuse{X
  }}%
  #3\penalty-10 }
```

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

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

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

```
\newcommand*{\parafootgroup}[1]{%
  \unvbox\csname #1footins\endcsname
  \ifcsstring{Xragged@#1}{L}{\RaggedLeft}{}%
  \ifcsstring{Xragged@#1}{R}{\RaggedRight}{}%
  \makehboxofhboxes
  \setbox0=\hbox{{\csuse{Xnotefontsize@#1}\csuse{Xtxtbeforenotes@#1}}\unhbox0 \removehboxes}%
  \csuse{Xnotefontsize@#1}
  \noindent\unhbox0\par%
  \global\hsize=\old@hsize%
  }%

```

`\mpparafootgroup`

The minipage version.

```
\newcommand*{\mpparafootgroup}[1]{{%
  \setXnoteswidthliketwocolumns@{#1}%
  \vskip\skip\@nameuse{mp#1footins}
  \ifl@dpairing\ifparledgroup%
    \leavevmode\marks\parledgroup@{begin}%
    \marks\parledgroup@series{#1}%
    \marks\parledgroup@type{Xfootnote}%
  \fi\fi\normalcolor
  \ifparledgroup%
    \ifl@dpairing%
    \else%
      \setXnoteswidthliketwocolumns@{#1}%
      \setXnotespositionliketwocolumns@{#1}%
      \print@Xfootnoterule{#1}%%
    \fi%
  \else%
      \setXnoteswidthliketwocolumns@{#1}%
      \setXnotespositionliketwocolumns@{#1}%
      \print@Xfootnoterule{#1}%
  \fi%
  \unvbox\csname mp#1footins\endcsname
  \ifcsstring{Xragged@#1}{L}{\RaggedLeft}{}%
  \ifcsstring{Xragged@#1}{R}{\RaggedRight}{}%
  \makehboxofhboxes
  \setbox0=\hbox{{\csuse{Xnotefontsize@#1}\csuse{Xtxtbeforenotes@#1}}\unhbox0 \removehboxes}%
  \csuse{Xnotefontsize@#1}
  \noindent\unhbox0\par}}

```

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

`\makehboxofhboxes`
`\removehboxes`

```
\newcommand*{\makehboxofhboxes}{\setbox0=\hbox{}%
  \loop
    \unpenalty
    \setbox2=\lastbox
  \ifhbox2
    \setbox0=\hbox{\box2\unhbox0}%
  \repeat}

\newcommand*{\removehboxes}{\setbox0=\lastbox
  \ifhbox0{\removehboxes}\unhbox0 \fi}

```

Insertion of the footnotes separator The command `\Xinsertparafootsep{`⟨*series*⟩`}` must be called at the beginning of `\parafootftm`.

`\prevpage@num`
`\Xinsertparafootsep`

```
\newcommand{\Xinsertparafootsep}[1]{%
   \ifnumequal{\csuse{#1prevpage@num}}{\page@num}%
        {\ifcsdef{prevline#1}% Be sur \prevline#1 exists.
      {\ifnumequal{\csuse{prevline#1}}{\line@num}%
         {\ifcsempty{Xsymlinenum@#1}{\csuse{Xparafootsep@#1}}{}}%
         {\csuse{Xparafootsep@#1}}%
      }%
      {\csuse{Xparafootsep@#1}}%
   }%
   {}%
   \global\csname #1prevpage@num\endcsname=\page@num%
}
```

XII.6.4 Columnar footnotes

Common tools

`\rigidbalance`
`\dosplits`
`\splitoff`
`\@h`
`\@k`

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

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

```
\newcount\@k \newdimen\@h
\newcommand*{\rigidbalance}[3]{\setbox0=\box#1 \@k=#2 \@h=#3
  \@@line{\splittopskip=\@h \vbadness=\@M \hfilneg
  \valign{##\vfil\cr\dosplits}}}

\newcommand*{\dosplits}{\ifnum\@k>0 \noalign{\hfil}\splitoff
  \global\advance\@k-1\cr\dosplits\fi}
```

```

\newcommand*{\splitoff}{\dimen0=\ht0
  \divide\dimen0 by\@k \advance\dimen0 by\@h
  \setbox2 \vsplit0 to \dimen0
  \unvbox2 }

```

Three columns

\Xarrangement@threecol

```
\newcommand*{\Xarrangement@threecol}[1]{%
  \csgdef{series@display#1}{threecol}
  \expandafter\let\csname v#1footnote\endcsname=\threecolvfootnote
  \expandafter\let\csname #1footfmt\endcsname=\threecolfootfmt
  \expandafter\let\csname #1footgroup\endcsname=\threecolfootgroup
  \dimen\csname #1footins\endcsname=\csuse{Xmaxhnotes@#1}%
  \skip\csname #1footins\endcsname=\csuse{Xbeforenotes@#1}%
  \advance\skip\csname #1footins\endcsname by\csuse{Xafterrule@#1}%
  \threecolfootsetup{#1}
```

The additional setup for minipages.

```
  \ifnoledgroup@\else
    \expandafter\let\csname mpv#1footnote\endcsname=\mpnormalvfootnote
    \expandafter\let\csname mp#1footgroup\endcsname=\mpthreecolfootgroup
    \skip\csname mp#1footins\endcsname=\csuse{Xbeforenotes@#1}%
    \advance\skip\csname mp#1footins\endcsname by\csuse{Xafterrule@#1}%
    \mpthreecolfootsetup{#1}
  \fi
}

```

The \footstart and \footnoterule macros for these notes assume the normal values (XII.6.2 p. 121 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.

```
\newcommand*{\threecolfootsetup}[1]{%
  \count\csname #1footins\endcsname 333
```

```
 \csxdef{default@#1footins}{333}%Use this to confine the  notes to one side only
 \multiply\dimen\csname #1footins\endcsname \thr@@}
```

`\mpthreecolfootsetup` The setup for minipages.

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

```

`\threecolvfootnote` `\threecolvfootnote` is the `\vfootnote` command for three-column notes. The call to `\Xnotefontsize@`$\langle s\rangle$ 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).

```
\notbool{parapparatus@}{\newcommand*}{\newcommand}{\threecolvfootnote}[2]{%
 \insert\csname #1footins\endcsname\bgroup
 \csuse{Xnotefontsize@#1}
 \footsplitskips
 \csname #1footfmt\endcsname #2[#1]\egroup}
```

`\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.

```
\notbool{parapparatus@}{\newcommandx*}{\newcommandx}{\threecolfootfmt}[4][4=Z]{%
 \normal@pars
 \hsize \csuse{Xhsizethreecol@#4}
 \nottoggle{Xparindent@#4}{\parindent=\z@}{}
 \tolerance=5000
 \hangindent=\csuse{Xhangindent@#4}
 \leavevmode
 \csuse{Xcolalign@#4}%
 \strut{\printlinefootnote{#1}{#4}}%
{\nottoggle{Xlemmadisablefontselection@#4}{\select@lemmafont#1|#2}{#2}}%
\iftoggle{nosep@}{\hskip\csuse{Xinplaceoflemmaseparator@#4}}{\ifcsempty{Xlemmaseparator@
   {\hskip\csuse{Xinplaceoflemmaseparator@#4}}%
  {\nobreak\hskip\csuse{Xbeforelemmaseparator@#4}\csuse{Xlemmaseparator@#4}\hskip\csuse{X
 }}%
 #3\strut\par\allowbreak}
```

`\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@`$\langle s\rangle$ 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.

```
\newcommand*{\threecolfootgroup}[1]{{\csuse{Xnotefontsize@#1}%
  \noindent\csuse{Xtxtbeforenotes@#1}\par%
  \splittopskip=\ht\strutbox
  \expandafter
  \rigidbalance\csname #1footins\endcsname \thr@@ \splittopskip}}
```

`\mpthreecolfootgroup` The setup for minipages.

```
\newcommand*{\mpthreecolfootgroup}[1]{{%
  \vskip\skip\@nameuse{mp#1footins}
  \ifl@dpairing\ifparledgroup%
    \leavevmode\marks\parledgroup@{begin}%
    \marks\parledgroup@series{#1}%
    \marks\parledgroup@type{Xfootnote}%
  \fi\fi\normalcolor
  \ifparledgroup%
    \ifl@dpairing%
    \else%
      \setXnoteswidthliketwocolumns@{#1}%
      \setXnotespositionliketwocolumns@{#1}%
      \print@Xfootnoterule{#1}%
    \fi%
  \else%
    \setXnoteswidthliketwocolumns@{#1}%
    \setXnotespositionliketwocolumns@{#1}%
    \print@Xfootnoterule{#1}%
  \fi%
  {\csuse{Xnotefontsize@#1}\noindent\csuse{Xtxtbeforenotes@#1}}\par
  \splittopskip=\ht\strutbox
  \expandafter
  \rigidbalance\csname mp#1footins\endcsname \thr@@ \splittopskip}}

```

Two columns

`\Xarrangement@twocol`

```
\newcommand*{\Xarrangement@twocol}[1]{%
  \csgdef{series@display#1}{twocol}
  \expandafter\let\csname v#1footnote\endcsname=\twocolvfootnote
  \expandafter\let\csname #1footfmt\endcsname=\twocolfootfmt
  \expandafter\let\csname #1footgroup\endcsname=\twocolfootgroup
  \dimen\csname #1footins\endcsname=\csuse{Xmaxhnotes@#1}%
  \skip\csname #1footins\endcsname=\csuse{Xbeforenotes@#1}%
  \advance\skip\csname #1footins\endcsname by\csuse{Xafterrule@#1}%
  \twocolfootsetup{#1}
```

The additional setup for minipages.

```
  \ifnoledgroup@\else
    \expandafter\let\csname mpv#1footnote\endcsname=\mpnormalvfootnote
    \expandafter\let\csname mp#1footgroup\endcsname=\mptwocolfootgroup
    \skip\csname mp#1footins\endcsname=\csuse{Xbeforenotes@#1}%
    \advance\skip\csname mp#1footins\endcsname by\csuse{Xafterrule@#1}%
    \mptwocolfootsetup{#1}
  \fi
}

```

\twocolfootsetup
\twocolvfootnote
\twocolfootfmt
\twocolfootgroup

Here is a series of macros which are very similar to their three-column counterparts. In this case, each note is assumed to contribute only a half a line of text. And the notes are set in columns giving a gap between them of one tenth of the `\hsize`.

```
\newcommand*{\twocolfootsetup}[1]{%
  \count\csname #1footins\endcsname 500
 \csxdef{default@#1footins}{500}%Use this to confine the  notes to one side only
  \multiply\dimen\csname #1footins\endcsname \tw@}
\notbool{parapparatus@}{\newcommand*}{\newcommand}{\twocolvfootnote}[2]{\insert\csname #1f
  \csuse{Xnotefontsize@#1}
  \footsplitskips
  \csname #1footfmt\endcsname #2[#1]\egroup}
\notbool{parapparatus@}{\newcommandx*}{\newcommandx}{\twocolfootfmt}[4][4=Z]{% 4th arg is c
  \normal@pars
  \hsize \csuse{Xhsizetwocol@#4}
  \nottoggle{Xparindent@#4}{\parindent=\z@}{}
  \tolerance=5000
  \hangindent=\csuse{Xhangindent@#4}
  \leavevmode
  \csuse{Xcolalign@#4}%
  \strut{\printlinefootnote{#1}{#4}}%
 {\nottoggle{Xlemmadisablefontselection@#4}{\select@lemmafont#1|#2}{#2}}%
 \iftoggle{nosep@}{\hskip\csuse{Xinplaceoflemmaseparator@#4}}{\ifcsempty{Xlemmaseparator@
    {\hskip\csuse{Xinplaceoflemmaseparator@#4}}%
  {\nobreak\hskip\csuse{Xbeforelemmaseparator@#4}\csuse{Xlemmaseparator@#4}\hskip\csuse{X
  }}%
  #3\strut\par\allowbreak}
\newcommand*{\twocolfootgroup}[1]{{\csuse{Xnotefontsize@#1}
  \noindent\csuse{Xtxtbeforenotes@#1}\par%
  \splittopskip=\ht\strutbox
  \expandafter
  \rigidbalance\csname #1footins\endcsname \tw@ \splittopskip}}

```

\mptwocolfootsetup
\mptwocolfootgroup

The versions for minipages.

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

```
\newcommand*{\mptwocolfootgroup}[1]{{%
  \vskip\skip\@nameuse{mp#1footins}
  \ifl@dpairing\ifparledgroup%
    \leavevmode\marks\parledgroup@{begin}%
    \marks\parledgroup@series{#1}%
    \marks\parledgroup@type{Xfootnote}%
  \fi\fi\normalcolor
  \ifparledgroup%
    \ifl@dpairing%
    \else%
      \setXnoteswidthliketwocolumns@{#1}%
      \setXnotespositionliketwocolumns@{#1}%
      \print@Xfootnoterule{#1}%
    \fi%
  \else%
    \setXnoteswidthliketwocolumns@{#1}%
    \setXnotespositionliketwocolumns@{#1}%
    \print@Xfootnoterule{#1}%
  \fi%
   {\csuse{Xnotefontsize@#1}\noindent\csuse{Xtxtbeforenotes@#1}}\par
  \splittopskip=\ht\strutbox
  \expandafter
  \rigidbalance\csname mp#1footins\endcsname \tw@ \splittopskip}}

```

XII.7 Critical notes presentation

Here, we define some commons macro which are used in order to print a critical notes, that is a note with 1) line number 2) lemma 3) lemma separator 4) text associated to the lemma.

XII.7.1 Font tools

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

The fonts that are used for printing notes might not have the character mapping we expect: for example, the Computer Modern font that contains old-style numerals does not contain an en-dash or square brackets, and its period and comma are in odd locations. To allow use of the standard footnote macros with such fonts, we use the following macros for certain characters.

The `\endashchar` macro is simply an en-dash from the normal font and is immune to changes in the surrounding font. The same goes for the full stop. These two are used in `\printlines`. The right bracket macro is the same again; it crops up in `\normalfootfmt` and the other footnote macros for controlling the format of the footnotes.

With polyglossia, each critical note has a `\footnote@lang` which shows the language of the lemma, and which can be used to switch the bracket from right to left.

```
\def\endashchar{\textnormal{--}}
\newcommand*{\fullstop}{\textnormal{.}}
```

```
\newcommand*{\rbracket}{\textnormal{%
   \csuse{text\csuse{footnote@lang}}{%
         \ifluatex%
      \ifdefstring{\footnote@luatextextdir}{TRT}{\thinspace[}{\thinspace]}%
            \else%
         \thinspace]%
         \fi}%
   }%
}

```

XII.7.2 Pstart number in footnote

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

```
\newcommand{\printpstart}[0]{%
   \ifboolexpr{bool{l@dpairing} or bool{l@dprintingpages} or bool{l@dprintingcolumns}}{%
        \ifledRcol%
            \thepstartR%
        \else%
            \thepstartL%
        \fi%
    }{%
        \thepstart%
    }%
}
```

XII.7.3 Line number printing

\printlinefootnote The `\printlinefootnote` macro is called in each `\<type>footfmt` command. It controls whether the line number is printed or not, according to the series options. Its first argument is the information about lines; its second is the series of the footnote. The printing of the line number is shared in `\printlinefootnotenumbers`.

```
\newcommand{\printlinefootnote}[2]{%
   \def\extractline@##1|##2|##3|##4|##5|##6|##7|{##2}%
   \def\extractsubline@##1|##2|##3|##4|##5|##6|##7|{##3}%
   \def\extractendline@##1|##2|##3|##4|##5|##6|##7|{##5}%
   \def\extractendsubline@##1|##2|##3|##4|##5|##6|##7|{##6}%
   \iftoggle{XnumberonlyfirstinXtwolines@#2}{%
     \edef\lineinfo@{\extractline@ #1| - \extractsubline@ #1| - \extractendline@ #1| - \extra
       }%
       {%
       \edef\lineinfo@{\extractline@ #1| - \extractsubline@ #1|}%
       }%
  \iftoggle{nonum@}{%Try if the line number must printed for this specific not (by default, ye
   \hspace{\csuse{Xinplaceofnumber@#2}}%
   }%
   {%
       {%
       \iftoggle{Xnonumber@#2}%Try if the line number must printed (by default, yes)
```

```
           {%
           \hspace{\csuse{Xinplaceofnumber@#2}}%
           }%
           {%
         {\iftoggle{Xnumberonlyfirstinline@#2}% If for this series the line number must be printed only i
                 {%
                 \ifcsdef{prevline#2}%
                    {%Be sure the \prevline exists.
                    \ifcsequal{prevline#2}{lineinfo@}%Try it
                       {%
                       \ifcsempty{Xsymlinenum@#2}%
                        {%
                            \hspace{\csuse{Xinplaceofnumber@#2}}%
                        }%
                {\hspace{\csuse{Xbeforesymlinenum@#2}}\csuse{Xnotenumfont@#2}%
                        \ifdimequal{\csuse{Xboxsymlinenum@#2}}{0pt}%
                            {\csuse{Xsymlinenum@#2}}%
                  {\hbox to \csuse{Xboxsymlinenum@#2}{\csuse{Xsymlinenum@#2}\hfill}}%
                        \hspace{\csuse{Xaftersymlinenum@#2}}}%
                       }%
                       {%
                        \printlinefootnotearea{#1}{#2}%
                        }%
                    }%
                    {%
                    \printlinefootnotearea{#1}{#2}%
                    }%
              }%
              {%
              \printlinefootnotearea{#1}{#2}%
              }%
              \csxdef{prevline#2}{\lineinfo@}%
           }%
        }%
     }%
   }%
}
```

\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.)

```
\newcommand{\printlinefootnotearea}[2]{%
  \printXbeforenumber{#2}%
  \csuse{Xnotenumfont@#2}%
  \boxfootnotenumbers{#1}{#2}%
  \printXafternumber{#2}%
}%
```

\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.

```
\newcommand{\boxfootnotenumbers}[2]{%
  \ifdimequal{\csuse{Xboxlinenum@#2}}{0pt}{%
      \printlinefootnotenumbers{#1}{#2}%
    }%
    {%
      \hbox to \csuse{Xboxlinenum@#2}%
        {%
        \IfSubStr{RC}{\csuse{Xboxlinenumalign@#2}}{\hfill}{}%
        \printlinefootnotenumbers{#1}{#2}%
        \IfSubStr{LC}{\csuse{Xboxlinenumalign@#2}}{\hfill}{}%
        }%
    }%
}%
```

`\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.

```
\newcommand{\printlinefootnotenumbers}[2]{%
  \xdef\@currentseries{#2}%
  \ifboolexpr{%
   (togl{Xpstart@#2} and bool{numberpstart})%
   or togl{Xpstarteverytime@#2}}%
  {\printpstart}{}%
  \iftoggle{Xonlypstart@#2}{}{\printlines#1|}%
}%
```

`\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.)

```
\newcommand{\printXbeforenumber}[1]{%
  \hspace{\csuse{Xbeforenumber@#1}}%
}%
```

`\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

```
\newcommand{\printXafternumber}[1]{%
  \iftoggle{Xnonbreakableafternumber@#1}{\nobreak}{}%
  \hspace{\csuse{Xafternumber@#1}}%
}%
```

If we have decided to print the line number in a specific notes, the `\printlines` macro prints the line numbers for a note—which, in the general case, is a rather complicated task. The seven parameters of the argument are the line numbers as stored in `\l@d@nums`, in the form described on V.9 p. 69: 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` `\ifl@d@ssub` `\ifl@d@elin` `\ifl@d@esl` `\ifl@d@dash` `\ifl@d@Xtwolines` `\ifl@d@Xmorethantwolines`

```
\newif\ifl@d@pnum
\newif\ifl@d@ssub
\newif\ifl@d@elin
\newif\ifl@d@esl
\newif\ifl@d@dash
\newif\ifl@d@Xtwolines%
\newif\ifl@d@Xmorethantwolines%
```

`\l@dparsefootspec` `\l@dp@rsefootspec` `\l@dparsedstartpage` `\l@dparsedstartline` `\l@dparsedstartsub` `\l@dparsedendpage` `\l@dparsedendline` `\l@dparsedendsub`

`\l@dparsefootspec{`⟨*spec*⟩`}{`⟨*lemma*⟩`}{`⟨*text*⟩`}` parses a footnote specification. ⟨*lemma*⟩ and ⟨*text*⟩ are the lemma and text respectively. ⟨*spec*⟩ is the line and page number and lemma font specifier in `\l@d@nums` style format. The real work is done by `\l@dp@rsefootspec` which defines macros holding the numeric values. Just a reminder of the arguments:

```
\printlines    #1      | #2 |  #3    |    #4    | #5 | #6     |   #7
\printlines start-page | line | subline | end-page | line | subline | font
```

```
\newcommand*{\l@dparsefootspec}[3]{\l@dp@rsefootspec#1|}
\def\l@dp@rsefootspec#1|#2|#3|#4|#5|#6|#7|{%
  \gdef\l@dparsedstartpage{#1}%
  \gdef\l@dparsedstartline{#2}%
  \gdef\l@dparsedstartsub{#3}%
  \gdef\l@dparsedendpage{#4}%
  \gdef\l@dparsedendline{#5}%
  \gdef\l@dparsedendsub{#6}%
}
```

Initialise the several number value macros.

```
\def\l@dparsedstartpage{0}%
\def\l@dparsedstartline{0}%
\def\l@dparsedstartsub{0}%
\def\l@dparsedendpage{0}%
\def\l@dparsedendline{0}%
\def\l@dparsedendsub{0}%

```

\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.

```
\newcommand*{\setprintlines}[6]{%
  \l@d@pnumfalse \l@d@dashfalse
```

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

```
  \ifbypage@
     \ifnum#4=#1 \else
        \l@d@pnumtrue
        \l@d@dashtrue
     \fi
  \fi
```

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.

```
  \ifl@d@pnum \l@d@elintrue \else \l@d@elinfalse \fi
  \ifnum#2=#5 \else
      \l@d@elintrue
      \l@d@dashtrue
  \fi
```

We print the starting sub-line if it is nonzero.

```
  \l@d@ssubfalse
  \ifnum#3=0 \else
      \l@d@ssubtrue
  \fi
```

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.

```
  \l@d@eslfalse
  \ifnum#6=0 \else
      \ifnum#6=#3
         \ifl@d@elin \l@d@esltrue \else \l@d@eslfalse \fi
      \else
         \l@d@esltrue
         \l@d@dashtrue
      \fi
  \fi%
```

However, if the \Xtwolines is set for the current series, we do not print the last line number.

```
  \ifl@d@dash%
   \ifboolexpr{togl{fulllines@} or test{\ifcsempty{Xtwolines@\@currentseries}}}%
      {}%
      {%
      \setistwofollowinglines{#1}{#2}{#4}{#5}%
        \ifboolexpr{%
          (%
               togl {Xtwolinesbutnotmore@\@currentseries}%
               and not%
               (%
                bool {istwofollowinglines@}%
               )%
        )%
        or%
        (%
           (not test{\ifnumequal{#1}{#4}})%
             and togl{Xtwolinesonlyinsamepage@\@currentseries}%
           )%
        }%
        {}%
        {%
        \l@d@dashfalse%
        \l@d@Xtwolinestrue%
        \l@d@elinfalse%
        \l@d@eslfalse%
        \ifcsempty{Xmorethantwolines@\@currentseries}%
          {}%
          {\ifistwofollowinglines@\else%
            \l@d@Xmorethantwolinestrue%
           \fi%
          }%
        }%
    }%
\fi%
```

End of `\setprintlines`.

```
}%
```

`\setistwofollowinglines` The `\ifistwofollowinglines` boolean, used by the `\Xtwolines` and related setting, is set to true by `\setistwofollowinglines`. This command takes the following arguments:

- `#1` First page number.
- `#2` First line number.
- `#3` Last page number.
- `#4` Last line number.

If `#3`-`#2` = 1, then that means the two lines are subsequent, and consequently `\ifistwofollowinglines` is set to true. However, if we use lineation by page, two given lines can be subsequent if:

- The first line number is equal to the last line number of the first page.
- The last line number is equal to 1.
- `#3`-`#1` is equal to 1.

```
\newif\ifistwofollowinglines@%
\newcommand{\setistwofollowinglines}[4]{%
    \ifcsdef{lastlinenumberon@#1}%
      {\numdef{\tmp}{\csuse{lastlinenumberon@#1}}}%
      {\numdef{\tmp}{0}}%
    \istwofollowinglines@false%
    \ifnumequal{#4-#2}{1}%
    {\istwofollowinglines@true}%
    {\ifbypage@%
      \ifnumequal{#3-#1}{1}%
      {%
      \ifnumequal{#2}{\tmp}%
        {\ifnumequal{#4}{1}{\istwofollowinglines@true}{}}%
        {}%
      }%
      {}%
     \fi%
    }%
}%
```

\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.

```
\def\printlines#1|#2|#3|#4|#5|#6|#7|{%
  \begingroup%
```

If we use LuaTEX, ensure we use good text's direction.

```
  \ifluatex%
    \luatextextdir TLT%
  \fi%
```

Decide which part of line number components we will print.

```
  \setprintlines{#1}{#2}{#3}{#4}{#5}{#6}%
```

One subtlety left here is when to print a period between numbers. But the only instance in which this is tricky is for the ending sub-line number: it could come after the starting sub-line number (in which case we want only the dash) or after an ending line number (in which case we need to insert a period). So, first, print the start line number.

```
  \ifdimequal{\csuse{Xboxstartlinenum@\@currentseries}}{0pt}%
    {\bgroup}%
   {\leavevmode\hbox to \csuse{Xboxstartlinenum@\@currentseries}\bgroup\hfill}%
  \ifl@d@pnum #1\fullstop\fi
  \linenumrep{#2}
  \ifl@d@ssub \fullstop \sublinenumrep{#3}\fi
  \egroup%
```

Then print the dash + end line number, or the range symbol.

```
  \ifdimequal{\csuse{Xboxendlinenum@\@currentseries}}{0pt}%
    {\bgroup}%
    {\hbox to \csuse{Xboxendlinenum@\@currentseries}\bgroup}%
  \ifl@d@Xtwolines%
```

```
    \ifl@d@Xmorethantwolines%
      \csuse{Xmorethantwolines@\@currentseries}%
    \else%
      \csuse{Xtwolines@\@currentseries}%
    \fi%
  \else%
    \ifl@d@dash \endashchar\fi%
    \ifl@d@pnum #4\fullstop\fi%
    \ifl@d@elin \linenumrep{#5}\fi%
    \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumrep{#6}\fi%
  \fi%
  \ifdimequal{\csuse{Xboxendlinenum@\@currentseries}}{0pt}%
    {}%
    {\hfill}%Prevent underfull hbox
  \egroup%
  \endgroup%
}%
```

XIII Familiar footnotes

XIII.1 Adjacent footnotes

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

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

Maïeul Rouquette has maintened this feature in reledmac, despite he thought that is not directly in relationship with the aim of reledmac.

