

1 *lineno.sty v4.41 2005/11/02*

2

3

4

A L^AT_EX package to attach
line numbers to paragraphs

5

Stephan I. Böttcher
Uwe Lück

6

boettcher@physik.uni-kiel.de
http://contact-ednotes.sty.de.vu

7

8 **Contents**

9	1	Introductions	2
10	1.1	Introduction to versions v < 4	3
11	1.2	Introduction to versions v4.00ff. (UL)	3
12	1.3	Availability	5
13	1.4	Introductory code	5
14	2	Put the line numbers to the lines	6
15	2.1	Basic code of <code>lineno.sty \output</code>	6
16	2.2	<code>\LineNoTest</code>	9
17	2.3	Other output routines (v4.2)	9
18	2.4	<code>\MakeLineNo</code> : Actually attach line number	11
19	3	Control line numbering	14
20	3.1	Inserting <code>\output</code> calls	14
21	3.2	Turning on/off	16
22	3.3	Display math	18
23	4	Line number references	20
24	4.1	Internals	20
25	4.2	The <code>\linelabel</code> command	22

1	5	The appearance of the line numbers	24
2	5.1	Basic code	24
3	5.2	Running line numbers	27
4	5.3	Pagewise line numbers	27
5	5.4	Twocolumn mode (New v3.06)	32
6	5.5	Numbering modulo m , starting at f	33
7	6	Package options	37
8	6.1	Extended referencing to line numbers. (v4.2)	37
9	6.2	<code>\linelabel</code> in math mode	38
10	6.3	Arrays, tabular environments (Revised v4.11)	38
11	6.4	Switch among settings	39
12	6.5	Compatibility with <code>hyperref</code>	41
13	6.6	A note on calling so many options	43
14	6.7	Execute options	43
15	7	Former package extensions	43
16	7.1	<code>displaymath</code>	43
17	7.2	Line numbers in internal vertical mode	44
18	7.3	Line number references with offset	45
19	7.4	Numbered quotation environments	46
20	7.5	Frame around a paragraph	47
21	8	Move <code>\vadjust</code> items (New v4.00)	48
22	8.1	Redefining <code>\vadjust</code>	48
23	8.2	Redefining the L ^A T _E X commands	49
24	8.3	Reminder on obsolescence	51
25	9	The final touch	52
26	10	The user commands	52
27	10.1	Customization hooks	55

28 1 Introductions

29 (New v4.00) Parts of former first section have been rendered separate sub-
 30 sections for package version v4.00. (/New v4.00)

1.1 Introduction to versions $v < 4$

This package provides line numbers on paragraphs. After $\text{T}_{\text{E}}\text{X}$ has broken a paragraph into lines there will be line numbers attached to them, with the possibility to make references through the $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ $\backslash\text{ref}$, $\backslash\text{pageref}$ cross reference mechanism. This includes four issues:

- attach a line number on each line,
- create references to a line number,
- control line numbering mode,
- count the lines and print the numbers.

The first two points are implemented through patches to the output routine. The third by redefining $\backslash\text{par}$, $\backslash\text{@par}$ and $\backslash\text{@@par}$. The counting is easy, as long as you want the line numbers run through the text. If they shall start over at the top of each page, the aux-file as well as $\text{T}_{\text{E}}\text{X}$'s memory have to carry a load for each counted line.

I wrote this package for my wife Petra, who needs it for transcriptions of interviews. This allows her to precisely refer to passages in the text. It works well together with $\backslash\text{marginpars}$, but not too well with displaymath . $\backslash\text{footnotes}$ are a problem, especially when they are split, but we may get there. (New v4.00 UL) Version v4.00 overcomes the problem, I believe. (/UL /New v4.00)

`lineno.sty` works surprisingly well with other packages, for example, `wrapfig.sty`. So please try if it works with whatever you need, and if it does, please tell me, and if it does not, tell me as well, so I can try to fix it.

1.2 Introduction to versions v4.00ff. (UL)

`lineno.sty` has been maintained by Stephan until version v3.14. From version v4.00 onwards, maintenance is shifting towards Uwe Lück (UL), who is the author of v4...code and of v4...changes in documentation. This came about as follows.

Since late 2002, Christian Tapp and Uwe Lück have employed `lineno.sty` for their `ednotes.sty`, a package supporting critical editions—cf.

<http://ednotes.sty.de.vu>

—while you find `ednotes.sty` and surrounding files in CTAN folder `/macros/latex/contrib/ednotes`.