`\multiplefootnotemarker` `\multfootsep` These macros may have been defined by the memoir class, are provided by the footmisc package and perhaps by other footnote packages. That is why we use `\providecommand` and not `\newcommand`.

```
\providecommand*{\multiplefootnotemarker}{3sp}
\providecommand*{\multfootsep}{\textsuperscript{\normalfont,}}

```

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

```
\providecommand*{\m@mmf@prepare}{%
  \kern-\multiplefootnotemarker
  \kern\multiplefootnotemarker\relax}
```

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

```
\providecommand*{\m@mmf@check}{%
  \ifdim\lastkern=\multiplefootnotemarker\relax
```

```
    \edef\@x@sf{\the\spacefactor}%
    \unkern
    \multfootsep
    \spacefactor\@x@sf\relax
  \fi}

```

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

```
\@ifclassloaded{memoir}{}{%
```

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

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

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

```

\patchcmd{\@footnotemark}
  {\nobreak}
  {\m@mmf@check
  \nobreak
  }
  {}{}
\patchcmd{\@footnotemark}
  {\@makefnmark}
  {\@makefnmark
  \m@mmf@prepare
  }
  {}{}
```

Finished the modifications for the non-memoir case.

```
}

```

XIII.2 Regular footnotes for numbered texts

`\l@doldold@footnotetext` `\@footnotetext` In order to enable the regular `\footnotes` in numbered text we have to play around with its `\@footnotetext`, using different forms for when in numbered or regular text.

```
\pretocmd{\@footnotetext}{%
  \ifnumberedpar@
    \edtext{}{\l@dbfnote{#1}}%
  \else
  }{}{}
\apptocmd{\@footnotetext}{\fi}{}{}%
```

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

```
\newcommand{\l@dbfnote}[1]{%
  \ifnumberedpar@
  \gdef\@tag{#1\relax}%
   \xright@appenditem{\noexpand\vl@dbfnote{{\expandonce\@tag}}{\@thefnmark}}%
                     \to\inserts@list
    \global\advance\insert@count \@ne
  \fi\ignorespaces}
\newcommand{\vl@dbfnote}[2]{%
  \def\@thefnmark{#2}%
  \@footnotetext{#1}%
  }%
```

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
\postbodyfootmark

Two convenience macros for use by `\...@footnotemark...` macros.

```
\newcommand*{\prebodyfootmark}{%
  \leavevmode
  \ifhmode
    \edef\@x@sf{\the\spacefactor}%
    \m@mmf@check
    \nobreak
  \fi}
\newcommand{\postbodyfootmark}{%
  \m@mmf@prepare
  \ifhmode\spacefactor\@x@sf\fi\relax}

```

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.

```
\newcommandx{\arrangementX}[2][1,usedefault]{%
  \def\do##1{%
    \csname arrangementX@#2\endcsname{##1}%
  }%
  \ifstrempty{#1}%
    {%
    \dolistloop{\@series}%
    }%
    {
    \docsvlist{#1}%
    }%
}%
```

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.

```
\newcommand*{\normal@footnotemarkX}[1]{%
  \prebodyfootmark
  \@nameuse{bodyfootmark#1}%
  \postbodyfootmark}

```

\normalbodyfootmarkX

The \normalbodyfootmarkX{⟨*series*⟩} *really* typesets the in-text marker. The style is the normal superscript.

```
\newcommand*{\normalbodyfootmarkX}[1]{%
  \hbox{\textsuperscript{\normalfont\@nameuse{@thefnmark#1}}}}
```

\normalvfootnoteX

\normalvfootnoteX{⟨*series*⟩}{⟨*text*⟩} does the \insert for the ⟨*series*⟩ and calls the series' \footfmt... to format the ⟨*text*⟩.

```
\notbool{parapparatus@}{\newcommand*}{\newcommand}{\normalvfootnoteX}[2]{%
  \insert\@nameuse{footins#1}\bgroup
    \csuse{bhooknoteX@#1}
    \csuse{notefontsizeX@#1}
    \footsplitskips
    \ifl@dpairing\ifl@dpaging\else%
      \setnoteswidthliketwocolumnsX@{#1}%
    \fi\fi%
    \setnotesXpositionliketwocolumns@{#1}%
    \spaceskip=\z@skip \xspaceskip=\z@skip
    \csuse{\csuse{footnote@dir}}\@nameuse{footfmt#1}{#1}{#2}\egroup}

```

\mpnormalvfootnoteX

The minipage version.

```
\newcommand*{\mpnormalvfootnoteX}[2]{%
  \global\setbox\@nameuse{mpfootins#1}\vbox{%
    \unvbox\@nameuse{mpfootins#1}
    \csuse{bhooknoteX@#1}
    \csuse{notefontsizeX@#1}
    \hsize\columnwidth
    \@parboxrestore
    \color@begingroup
    \@nameuse{footfmt#1}{#1}{#2}\color@endgroup}}

```

\normalfootfmtX

\normalfootfmtX{⟨*series*⟩}{⟨*text*⟩} typesets the footnote text, prepended by the marker.

```
\notbool{parapparatus@}{\newcommand*}{\newcommand}{\normalfootfmtX}[2]{%
 \ifluatex%
    \luatextextdir\footnote@luatextextdir%
    \luatexpardir\footnote@luatexpardir%
    \par%
 \fi%
```

```
  \protected@edef\@currentlabel{%
      \@nameuse{@thefnmark#1}%
  }%
 \ledsetnormalparstuffX{#1}%
 \hangindent=\csuse{hangindentX@#1}%
 {{\csuse{notenumfontX@#1}\@nameuse{footfootmark#1}}\strut%
   #2\strut\par}}

```

\normalfootfootmarkX

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

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

```

\normalfootstartX

`\normalfootstartX{`⟨*series*⟩`}` is the ⟨*series*⟩ footnote starting macro used in the output routine.

```
\newcommand*{\normalfootstartX}[1]{%
   \ifdimequal{0pt}{\prenotesX@}{}%
     {%
     \iftoggle{prenotesX@}{%
          \togglefalse{prenotesX@}%
          \skip\csname footins#1\endcsname=%
            \dimexpr\csuse{prenotesX@}+\csuse{afterruleX@#1}\relax%
          }%
         {}%
     }%
 \vskip\skip\csname footins#1\endcsname%
 \leftskip=\z@
 \rightskip=\z@
 \ifl@dpairing\else%
    \hsize=\old@hsize%
 \fi%
 \setnoteswidthliketwocolumnsX@{#1}%
 \setnotesXpositionliketwocolumns@{#1}%
 \print@footnoteXrule{#1}%
}%

```

\normalfootnoteruleX

The rule drawn before the footnote series group.

```
\let\normalfootnoteruleX=\footnoterule

```

\normalfootgroupX

`\normalfootgroupX{`⟨*series*⟩`}` sends the contents of the ⟨*series*⟩ insert box to the output page without alteration.

```
\newcommand*{\normalfootgroupX}[1]{%
 \unvbox\@nameuse{footins#1}%
 \hsize=\old@hsize%
 }%

```

\mpnormalfootgroupX The minipage version.

```
\newcommand*{\mpnormalfootgroupX}[1]{%
  \vskip\skip\@nameuse{mpfootins#1}
  \ifl@dpairing\ifparledgroup%
    \leavevmode\marks\parledgroup@{begin}%
    \marks\parledgroup@series{#1}%
    \marks\parledgroup@type{footnoteX}%
  \fi\fi\normalcolor
  \ifparledgroup%
    \ifl@dpairing%
    \else%
      \setnoteswidthliketwocolumnsX@{#1}%
      \setnotesXpositionliketwocolumns@{#1}%
      \print@footnoteXrule{#1}%
    \fi%
  \else%
    \setnoteswidthliketwocolumnsX@{#1}%
    \setnotesXpositionliketwocolumns@{#1}%
    \print@footnoteXrule{#1}%
  \fi%
  \unvbox\@nameuse{mpfootins#1}}

```

\normalbfnoteX

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

```

\vbfnoteX

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

```

\vnumfootnoteX

```
\newcommand{\vnumfootnoteX}[2]{%
  \ifnumberedpar@
    \edtext{}{\normalbfnoteX{#1}{#2}}%
  \else
    \@nameuse{regvfootnote#1}{#1}{#2}%
  \fi}

```

arrangementX@normal `\arrangementX@normal{`⟨*series*⟩`}` initialises the settings for the ⟨*series*⟩ footnotes. This should always be called for each series.

```
\newcommand*{\arrangementX@normal}[1]{%
  \csgdef{series@displayX#1}{normal}
  \expandafter\let\csname footstart#1\endcsname=\normalfootstartX
  \expandafter\newcount\csname prevpage#1@num\endcsname
  \@namedef{@footnotemark#1}{\normal@footnotemarkX{#1}}
  \@namedef{bodyfootmark#1}{\normalbodyfootmarkX{#1}}
  \expandafter\let\csname regvfootnote#1\endcsname=\normalvfootnoteX
  \expandafter\let\csname vfootnote#1\endcsname=\vnumfootnoteX
  \expandafter\let\csname footfmt#1\endcsname=\normalfootfmtX
  \@namedef{footfootmark#1}{\normalfootfootmarkX{#1}}
  \expandafter\let\csname footgroup#1\endcsname=\normalfootgroupX
  \expandafter\let\csname footnoterule#1\endcsname=\normalfootnoteruleX
  \count\csname footins#1\endcsname=1000
  \csxdef{default@footins#1}{1000}%Use to have note only for one side
  \dimen\csname footins#1\endcsname=\csuse{maxhnotesX@#1}
  \skip\csname footins#1\endcsname=\csuse{beforenotesX@#1}%
  \advance\skip\csname footins#1\endcsname by\csuse{afterruleX@#1}%
```

Additions for minipages.

```
  \ifnoledgroup@\else%
    \expandafter\let\csname mpvfootnote#1\endcsname=\mpnormalvfootnoteX
    \expandafter\let\csname mpfootgroup#1\endcsname=\mpnormalfootgroupX
    \count\csname mpfootins#1\endcsname=1000
    \dimen\csname mpfootins#1\endcsname=\csuse{maxhnotesX@#1}
    \skip\csname mpfootins#1\endcsname=\csuse{beforenotesX@#1}%
    \advance\skip\csname mpfootins#1\endcsname by\csuse{afterruleX@#1}%
  \fi
}

```

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

```
\newcommand*{\arrangementX@twocol}[1]{%
  \csgdef{series@displayX#1}{twocol}
  \expandafter\let\csname regvfootnote#1\endcsname=\twocolvfootnoteX
  \expandafter\let\csname footfmt#1\endcsname=\twocolfootfmtX
  \expandafter\let\csname footgroup#1\endcsname=\twocolfootgroupX
  \dimen\csname footins#1\endcsname=\csuse{maxhnotesX@#1}%
  \skip\csname footins#1\endcsname=\csuse{beforenotesX@#1}%
 \advance\skip\csname footins#1\endcsname by \csuse{afterruleX@#1}\relax%
  \twocolfootsetupX{#1}
  \ifnoledgroup@\else%
    \expandafter\let\csname mpvfootnote#1\endcsname=\mpnormalvfootnoteX
    \expandafter\let\csname mpfootgroup#1\endcsname=\mptwocolfootgroupX
    \skip\csname mpfootins#1\endcsname=\csuse{beforenotesX@#1}%
    \advance\skip\csname mpfootins#1\endcsname by\csuse{afterruleX@#1}
    \mptwocolfootsetupX{#1}
  \fi%
}

```

\twocolfootsetupX \mptwocolfootsetupX

\twocolfootsetupX{⟨*series*⟩}

```
\newcommand*{\twocolfootsetupX}[1]{%
  \count\csname footins#1\endcsname 500
 \csxdef{default@footins#1}{500}%Use this to confine the  notes to one side only
  \multiply\dimen\csname footins#1\endcsname by \tw@}
\newcommand*{\mptwocolfootsetupX}[1]{%
  \count\csname mpfootins#1\endcsname 500
  \multiply\dimen\csname mpfootins#1\endcsname by \tw@}

```

\twocolvfootnoteX

\twocolvfootnoteX{⟨*series*⟩}

```
\notbool{parapparatus@}{\newcommand*}{\newcommand}{\twocolvfootnoteX}[2]{%
  \insert\csname footins#1\endcsname\bgroup
    \csuse{notefontsizeX@#1}
    \footsplitskips
    \spaceskip=\z@skip \xspaceskip=\z@skip
    \@nameuse{footfmt#1}{#1}{#2}\egroup}

```

\twocolfootfmtX

\twocolfootfmtX{⟨*series*⟩}

```
\notbool{parapparatus@}{\newcommand*}{\newcommand}{\twocolfootfmtX}[2]{%
   \protected@edef\@currentlabel{%
       \@nameuse{@thefnmark#1}%
   }%
  \normal@pars
  \hangindent=\csuse{hangindentX@#1}%
  \hsize \csuse{hsizetwocolX@#1}
 \nottoggle{parindentX@#1}{\parindent=\z@}{}
  \tolerance=5000\relax
  \leavevmode
 \csuse{colalignX@#1}%
  {\csuse{notenumfontX@#1}\@nameuse{footfootmark#1}\strut%
    #2\strut\par}\allowbreak}

```

\twocolfootgroupX \mptwocolfootgroupX

\twocolfootgroupX{⟨*series*⟩}

```
\newcommand*{\twocolfootgroupX}[1]{{\csuse{notefontsizeX@#1}
  \splittopskip=\ht\strutbox
  \expandafter
  \rigidbalance\csname footins#1\endcsname \tw@ \splittopskip}}
\newcommand*{\mptwocolfootgroupX}[1]{{%
  \vskip\skip\@nameuse{mpfootins#1}
  \ifl@dpairing\ifparledgroup%
    \leavevmode\marks\parledgroup@{begin}%
    \marks\parledgroup@series{#1}%
    \marks\parledgroup@type{footnoteX}%
  \fi\fi\normalcolor
  \ifparledgroup%
    \ifl@dpairing%
    \else%
      \setnoteswidthliketwocolumnsX@{#1}%
      \setnotesXpositionliketwocolumns@{#1}%
      \print@footnoteXrule{#1}%
    \fi%
  \else%
    \setnoteswidthliketwocolumnsX@{#1}%
    \setnotesXpositionliketwocolumns@{#1}%
    \print@footnoteXrule{#1}%
  \fi%
  \splittopskip=\ht\strutbox
  \expandafter
  \rigidbalance\csname mpfootins#1\endcsname \tw@ \splittopskip}}

```

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

```
\newcommand*{\arrangementX@threecol}[1]{%
  \csgdef{series@displayX#1}{threecol}
  \expandafter\let\csname regvfootnote#1\endcsname=\threecolvfootnoteX
  \expandafter\let\csname footfmt#1\endcsname=\threecolfootfmtX
  \expandafter\let\csname footgroup#1\endcsname=\threecolfootgroupX
  \dimen\csname footins#1\endcsname=\csuse{maxhnotesX@#1}%
  \skip\csname footins#1\endcsname=\csuse{beforenotesX@#1}%
  \advance\skip\csname footins#1\endcsname by \csuse{afterruleX@#1}\relax%
  \threecolfootsetupX{#1}
  \ifnoledgroup@\else%
    \expandafter\let\csname mpvfootnote#1\endcsname=\mpnormalvfootnoteX
    \expandafter\let\csname mpfootgroup#1\endcsname=\mpthreecolfootgroupX
    \skip\csname mpfootins#1\endcsname=\csuse{beforenotesX@#1}%
    \advance\skip\csname mpfootins#1\endcsname by\csuse{afterruleX@#1}
    \mpthreecolfootsetupX{#1}
  \fi%
}

```

\threecolfootsetupX
\mpthreecolfootsetupX

\threecolfootsetupX{⟨*series*⟩}

```
\newcommand*{\threecolfootsetupX}[1]{%
  \count\csname footins#1\endcsname 333
  \csxdef{default@footins#1}{333}%Use this to confine the  notes to one side only
  \multiply\dimen\csname footins#1\endcsname by \thr@@}
\newcommand*{\mpthreecolfootsetupX}[1]{%
  \count\csname mpfootins#1\endcsname 333
  \multiply\dimen\csname mpfootins#1\endcsname by \thr@@}

```

\threecolvfootnoteX

`\threecolvfootnoteX{`⟨*series*⟩`}{`⟨*text*⟩`}`

```
\notbool{parapparatus@}{\newcommand*}{\newcommand}{\threecolvfootnoteX}[2]{%
  \insert\csname footins#1\endcsname\bgroup
    \csuse{notefontsizeX@#1}
    \footsplitskips
    \@nameuse{footfmt#1}{#1}{#2}\egroup}

```

\threecolfootfmtX

`\threecolfootfmtX{`⟨*series*⟩`}`

```
\notbool{parapparatus@}{\newcommand*}{\newcommand}{\threecolfootfmtX}[2]{%
   \protected@edef\@currentlabel{%
       \@nameuse{@thefnmark#1}%
   }%
  \hangindent=\csuse{hangindentX@#1}%
  \normal@pars
  \hsize \csuse{hsizethreecolX@#1}
  \nottoggle{parindentX@#1}{\parindent=\z@}{} %
  \tolerance=5000\relax
  \leavevmode
  \csuse{colalignX@#1}%
  {\csuse{notenumfontX@#1}\@nameuse{footfootmark#1}\strut%
    #2\strut\par}\allowbreak}

```

\threecolfootgroupX
\mpthreecolfootgroupX

`\threecolfootgroupX{`⟨*series*⟩`}`

```
\newcommand*{\threecolfootgroupX}[1]{{\csuse{notefontsizeX@#1}
  \splittopskip=\ht\strutbox
  \expandafter
  \rigidbalance\csname footins#1\endcsname \thr@@ \splittopskip}}
\newcommand*{\mpthreecolfootgroupX}[1]{{%
  \vskip\skip\@nameuse{mpfootins#1}
  \ifl@dpairing\ifparledgroup
    \leavevmode\marks\parledgroup@{begin}%
    \marks\parledgroup@series{#1}%
    \marks\parledgroup@type{footnoteX}%
  \fi\fi\normalcolor
  \ifparledgroup%
    \ifl@dpairing%
    \else%
      \setnoteswidthliketwocolumnsX@{#1}%
      \setnotesXpositionliketwocolumns@{#1}%
      \print@footnoteXrule{#1}%
    \fi%
  \else%
    \setnoteswidthliketwocolumnsX@{#1}%
    \setnotesXpositionliketwocolumns@{#1}%
    \print@footnoteXrule{#1}%
  \fi%
  \splittopskip=\ht\strutbox
  \expandafter
  \rigidbalance\csname mpfootins#1\endcsname \thr@@ \splittopskip}}

```

XIII.4.5 Paragraphed footnotes

The following macros set footnotes as one paragraph.

\arrangementX@threecol

`\footparagraphX{`⟨*series*⟩`}`

```
\newcommand*{\arrangementX@paragraph}[1]{%
  \csgdef{series@displayX#1}{paragraph}%
  \expandafter\newcount\csname #1prevpage@num\endcsname
  \expandafter\let\csname footstart#1\endcsname=\parafootstartX
  \expandafter\let\csname regvfootnote#1\endcsname=\para@vfootnoteX
  \expandafter\let\csname footfmt#1\endcsname=\parafootfmtX
  \expandafter\let\csname footgroup#1\endcsname=\para@footgroupX
  \expandafter\let\csname footnoterule#1\endcsname=\normalfootnoteruleX
  \count\csname footins#1\endcsname=1000
  \csxdef{default@footins#1}{1000}%Use this to confine the  notes to one side only
  \dimen\csname footins#1\endcsname=\csuse{maxhnotesX@#1}
  \skip\csname footins#1\endcsname=\csuse{beforenotesX@#1}%
  \advance\skip\csname footins#1\endcsname by\csuse{afterruleX@#1}%
  \para@footsetupX{#1}
  \ifnoledgroup@\else
    \expandafter\let\csname mpvfootnote#1\endcsname=\mppara@vfootnoteX
    \expandafter\let\csname mpfootgroup#1\endcsname=\mppara@footgroupX
    \count\csname mpfootins#1\endcsname=1000
    \dimen\csname mpfootins#1\endcsname=\csuse{maxhnotesX@#1}
    \skip\csname mpfootins#1\endcsname=\csuse{beforenotesX@#1}%
    \advance\skip\csname mpfootins#1\endcsname by\csuse{afterruleX@#1}%
  \fi
  }

```

\para@footsetupX

`\para@footsetupX{`⟨*series*⟩`}`

```
\newcommand*{\para@footsetupX}[1]{{\csuse{notefontsizeX@#1}
  \setnoteswidthliketwocolumnsX@{#1}%
  \dimen0=\baselineskip
  \multiply\dimen0 by 1024
  \divide\dimen0 by \columnwidth \multiply\dimen0 by \footfudgefiddle\relax%
  \expandafter
  \xdef\csname footfudgefactor#1\endcsname{%
    \expandafter\strip@pt\dimen0 }}}

```

\parafootstartX `\parafootstartX{`⟨*series*⟩`}`

```
\newcommand*{\parafootstartX}[1]{%
    \ifdimequal{0pt}{\prenotesX@}{}%
      {%
      \iftoggle{prenotesX@}{%
            \togglefalse{prenotesX@}%
            \skip\csname footins#1\endcsname=%
              \dimexpr\csuse{prenotesX@}+\csuse{afterruleX@#1}\relax%
            }%
          {}%
      }%
  \vskip\skip\csname footins#1\endcsname%
  \leftskip=\z@
  \rightskip=\z@
  \parindent=\z@
  \vskip\skip\@nameuse{footins#1}%
  \setnoteswidthliketwocolumnsX@{#1}%
  \setnotesXpositionliketwocolumns@{#1}%
  \print@footnoteXrule{#1}%
  }

```

\para@vfootnoteX `\para@vfootnoteX{`⟨*series*⟩`}{`⟨*text*⟩`}`
\mppara@vfootnoteX

```
\newcommand*{\para@vfootnoteX}[2]{%
  \insert\csname footins#1\endcsname
  \bgroup
    \csuse{bhooknoteX@#1}
    \csuse{notefontsizeX@#1}
    \footsplitskips
    \setbox0=\vbox{\hsize=\maxdimen
      \noindent\@nameuse{footfmt#1}{#1}{#2}}%
    \setbox0=\hbox{\unvxhX0[#1]}%
    \dp0=\z@
    \ht0=\csname footfudgefactor#1\endcsname\wd0
    \box0
    \penalty0
  \egroup}
\newcommand*{\mppara@vfootnoteX}[2]{%
  \global\setbox\@nameuse{mpfootins#1}\vbox{%
    \unvbox\@nameuse{mpfootins#1}
    \csuse{bhooknoteX@#1}
    \csuse{notefontsizeX@#1}
    \footsplitskips
    \setbox0=\vbox{\hsize=\maxdimen
    \noindent\color@begingroup\@nameuse{footfmt#1}{#1}{#2}\color@endgroup}%
    \setbox0=\hbox{\unvxhX0[#1]}%
    \dp0=\z@
    \ht0=\csname footfudgefactor#1\endcsname\wd0
    \box0
    \penalty0}}
```

\unvxhX

```
\newcommandx*{\unvxhX}[2][2=Z]{% 2th is optional for retro-compatibility
  \setbox0=\vbox{\unvbox#1%
    \global\setbox1=\lastbox}%
  \unhbox1
  \unskip             % remove \rightskip,
  \unskip             % remove \parfillskip,
  \unpenalty          % remove \penalty of 10000,
  \hskip\csuse{afternoteX@#2}}   % but add the glue to go between the notes

```

\parafootfmtX \parafootfmtX{⟨*series*⟩}

```
\newcommand*{\parafootfmtX}[2]{%
   \protected@edef\@currentlabel{%
       \@nameuse{@thefnmark#1}%
   }%
  \insertparafootsepX{#1}%
  \ledsetnormalparstuffX{#1}%
  {\csuse{notenumfontX@#1}%
   \csuse{notenumfontX@#1}%
   \@nameuse{footfootmark#1}%
   \strut%
    #2\penalty-10}}

```

\para@footgroupX \para@footgroupX{⟨*series*⟩}
\mppara@footgroupX

```
\newcommand*{\para@footgroupX}[1]{%
  \unvbox\csname footins#1\endcsname
  \ifcsstring{raggedX@#1}{L}{\RaggedLeft}{}%
  \ifcsstring{raggedX@#1}{R}{\RaggedRight}{}%
  \makehboxofhboxes
  \setbox0=\hbox{\unhbox0 \removehboxes}%
  \csuse{notefontsizeX@#1}
  \noindent\unhbox0\par}
\newcommand*{\mppara@footgroupX}[1]{{%
  \setnoteswidthliketwocolumnsX@{#1}%
  \vskip\skip\@nameuse{mpfootins#1}
  \ifl@dpairing\ifparledgroup
    \leavevmode%
    \leavevmode\marks\parledgroup@{begin}%
    \marks\parledgroup@series{#1}%
    \marks\parledgroup@type{footnoteX}%
  \fi\fi\normalcolor
  \ifparledgroup%
    \ifl@dpairing%
    \else%
      \setnoteswidthliketwocolumnsX@{#1}%
      \setnotesXpositionliketwocolumns@{#1}%
      \print@footnoteXrule{#1}%
    \fi%
  \else%
      \setnoteswidthliketwocolumnsX@{#1}%
      \setnotesXpositionliketwocolumns@{#1}%
      \print@footnoteXrule{#1}%
  \fi%
  \unvbox\csname mpfootins#1\endcsname
  \ifcsstring{raggedX@#1}{L}{\RaggedLeft}{}%
  \ifcsstring{raggedX@#1}{R}{\RaggedRight}{}%
  \makehboxofhboxes
  \setbox0=\hbox{\unhbox0 \removehboxes}%
  \csuse{notefontsizeX@#1}
  \noindent\unhbox0\par}}

```

Insertion of the footnotes separator The command `\insertparafootsepX{`⟨*series*⟩`}` must be called at the beginning of `\parafootftmX`.

`\prevpage@num`
`\Xinsertparafootsep`

```
\newcommand{\insertparafootsepX}[1]{%
  \ifnumequal{\csuse{prevpage#1@num}}{\page@num}%
    {\csuse{parafootsepX@#1}}%
    {}%
}
```

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.

`\ledsetnormalparstuff@common`
`\Xledsetnormalparstuff`
`\ledsetnormalparstuffX`

```
\newcommand*{\ledsetnormalparstuff@common}{%
  \ifluatex%
     \luatextextdir\footnote@luatextextdir%
   \luatexpardir\footnote@luatexpardir%
  \fi%
  \csuse{\csuse{footnote@dir}}%
  \normal@pars%
  \parfillskip \z@ \@plus 1fil}%

\newcommand*{\Xledsetnormalparstuff}[1]{%
  \ledsetnormalparstuff@common%
  \nottoggle{Xparindent@#1}{\noindent}{}%\noindent and and not \parindent=0pt to avoid to break the (bad)
}%

\newcommand*{\ledsetnormalparstuffX}[1]{%
  \ledsetnormalparstuff@common%
  \nottoggle{parindentX@#1}{\noindent}{}%\noindent and and not \parindent=0pt to avoid to break the (bad)
}%
```

XV Footnotes' width for two columns

We define here some commands which make sense only with reledpar, but must be called when defining notes parameters. These commands change the width of block notes to allow them to have the same size than two parallel columns.

`\old@hsize`
`noteswidthliketwocolumns@`
`oteswidthliketwocolumnsX@`

These two commands are called at the beginning of critical or familiar notes groups. They set, if the option is enabled, the `\hsize`. They are also called at the on the setup for paragraphed notes.