1 Soon, some weaknesses of `lineno.sty` showed up, mainly since Chris-
2 tian’s critical editions (using `ednotes.sty`) needed lots of `\linelabels` and
3 footnotes. (These weaknesses are due to weaknesses of L^AT_EX’s `\marginpar`
4 mechanism that Stephan used for `\linelabel`.) So we changed some
5 `lineno.sty` definitions in some extra files, which moreover offered new fea-
6 tures. We sent these files to Stephan, hoping he would take the changes into
7 `lineno.sty`. However, he was too short of time.

8 Writing a TUGboat article on Ednotes in 2004, we hoped to reduce the
9 number of files in the Ednotes bundle and so asked Stephan again. Now he
10 generously offered maintenance to me, so I could execute the changes on my
11 own.

12 The improvements are as follows:

13 (i) Footnotes placement approaches intentions better (footnotes formerly
14 liked to pile up at late pages).

15 (ii) The number of `\linelabels` in one paragraph is no longer limited to
16 18.

17 (iii) `\pagebreak`, `\nopagebreak`, `\vspace`, and the star and optional ver-
18 sions of `\` work as one would expect (section 8).

19 (iv) A command is offered which chooses the first line number to be printed
20 in the margin (subsection 5.5).

21 (v) (New v4.1) L^AT_EX tabular environments (optionally) get line numbers
22 as well, and you can refer to them in the usual automatic way. (It may
23 be considered a shortcoming that, precisely, *rows* are numbered, not
24 lines.—See subsection 6.3.)

25 (vi) We are moving towards referring to math items (subsection 6.2 and the
26 hooks in subsection 4.2). (/New v4.1)

27 (Thanks to Stephan for making this possible!)

28 Ednotes moreover profits from Stephan’s offer with regard to the doc-
29 umentation of our code which yielded these improvements formerly. This
30 documentation now becomes printable, being part of the `lineno.sty` docu-
31 mentation.

32 Of course, Stephan’s previous `lineno.sty` versions were a great and
33 ingenious work and exhibit greatest T_EXpertise. I never could have done
34 this. I learnt a lot in studying the code when Christian pointed out strange
35 output results and error messages, and there are still large portions of
36 `lineno.sty` which I don’t understand (consider only pagewise numbering

1 of lines). Fortunately, Stephan has offered future help if needed.—My code
 2 for attaching line numbers to *tabular environments* (as mentioned above,
 3 now still in `edtable.sty`) developed from macros which Stephan and Chris-
 4 tian experimented with in December 2002. Stephan built the basics. (How-
 5 ever, I then became too proud to follow his advice only to use and modify
 6 `longtable.sty`.)

7 There are some issues concerning use of counters on which I don't agree
 8 with Stephan and where I would like to change the code if `lineno.sty` is
 9 “mine” as Stephan offered. However, Stephan is afraid of compatibility prob-
 10 lems from which, in particular, his wife could suffer in the near future. So he
 11 demanded that I change as little as possible for my first version. Instead of
 12 executing changes that I plan I just offer my opinions at the single occasions.
 13 I hope to get in touch this way with users who consider subtle features vital
 14 which I consider strange.

15 On the other hand, the sections on improvements of the implementation
 16 have been blown up very much and may be tiring and litte understandable
 17 for mere *users*. These users may profit from the present presentation just by
 18 jumping to sections 6 and 10. There is a user's guide `ulineno.tex` which may
 19 be even more helpful, but it has not been updated for a while.

20 1.3 Availability

21 In case you have found the present file otherwise than from CTAN: A recent
 22 version and documentation of this package should be available from CTAN
 23 folder `/macros/latex/contrib/lineno`. Or mail to one of the addresses at top
 24 of file.

25 1.4 Introductory code

26 This style option is written for $\text{\LaTeX} 2_{\epsilon}$, November 1994 or later, since we
 27 need the `\protected@write` macro.

28 (New v4.00) And we use `\newcommand*` for controlling length of user
 29 macro arguments, which has been available since December 1994.

```
1 \NeedsTeXFormat{LaTeX2e}[1994/12/01]
2 \ProvidesPackage{lineno}
3 [\filedate\space line numbers on paragraphs \fileversion]
```

30 (/New v4.00)

1 2 Put the line numbers to the lines

2 (New v4.00) This section contained the most basic package code previously.
 3 For various purposes of version 4... , much of these basics have been to be
 4 modified. Much of my (UL's) reasoning on these modifications has been
 5 to be reported. Sorry, the present section has been blown up awfully thus
 6 and contains ramifications that may be difficult to trace. We add some
 7 `\subsection` commands in order to cope with the new situation. (/New
 8 v4.00)

9 2.1 Basic code of `lineno.sty` \output

10 The line numbers have to be attached by the output routine. We simply set
 11 the `\interlinepenalty` to -100000 . The output routine will be called after
 12 each line in the paragraph, except the last, where we trigger by `\par`. The
 13 `\linenopenalty` is small enough to compensate a bunch of penalties (e.g.,
 14 with `\samepage`).

15 (New v3.04) Longtable uses `\penalty-30000`. The `lineno` penalty range
 16 was shrunk to $-188000\dots -32000$. (/New v3.04) (New v4.00) New values
 17 are listed below (11111f.). (/New v4.00)

```
4 \newcount\linenopenalty\linenopenalty=-100000
```

18 (UL) Hm. It is never needed below that this is a counter.
 19 `\def\linenopenalty{-100000\relax}` would do. (I guess this consumes
 20 more memory, but it is more important to save counters than to save mem-
 21 ory.) I was frightened by `-\linenopenalty` below, but indeed \TeX interprets
 22 the string `--100000` as 100000 . Has any user or extension package writer ever
 23 called `\linenopenalty=xxx`, or could I really change this?—The counter is
 24 somewhat faster than the macro. Together with the compatibility question
 25 this seems to support keeping the counter. (???) (/UL)

```
5 \mathchardef\linenopenaltypar=32000
```

26 So let's make a hook to `\output`, the direct way. The \LaTeX macro
 27 `\@reinserts` puts the footnotes back on the page.

28 (New v3.01) `\@reinserts` badly screws up split footnotes. The bottom
 29 part is still on the recent contributions list, and the top part will be put back
 30 there after the bottom part. Thus, since `lineno.sty` does not play well with
 31 `\inserts` anyway, we can safely experiment with `\holdinginserts`, without
 32 making things much worse.

1 Or that's what I thought, but: Just activating `\holdinginserts` while
 2 doing the `\par` will not do the trick: The `\output` routine may be called
 3 for a real page break before all line numbers are done, and how can we get
 4 control over `\holdinginserts` at that point?

5 Let's try this: When the `\output` routine is run with `\holdinginserts=3`
 6 for a real page break, then we reset `\holdinginserts` and restart `\output`.

7 Then, again, how do we keep the remaining `\inserts` while doing further
 8 line numbers?

9 If we find `\holdinginserts=-3` we activate it again after doing `\output`.
 10 (/New v3.01)

11 (New v3.02) To work with `multicol.sty`, the original output routine is
 12 now called indirectly, instead of being replaced. When `multicol.sty` changes
 13 `\output`, it is a toks register, not the real thing. (/New v3.02)

14 (New v4.00) Two further complications are added.

15 (i) Problems with footnotes formerly resulted from L^AT_EX's `\@reinserts`
 16 in `\@specialoutput` which Stephan's `\linelabel` called via the
 17 `\marginpar` mechanism.

18 (ii) L^AT_EX commands using `\vadjust` formerly didn't work as one would
 19 have hoped. The problem is as follows: Printing the line num-
 20 ber results from a box that the output routine inserts at the
 21 place of the `\interlinepenalty`. `\vadjust` items appear *above* the
 22 `\interlinepenalty` (T_EXbook p. 105). So `\pagebreak`, e.g., for-
 23 merly sent the line number to the next page, while the penalty from
 24 `\nopagebreak` could not tie the following line, since it was screened
 25 off by the line number box.—Our trick is putting the `\vadjust` items
 26 into a list macro from which the output routine transfers them into the
 27 vertical list, below the line number box.

28 In this case (ii), like in case (i), footnotes would suffer if `\holdinginserts`
 29 were non-positive. Indeed, in both cases (i) and (ii) we tackle the foot-
 30 note problem by extending that part of Stephan's output routine that
 31 is active when `\holdinginserts` is positive. This extension writes the
 32 line number `\newlabel` to the `.aux` file (which was formerly done under
 33 `\holdinginserts = -3`) and handles the `\vadjust` items.—To trigger
 34 `\output` and its `\linelabel` or, resp., `\vadjust` part, the list of signal penal-
 35 ties started immediately before is increased here (first for `\linelabel`, second
 36 for postponed `\vadjust` items):