```

\newdimen\old@hsize%
\AtBeginDocument{\old@hsize=\hsize}%

\newcommand{\setXnoteswidthliketwocolumns@}[1]{%
  \global\let\hsize@fornote=\hsize%
  \global\old@hsize=\hsize%
  \iftoggle{Xnoteswidthliketwocolumns@#1}%
    {%
    \csuse{setwidthliketwocolumns@\columns@position}%
    \global\let\hsize@fornote=\hsize%
    }%
    {}%
  \let\hsize=\hsize@fornote%
  \let\columnwidth=\hsize@fornote%
}%

\newcommand{\setnoteswidthliketwocolumnsX@}[1]{%
  \global\let\hsize@fornote=\hsize%
  \global\old@hsize=\hsize%
  \iftoggle{noteswidthliketwocolumnsX@#1}%
    {%
    \csuse{setwidthliketwocolumns@\columns@position}%
    \global\let\hsize@fornote=\hsize%
    }%
    {}%
  \let\hsize=\hsize@fornote%
  \let\columnwidth=\hsize@fornote%
}%

```

\setXnotespositionliketwocolumns@
\setnotesXpositionliketwocolumns@

These two commands set the position of the critical / familiar footnotes, depending on the hooks `Xnoteswidthliketwocolumns` and `noteswidthliketwocolumnsX`. They call commands which are defined only in reledpar, because this feature has no sens without reledpar.

```
\newcommand{\setXnotespositionliketwocolumns@}[1]{%
  \iftoggle{Xnoteswidthliketwocolumns@#1}{%
    \csuse{setnotespositionliketwocolumns@\columns@position}%
  }{}%
}%

\newcommand{\setnotesXpositionliketwocolumns@}[1]{%
  \iftoggle{noteswidthliketwocolumnsX@#1}{%
    \csuse{setnotespositionliketwocolumns@\columns@position}%
  }{}%
}%

```

XVI Footnotes' order

\fnpos
\mpfnpos
\@fnpos
\@mpfnpos

The `\fnpos` and `\mpfnpos` simply place their arguments in `\@fnpos` and `\@mpfnpos`, which will be used later in the output routine.

```
\def\@fnpos{familiar-critical}
\def\@mpfnpos{critical-familiar}
\newcommand{\fnpos}[1]{\xdef\@fnpos{#1}}
\newcommand{\mpfnpos}[1]{\xdef\@mpfnpos{#1}}
```

XVII Footnotes' rule

Because the footnotes' rules can be shifted to the right when footnotes are set like two columns, we do not print them directly, but we put them in a `\vbox`.

\print@Xfootnoterule
\print@footnoteXrule

```
\newcommand{\print@Xfootnoterule}[1]{%
  \vskip-\csuse{Xafterrule@#1}%Because count in \dimen\csuse{#1footins}
  \nointerlineskip%
  \moveleft-\leftskip\vbox{\csuse{#1footnoterule}}%
  \nointerlineskip%
  \vskip\csuse{Xafterrule@#1}%
}%

\newcommand{\print@footnoteXrule}[1]{%
  \vskip-\csuse{afterruleX@#1}%Because count in \dimen\csuse{footins#1}
  \nointerlineskip%
  \moveleft-\leftskip\vbox{\csuse{footnoterule#1}}%
  \nointerlineskip%
  \vskip\csuse{afterruleX@#1}%
}%

```

XVIII Specific skip for first series of footnotes

XVIII.0.1 Overview

`\Xbeforenotes` inserts a specific skip for the first series of notes in a page. As we can't know in advance which series will be the first, we call `\prepare@preXnotes` before inserting any critical notes, in order to prevent page number overlapping.

1. If it is the first note of the current page, it changes the footnote skip for the series to the value specified to `\Xbeforenotes`. It also keeps the series of the note as the first one of the current page.
2. If it is not the first note of the current page:
   - If the current series is printed after the series kept as the first of the current page, then nothing happens.
   - If the current series is printed before the series kept as the first of the current page, then it changes the footnote skip of the current series to the value normally used by the series which was marked as the first of the page. It also keeps the current series as the new first one of the current page.

For example, suppose the series order is A,B. We call first a `\Bfootnote` and a `\Afootnote`. The only skips used are, finally, the skip specific to the first series of the page, and the skip for the B series. If we have not called `\Afootnote`, the only skip used is the skip specific to the first series of the page.

That is perfect.

The series skip and the first series of the current page are reset before the footnotes are printed. Then, the footstart macros manage the problem of the first series of the page.

After the rule, the space which is defined by `\Xafterrule` does not depend on whether the series is the first one of the page or not. So we use its normal value for each series.

And now, implementation !

XVIII.0.2 User level command

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

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

The same, but for familiar footnotes.

\preXnotes
\preXnotes@

```
\newtoggle{prenotesX@}
\toggletrue{prenotesX@}
\newcommand{\prenotesX@}{0pt}
\newcommand*{\prenotesX}[1]{\renewcommand{\prenotesX@}{#1}}
```

XVIII.0.3 Internal commands

firstXseries@
prepare@preXnotes

```
\gdef\firstXseries@{}
\newcommand{\prepare@preXnotes}[1]{%
  \ifdimequal{0pt}{\preXnotes@}%
  {}%
  {%
    \IfStrEq{\firstXseries@}{}{%
      \global\skip\csuse{#1footins}=\preXnotes@%
    \global\advance\skip\csname #1footins\endcsname by\csuse{Xafterrule@#1}%
      \gdef\firstXseries@{#1}%
    }%
    {%
     \ifseriesbefore{#1}{\firstXseries@}%
       {%
       \global\skip\csuse{#1footins}=\csuse{Xbeforenotes@\firstXseries@}%
     \global\advance\skip\csname #1footins\endcsname by\csuse{Xafterrule@#1}%
       \gdef\firstXseries@{#1}%
       }%
       {}%
    }%
  }%
}
```

The same thing is required for familiar notes and \prenotesX.

firstseriesX@
prepare@prenotesX

```
\gdef\firstseriesX@{}
\newcommand{\prepare@prenotesX}[1]{%
  \ifdimequal{0pt}{\prenotesX@}%
  {}%
  {%
    \IfStrEq{\firstseriesX@}{}{%
      \global\skip\csuse{footins#1}=\prenotesX@%
    \global\advance\skip\csname footins#1\endcsname by\csuse{afterruleX@#1}%
      \gdef\firstseriesX@{#1}%
    }%
    {%
     \ifseriesbefore{#1}{\firstseriesX@}%
       {%
       \global\skip\csuse{footins#1}=\csuse{beforenotesX@\firstseriesX@}%
     \global\advance\skip\csname footins#1\endcsname by\csuse{afterruleX@#1}%
       \gdef\firstXseries@{#1}%
       }%
       {}%
    }%
  }%
}
```

XIX Endnotes

First, check the noend option.

```
\ifbool{noend@}{}{%Used instead of \ifnoend@ to prevent expansion problem
```

`\l@d@end` `\ifl@dend@` `\l@dend@true` `\l@dend@false`

Endnotes of all varieties are saved up in a file, typically named ⟨*jobname*⟩`.end`. `\l@d@end` is the output stream number for this file, and `\ifl@dend@` is a flag that is `true` when the file is open.

```
\newwrite\l@d@end
\newif\ifl@dend@
```

`\l@dend@open` `\l@dend@close`

`\l@dend@open` and `\l@dend@close` are the macros that are used to open and close the endnote file. Note that all our writing to this file is `\immediate`: all page and line numbers for the endnotes are generated by the same mechanism we use for the footnotes, so that there is no need to defer any writing to catch information from the output routine.

```
\newcommand{\l@dend@open}[1]{\global\l@dend@true\immediate\openout\l@d@end=#1\relax}
\newcommand{\l@dend@close}{\global\l@dend@false\immediate\closeout\l@d@end}

```

`\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.

```
\newcommand{\l@dend@stuff}{%
 \ifl@dend@\relax\else
   \l@dend@open{\jobname.end}%
 \fi
 \immediate\write\l@d@end{\string\l@d@section{\the\section@num}}}

```

`\endprint` `\l@d@section`

The `\endprint` here is nearly identical in its functioning to `\normalfootfmt`.

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`.

```
\global\notbool{parapparatus@}{}{\long}\def\endprint#1#2#3#4#5{{%
  \ifXendinsertsep@%
    \hskip\csuse{Xendafternote@#4}%
    \csuse{Xendsep@#4}%
  \else%
    \iftoggle{Xendparagraph@#4}%
      {\global\Xendinsertsep@true}%
      {}%
  \fi%
  \xdef\@currentseries{#4}%
  \def\do##1{%
   \toggletrue{##1@}%
  }%
  \notblank{#5}{\docsvlist{#5}}{}%
  \csuse{Xendbhooknote@#4}%
  \csuse{Xendnotefontsize@#4}%
  {%
    \csuse{Xendnotenumfont@#4}%
    \ifdimequal{\csuse{Xendboxlinenum@#4}}{0pt}%
      {\printendlines#1|}%
      {\leavevmode%
       \hbox to \csuse{Xendboxlinenum@#4}%
       {%
       \IfSubStr{RC}{\csuse{Xendboxlinenumalign@#4}}{\hfill}{}%
       \printendlines#1|%
       \IfSubStr{LC}{\csuse{Xendboxlinenumalign@#4}}{\hfill}{}%
      }}%
  }%
  \enspace{%
    \nottoggle{Xendlemmadisablefontselection@#4}%
      {\select@lemmafont#1|#2}%
      {#2}%
  }%
  \ifboolexpr{%
    togl {nosep@}%
    or test{\ifcsempty{Xendlemmaseparator@#4}}%
    }%
    {\hskip\csuse{Xendinplaceoflemmaseparator@#4}}%
    {\nobreak%
     \hskip\csuse{Xendbeforelemmaseparator@#4}%
     \csuse{Xendlemmaseparator@#4}%
     \hskip\csuse{Xendafterlemmaseparator@#4}%
    }%
  #3%
  \nottoggle{Xendparagraph@#4}{\par}{}%
  \togglefalse{fulllines@}%
  \togglefalse{nosep@}%
}}%

\let\l@d@section=\@gobble

```

`\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.

```
\newcommand*{\setprintendlines}[6]{%
  \l@d@pnumfalse \l@d@dashfalse
  \ifnum#4=#1 \else
    \l@d@pnumtrue
    \l@d@dashtrue
  \fi
```

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.

```
  \ifl@d@pnum \l@d@elintrue \else \l@d@elinfalse \fi
  \ifnum#2=#5 \else
      \l@d@elintrue
      \l@d@dashtrue
  \fi
```

We print the starting sub-line if it is nonzero.

```
  \l@d@ssubfalse
  \ifnum#3=0 \else
      \l@d@ssubtrue
  \fi
```

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.

```
  \l@d@eslfalse
  \ifnum#6=0 \else
      \ifnum#6=#3
         \ifl@d@elin \l@d@esltrue \else \l@d@eslfalse \fi
      \else
         \l@d@esltrue
         \l@d@dashtrue
      \fi
  \fi%
```

```
  \ifl@d@dash%
   \ifboolexpr{togl{fulllines@} or test{\ifcsempty{Xendtwolines@\@currentseries}}}%
      {}%
```

```
    {%
    \setistwofollowinglines{#1}{#2}{#4}{#5}%
      \ifboolexpr{%
        (%
             togl {Xendtwolinesbutnotmore@\@currentseries}%
             and not%
               (%
                bool {istwofollowinglines@}%
               )%
        )%
        or%
        (%
           (not test{\ifnumequal{#1}{#4}})%
             and togl{Xendtwolinesonlyinsamepage@\@currentseries}%
           )%
        }%
        {}%
        {%
        \l@d@dashfalse%
        \l@d@Xtwolinestrue%
        \l@d@elinfalse%
        \l@d@eslfalse%
        \ifcsempty{Xendmorethantwolines@\@currentseries}%
          {}%
          {\ifistwofollowinglines@\else%
            \l@d@Xmorethantwolinestrue%
           \fi%
          }%
        }%
    }%
\fi%
```

End of `\setprintendlines`.

```
}%
```

`\printendlines` Now we are ready to print it all.

```
\def\printendlines#1|#2|#3|#4|#5|#6|#7|{\begingroup
  \setprintendlines{#1}{#2}{#3}{#4}{#5}{#6}%
```

The only subtlety left here is when to print a period between numbers. But the only instance in which this is tricky is for the ending sub-line number: it could be coming after the starting sub-line number (in which case we want only the dash) or after an ending line number (in which case we need to insert a period).

So, first, print the start lines.

```
  \ifdimequal{\csuse{Xendboxstartlinenum@\@currentseries}}{0pt}%
    {\bgroup}%
   {\leavevmode\hbox to \csuse{Xendboxstartlinenum@\@currentseries}\bgroup\hfill}%
  \printnpnum{#1}%
  \linenumrep{#2}%
  \ifl@d@ssub \fullstop \sublinenumrep{#3}\fi
```

```
 \egroup%
```

And now, print the dash + the end line number, or the line number range symbol.

```
 \ifdimequal{\csuse{Xendboxendlinenum@\@currentseries}}{0pt}%
   {\bgroup}%
   {\hbox to \csuse{Xendboxendlinenum@\@currentseries}\bgroup}%
 \ifl@d@Xtwolines%
   \ifl@d@Xmorethantwolines%
     \csuse{Xendmorethantwolines@\@currentseries}%
   \else%
     \csuse{Xendtwolines@\@currentseries}%
   \fi%
 \else%
   \ifl@d@dash \endashchar\fi%
   \ifl@d@pnum \printnpnum{#4}\fi%
   \ifl@d@elin \linenumrep{#5}\fi%
   \ifl@d@esl \ifl@d@elin \fullstop\fi \sublinenumrep{#6}\fi%
 \fi%
 \ifdimequal{\csuse{Xendboxendlinenum@\@currentseries}}{0pt}%
   {}%
   {\hfill}%Prevent underfull hbox
 \egroup%
 \endgroup%
}%

```

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

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

```

`\doendnotes` `\ifXendinsertsep@` `\doendnotes` is the command you use to print one series of endnotes; it takes one argument: the series letter of the note series you want to print. `\Xendinsertsep@` is set to true at the first note of the series, and to false at the last one.

```
\newif\ifXendinsertsep@%
\newcommand*{\doendnotes}[1]{\l@dend@close
  \begingroup
     \makeatletter
     \expandafter\let\csname #1end\endcsname=\endprint
     \input\jobname.end
     \global\Xendinsertsep@false%
  \endgroup}
```

`\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.

```
\newcommand*{\doendnotesbysection}[1]{%
 \l@dend@close%
 \global\expandafter\advance\csname #1end@bysection\endcsname by 1%
 \begingroup%
    \makeatletter%
    \def\l@d@section##1{%
      \ifnumequal{##1}{\csname #1end@bysection\endcsname}%
        {\cslet{#1end}{\endprint}}%
        {\cslet{#1end}{\@gobblefive}}%
    }%
    \input\jobname.end%
    \global\Xendinsertsep@false%
 \endgroup%
}%
```

End of section for end notes

```
}%
```

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@`.

```
\newcommand{\newseries}[1]{%
   \def\do##1{\newseries@{##1}}%
   \docsvlist{#1}
}
```

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

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

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

\newseries@

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

XX.1 Test if series is still existing

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

XX.2 Init specific to reledpar

When calling `\newseries@` after having loaded reledpar, we need to load specific setting.

```
     \ifdefined\newseries@par%
       \newseries@par{#1}%
     \fi%
```

XX.3 For critical footnotes

Critical footnotes are those which start with letters. We look for the `\nocritical` option of reledmac.

```
\unless\ifnocritical@
```

XX.3.1 Options

```
    \newtoggle{Xparindent@#1}
    \newtoggle{Xlemmadisablefontselection@#1}
    \csgdef{Xhangindent@#1}{0pt}%
    \csgdef{Xragged@#1}{}%
    \csgdef{Xhsizetwocol@#1}{0.45 \hsize}%
    \csgdef{Xhsizethreecol@#1}{.3 \hsize}%
    \csgdef{Xcolalign@#1}{\raggedright}%
    \csgdef{Xnotenumfont@#1}{\normalfont}%
    \csgdef{Xnotefontsize@#1}{\footnotesize}%
    \csgdef{Xbhooknote@#1}{}%

    \csgdef{Xboxlinenum@#1}{0pt}%
    \csgdef{Xboxlinenumalign@#1}{L}%

    \csgdef{Xboxstartlinenum@#1}{0pt}%
    \csgdef{Xboxendlinenum@#1}{0pt}%

    \csgdef{Xboxsymlinenum@#1}{0pt}%
    \newtoggle{Xnumberonlyfirstinline@#1}%
    \newtoggle{XnumberonlyfirstinXtwolines@#1}%
    \csgdef{Xtwolines@#1}{}%
    \csgdef{Xmorethantwolines@#1}{}%
    \newtoggle{Xtwolinesbutnotmore@#1}%
    \newtoggle{Xtwolinesonlyinsamepage@#1}%
    \newtoggle{Xonlypstart@#1}%
    \newtoggle{Xpsteverytime@#1}%
    \newtoggle{Xpstart@#1}%
    \csgdef{Xsymlinenum@#1}{}%
    \newtoggle{Xnonumber@#1}%
    \csgdef{Xbeforenumber@#1}{0pt}%
    \csgdef{Xafternumber@#1}{0.5em}%
    \newtoggle{Xnonbreakableafternumber@#1}%
    \csgdef{Xbeforesymlinenum@#1}{\csuse{Xbeforenumber@#1}}%
    \csgdef{Xaftersymlinenum@#1}{\csuse{Xafternumber@#1}}%
    \csgdef{Xinplaceofnumber@#1}{1em}%
    \global\cslet{Xlemmaseparator@#1}{\rbracket}%
    \csgdef{Xbeforelemmaseparator@#1}{0em}%
    \csgdef{Xafterlemmaseparator@#1}{0.5em}%
    \csgdef{Xinplaceoflemmaseparator@#1}{1em}%
    \csgdef{Xbeforenotes@#1}{1.2em \@plus .6em \@minus .6em}
    \csgdef{Xafterrule@#1}{0pt}
    \csgdef{Xtxtbeforenotes@#1}{}
    \csgdef{Xmaxhnotes@#1}{0.8\vsize}
    \newtoggle{Xnoteswidthliketwocolumns@#1}%
    \csgdef{Xparafootsep@#1}{}%
    \csgdef{Xafternote@#1}{1em plus.4em minus.4em}
```

XX.3.2 Create inserts, needed to add notes in foot

As regards inserts, see chapter 15 of *The TeXbook*by D. Knuth.

```
    \expandafter\newinsert\csname #1footins\endcsname%
    \unless\ifnoledgroup@%
      \expandafter\newinsert\csname mp#1footins\endcsname%
    \fi%
```

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.

```
  \global\notbool{parapparatus@}{\expandafter\newcommand\expandafter *}{\expandafter\new
        \ifnum\@edtext@level>0%
          \begingroup%
          \newcommand{\content}{##2}%
          \ifnumberedpar@%
              \ifledRcol%
                 \ifluatex%
                     \footnotelang@lua[R]%
                 \fi%
                 \@ifundefined{xpg@main@language}%if polyglossia
                        {}%
                        {\footnotelang@poly[R]}%
                  \footnoteoptions@[R]{##1}{true}%
                  \xright@appenditem{%
                    \noexpand\prepare@preXnotes{#1}%
                    \noexpand\prepare@edindex@fornote{\l@d@nums}%
             \unexpanded{\def\sw@list@inedtext}{\expandafter\unexpanded\expandafter{\sw@
                    \noexpand\csuse{v#1footnote}{#1}%
                  {{\l@d@nums}{\expandonce\@tag}{\expandonce\content}}%
                  }\to\inserts@listR
                  \footnoteoptions@[R]{##1}{false}%
                  \global\advance\insert@countR \@ne%
              \else%
                  \ifluatex%
                    \footnotelang@lua%
                  \fi%
                 \@ifundefined{xpg@main@language}%if polyglossia
                        {}%
                        {\footnotelang@poly}%
                  \footnoteoptions@{##1}{true}%
                  \xright@appenditem{%
                    \noexpand\prepare@preXnotes{#1}%
                    \noexpand\prepare@edindex@fornote{\l@d@nums}%
```

```
               \unexpanded{\def\sw@list@inedtext}{\expandafter\unexpanded\expandafter{\sw@inthisedtex
                      \noexpand\csuse{v#1footnote}{#1}%
                    {{\l@d@nums}{\expandonce\@tag}{\expandonce\content}}%
                     }\to\inserts@list
                    \global\advance\insert@count \@ne%
                    \footnoteoptions@{##1}{false}%
                \fi
            \else
                \csuse{v#1footnote}{#1}{{0|0|0|0|0|0|0}{}{##1}}%
            \fi%
            \endgroup%
          \else%
            \led@err@FootnoteWithoutEdtext%
          \fi%
   \ignorespaces%
          }
```

We need to be able to modify reledmac's footnote macros and restore their

```
      \global\csletcs{#1@@footnote}{#1footnote}
```

XX.3.4 Set standard display

```
      \Xarrangement@normal{#1}%
```

End of for critical footnotes.

```
    \fi
```

XX.4 For familiar footnotes

Familiar footnotes are those which end with letters. We look for the `nofamiliar` option of reledmac.

```
    \unless\ifnofamiliar@
```

XX.4.1 Options

```
      \newtoggle{parindentX@#1}
      \csgdef{hangindentX@#1}{0pt}%
      \csgdef{raggedX@#1}{}%
      \csgdef{hsizetwocolX@#1}{0.45 \hsize}%
      \csgdef{hsizethreecolX@#1}{.3 \hsize}%
      \csgdef{colalignX@#1}{\raggedright}%
      \csgdef{notenumfontX@#1}{\normalfont}%
      \csgdef{notefontsizeX@#1}{\footnotesize}%
      \csgdef{bhooknoteX@#1}{}%
      \csgdef{afterruleX@#1}{0pt}
      \csgdef{beforenotesX@#1}{1.2em \@plus .6em \@minus .6em}
      \csgdef{maxhnotesX@#1}{0.8\vsize}%
      \newtoggle{noteswidthliketwocolumnsX@#1}%
      \csgdef{parafootsepX@#1}{}%
      \csgdef{afternoteX@#1}{1em plus.4em minus.4em}
% End of for familiar footnotes.
% \subsubsection{Create inserts, needed to add notes in foot}
% As regards inserts, see chapter 15 of the TeXBook by D. Knuth.
%    \begin{macrocode}
      \expandafter\newinsert\csname footins#1\endcsname%
      \unless\ifnoledgroup@%
        \expandafter\newinsert\csname mpfootins#1\endcsname%
      \fi%
```

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.

```

      \global\expandafter\newcommand\csname footnote#1\endcsname[1]{%
          \begingroup%
              \prepare@prenotesX{#1}%
              \newcommand{\content}{##1}%
              \stepcounter{footnote#1}%
              \protected@csxdef{@thefnmark#1}{\csuse{thefootnote#1}}%
          \nottoggle{nomk@}%Nomk is set to true when using \footnoteXnomk with \parpackage
                  {\csuse{@footnotemark#1}}%
                  {}%
              \ifluatex%
                \xdef\footnote@luatextextdir{\the\luatextextdir}%
                \xdef\footnote@luatexpardir{\the\luatexpardir}%
              \fi%
              \csuse{vfootnote#1}{#1}{\expandonce\content}\m@mmf@prepare%
          \endgroup%
      }
```

Then define the counters.

```
      \newcounter{footnote#1}
     \global\expandafter\renewcommand\csname thefootnote#1\endcsname{\arabic{footnote#1}}
```

Do not forget to initialize series

```
\arrangementX@normal{#1}%
\fi
```

XX.5 The endnotes

Endnotes are commands like `\Xendnote`, where X is a series letter. First, we check for the `noend` options.

```
\unless\ifnoend@
```

XX.5.1 The main macro

The `\Xendnote` macro functions to write one endnote to the `.end` file. We change `\newlinechar` so that in the file every space becomes the start of a new line; this generally ensures that a long note doesn't exceed restrictions on the length of lines in files.

```

```

```
    \global\expandafter\newcommandx\csname #1endnote\endcsname[2][1,usedefault]{%
        \bgroup%
        \newlinechar='40%
        \global\@noneed@Footnotetrue%
        \newcommand{\content}{##2}%
        \immediate\write\l@d@end{%
          \expandafter\string\csname #1end\endcsname%
          {\ifnumberedpar@\l@d@nums\fi}%
          {\ifnumberedpar@\expandonce\@tag\fi}%
          {\expandonce\content}%
          {#1}%
          {##1}%
          \@percentchar%
        }%
        \egroup%
        \ignorespaces%
    }%
```

`\Xendnote` commands called `\Xend` commands on to the endnote file; these are analogous to the various `footfmt` commands above, and they take the same arguments. When we process this file, we want to pick out the notes of one series and ignore all the rest. To do that, we equate the end command for the series we want to `\endprint`, and leave the rest equated to `\@gobblefive`, which just skips over its five arguments.

```

     \global\cslet{#1end}{\@gobblefive}
```

We need to store the number of times `\doendnotesbysection` is called for one series.

```
    \global\expandafter\newcount\csname #1end@bysection\endcsname%
```

XX.5.2 The options

```
     \csgdef{Xendtwolines@#1}{}%
     \csgdef{Xendmorethantwolines@#1}{}%
     \newtoggle{Xendtwolinesbutnotmore@#1}{}%
     \newtoggle{Xendtwolinesonlyinsamepage@#1}{}%
     \newtoggle{Xendlemmadisablefontselection@#1}%
     \csgdef{Xendnotenumfont@#1}{\normalfont}%
     \csgdef{Xendnotefontsize@#1}{\footnotesize}%
     \csgdef{Xendbhooknote@#1}{}%

     \csgdef{Xendboxlinenum@#1}{0pt}%
     \csgdef{Xendboxlinenumalign@#1}{L}%

     \csgdef{Xendboxstartlinenum@#1}{0pt}%
     \csgdef{Xendboxendlinenum@#1}{0pt}%

     \csgdef{Xendlemmaseparator@#1}{}%
     \csgdef{Xendbeforelemmaseparator@#1}{0em}%
     \csgdef{Xendafterlemmaseparator@#1}{0.5em}%
     \csgdef{Xendinplaceoflemmaseparator@#1}{0.5em}%

     \newtoggle{Xendparagraph@#1}%
     \csgdef{Xendafternote@#1}{1em plus.4em minus.4em}%
     \csgdef{Xendsep@#1}{}%
```

End of endnotes declaration

```
  \fi%
```

Dump series in `\@series`

```
    \listxadd{\@series}{#1}
  }
}% End of \newseries
```

XX.6 Init standards series (A,B,C,D,E,Z)

```
\expandafter\newseries\expandafter{\default@series}
```

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.

```
\newcommand{\seriesatbegin}[1]{%
   \StrDel{\@series}{#1}[\@series]%
   \edef\@new{}%
   \listeadd{\@new}{#1}%
   \listeadd{\@new}{\@series}%
   \xdef\@series{\@new}%
}
```

`\seriesatend`

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

```
\newcommand{\seriesatend}[1]{%
   \StrDel{\@series}{#1}[\@series]%
   \edef\@new{}%
   \listeadd{\@new}{\@series}%
   \listeadd{\@new}{#1}%
   \xdef\@series{\@new}%
}
```

XXI.2 Test series order

`\ifseriesbefore`

`\ifseriesbefore{`⟨*seriesA*⟩`}{`⟨*seriesB*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}` expands ⟨*true*⟩ if ⟨*seriesA*⟩ is printed before ⟨*seriesB*⟩, expands ⟨*false*⟩ otherwise.

```
\newcommand{\ifseriesbefore}[4]{%
 \StrPosition{\@series}{#1}[\@first]%
 \StrPosition{\@series}{#2}[\@second]%
 \ifnumgreater{\@second}{\@first}{#3}{#4}%
}
```

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}{%
 \ifdefempty{\@series}%
   {\xdef\@firstseries{}}%
   {\StrChar{\@series}{1}[\@firstseries]}%
}%
```

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`.