6 `\mathchardef\@Mllbcodepen=11111`

7 `\mathchardef\@Mppvacodepen=11112`

1 (/New v4.00) (New v4.2) David Kastrup urges to use a private name instead
 2 of `\the\output` (LaTeX-L-list). Otherwise an `\output` routine loaded later
 3 and using `\newtoks\output` again may get lost entirely. So we change use of
 4 `\@LN@output`, using it for the former purpose. Reference to what appeared
 5 with the name of `\output` here lasts for a few lines and then is given away.

```
8 \let\@tempa\output
9 \newtoks\output
10 \let\@LN@output\output
11 \output=\expandafter{\the\@tempa}
```

6 Now we add two cases to Stephan’s output routine. (New v4.00)

```
12 \@tempa={%
```

7 (/New 4.2)

```
13         \LineNoTest
14         \if@tempswa
```

8 (New v4.00) We insert recognition of waiting `\linelabel` items—

```
15         \ifnum\outputpenalty=-\@Mllbcodepen
16         \WriteLineNo
```

9 —and of waiting `\vadjust` items:

```
17         \else
18         \ifnum\outputpenalty=-\@Mppvacodepen
19         \PassVadjustList
20         \else
```

10 (/New v4.00) (New v4.2) Outsource “Standard” output—which occurs so
 11 rarely—to subsection 2.3:

```
21         \LineNoLaTeXOutput
```

12 (/New v4.2) (New v4.00) Two new `\fis` for the `\linelabel` and `\vadjust`
 13 tests—

```
22         \fi
23         \fi
```

14 —and the remaining is Stephan’s code again: (/New v4.00)

```
24         \else
25         \MakeLineNo
26         \fi
27         }
```

15 (New v4.00) Our new macros `\WriteLineNo` and `\PassVadjustList` will be
 16 dealt with in sections 4 and 8.1. (/New v4.00)

1 2.2 \LineNoTest

2 The float mechanism inserts `\interlinepenalty`s during `\output`. So care-
 3 fully reset it before going on. Else we get doubled line numbers on every
 4 float placed in horizontal mode, e.g, from `\linelabel`.

5 Sorry, neither a `\linelabel` nor a `\marginpar` should insert a penalty,
 6 else the following linenummer could go to the next page. Nor should any
 7 other float. So let us suppress the `\interlinepenalty` altogether with the
 8 `\@nbreak` switch.

9 Since (ltspace.dtx, v1.2p)[1996/07/26], the `\@nbreaktrue` does it's job
 10 globally. We need to do it locally here.

```

28 \def\LineNoTest{%
29   \let\@par\@@par
30   \ifnum\interlinepenalty<-\linenopenaltypar
31     \advance\interlinepenalty-\linenopenalty
32     \@LN@nbreaktrue
33     \fi
34   \@tempwattrue
35   \ifnum\outputpenalty>-\linenopenaltypar\else
36     \ifnum\outputpenalty>-188000\relax
37       \@tempwafalse
38       \fi
39     \fi
40 }
41
42 \def\@LN@nbreaktrue{\let@if@nbreak@iftrue} % renamed v4.33

```

11 (UL) I thought here were another case of the save stack problem ex-
 12 plained in `TEXbook`, p. 301, namely through both local and global chang-
 13 ing `\if@nbreak`. However, `\@LN@nbreak` is called during `\@LN@output`
 14 only, while `\@nbreaktrue` is called by `LATEX`'s `\@startsection` only.
 15 The latter never happens during `\@LN@output`. So there is no local
 16 value of `\if@nbreak` on save stack when `\@nbreaktrue` acts, since
 17 `\the\@LN@output` (where `\@LN@output` is a new name for the original
 18 `\output`) is executed within a group (`TEXbook` p. 21). (/UL)

19 2.3 Other output routines (v4.2)

20 I had thought of dealing with bad interference of footnotes (and
 21 `\enlargethispage`) with (real) `\marginpars` and floats *here*. Yet this is
 22 done in

[http://\[CTAN\]/macros/latex/contrib/tamefloats/tameflts.sty](http://[CTAN]/macros/latex/contrib/tamefloats/tameflts.sty)