```
\newcommandx{\settoggle@series}[5][4,5,usedefault]{%
    \def\do##1{%
      \global\settoggle{#2@##1}{#3}%
      \ifstrequal{#4}{reload}%
        {%
        \csuse{Xarrangement@\csuse{series@display##1}}{##1}%
        \csuse{arrangementX@\csuse{series@displayX##1}}{##1}%
        }%
        {}%
      }%
    \ifstrempty{#1}{%
              \dolistloop{\@series}%
              \ifstrempty{#5}{}{%
                 \docsvlist{#5}%
               }
           }%
           {%
              \docsvlist{#1}%
           }%
}
```

\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`.

```
\newcommandx{\setcommand@series}[5][4,5,usedefault]{%
    \def\do##1{
        \csgdef{#2@##1}{#3}
    \ifstrequal{#4}{reload}{
        \csuse{Xarrangement@\csuse{series@display##1}}{##1}%
        \csuse{arrangementX@\csuse{series@displayX##1}}{##1}%
    }{}}
    \ifstrempty{#1}{%
              \dolistloop{\@series}%
              \ifstrempty{#5}{}{%
                 \docsvlist{#5}
               }
    }%
    {%
```

```
                \docsvlist{#1}%
     }%
}%
```

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`.

```
\newcommandx{\newhookcommand@series}[2][2,usedefault]{%
  \global\expandafter\newcommand\expandafter*\csname #1\endcsname[2][]{%
    \setcommand@series{##1}{#1}{##2}[][#2]%
  }%
  \ifstrempty{#2}{}{%
    \def\do##1{%
    \global\expandafter\newcommand\expandafter*\csname #1##1\endcsname[1]{%
        \csuse{#1}[##1]{####1}%
      }%
    }%
    \docsvlist{#2}%
  }%
}
```

\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`.

```
\newcommandx{\newhooktoggle@series}[2][2,usedefault]{%
  \global\expandafter\newcommandx\expandafter*\csname #1\endcsname[2][1,2={true},usedefault]{%
    \settoggle@series{##1}{#1}{##2}[][#2]%
    }%
  \ifstrempty{#2}{}{%
    \def\do##1{%
     \global\expandafter\newcommand\expandafter*\csname #1##1\endcsname{%
        \csuse{#1}[##1]%
      }%
    }%
    \docsvlist{#2}%
  }%
}
```

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.

```
\newcommand{\newhooktoggle@series@reload}[1]{%
  \global\expandafter\newcommandx\expandafter*\csname #1\endcsname[2][1,2={true},usedefault]{%
    \settoggle@series{##1}{#1}{##2}[reload]%
    }%
}%
```

\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.

```
\newcommand{\newhookcommand@series@reload}[1]{%
  \global\expandafter\newcommand\expandafter*\csname #1\endcsname[2][]{%
    \setcommand@series{##1}{#1}{##2}[reload]%
  }%
}
```

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.

```
\unless\ifnocritical@
  \newhooktoggle@series{Xparindent}
  \newhookcommand@series{Xtwolines}[appref]
  \newhookcommand@series{Xmorethantwolines}[appref]
  \newhooktoggle@series{Xtwolinesbutnotmore}[appref]
  \newhooktoggle@series{Xtwolinesonlyinsamepage}[appref]
  \newhookcommand@series{Xhangindent}
  \newhookcommand@series{Xragged}
  \newhookcommand@series{Xhsizetwocol}
  \newhookcommand@series{Xhsizethreecol}
  \newhookcommand@series{Xcolalign}%
  \newhookcommand@series{Xnotenumfont}
  \newhookcommand@series{Xbhooknote}
  \newhookcommand@series{Xboxsymlinenum}%
  \newhookcommand@series{Xsymlinenum}
  \newhookcommand@series{Xbeforenumber}
  \newhookcommand@series{Xafternumber}
  \newhookcommand@series{Xbeforesymlinenum}
  \newhookcommand@series{Xaftersymlinenum}
  \newhookcommand@series{Xinplaceofnumber}
  \newhookcommand@series{Xlemmaseparator}
  \newhookcommand@series{Xbeforelemmaseparator}
  \newhookcommand@series{Xafterlemmaseparator}
  \newhookcommand@series{Xinplaceoflemmaseparator}
  \newhookcommand@series{Xtxtbeforenotes}
  \newhookcommand@series@reload{Xafterrule}
  \newhooktoggle@series{Xnumberonlyfirstinline}
  \newhooktoggle@series{XnumberonlyfirstinXtwolines}
  \newhooktoggle@series{Xnonumber}
  \newhooktoggle@series{Xpstart}
  \newhooktoggle@series{Xpstarteverytime}%
  \newhooktoggle@series{Xonlypstart}
  \newhooktoggle@series{Xnonbreakableafternumber}
  \newhooktoggle@series{Xlemmadisablefontselection}
  \newhookcommand@series@reload{Xmaxhnotes}
  \newhookcommand@series@reload{Xbeforenotes}
  \newhooktoggle@series@reload{Xnoteswidthliketwocolumns}%
  \newhookcommand@series{Xnotefontsize}

  \newhookcommand@series{Xboxlinenum}%
  \newhookcommand@series{Xboxlinenumalign}%

  \newhookcommand@series{Xboxstartlinenum}%
  \newhookcommand@series{Xboxendlinenum}%

  \newhookcommand@series{Xafternote}%
  \newhookcommand@series{Xparafootsep}

\fi
```

XXI.3.5 Options for familiar notes

Before generating the optional commands for familiar notes, we check the `\nofamiliar` option.

```
\unless\ifnofamiliar@
  \newhooktoggle@series{parindentX}
  \newhookcommand@series{hangindentX}
  \newhookcommand@series{raggedX}
  \newhookcommand@series{hsizetwocolX}
  \newhookcommand@series{hsizethreecolX}
  \newhookcommand@series{colalignX}%
  \newhookcommand@series{notenumfontX}
  \newhookcommand@series{bhooknoteX}
  \newhookcommand@series@reload{beforenotesX}
  \newhookcommand@series@reload{maxhnotesX}
  \newhooktoggle@series@reload{noteswidthliketwocolumnsX}%
  \newhookcommand@series@reload{afterruleX}
  \newhookcommand@series{notefontsizeX}
  \newhookcommand@series{afternoteX}
  \newhookcommand@series{parafootsepX}
\fi
```

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.

```
\unless\ifnoend@
  \newhookcommand@series{Xendtwolines}[apprefwithpage]
  \newhookcommand@series{Xendmorethantwolines}[apprefwithpage]
  \newhooktoggle@series{Xendtwolinesbutnotmore}[apprefwithpage]
  \newhooktoggle@series{Xendtwolinesonlyinsamepage}[apprefwithpage]
  \newhookcommand@series{Xendnotenumfont}
  \newhookcommand@series{Xendbhooknote}
```

```

\newhookcommand@series{Xendboxlinenum}%
\newhookcommand@series{Xendboxlinenumalign}%

\newhookcommand@series{Xendboxstartlinenum}%
\newhookcommand@series{Xendboxendlinenum}%

\newhookcommand@series{Xendnotefontsize}
\newhooktoggle@series{Xendlemmadisablefontselection}
\newhookcommand@series{Xendlemmaseparator}
\newhookcommand@series{Xendbeforelemmaseparator}
\newhookcommand@series{Xendafterlemmaseparator}
\newhookcommand@series{Xendinplaceoflemmaseparator}

\newhooktoggle@series{Xendparagraph}
\newhookcommand@series{Xendafternote}
\newhookcommand@series{Xendsep}
\fi
```

XXI.4 Hooks for a particular footnote

`\fulllines@` `\fulllines@` toggle is used to print the full lines references, and not the abbreviated form defined by `\Xtwolines` and `\Xmorethantwolines`.

```
\newtoggle{fulllines@}%
```

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

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

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

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

`\nomk@` `\nomk@` toggle is used by reledpar to remove the footnote mark in the text when using `\footnoteXmk`. Read reledpar handbook.

```
\newtoggle{nomk@}%
```

XXI.5 Alias

`\noXlemmaseparator` `\noXlemmaseparator[⟨series⟩]` is just an alias for `\Xlemmaseparator[⟨series⟩]{}`.

```
\newcommandx*{\noXlemmaseparator}[1][1]{\Xlemmaseparator[#1]{}}
```

XXII Output routine

Now we begin the output routine and associated things.

XXII.0.1 Page number management

\pageno \advancepageno

\pageno is a page number, starting at 1, and \advancepageno increments the number.

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

```

XXII.0.2 Extra footnotes output

With luck we might only have to change \@makecol and \@reinserts of the LaTeX's kernel. Since reledmac, we use etoolbox's patching commands instead of overriding. It should provides better compatibility with other package which modify these 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.

```
\newcommand*{\l@ddoxtrafeet}{%
  \IfStrEq{familiar-critical}{\@fnpos}
    {\do@feetX\Xdo@feet}%
    {%
    \IfStrEq{critical-familiar}{\@fnpos}%
        {\Xdo@feet\do@feetX}%
        {\do@feetX\Xdo@feet}%
    }%
}%

```

\Xdo@feet

\Xdo@feet is the code extending \@makecol to cater for the extra critical feet.

```
\newcommand*{\Xdo@feet}{%
  \setbox\@outputbox \vbox{%
    \unvbox\@outputbox
    \@opXfeet}}
```

\@opXfeet \print@Xnotes

The extra critical feet to be added to the output. The normal way to add one series, \print@Xnotes, is replaced by reledpar when using \Pages.

```
\newcommand\print@Xnotes[1]{%
  \csuse{#1footstart}{#1}%
  \csuse{#1footgroup}{#1}%%
}%
```

We print all series of notes by looping on them. We check before printing them that they are not voided.

```
\newcommand*{\@opXfeet}{%
  \unless\ifnocritical@%
    \gdef\firstXseries@{}%
    \def\do##1{%
      \ifvoid\csuse{##1footins}\else%
```

```
          \global\skip\csuse{##1footins}=\csuse{Xbeforenotes@##1}%
         \global\advance\skip\csuse{##1footins} by\csuse{Xafterrule@##1}%
          \print@Xnotes{##1}%
       \fi%
      }%
    \dolistloop{\@series}%
  \fi%
}%
```

\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`.

```
\newcommand*{\l@ddodoreinxtrafeet}{%
  \IfStrEq{familiar-critical}{\@fnpos}
    {\@doreinfeetX\X@doreinfeet}%
    {%
    \IfStrEq{critical-familiar}{\@fnpos}%
        {\X@doreinfeet\@doreinfeetX}%
        {\@doreinfeetX\X@doreinfeet}%
    }%
}

```

\X@doreinfeet

`\X@doreinfeet` is the code for catering for the extra critical footnotes within `\@reinserts`.

```
\newcommand*{\X@doreinfeet}{%
  \unless\ifnocritical@%
    \def\do##1{%
      \ifvoid\csuse{##1footins}\else%
        \insert\csuse{##1footins}{\unvbox\csuse{##1footins}}%
      \fi}%
    \dolistloop{\@series}
  \fi%
}

```

\print@notesX
\do@feetX
\@doreinfeetX

We have to add all the new kinds of familiar footnotes to the output routine. The normal way to add one series. `\print@Xnotes` is replaced by reledpar when using `\Pages`.

```
\newcommand\print@notesX[1]{%
  \csuse{footstart#1}{#1}%
  \csuse{footgroup#1}{#1}%
}%
```

We print all the series of notes by looping on them. We check before printing them that they are not voided.

```
\newcommand*{\do@feetX}{%
  \unless\ifnofamiliar@%
    \gdef\firstseriesX@{}%
    \setbox\@outputbox \vbox{%
      \unvbox\@outputbox%
      \def\do##1{%
      \ifvoid\csuse{footins##1}\else%
```

```
          \global\skip\csuse{footins##1}=\csuse{beforenotesX@##1}%
         \global\advance\skip\csuse{footins##1} by\csuse{afterruleX@##1}%
          \print@notesX{##1}%
      \fi%
      }%
      \dolistloop{\@series}}%
  \fi%
}%

\newcommand{\@doreinfeetX}{%
  \unless\ifnofamiliar@%
    \def\do##1{%
      \ifvoid\csuse{footins##1}\else
        \insert%
          \csuse{footins##1}
          {\unvbox\csuse{footins##1}}%
       \fi%
    }%
    \dolistloop{\@series}%
  \fi%
}%

```

XXII.0.3 Standard output's commands patching

The memoir class does not use the 'standard' versions of `\@makecol` and `\@reinserts`, due to its sidebar insert. We had better add that code if memoir is used. (It can be awkward dealing with `\if` code within `\if` code, so don't use `\ifl@dmemoir` here.)

```
\@ifclassloaded{memoir}{%
```

memoir is loaded so we use memoir's built in hooks.

```
  \g@addto@macro{\m@mdoextrafeet}{\l@ddoxtrafeet}%
  \g@addto@macro{\m@mdodoreinextrafeet}{\l@ddodoreinxtrafeet}%
  }{%
```

memoir has not been loaded, so patch `\@makecol` and `\@reinserts`.

```
  \@ifpackageloaded{fancyhdr}{%
    \patchcmd%
      {\latex@makecol}%
      {\xdef\@freelist{\@freelist\@midlist}}%
      {\xdef\@freelist{\@freelist\@midlist}\l@ddoxtrafeet}%
      {}%
      {\led@error@fail@patch@@makecol}%
    }{%
    \patchcmd%
      {\@makecol}%
      {\xdef\@freelist{\@freelist\@midlist}}%
      {\xdef\@freelist{\@freelist\@midlist}\l@ddoxtrafeet}%
      {}%
      {\led@error@fail@patch@@makecol}%
    }%

  \patchcmd%
    {\@reinserts}%
    {\ifvbox}%
    {\l@ddodoreinxtrafeet\ifvbox}%
    {}%
    {\led@error@fail@patch@@reinserts}%
}

```

It turns out that `\@doclearpage` also needs modifying.

`\if@led@nofoot` We have to check if there are any leftover feet.

```
\newif\if@led@nofoot

\@ifclassloaded{memoir}{%
```

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

`\@mem@extranofeet`

```
\g@addto@macro{\@mem@extranofeet}{%%
  \def\do#1{%
    \unless\ifnocritical@%
      \ifvoid\csuse{#1footins}\else\@mem@nofootfalse\fi%
    \fi%
    \unless\ifnofamiliar@%
      \ifvoid\csuse{footins#1}\else\@mem@nofootfalse\fi%
    \fi%
    }
  \dolistloop{\@series}%
  }%
}{%
```

As memoir is not loaded we have patch `\@doclearpage`.

`\@led@testifnofoot`
`\@doclearpage`

```
\newcommand*{\@led@testifnofoot}{%
  \@led@nofoottrue%
  \ifvoid\footins\else%
    \@led@nofootfalse%
  \fi%
  \def\do##1{%
    \unless\ifnocritical@%
      \ifvoid\csuse{##1footins}\else%
        \@led@nofootfalse%
      \fi%
    \fi%
    \unless\ifnofamiliar@%
      \ifvoid\csuse{footins##1}\else%
```

```
        \@led@nofootfalse%
      \fi%
    \fi%
  }%
  \dolistloop{\@series}%
}%

\pretocmd%
 {\@doclearpage}%
 {\@led@testifnofoot}%
 {}%
 {\led@error@fail@patch@@doclearpage}%

\patchcmd%
 {\@doclearpage}%
 {\ifvoid\footins}%
 {\if@led@nofoot}%
 {}%
 {\led@error@fail@patch@@doclearpage}%

}

```

XXIII Cross referencing

You can mark a place in the text using a command of the form `\edlabel{`⟨*foo*⟩`}`, and later refer to it using the label ⟨*foo*⟩ by saying `\edpageref{`⟨*foo*⟩`}`, or `\lineref{`⟨*foo*⟩`}` or `\sublineref{`⟨*foo*⟩`}` or `\pstartref`. These reference commands will produce, respectively, the page, line sub-line and pstart on which the `\edlabel{`⟨*foo*⟩`}` command occurred.

The reference macros warn you if a reference is made to an undefined label. If `{`⟨*foo*⟩`}` has been used as a label before, the `\edlabel{`⟨*foo*⟩`}` command will issue a complaint; subsequent `\edpageref` and `\edlineref` commands will refer to the latest occurrence of `\edlabel{`⟨*foo*⟩`}`.

`\labelref@list` Set up a new list, `\labelref@list`, to hold the page, line and sub-line numbers for each label.

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

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

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

```

`\edlabel` The `\edlabel` command first writes a `\@lab` macro to the `\linenum@out` file. It then checks to see that the `\labelref@list` actually has something in it (if not, it creates a dummy entry), and pops the next value for the current label, storing it in `\label@refs`.

Finally it defines the label to be `\empty` so that any future check will turn up the fact that it has been used.[31]

This version of the original edmac `\label` uses `\@bsphack` and `\@esphack` to eliminate extra space problems and also use the LaTeX write methods for the `.aux` file.

Jesse Billett[32] found that the original code could be off by several pages. This version, hopefully cures that, and also allows for non-arabic page numbering.

```
\newcommand*{\edlabel}[1]{%
  \ifl@dpairing\ifautopar%
    \strut%
  \fi\fi%
  \@bsphack%
  \ifledRcol%
    \write\linenum@outR{\string\@lab}%
    \ifx\labelref@listR\empty%
      \xdef\label@refs{\zz@@@}%
    \else%
      \gl@p\labelref@listR\to\label@refs%
    \fi%
    \ifvmode%
        \advancelabel@refs%
    \fi%
```

Use code from the kernel `\label` command to write the correct page number. Also define an hypertarget if hyperref package is loaded.

```
    \protected@write\@auxout{}%
    {\string\l@dmake@labelsR\space\thepage|\label@refs|\the\c@pstartR|{#1}}%
    \ifdef{\hypertarget}{\Hy@raisedlink{\hypertarget{#1}{}}}{}%
  \else%
    \write\linenum@out{\string\@lab}%
    \ifx\labelref@list\empty%
      \xdef\label@refs{\zz@@@}%
    \else%
      \gl@p\labelref@list\to\label@refs%
    \fi%
    \ifvmode%
        \advancelabel@refs%
    \fi%
  \protected@write\@auxout{}%
   {\string\l@dmake@labels\space\thepage|\label@refs|\the\c@pstart|{#1}}%
    \ifdef{\hypertarget}{\Hy@raisedlink{\hypertarget{#1}{}}}{}%
  \fi%
\@esphack}%

```

`\advancelabel@refs`
`\labelrefsparseline`
`\labelrefsparsesubline`

In cases where `\edlabel` is the first element in a paragraph, we have a problem with line counts, because line counts change only at the first horizontal box of the paragraph.

[31]The remaining macros in this section were kindly revised by Wayne Sullivan, who substantially improved their efficiency and flexibility.

[32](`jdb43@cam.ac.uk`) via the `ctt` thread ‘ledmac cross referencing’, 25 August 2003.

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.

```
\newcounter{line}%
\newcounter{subline}%
\newcommand{\advancelabel@refs}{%
    \setcounter{line}{\expandafter\labelrefsparseline\label@refs}%
    \stepcounter{line}%
    \ifsublines@%
     \setcounter{subline}{\expandafter\labelrefsparsesubline\label@refs}%
        \stepcounter{subline}{1}%
        \def\label@refs{\theline|\thesubline}%
    \else%
        \def\label@refs{\theline|0}%
    \fi%
}
\def\labelrefsparseline#1|#2{#1}
\def\labelrefsparsesubline#1|#2{#2}
```

`\l@dmake@labels` The `\l@dmake@labels` macro gets executed when the labels file is read. For each label it defines a macro, whose name is made up partly from the label you supplied, that contains the page, line and sub-line numbers. But first it checks to see whether the label has already been used (and complains if it has).

The initial use of `\newcommand` is to catch if `\l@dmake@labels` has been previously defined (by a class or package).

#1 page number, #2 line number, #3 sub-line number, #4 pstart number, #5 label.

```
\newcommand*{\l@dmake@labels}{}
\def\l@dmake@labels#1|#2|#3|#4|#5{%
  \expandafter\ifx\csname the@label#5\endcsname \relax\else
    \led@warn@DuplicateLabel{#5}%
  \fi
  \expandafter\gdef\csname the@label#5\endcsname{#1|#2|#3|#4}%
  \ignorespaces}

```

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.

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

```

`\@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`.

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

```

`\applabel` `\applabel`, if called in `\edtext` will insert automatically both a start and an end label for the current edtext lines.

```
\newcommand*{\applabel}[1]{%
  \ifnum\@edtext@level>0%
```

Label should not be already defined.

```
      \ifcsundef{the@label#1}{%
        \csdef{the@label#1}{applabel}%
       }%
       {%
         \led@warn@DuplicateLabel{#1 (applabel)}%
       }%
```

Parse the `\edtext` line numbers.

```
     \expandafter\l@dp@rsefootspec\l@d@nums|%
```

Use the LaTeX standard hack for label.

```
     \@bsphack%
```

And now, write the data in the auxiliary file.

```
      \ifledRcol%
        \protected@write\@auxout{}%
       {\string\l@dmake@labelsR\space\l@dparsedstartpage|\l@dparsedstartline|\l@dparsedsta
        \ifdef{\hypertarget}{\Hy@raisedlink{\hypertarget{#1:start}{}}}{}%
        \protected@write\@auxout{}%
       {\string\l@dmake@labelsR\space\l@dparsedendpage|\l@dparsedendline|\l@dparsedendsub|
      \else%
        \protected@write\@auxout{}%
       {\string\l@dmake@labels\space\l@dparsedstartpage|\l@dparsedstartline|\l@dparsedstar
        \ifdef{\hypertarget}{\Hy@raisedlink{\hypertarget{#1:start}{}}}{}%
        \protected@write\@auxout{}%
       {\string\l@dmake@labels\space\l@dparsedendpage|\l@dparsedendline|\l@dparsedendsub|\
      \fi%
```

Use the LaTeX standard hack for label.

```
      \@esphack%
```

Warning if `\applabel` is called outside of `\edtext`.

```
  \else%
    \led@warn@AppLabelOutEdtext{#1}%
  \fi%
```

End of `\applabel`

```
}%
```

\wrap@edcrossref

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

```
\newrobustcmd{\wrap@edcrossref}[2]{%
  \ifdef{\hyperlink}%
    {\hyperlink{#1}{#2}}%
    {#2}%
}
```

\edpageref
\xpageref

If the specified label exists, `\edpageref` gives its page number.

For this reference command, as for the other two, a special version with prefix x is provided for use in places where the command is to be scanned as a number, as in `\linenum`. These special versions have two limitations: they do not print error messages if the reference is unknown, and they can't appear as the first label or reference command in the file; you must ensure that a `\edlabel` or a normal reference command appears first, or these x-commands will always return zeros.

LaTeX already defines a `\pageref`, so changing the name to `\edpageref`.

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

```

\edlineref
\xlineref

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

```
\newcommand*{\edlineref}[1]{\l@dref@undefined{#1}\wrap@edcrossref{#1}{\l@dgetref@num{2}{#1}}}%
\newcommand*{\xlineref}[1]{\l@dgetref@num{2}{#1}}%

```

\sublineref
\xsublineref

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

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

```

\pstarteref
\xpstartref

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

```
\newcommand*{\pstartref}[1]{\l@dref@undefined{#1}\wrap@edcrossref{#1}{\l@dgetref@num{4}{#1}}}
\newcommand*{\xpstartref}[1]{\l@dgetref@num{4}{#1}}

```

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.

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

```

\l@dgetref@num Next, \l@dgetref@num fetches the number we want. It has two arguments: the first is simply a digit, specifying whether to fetch a page (1), line (2), sub-line (3) or (4) pstart number. (This switching is done by calling \l@dlabel@parse.) The second argument is the label-macro, which because of the \@lab macro above is defined to be a string of the type 123|456|789.

```
\newcommand*{\l@dgetref@num}[2]{%
  \expandafter
  \ifx\csname the@label#2\endcsname \relax
    000%
  \else
    \expandafter\expandafter\expandafter
    \l@dlabel@parse\csname the@label#2\endcsname|#1%
  \fi}

```

\l@dlabel@parse Notice that we slipped another | delimiter into the penultimate line of \l@dgetref@num, to keep the 'switch-number' separate from the reference numbers. This | is used as another parameter delimiter by \l@dlabel@parse, which extracts the appropriate number from its first arguments. The |-delimited arguments consist of the expanded label-macro (three reference numbers), followed by the switch-number (1, 2, 3 or 4) which defines which of the earlier three numbers to pick out. (It was earlier given as the first argument of \l@dgetref@num.)

```
\newcommand*{\l@dlabel@parse}{}
\def\l@dlabel@parse#1|#2|#3|#4|#5{%
  \ifcase #5%
  \or #1%
  \or #2%
  \or #3%
  \or #4%
  \fi}
```

\xxref The \xxref command takes two arguments, both of which are labels, e.g., \xxref{mouse}{elephant}. It first does some checking to make sure that the labels do exist (if one does not, those numbers are set to zero). Then it calls \linenum and sets the beginning page, line, and sub-line numbers to those of the place where \label{mouse} was placed, and the ending numbers to those at {elephant}. The point of this is to be able to manufacture footnote line references to passages which can not be specified in the normal way as the first argument to \edtext for one reason or another. Using \xxref in the second argument of \edtext lets you set things up at least semi-automatically.

```
\newcommand*{\xxref}[2]{%
  {%
   \expandafter\ifx\csname the@label#1\endcsname \relax%
     \expandafter\let\csname the@@label#1\endcsname\zz@@@%
    \else%
    \expandafter\def\csname the@@label#1\endcsname{\l@dgetref@num{1}{#1}|\l@dgetref@num{2}
   \fi%
   \expandafter\ifx\csname the@label#2\endcsname \relax%
     \expandafter\let\csname the@@label#2\endcsname\zz@@@%
   \else%
    \expandafter\def\csname the@@label#2\endcsname{\l@dgetref@num{1}{#2}|\l@dgetref@num{2}{#2}|\l@dge
   \fi%
   \ifdefined\@Rlineflag%
   \StrDel{\csuse{the@@label#1}}{\@Rlineflag}[\@tempa]%
     \StrDel{\csuse{the@@label#2}}{\@Rlineflag}[\@tempb]%
  \else%
   \letcs{\@tempa}{the@@label#1}%
   \letcs{\@tempb}{the@@label#2}%
 \fi%
   \linenum{\@tempa|%
    \@tempb}}}%

```

\appref
\apprefwithpage
\apprefprefixsingle
\apprefprefixmore

\appref prints a crossref to some lines of the apparatus defined by \applabel. It prints the lines as they should be printed in the apparatus.

If \apprefprefixsingle is not empty, it prints it before the line number. If \apprefprefixmore is not empty, it prints it before the line numbers when the first line is not the same as the last line. \apprefwithpage prints a crossref to some lines of the apparatus defined by \applabel. It always prints the page number, as it should be printed in the end notes. The \Xtwolinesappref and \Xmorethantwolinesappref are similar to the footnote hooks and \Xtwolines \Xmorethantwolines.

So, first declare the default value of the hooks for the pseudo-series appref. Also declare the internal toggle which are switch by reledmac.

```
\xdef\Xtwolines@appref{}%
\xdef\Xmorethantwolines@appref{}%
\newtoggle{Xtwolinesbutnotmore@appref}%
\newtoggle{Xtwolinesonlyinsamepage@appref}%

\xdef\Xendtwolines@apprefwithpage{}%
\xdef\Xendmorethantwolines@apprefwithpage{}%
\newtoggle{Xendtwolinesbutnotmore@apprefwithpage}%
\newtoggle{Xendtwolinesonlyinsamepage@apprefwithpage}%

```

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.

```

\xdef\Xboxstartlinenum@appref{0pt}
\xdef\Xboxendlinenum@appref{0pt}

\xdef\Xendboxstartlinenum@apprefwithpage{0pt}
\xdef\Xendboxendlinenum@apprefwithpage{0pt}

```

Now, declare the default value of \apprefprefixsingle and \apprefprefixmore.

```
\newcommand\apprefprefixsingle{}%
\newcommand\apprefprefixmore{}%

```

And now, the main commands: `\appref` and `\apprefwithpage`. These commands call `\printlines` and `\printendlines`. That is why we have previously declared all hooks values tested inside these last commands.

```
\newcommandx{\appref}[2][1,usedefault]{%
  \IfStrEq{#1}{fulllines}%
    {\toggletrue{fulllines@}}%
    {}%
  \xdef\@currentseries{appref}%
  \ifdefempty{\apprefprefixmore}%
    {\apprefprefixsingle}%
    {%
      \IfEq{\xlineref{#2:start}}{\xlineref{#2:end}}%
        {\apprefprefixsingle}%
        {\apprefprefixmore}%
    }%
  \printlines\xpageref{#2:start}|\xlineref{#2:start}|\xsublineref{#2:start}|\xpageref{#2:en
  \togglefalse{fulllines@}%
}%

% \changes{v1.23.0}{2015/05/18}{Debug \cs{Xendtwolines}, \cs{Xendmorethantwolines}, \cs{Xen
\newcommandx{\apprefwithpage}[2][1,usedefault]{%
  \IfStrEq{#1}{fulllines}%
    {\toggletrue{fulllines@}}%
    {}%
  \xdef\@currentseries{apprefwithpage}%
  \printendlines\xpageref{#2:start}|\xlineref{#2:start}|\xsublineref{#2:start}|\xpageref{#
  \togglefalse{fulllines@}%
}%
```

`\edmakelabel` Sometimes the `\edlabel` command cannot be used to specify exactly the page and line desired; you can use the `\edmakelabel` macro make your own label. For example, if you say `\edmakelabel{elephant}{10|25|0}` you will have created a new label, and a later call to `\edpageref{elephant}` would print '10' and `\lineref{elephant}` would print '25'. The sub-line number here is zero. `\edmakelabel` takes a label, followed by a page and a line number(s) as arguments. LaTeX defines a `\makelabel` macro which is used in lists. Peter Wilson has changed the name to `\edmakelabel`.

```
\newcommand*{\edmakelabel}[2]{\expandafter\xdef\csname the@label#1\endcsname{#2}}

```

(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. 92 and V.9 p. 69), since `\xxref` makes a call to `\linenum` in order to do its work.)

XXIV Side notes

Regular `\marginpar`s 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 `\marginpar`s in numbered text do not disturb the flow.

```
\pretocmd{\@xympar}%
  {\ifnumberedpar@
    \led@warn@NoMarginpars
    \@esphack
  \else}%
  {}%
  {}%

\apptocmd{\@xympar}%
  {\fi}%
  {}
  {}

```

We provide side notes as replacement for `\marginpar` in numbered text.

`\sidenote@margin` `\sidenotemargin` `\l@dgetsidenote@margin` These are the sidenote equivalents to `\line@margin` and `\linenummargin` for specifying which margin. The default is the right margin (opposite to the default for line numbers). `\l@dgetsidenote@margin` returns the number associated to side note margin:

left : 0

right : 1

outer : 2

inner : 3

```
\newcount\sidenote@margin
\newcommand*{\sidenotemargin}[1]{{%
  \l@dgetsidenote@margin{#1}%
  \ifnum\@l@dtempcntb>\m@ne
    \ifledRcol
      \global\sidenote@marginR=\@l@dtempcntb
    \else
      \global\sidenote@margin=\@l@dtempcntb
    \fi
  \fi}}
\newcommand*{\l@dgetsidenote@margin}[1]{%
  \def\@tempa{#1}\def\@tempb{left}%
  \ifx\@tempa\@tempb
    \@l@dtempcntb \z@
  \else
    \def\@tempb{right}%
    \ifx\@tempa\@tempb
      \@l@dtempcntb \@ne
    \else
```

```
      \def\@tempb{outer}%
      \ifx\@tempa\@tempb
        \@l@dtempcntb \tw@
      \else
        \def\@tempb{inner}%
        \ifx\@tempa\@tempb
          \@l@dtempcntb \thr@@
        \else
          \led@warn@BadSidenotemargin
          \@l@dtempcntb \m@ne
        \fi
      \fi
    \fi
  \fi}
\sidenotemargin{right}

```

\l@dlp@rbox
\l@drp@rbox

We need two boxes to store sidenote texts.

```
\newbox\l@dlp@rbox
\newbox\l@drp@rbox

```

\ledlsnotewidth
\ledrsnotewidth
\ledlsnotesep
\ledrsnotesep
\ledlsnotefontsetup
\ledrsnotefontsetup

These specify the width of the left/right boxes (initialised to \marginparwidth), their distance from the text (initialised to \linenumsep), and the fonts used.

```
\newdimen\ledlsnotewidth \ledlsnotewidth=\marginparwidth
\newdimen\ledrsnotewidth \ledrsnotewidth=\marginparwidth
\newdimen\ledlsnotesep \ledlsnotesep=\linenumsep
\newdimen\ledrsnotesep \ledrsnotesep=\linenumsep
\newcommand*{\ledlsnotefontsetup}{\raggedleft\footnotesize}
\newcommand*{\ledrsnotefontsetup}{\raggedright\footnotesize}

```

\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.

```
\newcommand*{\ledleftnote}[1]{\edtext{}{\l@dlsnote{#1}}}
\newcommand*{\ledrightnote}[1]{\edtext{}{\l@drsnote{#1}}}

\newcommand*{\ledinnernote}[1]{%
  \ifodd\c@page% Do not use \page@num, because it is not yet calculated when command is called
    \ledleftnote{#1}%
  \else%
    \ledrightnote{#1}%
  \fi%
}

\newcommand*{\ledouternote}[1]{%
  \ifodd\c@page% Do not use \page@num, because it is not yet calculated when command is called
```

```
    \ledrightnote{#1}%
  \else%
    \ledleftnote{#1}%
  \fi%
}

\newcommand*{\ledsidenote}[1]{\edtext{}{\l@dcsnote{#1}}}
```

\l@dlsnote
\l@drsnote
\l@dcsnote

. The 'footnotes' for left, right, and moveable sidenotes. The whole scheme is reminiscent of the critical footnotes code.

```
\newif\ifrightnoteup
  \rightnoteuptrue

\newcommand*{\l@dlsnote}[1]{%
  \begingroup%
  \newcommand{\content}{#1}%
  \ifnumberedpar@
    \ifledRcol%
      \xright@appenditem{\noexpand\vl@dlsnote{\expandonce\content}}%
                         \to\inserts@listR
      \global\advance\insert@countR \@ne%
    \else%
      \xright@appenditem{\noexpand\vl@dlsnote{\expandonce\content}}%
                         \to\inserts@list
      \global\advance\insert@count \@ne%
    \fi
  \fi\ignorespaces\endgroup}

\newcommand*{\l@drsnote}[1]{%
  \begingroup%
  \newcommand{\content}{#1}%
  \ifnumberedpar@
    \ifledRcol%
      \xright@appenditem{\noexpand\vl@drsnote{\expandonce\content}}%
                         \to\inserts@listR
       \global\advance\insert@countR \@ne%
    \else%
      \xright@appenditem{\noexpand\vl@drsnote{\expandonce\content}}%
                         \to\inserts@list
      \global\advance\insert@count \@ne%
    \fi
  \fi\ignorespaces\endgroup}

\newcommand*{\l@dcsnote}[1]{%
  \begingroup%
  \newcommand{\content}{#1}%
  \ifnumberedpar@
    \ifledRcol%
      \xright@appenditem{\noexpand\vl@dcsnote{\expandonce\content}}%
                          \to\inserts@listR
      \global\advance\insert@countR \@ne%
    \else%
      \xright@appenditem{\noexpand\vl@dcsnote{\expandonce\content}}%
                          \to\inserts@list
      \global\advance\insert@count \@ne%
    \fi
  \fi\ignorespaces\endgroup}

```

\vl@dlsnote \vl@drsnote \vl@dcsnote

Put the left/right text into boxes, but just save the moveable text. `\l@dcsnotetext`, `\l@dcsnotetext@l` and `\l@dcsnotetext@r` are etoolbox's lists which will store the content of side notes. We store the content in lists, because we need to loop later on them, in case many sidenote co-exist for the same line. That is there some special test to do, in order to:

- Store the content of `\ledsidenote` to `\l@dcsnotetext` in any cases.
- Store the content of `\rightsidenote` to:
  - `\l@dcsnotetext` if `\ledsidenote` is to be put on right.
  - `\l@dcsnotetext@r` if `\ledsidenote` is to be put on left.
- Store the content of `\leftsidenote` to:
  - `\l@dcsnotetext` if `\ledsidenote` is to be put on left.
  - `\l@dcsnotetext@l` if `\ledsidenote` is to be put on right.

```
\newcommand*{\vl@dlsnote}[1]{%
  \ifledRcol@%
    \@l@dtempcntb=\sidenote@marginR%
    \ifnum\@l@dtempcntb>\@ne%
      \advance\@l@dtempcntb by\page@numR%
    \fi%
  \else%
    \@l@dtempcntb=\sidenote@margin%
    \ifnum\@l@dtempcntb>\@ne%
      \advance\@l@dtempcntb by\page@num%
    \fi%
  \fi%
  \ifodd\@l@dtempcntb%
    \listgadd{\l@dcsnotetext@l}{#1}%
  \else%
    \listgadd{\l@dcsnotetext}{#1}%
  \fi
}
\newcommand*{\vl@drsnote}[1]{%
  \ifledRcol@%
    \@l@dtempcntb=\sidenote@marginR%
    \ifnum\@l@dtempcntb>\@ne%
      \advance\@l@dtempcntb by\page@numR%
    \fi%
  \else%
    \@l@dtempcntb=\sidenote@margin%
    \ifnum\@l@dtempcntb>\@ne%
      \advance\@l@dtempcntb by\page@num%
    \fi%
  \fi%
  \ifodd\@l@dtempcntb%
    \listgadd{\l@dcsnotetext}{#1}%
  \else%
    \listgadd{\l@dcsnotetext@r}{#1}%
  \fi%
}
\newcommand*{\vl@dcsnote}[1]{\listgadd{\l@dcsnotetext}{#1}}

```

`\setl@dlp@rbox` `\setl@drpr@box` `\setl@dlprbox{`⟨*lednums*⟩`}{`⟨*tag*⟩`}{`⟨*text*⟩`}` puts ⟨*text*⟩ into the `\l@dlp@rbox` box. And similarly for the right side box. It is these boxes that finally get displayed in the margins.

```
\newcommand*{\setl@dlp@rbox}[1]{%
  {\parindent\z@\hsize=\ledlsnotewidth\ledlsnotefontsetup
   \global\setbox\l@dlp@rbox
   \ifleftnoteup
     =\vbox to\z@{\vss #1}%
   \else
     =\vbox to 0.70\baselineskip{\strut#1\vss}%
   \fi}}
\newcommand*{\setl@drp@rbox}[1]{%
  {\parindent\z@\hsize=\ledrsnotewidth\ledrsnotefontsetup
   \global\setbox\l@drp@rbox
   \ifrightnoteup
     =\vbox to\z@{\vss#1}%
   \else
     =\vbox to0.7\baselineskip{\strut#1\vss}%
   \fi}}
\newif\ifleftnoteup
  \leftnoteuptrue
```

`\sidenotesep` This macro is used to separate sidenotes of the same line.

```
\newcommand{\sidenotesep}{, }
```

`\affixside@note` This macro puts any moveable sidenote text into the left or right sidenote box, depending on which margin it is meant to go in. It's a very much stripped down version of `\affixlin@num`.

Before do it, we concatenate all moveable sidenotes of the line, using `\sidenotesep` as separator. It is the result that we put on the sidenote.

```
\newcommand*{\affixside@note}{%
    \def\sidenotecontent@{}%
    \numgdef{\itemcount@}{0}%
    \def\do##1{%
        \ifnumequal{\itemcount@}{0}%
            {%
        \appto\sidenotecontent@{##1}}% Not print not separator before the 1st note
            {\appto\sidenotecontent@{\sidenotesep ##1}%
            }%
            \numgdef{\itemcount@}{\itemcount@+1}%
    }%
    \dolistloop{\l@dcsnotetext}%
    \ifnumgreater{\itemcount@}{1}{\led@err@ManySidenotes}{}%
```

And we do the same for left and right notes (not movable).

```
  \gdef\@templ@d{}%
  \gdef\@templ@n{\l@dcsnotetext\l@dcsnotetext@l\l@dcsnotetext@r}%
  \ifx\@templ@d\@templ@n \else%
    \if@twocolumn%
      \if@firstcolumn%
        \setl@dlp@rbox{##1}{\sidenotecontent@}%
      \else%
        \setl@drp@rbox{\sidenotecontent@}%
      \fi%
    \else%
      \@l@dtempcntb=\sidenote@margin%
      \ifnum\@l@dtempcntb>\@ne%
        \advance\@l@dtempcntb by\page@num%
      \fi%
      \ifodd\@l@dtempcntb%
        \setl@drp@rbox{\sidenotecontent@}%
        \gdef\sidenotecontent@{}%
        \numgdef{\itemcount@}{0}%
        \dolistloop{\l@dcsnotetext@l}%
        \ifnumgreater{\itemcount@}{1}{\led@err@ManyLeftnotes}{}%
        \setl@dlp@rbox{\sidenotecontent@}%
      \else%
        \setl@dlp@rbox{\sidenotecontent@}%
        \gdef\sidenotecontent@{}%
        \numgdef{\itemcount@}{0}%
        \dolistloop{\l@dcsnotetext@r}%
        \ifnumgreater{\itemcount@}{1}{\led@err@ManyRightnotes}{}%
        \setl@drp@rbox{\sidenotecontent@}%
      \fi%
    \fi%
  \fi%
}
```

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` `\l@dfeetendmini` These will be the hooks in `\@iiiminpage` and `\endminipage`.

They can be extended to handle other things if necessary.

```
\ifnoledgroup@\else%
\newcommand*{\l@dfeetbeginmini}{\l@dedbeginmini\l@dfambeginmini}
\newcommand*{\l@dfeetendmini}{%
    \IfStrEq{critical-familiar}{\@mpfnpos}%
        {\l@dedendmini\l@dfamendmini}%
        {%
            \IfStrEq{familiar-critical}{\@mpfnpos}%
                {\l@dfamendmini\l@dedendmini}%
                {\l@dedendmini\l@dfamendmini}%
        }%
    }%
```

`\l@dedbeginmini` `\l@dedendmini` These handle the initiation and closure of critical footnotes in a minipage environment.

```
\newcommand*{\l@dedbeginmini}{%
  \unless\ifnocritical@%
    \def\do##1{\csletcs{v##1footnote}{mpv##1footnote}}%
    \dolistloop{\@series}%
  \fi%
  }
\newcommand*{\l@dedendmini}{%
  \unless\ifnocritical@%
    \ifl@dpairing%
      \ifledRcol%
        \flush@notesR%
      \else%
        \flush@notes%
      \fi%
    \fi
    \def\do##1{%
      \ifvoid\csuse{mp##1footins}\else%
        \ifl@dpairing\ifparledgroup%
          \ifledRcol%
        \dimgdef{\parledgroup@beforenotesR}{\parledgroup@beforenotesR+\skip\@nameuse{mp##1footins}}%
          \else%
         \dimgdef{\parledgroup@beforenotesL}{\parledgroup@beforenotesL+\skip\@nameuse{mp##1footins}}%
          \fi%
         \fi\fi%
         \csuse{mp##1footgroup}{##1}%
    \fi}%
    \dolistloop{\@series}%
  \fi%
}%

```

`\l@dfambeginmini` `\l@dfamendmini` These handle the initiation and closure of familiar footnotes in a minipage environment.

```
\newcommand*{\l@dfambeginmini}{%
 \unless\ifnofamiliar@%
   \def\do##1{\csletcs{vfootnote##1}{mpvfootnote##1}}%
     \dolistloop{\@series}%
 \fi%
}%

\newcommand*{\l@dfamendmini}{%
 \unless\ifnofamiliar@%
   \def\do##1{%
     \ifvoid\csuse{mpfootins##1}\else%
     \csuse{mpfootgroup##1}{##1}%
   \fi}%
   \dolistloop{\@series}%
 \fi%
}%
```

\@iiiminipage

This is our extended form of the kernel `\@iiiminipage` defined in `ltboxes.dtx`.

```
\patchcmd%
  {\@iiiminipage}%
  {\let\@footnotetext\@mpfootnotetext}%
  {\let\@footnotetext\@mpfootnotetext\l@dfeetbeginmini}%
  {}%
  {\led@error@fail@patch@@iiiminipage}%
```

\endminipage

This is our extended form of the kernel `\endminipage` defined in `ltboxes.dtx`.

```
\patchcmd%
  {\endminipage}%
  {\footnoterule}%
  {\footnoterule\l@advance@parledgroup@beforenormalnotes}%
  {}%
  {\led@error@fail@patch@endminipage}

\patchcmd%
  {\endminipage}%
  {\@minipagefalse}%
  {\l@dfeetendmini\@minipagefalse}%
  {}%
  {\led@error@fail@patch@endminipage}

```

\l@dunboxmpfoot
dvance@parledgroup@beforenormalnotes

`\@ldunboxmpfoot` insert normal footnotes for ledgroup.

```
\newcommand*{\l@dunboxmpfoot}{%
    \vskip\skip\@mpfootins
    \normalcolor
    \footnoterule
    \l@advance@parledgroup@beforenormalnotes
    \unvbox\@mpfootins%
}
```

When using parallel ledgroup, we need to store the vertical space added before footnote, in order to compensate them between left and right pages.

```
\newcommand{\l@advance@parledgroup@beforenormalnotes}{%
  \ifparledgroup
    \ifl@dpairing
      \ifledRcol
       \dimgdef{\parledgroup@beforenotesR}{\parledgroup@beforenotesR+\skip\@mpfootins}
      \else
       \dimgdef{\parledgroup@beforenotesL}{\parledgroup@beforenotesL+\skip\@mpfootins}
      \fi
    \fi
  \fi
}
```

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.

```
\newenvironment{ledgroup}{%
  \resetprevpage@num%
  \def\@mpfn{mpfootnote}\def\thempfn{\thempfootnote}\c@mpfootnote\z@%
  \let\@footnotetext\@mpfootnotetext
  \l@dfeetbeginmini%
}{%
  \par
  \unskip
  \ifvoid\@mpfootins\else
    \l@dunboxmpfoot
  \fi
  \l@dfeetendmini%
}

```

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.

```
\newenvironment{ledgroupsized}[2][l]{%
```

Set the various text measures.

```
  \hsize #2\relax
%%  \textwidth #2\relax
%%  \columnwidth #2\relax
```

Initialize fills for centering.

```
  \let\ledllfill\hfil
  \let\ledrlfill\hfil
  \def\@tempa{#1}\def\@tempb{l}%
```

Left adjusted numbered lines

```
      \ifx\@tempa\@tempb
    \let\ledllfill\relax
```

```
  \else
    \def\@tempb{r}%
    \ifx\@tempa\@tempb
```

Right adjusted numbered lines

```
      \let\ledrlfill\relax
    \fi
  \fi
```

Set up the footnoting.

```
  \def\@mpfn{mpfootnote}\def\thempfn{\thempfootnote}\c@mpfootnote\z@
  \let\@footnotetext\@mpfootnotetext
  \l@dfeetbeginmini%
}{%
  \par
  \unskip
  \ifvoid\@mpfootins\else
    \l@dunboxmpfoot
  \fi
  \l@dfeetendmini%
}

```

Close the `\ifnoledgroup@\else`.

```
\fi%
```

`\ifledgroupnotesL@` `\ifledgroupnotesR@` These boolean tests check if we are in the notes of a ledgroup. If we are, we do not number the lines. It could be useful for parallel ledgroup of reledpar.

```
\newif\ifledgroupnotesL@
\newif\ifledgroupnotesR@
```

XXVI Indexing

Here is some code for indexing using page and line numbers.

First, ensure that imakeidx or indextools is loaded *before* eledmac.

```
\AtBeginDocument{%
  \unless\ifl@imakeidx%
    \@ifpackageloaded{imakeidx}{\led@error@ImakeidxAfterEledmac}{}%
  \fi%
  \unless\ifl@indextools%
    \@ifpackageloaded{indextools}{\led@error@indextoolsAfterEledmac}{}%
  \fi%
}
```

`\pagelinesep` `\edindexlab` `\c@labidx` In order to get a correct line number we have to use the label/ref mechanism. These macros are for that.

```
\newcommand{\pagelinesep}{-}
\newcommand{\edindexlab}{$&}
```

```
\newcounter{labidx}
\setcounter{labidx}{0}

```

\doedindexlabel This macro sets an \edlabel.

```
\newcommand{\doedindexlabel}{%
  \stepcounter{labidx}%
  \edlabel{\edindexlab\thelabidx}%
}

```

\thepageline This macro makes up the page/line number combo from the label/ref.

```
\newcommand{\thepageline}{%
  \thepage%
  \pagelinesep%
  \xlineref{\edindexlab\thelabidx}%
}
```

\thestartpageline
\theendpageline These macros make up the page/line start/end number when the \edindex command is called in critical notes.

```
\newcommand{\thestartpageline}{%
  \l@dparsedstartpage%
  \pagelinesep%
  \l@dparsedstartline%
}
\newcommand{\theendpageline}{%
  \l@dparsedendpage%
  \pagelinesep%
  \l@dparsedendline%
}
```

\if@edindex@fornote@true This boolean test is switching at the beginning of each critical note, to allow index referring to this note.

```
\newif\if@edindex@fornote@
```

\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 |.

```
\newcommand{\prepare@edindex@fornote}[1]{%
    \l@dp@rsefootspec#1|%
    \@edindex@fornote@true%
}
```

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.

```
\newcommand{\get@edindex@ledinnote@command}{%
  \ifxindy@%
    \gdef\@ledinnote@command{%
      ledinnote\thelabidx%
    }%
    \ifxindyhyperref@%
      \immediate\write\eledmac@xindy@out{%
        (define-attributes ("ledinnote\thelabidx"))^^J
        \space\space(markup-locref^^J
          \eledmacmarkuplocrefdepth^^J
       :open "\string\ledinnote[\edindexlab\thelabidx]{\@index@command}{"^^J
          :close "}"^^J
          :attr "ledinnote\thelabidx"^^J
          )
        }%
    \else%
      \immediate\write\eledmac@xindy@out{%
        (define-attributes ("ledinnote\thelabidx"))^^J
        \space\space(markup-locref^^J
          \eledmacmarkuplocrefdepth^^J
          :open "\string\ledinnote{\@index@command}{"^^J
          :close "}"^^J
          :attr "ledinnote\thelabidx"^^J
          )
        }%
    \fi%
```

If we do not use xindy option, \@ledinnote@command will produce something like ledinnote{formatingcommand}.

```
  \else%
    \gdef\@ledinnote@command{%
       ledinnote[\edindexlab\thelabidx]{\@index@command}%
    }%
  \fi%
}
```

\get@index@command This macro is used to analyse if a text to be indexed has a command after a |.

```
\def\get@index@command#1|#2+{%
  \gdef\@index@txt{#1}%
  \gdef\@index@command{#2}%
  \xdef\@index@parenthesis{}%
  \IfBeginWith{\@index@command}{(}{%
    \StrGobbleLeft{\@index@command}{1}[\@index@command@]%
    \global\let\@index@command\@index@command@%
    \xdef\@index@parenthesis{(}%
    }{}%
  \IfBeginWith{\@index@command}{)}{%
    \StrGobbleLeft{\@index@command}{1}[\@index@command@]%
    \global\let\@index@command\@index@command@%
    \xdef\@index@parenthesis{)}%
    }{}%
}
```

\ledinnote \ledinnotehyperpage \ledinnotemark These macros are used to specify that an index reference points to a note. Arguments of \ledinnote are: #1 (optional): the label for the hyperlink, #2: command applied to the number, #3: the number itself.

```
\newcommandx{\ledinnote}[3][1,usedefault]{%
  \ifboolexpr{%
    test{\ifdefequal{\iftrue}{\ifHy@hyperindex}}%
    or%
    bool {xindyhyperref@}%
    }%
    {%
    \csuse{#2}{\hyperlink{#1}{\ledinnotemark{#3}}}%
    }%
    {%
    \csuse{#2}{\ledinnotemark{#3}}%
    }%
}%
\newcommand{\ledinnotehyperpage}[2]{\csuse{#1}{\ledinnotemark{\hyperpage{#2}}}}%
\newcommand{\ledinnotemark}[1]{#1\emph{n}}%
```

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 \@wredindex Write the index information to the idx file.

```
\newcommandx{\@wredindex}[2][1=\expandonce\jobname,usedefault]{%#1 = the index name, #2 = the text
  \global\let\old@Rlineflag\@Rlineflag%
  \gdef\@Rlineflag{}%
  \ifl@imakeidx%
    \if@edindex@fornote@%
      \IfSubStr[1]{#2}{|}{\get@index@command#2+}{\get@index@command#2|+}%
      \get@edindex@ledinnote@command%
    \expandafter\imki@wrindexentry{#1}{\@index@txt|(\@ledinnote@command}{\thestartpageline}%
    \expandafter\imki@wrindexentry{#1}{\@index@txt|)\@ledinnote@command}{\theendpageline}%
    \else%
      \get@edindex@hyperref{#2}%
     \imki@wrindexentry{#1}{\@index@txt\@edindex@hyperref}{\thepageline}%
    \fi%
  \else%
    \if@edindex@fornote@%
      \IfSubStr[1]{#2}{|}{\get@index@command#2+}{\get@index@command#2|+}%
      \get@edindex@ledinnote@command%
      \expandafter\protected@write\@indexfile{}%
{\string\indexentry{\@index@txt|(\@ledinnote@command}{\thestartpageline}
}%
      \expandafter\protected@write\@indexfile{}%
{\string\indexentry{\@index@txt|)\@ledinnote@command}{\theendpageline}
       }%
    \else%
      \protected@write\@indexfile{}%
{\string\indexentry{#2}{\thepageline}
}%
    \fi%
  \fi%
  \endgroup
  \global\let\@Rlineflag\old@Rlineflag%
  \@esphack%
}
```

Need to add the definition of `\edindex` to `\makeindex`, and initialise `\edindex` to do nothing.

```
\pretocmd{\makeindex}{%
  \def\edindex{\@bsphack
  \doedindexlabel
  \begingroup
  \@sanitize
  \@wredindex}}{}{}
\newcommand{\edindex}[1]{\@bsphack\@esphack}
```

`\hyperlinkformat`

`\hyperlinkformat` command is to be used to have both a internal hyperlink and a format, when indexing.

```
\newcommand{\hyperlinkformat}[3]{%
  \ifstrempty{#1}%
    {\hyperlink{#2}{#3}}%
    {\csuse{#1}{\hyperlink{#2}{#3}}%
  }}
```

`\hyperlinkR`

`\hyperlinkR` command is to be used to create a internal hyperlink and `\ledRflag`, when indexing.

```
\newcommand{\hyperlinkR}[2]{%
 \hyperlink{#1}{#2\@Rlineflag}%
}%

```

`\hyperlinkformatR`

`\hyperlinkformatR` command is to be used to create a internal hyperlink, a format and a `\@Rlineflag`, when indexing.

```
\newcommand{\hyperlinkformatR}[3]{%
  \hyperlinkformat{#1}{#2}{#3\@Rlineflag}%
}%

```

`\get@edindex@hyperref`
`\@edindex@hyperref`

`\get@edindex@hyperref` is to be used to define the `\@edindex@hyperref` macro, which, in index, links to the point where the index was called (with hyperref.

```
\newcommand{\get@edindex@hyperref}[1]{%
```

We have to disable temporary spaces to work through a xstring bug (or feature?)

```
\edef\temp@{%
\catcode`\ =9 %space need for catcode
#1%
\catcode`\ =10 % space need for catcode
}%
```

Now, we define `\@edindex@hyperref` if the `hyperindex` of hyperref is enabled.

```
\ifdefequal{\iftrue}{\ifHy@hyperindex}{%
  \IfSubStr{\temp@}{|}%
    {\get@index@command#1+%
    \ifledRcol%
      \gdef\@edindex@hyperref{|\@index@parenthesis %space kept
      hyperlinkformatR{\@index@command}%
      {\edindexlab\thelabidx}}%
    \else%
      \gdef\@edindex@hyperref{|\@index@parenthesis %space kept
      hyperlinkformat{\@index@command}%
      {\edindexlab\thelabidx}}%
    \fi%
    }%
    {\get@index@command#1|+%
     \ifledRcol%
      \gdef\@edindex@hyperref{|hyperlinkR{\edindexlab\thelabidx}}%
     \else%
      \gdef\@edindex@hyperref{|hyperlink{\edindexlab\thelabidx}}%
     \fi%
    }%
  }%
% If we use both xindy and hyperref, first get the \cs{index@command} command.
% Then define \cs{@edindex@hyperref} in the form \verb+eledmacXXX+
%    \begin{macrocode}
  {\ifxindyhyperref@%
    \IfSubStr{\temp@}{|}%
      {\get@index@command#1+}%
      {\get@index@command#1|+}%
   \gdef\@edindex@hyperref{|eledmac\thelabidx}%
```

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`.

```
    \IfStrEq{\@index@parenthesis}{(}%
      {%
      \csxdef{xindyparenthesis@\@index@txt}{\thelabidx}%
      \gdef\@edindex@hyperref{|(eledmac\thelabidx}%
      }%
      {}%
```

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`.

```
      \IfStrEq{\@index@parenthesis}{)}%
        {%
     \xdef\@edindex@hyperref{|)eledmac\csuse{xindyparenthesis@\@index@txt}}%
        \global\csundef{xindyparenthesis@\@index@txt}%
        }%
```

Write in the .xdy file the attributes of the location.

```
        {
        \immediate\write\eledmac@xindy@out{%
          (define-attributes ("eledmac\thelabidx"))^^J
          \space\space(markup-locref^^J
            \eledmacmarkuplocrefdepth^^J
            :open "\string\hyperlink%
                   \ifledRcol R\fi%
                   {\edindexlab\thelabidx}%
                   {\ifdefempty{\@index@command}%
                     {}%
                     {\@backslashchar\@index@command}%
                   {"^^J
            :close "}}"^^J
            :attr "eledmac\thelabidx"^^J
            )
          }%
        }%
```

And now, in any other case.

```
    \else%
      \gdef\@index@txt{#1}%
      \gdef\@edindex@hyperref{}%
    \fi%
    }%
}
```

XXVII Verse

The original code is principally Wayne Sullivan's code from edstanza. However, the code has been many time modified by Maïeul Rouquette in order to obtain new features and improved compatibility with reledpar.

XXVII.1 Hanging symbol management

`\@hangingsymbol`
`\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.

```
\def\@hangingsymbol{}
\newcommand*{\sethangingsymbol}[1]{%
  \gdef\@hanginsymbol{#1}%
}%
\newif\ifinstanza
```

`\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.

```
\newif\ifinserthangingsymbol
\newcommand{\inserthangingsymbol}{%
\ifinserthangingsymbol%
  \ifinstanza%
    \@hangingsymbol%
  \fi%
\fi%
}
```

XXVII.2 Using & character

`\ampersand`

Within a stanza the `\&` macro is going to be usurped. We need an alias in case an & needs to be typeset in a stanza. Define it rather than letting it in case some other package has already defined it.

```
\newcommand*{\ampersand}{\char`\&}

```

XXVII.3 Code category setting

`\stanza@count` `\stanzaindentbase`

Before we can define the main macros we need to save and reset some category codes. To save the current values we use `\next` and `\body` from the `\loop` macro.

```
  \chardef\body=\catcode`\@
  \catcode`\@=11
  \chardef\next=\catcode`\&
  \catcode`\&=\active

```

XXVII.4 Stanza count and indent

A count register is allocated for counting lines in a stanza; also allocated is a dimension register which is used to specify the base value for line indentation; all stanza indentations are multiples of this value. The default value of `\stanzaindentbase` is 20pt.

```
  \newcount\stanza@count
  \newlength{\stanzaindentbase}
  \setlength{\stanzaindentbase}{20pt}

```

\strip@szacnt
\setstanzavalues

The indentations of stanza lines are non-negative integer multiples of the unit called `\stanzaindentbase`. To make it easier for the user to specify these numbers, some list macros are defined. These take numerical values in a list separated by commas and assign the values to special control sequences using `\mathchardef`. Though this does limit the range from 0 to 32767, it should suffice for most applications, including *penalties*, which will be discussed below.

```
\def\strip@szacnt#1,#2|{\def\@tempb{#1}\def\@tempa{#2|}}
\newcommand*{\setstanzavalues}[2]{\def\@tempa{#2,,|}%
        \stanza@count\z@
     \def\next{\expandafter\strip@szacnt\@tempa
            \ifx\@tempb\empty\let\next\relax\else
            \expandafter\mathchardef\csname #1@\number\stanza@count
            @\endcsname\@tempb\relax
            \advance\stanza@count\@ne\fi\next}%
      \next}

```

\setstanzaindents
\setstanzapenalties

In the original edmac, `\setstanzavalues{sza}{⟨...⟩}` had to be called to set the indents, 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).

```
\newcommand*{\setstanzaindents}[1]{\setstanzavalues{sza}{#1}}
\newcommand*{\setstanzapenalties}[1]{\setstanzavalues{szp}{#1}}
%
```

\managestanza@modulo

Since version 0.13, the `stanzaindentsrepetition` counter can be used when the indentation is repeated every n verses. The `\managestanza@modulo` is a command which modifies the counter stanza@modulo. The command adds 1 to stanza@modulo, but if stanza@modulo is equal to the stanzaindentsrepetition counter, the command restarts it.

```
\newcounter{stanzaindentsrepetition}
\newcount\stanza@modulo

\newcommand*{\managestanza@modulo}[0]{
    \advance\stanza@modulo\@ne
    \ifnum\stanza@modulo>\value{stanzaindentsrepetition}
     \stanza@modulo\@ne
    \fi
}
```

\stanzaindent
\stanzaindent*

The macro `\stanzaindent`, when called at the beginning of a verse, changes the indentation normally defined for this verse by `\setstanzaindent`. The starred version skips the current verse for the repetition of stanza indent.

```
\newcommand{\stanzaindent}[1]{%
  \hspace{\dimexpr#1\stanzaindentbase-\parindent\relax}%
  \ignorespaces%
}%
\WithSuffix\newcommand\stanzaindent*[1]{%
  \stanzaindent{#1}%
  \global\advance\stanza@modulo-\@ne%
  \ifnum\stanza@modulo=0%
    \global\stanza@modulo=\value{stanzaindentsrepetition}%
  \fi%
  \ignorespaces%
}%
```

XXVII.5 Main work

\stanza@line
\stanza@hang
\sza@penalty

Now we arrive at the main works. `\stanza@line` sets the indentation for the line and starts a numbered paragraph—each line is treated as a paragraph. `\stanza@hang` sets the hanging indentation to be used if the stanza line requires more than one print line.

If it is known that each stanza line will fit on one print line, it is advisable to set the hanging indentation to zero. `\sza@penalty` places the specified penalty following each stanza line. By default, this facility is turned off so that no penalty is included. However, the user may initiate these penalties to indicate good and bad places in the stanza for page breaking.

```
\newcommandx{\stanza@line}[1][1]{
    \ifnum\value{stanzaindentsrepetition}=0
        \parindent=\csname sza@\number\stanza@count
                 @\endcsname\stanzaindentbase
    \else
        \parindent=\csname sza@\number\stanza@modulo
                 @\endcsname\stanzaindentbase
        \managestanza@modulo
    \fi
    \pstart[#1]\stanza@hang\ignorespaces}
\xdef\stanza@hang{\noexpand\leavevmode\noexpand\startlock
            \hangindent\expandafter
            \noexpand\csname sza@0@\endcsname\stanzaindentbase
            \hangafter\@ne}
\def\sza@penalty{\count@\csname szp@\number\stanza@count @\endcsname
          \ifnum\count@>\@M\advance\count@-\@M\penalty-\else
          \penalty\fi\count@}
```

\@startstanza
\stanza
\@stopstanza
\newverse

Now we have the components of the `\stanza` macro, which appears at the start of a group of lines. This macro initializes the count and checks to see if hanging indentation and penalties are to be included. Hanging indentation suspends the line count, so that the enumeration is by verse line rather than by print line. If the print line count is desired, invoke `\let\startlock\relax` and do the same for `\endlock`. Here and above we have used `\xdef` to make the stored macros take up a bit less space, but it also makes them more obscure to the reader. Lines of the stanza are delimited by ampersands &. The last line of the stanza must end with `\&`.

```
\xdef\@startstanza[#1]{%
  \noexpand\instanzatrue\expandafter
  \begingroup%
  \catcode`\noexpand\&\active%
  \global\stanza@count\@ne\stanza@modulo\@ne
  \noexpand\ifnum\expandafter\noexpand
  \csname sza@0@\endcsname=\z@\let\noexpand\stanza@hang\relax
  \let\noexpand\endlock\relax\noexpand\else\interlinepenalty
  \@M\rightskip\z@ plus 1fil\relax\noexpand\fi\noexpand\ifnum
  \expandafter\noexpand\csname szp@0@\endcsname=\z@
  \let\noexpand\sza@penalty\relax\noexpand\fi%
  \def\noexpand&{%
        \noexpand\newverse[][]}%
  \def\noexpand\&{\noexpand\@stopstanza}%
  \noexpand\stanza@line[#1]}

\newcommandx{\stanza}[1][1,usedefault]{\@startstanza[#1]}

\newcommandx{\@stopstanza}[1][1,usedefault]{%
  \unskip%
  \endlock%
  \pend[#1]%
  \endgroup%
  \instanzafalse%
}

\newcommandx*{\newverse}[2][1,2,usedefault]{%
  \unskip%
  \endlock\pend[#1]\sza@penalty\global%
  \advance\stanza@count\@ne\stanza@line[#2]%
  }

```

`\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`.

```
\newcommand*{\flagstanza}[2][\stanzaindentbase]{%
  \hskip -#1\llap{#2}\hskip #1\ignorespaces}

```

XXVII.6 Restore catcode and penalties

The ampersand & is used to mark the end of each stanza line, except the last, which is marked with `\&`. This means that `\halign` may not be used directly within a stanza line. This does not affect macros involving alignments defined outside `\stanza`   `\&`. Since these macros usurp the control sequence `\&`, the replacement `\ampersand` is defined to be used if this symbol is needed in a stanza. Also we reset the modified category codes and initialize the penalty default.

```
  \catcode`\&=\next
  \catcode`\@=\body
  \setstanzavalues{szp}{0}

```

XXVIII Arrays and tables

XXVIII.1 Preamble: macro as environment

The following is borrowed, and renamed, from the amsmath package. See also the CTT thread 'eeq and amstex', 1995/08/31, started by Keith Reckdahl and ended definitively by David M. Jones.

Several of the [math] macros scan their body twice. This means we must collect all text in the body of an environment form before calling the macro.

`\@emptytoks` This is actually defined in the amsgen package.

```
\newtoks\@emptytoks

```

The rest is from amsmath.

`\l@denvbody` A token register to contain the body.

```
\newtoks\l@denvbody

```

`\addtol@denvbody` `\addtol@denvdody{arg}` adds arg to the token register `\l@denvbody`.

```
\newcommand{\addtol@denvbody}[1]{%
  \global\l@denvbody\expandafter{\the\l@denvbody#1}}

```

`\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`.

```
\newcommand{\l@dcollect@body}[1]{%
  \l@denvbody{\expandafter#1\expandafter{\the\l@denvbody}}%
  \edef\processl@denvbody{\the\l@denvbody\noexpand\end{\@currenvir}}%
  \l@denvbody\@emptytoks \def\l@dbegin@stack{b}%
  \begingroup
    \expandafter\let\csname\@currenvir\endcsname\l@dcollect@@body
   \edef\processl@denvbody{\expandafter\noexpand\csname\@currenvir\endcsname}%
    \processl@denvbody%
    }%

```

`\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.

```
\def\l@dpush@begins#1\begin#2{%
  \ifx\end#2\else b\expandafter\l@dpush@begins\fi}

```

`\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.

```
\def\l@dcollect@@body#1\end#2{%
  \edef\l@dbegin@stack{\l@dpush@begins#1\begin\end
                        \expandafter\@gobble\l@dbegin@stack}%
  \ifx\@empty\l@dbegin@stack
    \endgroup
    \@checkend{#2}%
    \addtol@denvbody{#1}%
  \else
    \addtol@denvbody{#1\end{#2}}%
  \fi
  \processl@denvbody % A little tricky! Note the grouping
}

```

There was a question on CTT about how to use `\collect@body` for a macro taking an argument. The following is part of that thread.

```
From: Heiko Oberdiek <oberdiek@uni-freiburg.de>
Newsgroups: comp.text.tex
Subject: Re: Using \collect@body with commands that take >1 argument
Date: Fri, 08 Aug 2003 09:03:20 +0200

eed132@psu.edu (Evan) wrote:
> I'm trying to make a new Latex environment that acts like the>
 \colorbox command that is part of the color package. I looked through
> the FAQ and ran across this bit about using the \collect@body command
> that is part of AMSLaTeX:
> http://www.tex.ac.uk/cgi-bin/texfaq2html?label=cmdasenv
>
> It almost works. If I do something like the following:
>   \newcommand{\redbox}[1]{\colorbox{red}{#1}}
>
>   \makeatletter
>   \newenvironment{redbox}{\collect@body \redbox}{}

You will get an error message: Command \redbox already defined.
Thus you must rename either the command \redbox or the environment
name.

>   \begin{coloredbox}{blue}
>     Yadda yadda yadda... this is on a blue background...
>   \end{coloredbox}
> and can't figure out how to make the \collect@body take this.
>   \collect@body \colorbox{red}
>   \collect@body {\colorbox{red}}

The argument of \collect@body has to be one token exactly.

\documentclass{article}
\usepackage{color}
\usepackage{amsmath}

\newcommand{\redbox}[1]{\colorbox{red}{#1}}
\makeatletter
\newenvironment{coloredbox}[1]{%
  \def\next@{\colorbox{#1}}%
  \collect@body\next@
}{}

% ignore spaces at begin and end of environment
\newenvironment{coloredboxII}[1]{%
  \def\next@{\mycoloredbox{#1}}%
  \collect@body\next@
}{}
\newcommand{\mycoloredbox}[2]{%
  \colorbox{#1}{\ignorespaces#2\unskip}%
}

% support of optional color model argument
\newcommand\coloredboxIII\endcsname{}
\def\coloredboxIII#1#{%
  \@coloredboxIII{#1}%
}
\def\@coloredboxIII#1#2{%
  \def\next@{\mycoloredboxIII{#1}{#2}}%
  \collect@body\next@
}
\newcommand{\mycoloredboxIII}[3]{%
  \colorbox#1{#2}{\ignorespaces#3\unskip}%
}

\makeatother

\begin{document}
  Black text before
  \begin{coloredbox}{blue}
   Hello World
  \end{coloredbox}
  Black text after

  Black text before
  \begin{coloredboxII}{blue}
   Hello World
  \end{coloredboxII}
  Black text after

  Black text before
  \begin{coloredboxIII}[rgb]{0,0,1}
   Hello World
  \end{coloredboxIII}
  Black text after

\end{document}

Yours sincerely
  Heiko <oberdiek@uni-freiburg.de>
```

XXVIII.2 Tabular environments

This is based on the work by Herbert Breger in developing `tabmac.tex`.

The original `tabmac.tex` file was void of comments or any explanatory text other than the above notice. The algorithm is Breger's. Peter Wilson have made some cosmetic changes to the original code and reimplemented some things so they are more LaTeX-like. All the commentary are from Peter Wilson, as are any mistake or errors.

However, Maïeul Rouquette has modified code in order to add new features of eledmac and reledmac.

XXVIII.2.1 Disabling and restoring commands

`\l@dtabnoexpands` More no expansion for critical and familiar footnotes in tabular environment.

```
\newcommand*{\l@dtabnoexpands}{%
  \let\rtab=0%
  \let\ctab=0%
  \let\ltab=0%
  \let\rtabtext=0%
  \let\ltabtext=0%
  \let\ctabtext=0%
  \let\edbeforetab=0%
  \let\edaftertab=0%
  \let\edatleft=0%
  \let\edatright=0%
  \let\edvertline=0%
  \let\edvertdots=0%
  \let\edrowfill=0%
}

```

`\disable@familiarnotes` `\restore@familiarnotes` Macros to disable and restore familiar notes, to prevent them from printing multiple times in edtabularx and edarrayx environments.

```
\newcommand{\disable@familiarnotes}{%
```

```
  \unless\ifnofamiliar@%
    \def\do##1{%
        \csletcs{footnote@@##1}{footnote##1}%
        \expandafter\renewcommand \csname footnote##1\endcsname[1]{%
          \protected@csxdef{@thefnmark##1}{\csuse{thefootnote##1}}%
          \csuse{@footnotemark##1}%
        }%
    }%
    \dolistloop{\@series}%
  \fi%
}%
\newcommand{\restore@familiarnotes}{%
  \unless\ifnofamiliar@%
    \def\do##1{%
        \csletcs{footnote##1}{footnote@@##1}%
    }%
    \dolistloop{\@series}%
  \fi%
}%

```

`\disable@sidenotes` `\restore@sidenotes` The sames, for side notes.

```
\newcommand{\disable@sidenotes}{%
  \let\@@ledrightnote\ledrightnote%
  \let\@@ledleftnote\ledleftnote%
  \let\@@ledsidenote\ledsidenote%
  \let\ledrightnote\@gobble%
  \let\ledleftnote\@gobble%
  \let\ledsidenote\@gobble%
}%
\newcommand{\restore@sidenotes}{%
  \let\ledrightnote\@@ledrightnote%
  \let\ledleftnote\@@ledleftnote%
  \let\ledsidenote\@@ledsidenote%
}%
```

`\disable@notes` `\restore@notes` Disable/restore side and familiar notes.

```
\newcommand{\disable@notes}{%
  \disable@sidenotes%
  \disable@familiarnotes%
}%
\newcommand{\restore@notes}{%
  \restore@sidenotes%
  \restore@familiarnotes%
}%
```

`\EDTEXT` `\xedtext` We need to be able to modify the `\edtext` macros and also restore their original definitions.

```
\let\EDTEXT=\edtext
\newcommand{\xedtext}[2]{\EDTEXT{#1}{#2}}
```

\EDLABEL
\xedlabel

We need to be able to modify and restore the `\edlabel` macro.

```
\let\EDLABEL=\edlabel
\newcommand*{\xedlabel}[1]{\EDLABEL{#1}}
```

\EDINDEX
\xedindex
\nulledindex

Macros supporting modification and restoration of `\edindex`.

```
\let\EDINDEX=\edindex
\newcommand{\xedindex}{\@bsphack%
  \@ifnextchar [{\l@d@index}{\l@d@index[\jobname]}}
\newcommand{\nulledindex}[2][\jobname]{\@bsphack\@esphack}

```

\@line@@num

Macro supporting restoration of `\linenum`.

```
\let\@line@@num=\linenum
```

\l@dgobblearg

`\l@dgobbleoptarg[`⟨*arg*⟩`]{`⟨*arg*⟩`}` replaces these two arguments (first is optional) by `\relax`.

```
\newcommand*{\l@dgobbleoptarg}[2][]{\relax}%

```

\Relax
\NEXT

```
\let\Relax=\relax
\let\NEXT=\next

```

\l@dmodforedtext
\l@drestoreforedtext

Modify and restore various macros for when `\edtext` is used.

```
\newcommand{\l@dmodforedtext}{%
  \let\edtext\relax
  \def\do##1{\global\csletcs{##1footnote}{l@dgobbleoptarg}}%
  \dolistloop{\@series}%
  \let\edindex\nulledindex
  \let\linenum\@gobble}
\newcommand{\l@drestoreforedtext}{%
  \def\do##1{\global\csletcs{##1footnote}{##1@@footnote}}
  \dolistloop{\@series}%
  \let\edindex\xedindex}
```

\l@dnullfills
\l@drestorefills

Nullify and restore some column fillers, etc.

```
\newcommand{\l@dnullfills}{%
  \def\edlabel##1{}%
  \def\edrowfill##1##2##3{}%
}
\newcommand{\l@drestorefills}{%
  \def\edrowfill##1##2##3{\@EDROWFILL@{##1}{##2}{##3}}%
}

```

\letsforverteilen

Gathers some lets and other code that is common to the `*verteilen*` macros.

```
\newcommand{\letsforverteilen}{%
```

```
  \let\edtext\xedtext
  \let\edindex\xedindex
  \def\do##1{\global\csletcs{##1footnote}{##1@@footnote}}
  \dolistloop{\@series}%
  \let\linenum\@line@@num
  \hilfsskip=\l@dcolwidth%
  \advance\hilfsskip by -\wd\hilfsbox
  \def\edlabel##1{\xedlabel{##1}}}

```

`\disablel@dtabfeet` `\enablel@dtabfeet`

Declarations for using or using `\edtext` inside tabulars. The default at this point is for `\edtext`.

```
\newcommand\disablel@dtabfeet{\l@dmodforedtext}%
\newcommand\enablel@dtabfeet{\l@drestoreforedtext}%
```

XXVIII.2.2 Counters, boxes and lengths

`\l@dampcount` `\l@dcolcount`

`\l@dampcount` is a counter for the & column dividers and `\l@dcolcount` is a counter for the columns.

```
\newcount\l@dampcount
  \l@dampcount=1\relax
\newcount\l@dcolcount
  \l@dcolcount=0\relax

```

`\hilfsbox` `\hilfsskip` `\Hilfsbox` `\hilfscount`

Some (temporary) helper items.

```
\newbox\hilfsbox
\newskip\hilfsskip
\newbox\Hilfsbox
\newcount\hilfscount

```

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).

```
\newdimen\dcoli
\newdimen\dcolii
\newdimen\dcoliii
\newdimen\dcoliv
\newdimen\dcolv
\newdimen\dcolvi
\newdimen\dcolvii
\newdimen\dcolviii
\newdimen\dcolix
\newdimen\dcolx
\newdimen\dcolxi
\newdimen\dcolxii
\newdimen\dcolxiii
\newdimen\dcolxiv
\newdimen\dcolxv
```

```
\newdimen\dcolxvi
\newdimen\dcolxvii
\newdimen\dcolxviii
\newdimen\dcolxix
\newdimen\dcolxx
\newdimen\dcolxxi
\newdimen\dcolxxii
\newdimen\dcolxxiii
\newdimen\dcolxxiv
\newdimen\dcolxxv
\newdimen\dcolxxvi
\newdimen\dcolxxvii
\newdimen\dcolxxviii
\newdimen\dcolxxix
\newdimen\dcolxxx
\newdimen\dcolerr   % added for error handling

```

`\l@dcolwidth` This is a cunning way of storing the columnwidths indexed by the column number `\l@dcolcount`, like an array. (was `\Dimenzuordnung`)

```
\newcommand{\l@dcolwidth}{\ifcase \the\l@dcolcount \dcoli %???
  \or \dcoli \or \dcolii \or \dcoliii
  \or \dcoliv \or \dcolv \or \dcolvi
  \or \dcolvii \or \dcolviii \or \dcolix \or \dcolx
  \or \dcolxi \or \dcolxii \or \dcolxiii
  \or \dcolxiv \or \dcolxv \or \dcolxvi
  \or \dcolxvii \or \dcolxviii \or \dcolxix \or \dcolxx
  \or \dcolxxi \or \dcolxxii \or \dcolxxiii
  \or \dcolxxiv \or \dcolxxv \or \dcolxxvi
  \or \dcolxxvii \or \dcolxxviii \or \dcolxxix \or \dcolxxx
  \else \dcolerr \fi}

```

`\stepl@dcolcount` This increments the column counter, and issues an error message if it is too large.

```
\newcommand*{\stepl@dcolcount}{\advance\l@dcolcount\@ne
  \ifnum\l@dcolcount>30\relax
    \led@err@TooManyColumns
  \fi}

```

`\l@dsetmaxcolwidth` Sets the column width to the maximum value seen so far.

```
\newcommand{\l@dsetmaxcolwidth}{%
  \ifdim\l@dcolwidth < \wd\hilfsbox
    \l@dcolwidth = \wd\hilfsbox
  \else \relax \fi}

```

`\measuremcell` Measure (recursively) the width required for a math cell.

```
\def\measuremcell #1&{%
```

```
    \ifx #1\\ \ifnum\l@dcolcount=0\let\NEXT\relax%
              \else\l@dcheckcols%
                    \l@dcolcount=0%
                    \let\NEXT\measuremcell%
              \fi%
    \else\setbox\hilfsbox=\hbox{$\displaystyle{#1}$}%
      \stepl@dcolcount%
      \l@dsetmaxcolwidth%
      \let\NEXT\measuremcell%
    \fi\NEXT}

```

`\measuretcell` Measure (recursively) the width required for a text cell.

```
\def\measuretcell #1&{%
    \ifx #1\\ \ifnum\l@dcolcount=0\let\NEXT\relax%
              \else\l@dcheckcols%
                    \l@dcolcount=0%
                    \let\NEXT\measuretcell%
              \fi%
    \else\setbox\hilfsbox=\hbox{#1}%
      \stepl@dcolcount%
      \l@dsetmaxcolwidth%
      \let\NEXT\measuretcell%
    \fi\NEXT}

```

`\measuremrow` Measure (recursively) the width required for a math row.

```
\def\measuremrow #1\\{%
   \ifx #1&\let\NEXT\relax%
   \else\measuremcell #1&\\&\\&%
      \let\NEXT\measuremrow%
   \fi\NEXT}
```

`\measuretrow` Measure (recursively) the width required for a text row.

```
\def\measuretrow #1\\{%
   \ifx #1&\let\NEXT\relax%
   \else\measuretcell #1&\\&\\&%
     \let\NEXT\measuretrow%
   \fi\NEXT}

```

`\edtabcolsep` The length `\edtabcolsep` controls the distance between columns.

```
\newskip\edtabcolsep
\global\edtabcolsep=10pt

```

`\variab`

```
\newcommand{\variab}{\relax}

```

\l@dcheckcols Check that the number of columns is consistent.

```
\newcommand*{\l@dcheckcols}{%
  \ifnum\l@dcolcount=1\relax
  \else
    \ifnum\l@dampcount=1\relax
    \else
      \ifnum\l@dcolcount=\l@dampcount\relax
      \else
        \l@d@err@UnequalColumns
      \fi
    \fi
    \l@dampcount=\l@dcolcount
  \fi}

```

\edfilldimen A length.

```
\newdimen\edfilldimen
\edfilldimen=0pt

```

\c@addcolcount \theaddcolcount A counter to hold the number of a column. We use a roman number so that we can grab the column dimension from `\dcol`.

```
\newcounter{addcolcount}
  \renewcommand{\theaddcolcount}{\roman{addcolcount}}
```

XXVIII.2.3 Tabular typesetting

\setmcellright Typeset (recursively) cells of display math right justified.

```
\def\setmcellright #1&{\def\edlabel##1{}%
        \let\edindex\nulledindex
        \ifx #1\\ \ifnum\l@dcolcount=0%\removelastskip
                       \let\Next\relax%
                  \else\l@dcolcount=0%
                        \let\Next=\setmcellright%
                  \fi%
        \else%
            \disablel@dtabfeet%
            \stepl@dcolcount%
            \disable@notes%
            \setbox\hilfsbox=\hbox{$\displaystyle{#1}$}%
            \restore@notes%
            \letsforverteilen%
            \hskip\hilfsskip$\displaystyle{#1}$%
            \hskip\edtabcolsep%
            \let\Next=\setmcellright%
        \fi\Next}

```

\settcellright Typeset (recursively) cells of text right justified.

```
\def\settcellright #1&{\def\edlabel##1{}%
       \let\edindex\nulledindex
       \ifx #1\\ \ifnum\l@dcolcount=0%\removelastskip
                    \let\Next\relax%
                \else\l@dcolcount=0%
                      \let\Next=\settcellright%
                \fi%
       \else%
           \disablel@dtabfeet%
           \stepl@dcolcount%
           \disable@notes%
           \setbox\hilfsbox=\hbox{#1}%
           \restore@notes%
           \letsforverteilen%
           \hskip\hilfsskip#1%
           \hskip\edtabcolsep%
           \let\Next=\settcellright%
       \fi\Next}
```

`\setmcellleft` Typeset (recursively) cells of display math left justified.

```
\def\setmcellleft #1&{\def\edlabel##1{}%
    \let\edindex\nulledindex
    \ifx #1\\ \ifnum\l@dcolcount=0 \let\Next\relax%
              \else\l@dcolcount=0%
                    \let\Next=\setmcellleft%
              \fi%
     \else   \disablel@dtabfeet%
              \stepl@dcolcount%
              \disable@notes%
              \setbox\hilfsbox=\hbox{$\displaystyle{#1}$}%
              \restore@notes%
              \letsforverteilen%
              $\displaystyle{#1}$\hskip\hilfsskip\hskip\edtabcolsep%
              \let\Next=\setmcellleft%
     \fi\Next}

```

`\settcellleft` Typeset (recursively) cells of text left justified.

```
\def\settcellleft #1&{\def\edlabel##1{}%
    \let\edindex\nulledindex
    \ifx #1\\ \ifnum\l@dcolcount=0 \let\Next\relax%
              \else\l@dcolcount=0%
                    \let\Next=\settcellleft%
              \fi%
     \else   \disablel@dtabfeet%
              \stepl@dcolcount%
              \disable@notes%
              \setbox\hilfsbox=\hbox{#1}%
              \restore@notes%
              \letsforverteilen%
```

```
                #1\hskip\hilfsskip\hskip\edtabcolsep%
                \let\Next=\settcellleft%
       \fi\Next}
```

`\setmcellcenter` Typeset (recursively) cells of display math centered.

```
\def\setmcellcenter #1&{\def\edlabel##1{}%
      \let\edindex\nulledindex
      \ifx #1\\ \ifnum\l@dcolcount=0\let\Next\relax%
               \else\l@dcolcount=0%
                     \let\Next=\setmcellcenter%
               \fi%
      \else    \disablel@dtabfeet%
               \stepl@dcolcount%
               \disable@notes%
               \setbox\hilfsbox=\hbox{$\displaystyle{#1}$}%
               \restore@notes%
               \letsforverteilen%
               \hskip 0.5\hilfsskip$\displaystyle{#1}$\hskip0.5\hilfsskip%
               \hskip\edtabcolsep%
               \let\Next=\setmcellcenter%
       \fi\Next}

```

`\settcellcenter` Typeset (recursively) cells of text centered.

```
\def\settcellcenter #1&{\def\edlabel##1{}%
      \let\edindex\nulledindex
      \ifx #1\\ \ifnum\l@dcolcount=0 \let\Next\relax%
                \else\l@dcolcount=0%
                      \let\Next=\settcellcenter%
                \fi%
       \else   \disablel@dtabfeet%
                \stepl@dcolcount%
                \disable@notes%
                \setbox\hilfsbox=\hbox{#1}%
                \restore@notes%
                \letsforverteilen%
                \hskip 0.5\hilfsskip #1\hskip 0.5\hilfsskip%
                \hskip\edtabcolsep%
                \let\Next=\settcellcenter%
       \fi\Next}

```

`\NEXT`

```
\let\NEXT=\relax

```

`\setmrowright` Typeset (recursively) rows of right justified math.

```
\def\setmrowright #1\\{%
      \ifx #1& \let\NEXT\relax
```

```
    \else \centerline{\setmcellright #1&\\&\\&}
          \let\NEXT=\setmrowright
    \fi\NEXT}
```

\settrowright Typeset (recursively) rows of right justified text.

```
\def\settrowright #1\\{%
    \ifx #1& \let\NEXT\relax
    \else \centerline{\settcellright #1&\\&\\&}
          \let\NEXT=\settrowright
    \fi\NEXT}

```

\setmrowleft Typeset (recursively) rows of left justified math.

```
\def\setmrowleft #1\\{%
    \ifx #1&\let\NEXT\relax
    \else \centerline{\setmcellleft #1&\\&\\&}
          \let\NEXT=\setmrowleft
    \fi\NEXT}
```

\settrowleft Typeset (recursively) rows of left justified text.

```
\def\settrowleft #1\\{%
    \ifx #1& \let\NEXT\relax
    \else \centerline{\settcellleft #1&\\&\\&}
          \let\NEXT=\settrowleft
    \fi\NEXT}

```

\setmrowcenter Typeset (recursively) rows of centered math.

```
\def\setmrowcenter #1\\{%
    \ifx #1& \let\NEXT\relax%
    \else \centerline{\setmcellcenter #1&\\&\\&}
          \let\NEXT=\setmrowcenter
     \fi\NEXT}
```

\settrowcenter Typeset (recursively) rows of centered text.

```
\def\settrowcenter #1\\{%
    \ifx #1& \let\NEXT\relax
    \else \centerline{\settcellcenter #1&\\&\\&}
          \let\NEXT=\settrowcenter
    \fi\NEXT}

```

\nullsetzen

```
\newcommand{\nullsetzen}{%
    \stepl@dcolcount%
    \l@dcolwidth=0pt%
    \ifnum\l@dcolcount=30\let\NEXT\relax%
          \l@dcolcount=0\relax
    \else\let\NEXT\nullsetzen%
    \fi\NEXT}

```

\edatleft `\edatleft[`⟨*math*⟩`]{`⟨*symbol*⟩`}{`⟨*len*⟩`}`. Left ⟨*symbol*⟩, 2⟨*len*⟩ high with prepended ⟨*math*⟩ vertically centered.

```
\newcommand{\edatleft}[3][\@empty]{%
  \ifx#1\@empty
    \vbox to 10pt{\vss\hbox{$\left#2\vrule width0pt height #3
                          depth 0pt \right. $\hss}\vfil}
  \else
    \vbox to 4pt{\vss\hbox{$#1\left#2\vrule width0pt height #3
                depth 0pt \right. $}\vfil}
  \fi}
```

\edatright `\edatright[`⟨*math*⟩`]{`⟨*symbol*⟩`}{`⟨*len*⟩`}`. Right ⟨*symbol*⟩, 2⟨*len*⟩ high with appended ⟨*math*⟩ vertically centered.

```
\newcommand{\edatright}[3][\@empty]{%
  \ifx#1\@empty
    \vbox to 10pt{\vss\hbox{$\left.\vrule width0pt height #3
                depth 0pt \right#2 $\hss}\vfil}
  \else
    \vbox to 4pt{\vss\hbox{$\left.\vrule width0pt height #3
                depth 0pt \right#2 #1 $}\vfil}
  \fi}

```

\edvertline `\edvertline{`⟨*len*⟩`}` vertical line ⟨*len*⟩ high.

```
\newcommand{\edvertline}[1]{\vbox to 8pt{\vss\hbox{\vrule height #1}\vfil}}

```

\edvertdots `\edvertdots{`⟨*len*⟩`}` vertical dotted line ⟨*len*⟩ high.

```
\newcommand{\edvertdots}[1]{\vbox to 1pt{\vss\vbox to #1%
        {\cleaders\hbox{$\m@th\hbox{.}\vbox to 0.5em{ }$}\vfil}}}

```

\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@`.

```
\newcommand{\l@dtabaddcols}[2]{%
  \l@dcheckstartend{#1}{#2}%
  \ifl@dstartendok
  \setcounter{addcolcount}{#1}%
  \@whilenum \value{addcolcount}<#2\relax \do
  {\advance\edfilldimen by \the \csname dcol\theaddcolcount\endcsname
   \advance\edfilldimen by \edtabcolsep
   \stepcounter{addcolcount}}%
  \advance\edfilldimen by \the \csname dcol\theaddcolcount\endcsname
  \fi
}

```

`\ifl@dstartendok` `\l@dcheckstartend`

`\l@dcheckstartend{`⟨*startcol*⟩`}{`⟨*endcol*⟩`}` checks that the values of ⟨*startcol*⟩ and ⟨*endcol*⟩ are sensible. If they are then `\ifl@dstartendok` is set TRUE, otherwise it is set FALSE.

```
\newif\ifl@dstartendok
\newcommand{\l@dcheckstartend}[2]{%
  \l@dstartendoktrue
  \ifnum #1<\@ne
    \l@dstartendokfalse
    \led@err@LowStartColumn
  \fi
  \ifnum #2>30\relax
    \l@dstartendokfalse
    \led@err@HighEndColumn
  \fi
  \ifnum #1>#2\relax
    \l@dstartendokfalse
    \led@err@ReverseColumns
  \fi
}

```

`\edrowfill` `\@edrowfill@` `\@EDROWFILL@`

`\edrowfill{`⟨*startcol*⟩`}{`⟨*endcol*⟩`}`fill fills columns ⟨*startcol*⟩ to ⟨*endcol*⟩ inclusive with ⟨*fill*⟩ (e.g. `\hrulefill`, `\upbracefill`). This is a LaTeX style reimplementation and generalization of the original `\waklam`, `\Waklam`, `\waklamec`, `\wastricht` and `\wapunktel` macros.

```
\newcommand*{\edrowfill}[3]{%
  \l@dtabaddcols{#1}{#2}%
  \hb@xt@ \the\l@dcolwidth{\hb@xt@ \the\edfilldimen{#3}\hss}}
\let\@edrowfill@=\edrowfill
\def\@EDROWFILL@#1#2#3{\@edrowfill@{#1}{#2}{#3}}

```

`\edbeforetab` `\edaftertab`

The macro `\edbeforetab{`⟨*text*⟩`}{`⟨*math*⟩`}` puts ⟨*text*⟩ at the left margin before 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`.

```
\newcommand{\leftltab}[1]{%
  \hb@xt@\z@{\vbox{\edtabindent%
  \moveleft\Hilfsskip\hbox{\ #1}}\hss}}

```

`\leftrtab`

`\leftrtab{`⟨*text*⟩`}{`⟨*math*⟩`}` for `\edbeforetab` in `\rtab`.

```
\newcommand{\leftrtab}[2]{%
  #2\hb@xt@\z@{\vbox{\edtabindent%
    \advance\Hilfsskip by\dcoli%
    \moveleft\Hilfsskip\hbox{\ #1}}\hss}}

```

\leftctab

\leftctab{⟨*text*⟩}{⟨*math*⟩} for \edbeforetab in \ctab.

```
\newcommand{\leftctab}[2]{%
        \hb@xt@\z@{\vbox{\edtabindent\l@dcolcount=\l@dampcount%
        \advance\Hilfsskip by 0.5\dcoli%
        \setbox\hilfsbox=\hbox{\def\edlabel##1{}%
        \disablel@dtabfeet$\displaystyle{#2}$}%
        \advance\Hilfsskip by -0.5\wd\hilfsbox%
        \moveleft\Hilfsskip\hbox{\ #1}}\hss}%
        #2}

```

\rightctab

\rightctab{⟨*math*⟩}{⟨*text*⟩} for \edaftertab in \ctab.

```
\newcommand{\rightctab}[2]{%
        \setbox\hilfsbox=\hbox{\def\edlabel##1{}%
        \disablel@dtabfeet#2}\l@dampcount=\l@dcolcount%
        #1\hb@xt@\z@{\vbox{\edtabindent\l@dcolcount=\l@dampcount%
        \advance\Hilfsskip by 0.5\l@dcolwidth%
        \advance\Hilfsskip by -\wd\hilfsbox%
        \setbox\hilfsbox=\hbox{\def\edlabel##1{}%
        \disablel@dtabfeet$\displaystyle{#1}$}%
        \advance\Hilfsskip by -0.5\wd\hilfsbox%
        \advance\Hilfsskip by \edtabcolsep%
        \moveright\Hilfsskip\hbox{ #2}}\hss}%
        }

```

\rightltab

\rightltab{⟨*math*⟩}{⟨*text*⟩} for \edaftertab in \ltab.

```
\newcommand{\rightltab}[2]{%
        \setbox\hilfsbox=\hbox{\def\edlabel##1{}%
        \disablel@dtabfeet#2}\l@dampcount=\l@dcolcount%
        #1\hb@xt@\z@{\vbox{\edtabindent\l@dcolcount=\l@dampcount%
        \advance\Hilfsskip by\l@dcolwidth%
        \advance\Hilfsskip by-\wd\hilfsbox%
        \setbox\hilfsbox=\hbox{\def\edlabel##1{}%
        \disablel@dtabfeet$\displaystyle{#1}$}%
        \advance\Hilfsskip by-\wd\hilfsbox%
        \advance\Hilfsskip by\edtabcolsep%
        \moveright\Hilfsskip\hbox{ #2}}\hss}%
        }

```

\rightrtab

\rightrtab{⟨*math*⟩}{⟨*text*⟩} for \edaftertab in \rtab.

```
\newcommand{\rightrtab}[2]{%
        \setbox\hilfsbox=\hbox{\def\edlabel##1{}%
        \disablel@dtabfeet#2}%
        #1\hb@xt@\z@{\vbox{\edtabindent%
        \advance\Hilfsskip by-\wd\hilfsbox%
        \advance\Hilfsskip by\edtabcolsep%
        \moveright\Hilfsskip\hbox{ #2}}\hss}%
```

```
          }

```

\rtab
\edbeforetab
\edaftertab

`\rtab{`⟨*body*⟩`}` typesets ⟨*body*⟩ as an array with the entries right justified.

The process is first to measure the ⟨*body*⟩ to get the column widths, and then in a second pass to typeset the body.

```
\newcommand{\rtab}[1]{%
 \l@dnullfills
   \def\edbeforetab##1##2{\leftrtab{##1}{##2}}%
   \def\edaftertab##1##2{\rightrtab{##1}{##2}}%
   \measurembody{#1}%
 \l@drestorefills
   \variab
   \setmrowright #1\\&\\%
   \enablel@dtabfeet}

```

\measurembody

`\measurembody{`⟨*body*⟩`}` measures the array ⟨*body*⟩.

```
\newcommand{\measurembody}[1]{%
 \disablel@dtabfeet%
 \l@dcolcount=0%
 \nullsetzen%
 \l@dcolcount=0
 \measuremrow #1\\&\\%
 \global\l@dampcount=1}

```

\rtabtext

`\rtabtext{`⟨*body*⟩`}` typesets ⟨*body*⟩ as a tabular with the entries right justified.

```
\newcommand{\rtabtext}[1]{%
 \l@dnullfills
   \measuretbody{#1}%
 \l@drestorefills
   \variab
   \settrowright #1\\&\\%
   \enablel@dtabfeet}

```

\measuretbody

`\measuretbody{`⟨*body*⟩`}` measures the tabular ⟨*body*⟩.

```
\newcommand{\measuretbody}[1]{%
 \disable@notes%
 \disablel@dtabfeet%
 \l@dcolcount=0%
 \nullsetzen%
 \l@dcolcount=0
 \measuretrow #1\\&\\%
 \restore@notes%
 \global\l@dampcount=1}

```

\ltab \edbeforetab \edaftertab

Array with entries left justified.

```
\newcommand{\ltab}[1]{%
 \l@dnullfills
    \def\edbeforetab##1##2{\leftltab{##1}{##2}}%
    \def\edaftertab##1##2{\rightltab{##1}{##2}}%
    \measurembody{#1}%
  \l@drestorefills
    \variab
    \setmrowleft #1\\&\\%
    \enablel@dtabfeet}

```

\ltabtext

Tabular with entries left justified.

```
\newcommand{\ltabtext}[1]{%
  \l@dnullfills
    \measuretbody{#1}%
  \l@drestorefills
    \variab
    \settrowleft #1\\&\\%
    \enablel@dtabfeet}

```

\ctab \edbeforetab \edaftertab

Array with centered entries.

```
\newcommand{\ctab}[1]{%
  \l@dnullfills
    \def\edbeforetab##1##2{\leftctab{##1}{##2}}%
    \def\edaftertab##1##2{\rightctab{##1}{##2}}%
    \measurembody{#1}%
  \l@drestorefills
    \variab
    \setmrowcenter #1\\&\\%
    \enablel@dtabfeet}

```

\ctabtext

Tabular with entries centered.

```
\newcommand{\ctabtext}[1]{%
  \l@dnullfills
    \measuretbody{#1}%
  \l@drestorefills
    \variab
    \settrowcenter #1\\&\\%
    \enablel@dtabfeet}

```

\spreadtext

```
\newcommand{\spreadtext}[1]{%\l@dcolcount=\l@dampcount%
   \hb@xt@ \the\l@dcolwidth{\hbox{#1}\hss}}
```

\spreadmath

```
\newcommand{\spreadmath}[1]{%
  \hb@xt@ \the\l@dcolwidth{\hbox{$\displaystyle{#1}$}\hss}}

```

\HILFSskip \Hilfsskip More helpers.

```
\newskip\HILFSskip
\newskip\Hilfsskip

```

\EDTABINDENT

```
\newcommand{\EDTABINDENT}{%
    \ifnum\l@dcolcount=30\let\NEXT\relax\l@dcolcount=0%
    \else\stepl@dcolcount%
         \advance\Hilfsskip by\l@dcolwidth%
         \ifdim\l@dcolwidth=0pt\advance\hilfscount\@ne
         \else\advance\Hilfsskip by \the\hilfscount\edtabcolsep%
         \hilfscount=1\fi%
         \let\NEXT=\EDTABINDENT%
    \fi\NEXT}%
```

\edtabindent (was \tabindent)

```
\newcommand{\edtabindent}{%
    \l@dcolcount=0\relax
    \Hilfsskip=0pt%
    \hilfscount=1\relax
    \EDTABINDENT%
    \hilfsskip=\hsize%
    \advance\hilfsskip -\Hilfsskip%
    \Hilfsskip=0.5\hilfsskip%
    }%

```

\EDTAB (was \TAB)

```
\def\EDTAB #1|#2|{%
    \setbox\tabhilfbox=\hbox{$\displaystyle{#1}$}%
    \setbox\tabHilfbox=\hbox{$\displaystyle{#2}$}%
    \advance\tabelskip -\wd\tabhilfbox%
    \advance\tabelskip -\wd\tabHilfbox%
    \unhbox\tabhilfbox\hskip\tabelskip%
    \unhbox\tabHilfbox}%

```

\EDTABtext (was \TABtext)

```
\def\EDTABtext #1|#2|{%
    \setbox\tabhilfbox=\hbox{#1}%
    \setbox\tabHilfbox=\hbox{#2}%
    \advance\tabelskip -\wd\tabhilfbox%
    \advance\tabelskip -\wd\tabHilfbox%
    \unhbox\tabhilfbox\hskip\tabelskip%
    \unhbox\tabHilfbox}%
```

`\tabhilfbox` `\tabHilfbox` Further helpers.

```
\newbox\tabhilfbox
\newbox\tabHilfbox

```

XXVIII.2.4 Environments

`edarrayl` `edarrayc` `edarrayr` The ‘environment’ forms for `\ltab`, `\ctab` and `\rtab`.

```
\newenvironment{edarrayl}{\l@dcollect@body\ltab}{}
\newenvironment{edarrayc}{\l@dcollect@body\ctab}{}
\newenvironment{edarrayr}{\l@dcollect@body\rtab}{}

```

`edtabularl` `edtabularc` `edtabularr` The ‘environment’ forms for `\ltabtext`, `\ctabtext` and `\rtabtext`.

```
\newenvironment{edtabularl}{\l@dcollect@body\ltabtext}{}
\newenvironment{edtabularc}{\l@dcollect@body\ctabtext}{}
\newenvironment{edtabularr}{\l@dcollect@body\rtabtext}{}

```

XXIX Quotation's commands

`\initnumbering@quote` This macro, called at the begin of any numbered section, redefine locally the `quotation` and `quote` environment, in order to allow to use it inside numbered section.

`\quotation` `\endquotation` `\quote` `\endquote` `\initnumbering@quote` defines quotation environment.

```
\newcommand{\initnumbering@quote}{
    \ifnoquotation@\else
    \renewcommand{\quotation}{\par\leavevmode%
                                \parindent=1.5em%
                                \skipnumbering%
                                \ifautopar%
                                    \vskip-\parskip%
                                \else%
                                    \vskip\topsep%
                                \fi%
                                \global\leftskip=\leftmargin%
                                \global\rightskip=\leftmargin%
                            }
    \renewcommand{\endquotation}{\par%
                             \global\leftskip=0pt%
                                \global\rightskip=0pt%
                                \leavevmode%
                                \skipnumbering%
                                \ifautopar%
                                    \vskip-\parskip%
                                \else%
                                    \vskip\topsep%
                                \fi%
```

```
                                 }
    \renewcommand{\quote}{\par\leavevmode%
                                 \parindent=0pt%
                                 \skipnumbering%
                                 \ifautopar%
                                     \vskip-\parskip%
                                 \else%
                                     \vskip\topsep%
                                 \fi%
                                 \global\leftskip=\leftmargin%
                                 \global\rightskip=\leftmargin%
    }
    \renewcommand{\endquote}{\par%
                                 \global\leftskip=0pt%
                                 \global\rightskip=0pt%
                                 \leavevmode%
                                 \skipnumbering%
                                 \ifautopar%
                                     \vskip-\parskip%
                                 \else%
                                     \vskip\topsep%
                                 \fi%
                                 }
    \fi
}
```

XXX Section’s title commands

XXX.1 Commands to disable some feature

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

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

`\ledsectnomark` The `\ledsectnomark` only disables the `\chaptermark`, `\sectionmark` and `\subsectionmark` macros.

```
\newcommand{\ledsectnomark}{%
  \let\chaptermark\@gobble%
  \let\sectionmark\@gobble%
  \let\subsectionmark\@gobble%
}
```

XXX.2 General overview

The system of `\eledxxxx` commands to section text work like this:

1. When one of these commands is called, reledmac writes to an auxiliary files:
    - The section level.

  - The section title.
  - The side (when eledpar is used).
  - The pstart where the command is called.
  - If we have starred version or not.

2. reledmac adds the title of the section to pstart, as normal content. This is to enable critical notes.
3. When LaTeX is run a other time, this file is read. That:
   - Adds the pstart number to a list of pstarts where a sectioning command is used.
   - Defines a command, the name of which contains the pstart number, and which calls the normal LaTeX sectioning command.
4. This last command is called when the pstart is effectively printed.

XXX.3 `\beforeeledchapter` command

We do not define commands for `\eledsection` and related if the `noeledsec` option is loaded. We use etoolbox tests and not the `\ifxxx`...`\else`...`\fi` structure to prevent problem of expansions with command after the `\ifxxx` which contains `\fi`.

```
\notbool{@noeled@sec}{%
```

`\beforeeledchapter` For technical reasons, not yet solved, page-breaking before chapters can't be made automatically by eledmac. Users have to use `\beforeeledchapter`.

```
\ifl@dmemoir
  \newcommand\beforeeledchapter{%
  \clearforchapter%
}
\else
  \newcommand\beforeeledchapter{%
  \if@openright%
   \cleardoublepage%
  \else%
   \clearpage%
  \fi%
}
\fi
```

XXX.4 Auxiliary commands

`\if@eled@sectioning` The boolean `\if@eled@sectioning` is set to true when a sectioning command is called by a `\eledxxx` command, and set to false after. It is used to enable/disable line number printing.

```
\newif\if@eled@sectioning
```

nt@leftmargin@eledsection
t@rightmargin@eledsection

`\print@leftmargin@eledsection` and `\print@rightmargin@eledsection` are added by reledmac inside the code of sectioning command, in order to affix lines numbers. They include tests for RTL languages.

```
\def\print@rightmargin@eledsection{%
  \if@eled@sectioning%
    \begingroup%
    \if@RTL%
        \let\llap\rlap%
        \let\leftlinenum\rightlinenum%
        \let\leftlinenumR\rightlinenumR%
        \let\l@drd@ta\l@dld@ta%
        \let\l@drsn@te\l@dlsn@te%
    \fi%
    \hfill\l@drd@ta \csuse{LR}{\l@drsn@te}%
    \endgroup%
  \fi%
}%

\def\print@leftmargin@eledsection{%
  \if@eled@sectioning%
    \leavevmode%
    \begingroup%
    \if@RTL%
        \let\rlap\llap%
        \let\rightlinenum\leftlinenum%
        \let\rightlinenumR\leftlinenumR%
        \let\l@dld@ta\l@drd@ta%
        \let\l@dlsn@te\l@drsn@te%
    \fi%
    \l@dld@ta\csuse{LR}{\l@dlsn@te}%
    \endgroup%
  \fi%
}%

```

XXX.5 Patching standard commands

\chapter
\M@sect
\@mem@old@ssect
\@makechapterhead
\@makechapterhead
\@makeschapterhead
\@sect
\@ssect

We have to patch LaTeX, book and memoir sectioning commands in order to:

- Disable `\edtext` inside.
- Disable page breaking (for `\chapter`).
- Add line numbers and sidenotes.

Unfortunately, Maïeul Rouquette was not able to try if memoir is loaded. That is why eledmac tries to define for both standard class and memoir class.

```
\catcode`\#=12 % Space NEEDS by \catcode
\AtBeginDocument{%
\patchcmd{\chapter}{\clearforchapter}{%
```

```
  \if@eled@sectioning\else%
    \ifl@dprintingpages\else%
      \clearforchapter%
    \fi%
  \fi%
  }
  {}
  {}

\pretocmd{\M@sect}
  {\let\old@edtext=\edtext%
  \let\edtext=\dummy@edtext@showlemma%
  }
  {}
  {}

\apptocmd{\M@sect}
  {\let\edtext=\old@edtext}
  {}
  {}

\patchcmd{\M@sect}
  { #9}
  { #9%
  \print@rightmargin@eledsection%
  }
  {}
  {}

\patchcmd{\M@sect}
  {\hskip #3\relax}
  {\hskip #3\relax%
  \print@leftmargin@eledsection%
  }
  {}
  {}

\patchcmd{\@mem@old@ssect}
  {#5}
  {#5%
  \print@leftmargin@eledsection%
  }
  {}
  {}

\patchcmd{\@mem@old@ssect}
  {\hskip #1}
  {\hskip #1%
  \print@rightmargin@eledsection%
  }
  {}
  {}

\patchcmd{\chapter}{\if@openright\cleardoublepage\else\clearpage\fi}{%
  \if@eled@sectioning\else%
    \ifl@dprintingpages\else%
    \if@openright\cleardoublepage\else\clearpage\fi%No clearpage inside a \eledsection: will keep criti
    \fi%
  \fi%
  }%
  {}%
  {}%

\patchcmd{\@makechapterhead}
  {#1}
  {\print@leftmargin@eledsection%
    #1%
  \print@rightmargin@eledsection%
  }
  {}
  {}

\patchcmd{\@makechapterhead}% For BIDI
  {\if@RTL\raggedleft\else\raggedright\fi}%
  {\if@eled@sectioning\else%
    \if@RTL\raggedleft\else\raggedright\fi%
  \fi%
  }%
  {}%
  {}%

\patchcmd{\@makeschapterhead}
  {#1}
  {\print@leftmargin@eledsection%
    #1%
   \print@rightmargin@eledsection%
  }
  {}
  {}

\pretocmd{\@sect}
  {\let\old@edtext=\edtext
  \let\edtext=\dummy@edtext@showlemma%
  }
  {}
  {}

\apptocmd{\@sect}
  {\let\edtext=\old@edtext}
  {}
  {}

\pretocmd{\@ssect}
  {\let\old@edtext=\edtext%
  \let\edtext=\dummy@edtext@showlemma%
  }
  {}
  {}

\apptocmd{\@ssect}
  {\let\edtext=\old@edtext}
  {}
  {}

```

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.

```
\@ifpackageloaded{nameref}{

  \patchcmd{\NR@sect}
    {#8}
    {#8%
    \print@rightmargin@eledsection%
    }
    {}
    {}

  \patchcmd{\NR@sect}
    {\hskip #3\relax}
    {\hskip #3\relax%
    \print@leftmargin@eledsection%
    }
    {}
    {}

  \patchcmd{\NR@ssect}
    {#5}
    {#5%
    \print@rightmargin@eledsection%
    }
    {}
    {}

  \patchcmd{\NR@ssect}
    {\hskip #1}
    {\hskip #1%
    \print@leftmargin@eledsection%
    }
    {}
    {}
```

```
 }%
 {
 \patchcmd{\@sect}
   {#8}
   {#8%
   \print@rightmargin@eledsection%
   }
   {}
   {}

 \patchcmd{\@sect}
   {\hskip #3\relax}
   {\hskip #3\relax%
   \print@leftmargin@eledsection%
   }
   {}
   {}

 \patchcmd{\@ssect}
   {#5}
   {#5%
   \print@rightmargin@eledsection%
   }
   {}
   {}

 \patchcmd{\@ssect}
   {\hskip #1}
   {\hskip #1%
   \print@leftmargin@eledsection%
   }
   {}
   {}
 }%
}
\catcode`\#=6 %Space NEEDS by \catcode
```

XXX.6 Main code of \eledxxx commands

\eled@sectioning@out

\eled@sectioning@out is the output file, to dump the pstarts where a sectioning command is used.

```
\newwrite\eled@sectioning@out
```

\eledchapter
\eledsection
\eledsubsection
\eledsubsubsection
\eledchapter*
\eledsection*
\eledsubsection*
\eledsubsubsection*

And now, the user sectioning commands, which write to the file, and also add content as a "normal" line.

```
\newcommand{\eledchapter}[2][]{%
  #2%
  \ifledRcol%
    \immediate\write\eled@sectioningR@out{%
```

```
      \string\eled@chapter{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR}{}{R}
      }%
  \else%
    \immediate\write\eled@sectioning@out{%
      \string\eled@chapter{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL}{}{}
      }%
  \fi%
}

\newcommand{\eledsection}[2][]{%
  #2%
  \ifledRcol%
    \immediate\write\eled@sectioningR@out{%
      \string\eled@section{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR}{}{R}
      }%
  \else%
    \immediate\write\eled@sectioning@out{%
      \string\eled@section{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL}{}{}
      }%
  \fi%
}

\newcommand{\eledsubsection}[2][]{%
  #2%
  \ifledRcol%
    \immediate\write\eled@sectioningR@out{%
    \string\eled@subsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR}{}{R}
      }%
  \else%
    \immediate\write\eled@sectioning@out{%
    \string\eled@subsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL}{}{}
      }%
  \fi%
}
\newcommand{\eledsubsubsection}[2][]{%
  #2%
  \ifledRcol%
    \immediate\write\eled@sectioningR@out{%
    \string\eled@subsubsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR}{}{R}
      }%
  \else%
    \immediate\write\eled@sectioning@out{%
    \string\eled@subsubsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL}{}{}
      }%
  \fi%
}


\WithSuffix\newcommand\eledchapter*[2][]{%
  #2%
  \ifledRcol%
    \immediate\write\eled@sectioningR@out{%
     \string\eled@chapter{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR}{*}{R}
      }%
  \else%
    \immediate\write\eled@sectioning@out{%
      \string\eled@chapter{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL}{*}{}
      }%
  \fi%
}

\WithSuffix\newcommand\eledsection*[2][]{%
  #2%
  \ifledRcol%
    \immediate\write\eled@sectioningR@out{%
     \string\eled@section{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR}{*}{R}
      }%
  \else%
    \immediate\write\eled@sectioning@out{%
      \string\eled@section{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL}{*}{}
      }%
  \fi%
}

\WithSuffix\newcommand\eledsubsection*[2][]{%
  #2%
  \ifledRcol%
    \immediate\write\eled@sectioningR@out{%
    \string\eled@subsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR}{*}{R}
      }%
  \else%
    \immediate\write\eled@sectioning@out{%
    \string\eled@subsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL}{*}{}
      }%
  \fi%
}

\WithSuffix\newcommand\eledsubsubsection*[2][]{%
  #2%
  \ifledRcol%
    \immediate\write\eled@sectioningR@out{%
    \string\eled@subsubsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsR}{*}{R}
      }%
  \else%
    \immediate\write\eled@sectioning@out{%
    \string\eled@subsubsection{#1}{\unexpanded{#2}}{\the\l@dnumpstartsL}{*}{}
      }%
  \fi%
}
```

XXX.7 Macros written in the auxiliary file

\eled@chapter
\eled@section
\eled@subsection
\eled@subsubsection

The sectioning macros, called in the auxiliary file. They have five arguments:

1. Optional arguments of LaTeX sectioning command.
2. Mandatory arguments of LaTeX sectioning command.
3. Pstart number.
4. Side: R if right, nothing if left.
5. Starred or not.

```
\def\eled@chapter#1#2#3#4#5{%
    \ifstrempty{#4}%
      {%
      \ifstrempty{#1}%
        {%
     \global\csdef{eled@sectioning@#3#5}{\let\edtext=\dummy@edtext@showlemma\chapter{#2}}
     \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext{}\chaptermark{#2}}%
        }%Need for \pairs, because of using parbox.
        {%
     \global\csdef{eled@sectioning@#3#5}{\let\edtext=\dummy@edtext@showlemma\chapter[#1]{
     \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext{}\chaptermark{#2}}%Need f
        }%
      }%
      {%
      \ifstrempty{#1}%
     {\global\csdef{eled@sectioning@#3#5}{\let\edtext=\dummy@edtext@showlemma\chapter*{#2
     {\global\csdef{eled@sectioning@#3#5}{\let\edtext=\dummy@edtext@showlemma\chapter*[#1
      }%
  \listcsgadd{eled@sections#5@@}{#3}%
    }
\def\eled@section#1#2#3#4#5{%
    \ifstrempty{#4}%
      {\ifstrempty{#1}%
        {%
        \global\csdef{eled@sectioning@#3#5}{\section{#2}}%
     \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext{}\sectionmark{#2}}%Need f
        }%
        {%
        \global\csdef{eled@sectioning@#3#5}{\section[#1]{#2}}%
     \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext{}\sectionmark{#1}}%Need f
        }%
      }%
      {\ifstrempty{#1}%
        {\global\csdef{eled@sectioning@#3#5}{\section*{#2}}}%
     {\global\csdef{eled@sectioning@#3#5}{\section*[#1]{#2}}}%Bug in LaTeX!
      }
  \listcsgadd{eled@sections#5@@}{#3}%
    }
```

```
\def\eled@subsection#1#2#3#4#5{%
    \ifstrempty{#4}%
      {\ifstrempty{#1}%
        {%
        \global\csdef{eled@sectioning@#3#5}{\subsection{#2}}%
     \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext{}\csuse{subsectionmark}{#2}}%Need fo
        }%
        {%
        \global\csdef{eled@sectioning@#3#5}{\subsection[#1]{#2}}%
     \global\csdef{eled@sectmark@#3#5}{\let\edtext=\dummy@edtext{}\csuse{subsectionmark}{#1}}%Need fo
        }%
      }%
      {\ifstrempty{#1}%
        {\global\csdef{eled@sectioning@#3#5}{\subsection*{#2}}}%
     {\global\csdef{eled@sectioning@#3#5}{\subsection*[#1]{#2}}}%Bug in LaTeX!
      }
  \listcsgadd{eled@sections#5@@}{#3}%
    }
\def\eled@subsubsection#1#2#3#4#5{%
    \ifstrempty{#4}%
      {\ifstrempty{#1}%
        {\global\csdef{eled@sectioning@#3#5}{\subsubsection{#2}}}%
        {\global\csdef{eled@sectioning@#3#5}{\subsubsection[#1]{#2}}}%
      }%
      {\ifstrempty{#1}%
        {\global\csdef{eled@sectioning@#3#5}{\subsubsection*{#2}}}%
     {\global\csdef{eled@sectioning@#3#5}{\subsubsection*[#1]{#2}}}%Bug in LaTeX!
      }
  \listcsgadd{eled@sections#5@@}{#3}%
    }

```

End of the conditional test about `noeledsec` option.

```
}{}
```

XXXI Page breaking or no page breaking depending of specific lines

By default, page breaks are automatic. However, the user can define lines which will force page breaks, or prevent page breaks around one specific line. On the first run, the line-list file records the line number of where the page break is being changed (either forced, or prevented). On the next run, page breaks occur either before or after this line, depending on how the user sets the command. The default setting is after the line.

`\normal@page@break` `\normal@page@break` is an etoolbox list which contains the absolute line number of the last line, for each page.

```
\def\normal@page@break{}
```

\prev@pb \prev@nopb The \l@prev@pb macro is a etoolbox list, which contains the lines in which page breaks occur (before or after). The \l@prev@nopb macro is a etoolbox list, which contains the lines with NO page break before or after.

```
\def\l@prev@pb{}
\def\l@prev@nopb{}
```

\ledpb \ledpbnum \lednopb \lednopbnum The \ledpb macro writes the call to \led@pb in line-list file. The \ledpbnum macro writes the call to \led@pbnum in line-list file. The \lednopb macro writes the call to \led@nopb in line-list file. The \lednopbnum macro writes the call to \led@nopbnum in line-list file.

```
\newcommand{\ledpb}{\write\linenum@out{\string\led@pb}}
\newcommand{\ledpbnum}[1]{\write\linenum@out{\string\led@pbnum{#1}}}
\newcommand{\lednopb}{\write\linenum@out{\string\led@nopb}}
\newcommand{\lednopbnum}[1]{\write\linenum@out{\string\led@nopbnum{#1}}}
```

\led@pb \led@pbnum \led@nopb \led@nopbnum The \led@pb adds the absolute line number in the \prev@pb list. The \led@pbnum adds the argument in the \prev@pb list. The \led@nopb adds the absolute line number in the \prev@nopb list. The \led@nopbnum adds the argument in the \prev@nopb list.

```
\newcommand{\led@pb}{\listxadd{\l@prev@pb}{\the\absline@num}}
\newcommand{\led@pbnum}[1]{\listxadd{\l@prev@pb}{#1}}
\newcommand{\led@nopb}{\listxadd{\l@prev@nopb}{\the\absline@num}}
\newcommand{\led@nopbnum}[1]{\listxadd{\l@prev@nopb}{#1}}
```

\ledpbsetting \led@pb@setting The \ledpbsetting macro only changes the value of \led@pb@macro, for which the default value is before.

```
\def\led@pb@setting{before}
\newcommand{\ledpbsetting}[1]{\gdef\led@pb@setting{#1}}
```

\led@check@pb \led@check@nopb The \led@check@pb and \led@check@nopb are called before or after each line. They check if a page break must occur, depending on the current line and on the content of \l@pb.

```
\newcommand{\led@check@pb}{\xifinlist{\the\absline@num}{\l@prev@pb}{\pagebreak[4]}{}}
\newcommand{\led@check@nopb}{%
    \IfStrEq{\led@pb@setting}{before}{%
      \xifinlist{\the\absline@num}{\l@prev@nopb}%
         {\numdef{\abs@prevline}{\the\absline@num-1}%
         \xifinlist{\abs@prevline}{\normal@page@break}%
           {\nopagebreak[4]\enlargethispage{\baselineskip}}%
           {}}%
         {}}%
      {}%
    {}%
    \IfStrEq{\led@pb@setting}{after}{%
      \xifinlist{\the\absline@num}{\l@prev@nopb}{%
        \xifinlist{\the\absline@num}{\normal@page@break}%
          {\nopagebreak[4]\enlargethispage{\baselineskip}}%
            {}%
}%
```

```
         {}}%
       {}%
     {}%
}
```

XXXII Long verse: prevents being separated by a page break

\iflednopbinverse The \lednopbinverse boolean is set to false by default. If set to true, reledmac will automatically prevent page breaks inside verse. The declaration is made at the beginning of the file, because it is used as a package option.

\check@pb@in@verse The \check@pb@in@verse checks if a verse is broken in two page. If true, it adds:

- The absolute line number of the first line of the verse -1 in the \led@pb list, if the page break must occur before the verse.
- The absolute line number of the first line of the verse -1 in the \led@nopb list, if the page break must occur after the verse.

```
\newcommand{\check@pb@in@verse}{%
   \ifinstanza\iflednopbinverse\ifinserthangingsymbol% Using stanzas and enabling page breaks in verse c
        \ifnum\page@num=\last@page@num\else%If we have change page
          \IfStrEq{\led@pb@setting}{before}{%
              \numgdef{\abs@line@verse}{\the\absline@num-1}%
              \ledpbnum{\abs@line@verse}%
            }{}%
          \IfStrEq{\led@pb@setting}{after}{%
              \numgdef{\abs@line@verse}{\the\absline@num-1}%
              \lednopbnum{\abs@line@verse}%
          }{}%
        \fi%
    \fi\fi\fi%
}
```

XXXIII Compatibility with eledmac

Here, we define some command for the eledmac-compat option.

```
\ifeledmaccompat@%

\newcommand{\footnormalX}[1]{\arrangementX[#1]{normal}}%
\newcommand{\footparagraphX}[1]{\arrangementX[#1]{paragraph}}%
\newcommand{\foottwocolX}[1]{\arrangementX[#1]{twocol}}%
\newcommand{\footthreecolX}[1]{\XarrangementX[#1]{threecol}}%

\unless\ifnocritical@
  \newcommand{\footnormal}[1]{\Xarrangement[#1]{normal}}%
```

```
\newcommand{\footparagraph}[1]{\Xarrangement[#1]{paragraph}}%
\newcommand{\foottwocol}[1]{\Xarrangement[#1]{twocol}}%
\newcommand{\footthreecol}[1]{\Xarrangement[#1]{threecol}}%
\let\hsizetwocol\Xhsizetwocol
\let\hsizethreecol\Xhsizethreecol
\let\bhookXnote\Xbhooknote
\let\boxsymlinenum\Xboxsymlinenum
\let\symlinenum\Xsymlinenum
\let\beforenumberinfootnote\Xbeforenumber
\let\afternumberinfootnote\Xafternumber
\let\beforeXsymlinenum\Xbeforesymlinenum
\let\afterXsymlinenum\Xaftersymlinenum
\let\inplaceofnumber\Xinplaceofnumber
\let\Xlemmaseparator\lemmaseparator
\let\afterlemmaseparator\Xafterlemmaseparator
\let\beforelemmaseparator\Xbeforelemmaseparator
\let\inplaceoflemmaseparator\Xinplaceoflemmaseparator
\let\txtbeforeXnotes\Xtxtbeforenotes
\let\afterXrule\Xafterrule
\let\numberonlyfirstinline\Xnumberonlyfirstinline
\let\numberonlyfirstintwolines\Xnumberonlyfirstintwolines
\let\nonumberinfootnote\Xnonumberinfootnote
\let\pstartinfootnote\Xpstart
\let\pstartinfootnoteeverytime\Xpstarteverytime
\let\onlyXpstart\Xonlypstart
\let\Xnonumberinfootnote\Xnonumber
\let\nonbreakableafternumber\Xnonbreakableafternumber
\let\maxhXnotes\Xmaxhnotes
\let\beforeXnotes\Xbeforenotes
\let\boxlinenum\Xboxlinenum
\let\boxlinenumalign\Xboxlinenumaligm
\let\boxstartlinenum\Xboxstartlinenum
\let\boxendlinenum\Xboxendlinenum
\let\twolines\Xtwolines
\let\morethantwolines\Xmorethantwolines
\let\twolinesbutnotmore\Xtwolinesbutnotmore
\let\twolinesonlyinsamepage\Xtwolinesonlyinsamepage
\fi

\unless\ifnofamiliar@
\let\notesXwidthliketwocolumns\noteswidthliketwocolumnsX
\fi
\newcommandx[2][1,usedefault]{\parafootsep}{%
  \Xparafootsep[#1]{#2}%
  \parafootsepX[#1]{#2}
}%

\newcommandx[2][1,usedefault]{\afternote}{%
  \Xafternote[#1]{#2}%
  \afternoteX[#1]{#2}%
}%

\unless\ifnoend@
\let\XendXtwolines\Xendtwolines
\let\XendXmorethantwolines\Xendmorethantwolines
\let\bhookXendnote\Xendbhooknote
\let\boxXendlinenum\Xendboxlinenum%
\let\boxXendlinenumalign\Xendboxlinenumalign%
\let\boxXendstartlinenum\Xendboxstartlinenum%
\let\boxXendendlinenum\Xendboxendlinenum%
\let\XendXlemmaseparator\Xendlemmaseparator
\let\XendXbeforelemmaseparator\Xendbeforelemmaseparator
\let\XendXafterlemmaseparator\Xendafterlemmaseparator
\let\XendXinplaceoflemmaseparator\Xendinplaceoflemmaseparator
\fi

\AtBeginDocument{%
  \ifdef\lineref{}{\let\lineref\edlineref}%
}%


\fi%
```

</code>

Appendix A Some things to do when changing version

Appendix A.1 Migrating from edmac

Appendix A.2 Migration from ledmac to eledmac

In eledmac, some changes were made in the code to allow for easy customization. This can cause problems for people who have made their own customizations. The next sections explain how to correct this.

If you have created your own series using `\addfootins` and `\addfootinsX`, you should use instead the `\newseries` command (see 5.5.1 p. 19). You must remove your `\Xfootnote` command.

If you have customized the `\XXXXXXfmt` command, you should check if commands for display options (6 p. 20) and options in `\Xfootnote` (5.2.2 p. 14) cannot do the same thing. If not, you can add a new ticket in Github to request a new function for doing this.[33]

If for some reason you do not want to make the modifications to use eledmac new functions, you can continue using your own `\XXXXXfmt` command, but you must replace:

```
\renewcommand*{XXXXfmt}[3]
```

with

```
\renewcommandx*{XXXXfmt}[4][4=Z]
```

If you do not make that, you will see a spurious `[X]`, where X is series letter.

If you used a `\protect` command inside a `\footnote` command inside a numbered section, you must change the `\protect` to `\noexpand`. If you do not, the command after the `\protect` will not be used.

Appendix A.3 Migration to eledmac 1.5.1

The version 1.5.1 corrects a bug with `stanzaindentsrepetition` (cf. 8.3 p. 31). 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 value equal to 2, you must change your `\setstanzaindents`. Explanation:

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

[33] `https://github.com/maieul/ledmac/issues`

This code, in a version older than 1.5.1, made that the first verse had an indent of 0, the secund verse of 1, the third verse of 0, the fourth verse of 1 etc.

But instead the code should have assigned the reverse: the first verse had an indent of 1, the secund verse of 0, the third verse of 1, the fourth verse of 0 etc.

So version 1.5.1 corrected this bug. If you want to keep the older presentation, you must change:

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

by:

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

Appendix A.4 Migration to eledmac 1.12.0

The migration to eledmac 1.12.0 is easy:

- You must delete all the auxiliary files, and so one, make the normal three runs.
- If you have modified `\l@reg`, which is not advisable, you must rename it to `\@nl@reg`.

Anyway, there is another problem. If you have text in brackets just after `\pstart` or `\pend`, the text will be considered an optional argument of `\pstart` or `\pend` (see 4.2.3 p. 8). In this case, just add a `\relax` between `\pstart`/`\pend` and the brackets.

The version 1.12.0 adds a new better way to manage section titles inside numbered text. Please read § 14.2 (14.2 p. 43).

Appendix A.5 Migration to eledmac 17.1

The version change the default behavior of `\Xpstart`. Henceforth, the pstart will be printed if footnote only for the section of text where you have called `\numberpstarttrue`.

We do not see any reason to print it in other section. However, if you want to print the pstart number in all footnote, with or without `\numberpstarttrue`, you can use `\Xpstarteverytime`.

Appendix A.6 Migration to eledmac 1.21.0

Appendix A.6.1 `\Xledsetnormalparstuff` and `\ledsetnormalparstuffX`

The `\ledsetnormalparstuff` has been split in two different commands:

- `\Xledsetnormalparstuff` for critical notes;
- `\ledsetnormalparstuffX` for familiar notes.

The new commands take an optional argument which is the series letter. If you have redefined `\ledsetnormalparstuff` or commands which call them, you must make the appropriate change

Appendix A.6.2 Endnotes

In any case, clean the `.end` file before the next run.

The previous version of Eledmac had a bug: there were two spaces between the start page number and the start line number, but only one space between the end page number and the end line number.

Indeed, 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, just load the package with the `oldprintnpnumspace` option.

If you have redefined `\endprint`, you must:

- Contact us to 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` commands takes now a first optional argument, which is the label for the hyperreference. If you have redefined it, change your redefinition, and check if you can avoid this redefinition by redefining only `\ledinnotemark`.

Appendix A.8 Migration to eledmac 1.23.0

People must delete the numbered auxiliary file before new run after update of eledmac.

Appendix A.9 Migration from eledmac to reledmac

There are four types of change in reledmac which implies user intervention.

Some tools specific to memoir has been replaced to more generic tools, in order to obtain a more manageable code. Some commands and options have been deleted because they are deprecated. Some customization by `\renewcommand` have been replaced by commands. Some commands name have been changed in order to have a more logical and uniform.

Appendix A.9.1 Multiple index with memoir

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.

Consequently: memoir's users are invited to use indextool and imakeidx in order to have multiple index.

Appendix A.9.2 Deprecated commands and options

Here, you will find a tabular of deprecated commands and their alternative. Notes that the use of these command can have been changed. Please read the handbook.

Deprecated command	*Replaced by*
`\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 deleted, because linked to deprecated commands.

The `oldprintnpnumspace` option has also been deleted, because linked to an historical bug. The `\usingedtext` and `\usingcritext` commands are also deprecated.

Appendix A.9.3 `\renewcommand` replaced by command

Many use of `\renewcommand` has been replaced by use of specific commands. Please read handbook on specific command.

Deprecated `\renewcommand`	*Replaced by*
`\@led@extranofeet`	`\newseries`
`\endstanzaextra`	Optional argument of `\&`
`\hangingsymbol`	`\sethangingsymbol`
`\ledfootinsdim`	`\Xmaxhnotes` and `\maxhnotesX`
`\parafootftmsep`	`\Xparafootsep` and `\parafootsepX`
`\notenumfont`	`\Xnotenumfont`, `\Xendnotenumfont` and `\notenumfontX`
`\notefontsetup`	`\Xnotefontsize`, `\Xendnotefontsize` and `\notefontsizeX`
`\startstanzahook`	Optional argument of `\stanza`
`\symplinenum`	`\Xsymlinenum`

Appendix A.9.4 Commands which names have changed

In order to help migration from eledmac to reledmac, you could load reledmac with `eledmac-compat` option. However, it is better to change the command names. In many case, you use only few of them, except for the command like `\footparagraph`.

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`
`\bhookXnote`	`\Xbhookendnote`
`\bhookXnote`	`\Xbooknote`
`\boxendlinenum`	`\Xboxendlinenum`
`\boxlinenum`	`\Xboxlinenum`
`\boxlinenumalign`	`\Xboxlinenumalign`
`\boxstartlinenum`	`\Xboxstartlinenum`
`\boxsymlinenum`	`\Xboxsymlinenum`
`\boxXendlinenum`	`\Xendboxlinenum`
`\boxXendlinenumalign`	`\Xendboxlinenumalign`
`\boxXendstartlinenum`	`\boxXendstartlinenum`
`\letboxXendendlinenum`	`\Xendletboxendlinenum`
`\hsizetwocol`	`\Xhsizetwocol`
`\hsizethreecol`	`\Xhsizethreecol`
`\inplaceoflemmaseparator`	`\Xinplaceoflemmaseparator`
`\inplaceofnumber`	`\Xinplaceofnumber`
`\lemmaseparator`	`\Xlemmaseparator`
`\maxhXnotes`	`\Xmaxhnotes`
`\morethantwolines`	`\Xmorethantwolines`
`\nonumberinfootnote`	`\Xnonumber`
`\notesXwidthliketwocolumns`	`\noteswidthliketwocolumnsX`
`\numberonlyfirstinline`	`\Xnumberonlyfirstinline`
`\numberonlyfirstintwolines`	`\Xnumberonlyfirstintwolines`

Old command	*New command*
`\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`