1 now, and I prefer striving for compatibility with the latter. (See there for ex-
 2 panding on the problem.) This requires returning the special absolute value
 3 of `\holdinginserts` that `lineno.sty` finds at the end of a newly type-
 4 set paragraph—now done in subsection 3.1 (`\linenumberpar`). The former
 5 `\LineNoHoldInsertsTest` has been filled into here. Note: when the follow-
 6 ing code is invoked, we have `\if@tempswa = \iftrue`. WARNING: I am
 7 still not sure whether the present code is good for cooperating with other
 8 packages that use `\holdinginserts`.

```

43 \def\LineNoLaTeXOutput{%
44   \ifnum \holdinginserts=\thr@@   % v4.33 without \@tempswafalse
45     \global\holdinginserts=\thr@@
46     \unvbox\@cclv
47     \ifnum \outputpenalty=\@M \else \penalty\outputpenalty \fi
48   \else
49     \if@twocolumn \let\@makecol\@LN@makecol \fi
50     \the\@LN@output % finally following David Kastrup's advice.
51     \ifnum \holdinginserts=-\thr@@
52       \global\holdinginserts=\thr@@ \fi
53   \fi
54 }

```

9 *More on dealing with output routines from other packages:* Since
 10 `lineno.sty`'s output routine is called at least once for each output line,
 11 I think it should be in \TeX 's original `\output`, while output routines deal-
 12 ing with building pages and with floats etc. should be filled into registers
 13 addressed by `\output` after `\newtoks\output`. Therefore

- 14 1. `tameflts.sty` should be loaded *after* `lineno.sty`;
- 15 2. if a class changes `\output` (APS journal class `revtex4`, e.g.),
 16 `lineno.sty` should be loaded by `\RequirePackage` [here pre-
 17 sumably following some options in brackets]{`lineno`} *preceding*
 18 `\documentclass`.
- 19 3. If you actually maintain such a class, please consider loading
 20 `lineno.sty` on some draft option. The bunch of `lineno`'s package op-
 21 tions may be a problem, but perhaps the purpose of your class is offering
 22 only very few of `lineno`'s options anyway, maybe just one.

23 The latter may also be needed with classes that don't follow David Kastrup's
 24 rule on changing `\output`.

1 2.4 \MakeLineNo: Actually attach line number

2 We have to return all the page to the current page, and add a box with the
 3 line number, without adding breakpoints, glue or space. The depth of our
 4 line number should be equal to the previous depth of the page, in case the
 5 page breaks here, and the box has to be moved up by that depth.

6 The `\interlinepenalty` comes after the `\vadjust` from a `\linelabel`,
 7 so we increment the line number *after* printing it. The macro
 8 `\makeLineNumber` produces the text of the line number, see section 5.

9 (UL) I needed a while to understand the sentence on incrementing. Cor-
 10 rectly: writing the `\newlabel` to the `.aux` file is triggered by the signal
 11 penalty that `\end@float` inserts via `\vadjust`. However, this could be
 12 changed by our new `\PostponeVadjust`. After `\c@linenumber` has been in-
 13 troduced as a L^AT_EX counter, it might be preferable that it behaved like stan-
 14 dard L^AT_EX counters which are incremented shortly before printing. But this
 15 may be of little practical relevance in this case, as `\c@linenumber` is driven in
 16 a very non-standard way.—However still, this behaviour of `\c@linenumber`
 17 generates a problem with our `edtable.sty`. (/UL).

18 Finally we put in the natural `\interlinepenalty`, except after the last
 19 line.

20 (New v3.10) Frank Mittelbach points out that `box255` may be less deep
 21 than the last box inside, so he proposes to measure the page depth with
 22 `\boxmaxdepth=\maxdimen`. (/New v3.10)

23 (UL, New v4.00) We also resume the matter of `\vadjust` items that was
 24 started in section 2.1.

25 T_EX puts only nonzero interline penalties into the vertical list (T_EXbook
 26 p. 105), while `lineno.sty` formerly replaced the signal interline penalty by
 27 something closing with an explicit penalty of the value that the interline
 28 penalty would have without `lineno.sty`. This is usually 0. Now, ex-
 29 plicit vertical penalties can be very nasty with respect to `\nopagebreak`,
 30 e.g., a low (even positive) `\widowpenalty` may force a widow where you
 31 explicitly tried to forbid it by `\nopagebreak` (see explanation soon below).
 32 The `\nopagebreak` we create here would never work if all those zero penal-
 33 ties were present.—On the other hand, we cannot just omit Stephan's zero
 34 penalties, because T_EX puts a penalty of 10000 after what `lineno.sty` in-
 35 serts (T_EXbook p. 125). This penalty must be overridden to allow page
 36 breaks between ordinary lines. To revive `\nopagebreak`, we therefore re-
 37 place those zero (or low) penalties by penalties that the user demanded by
 38 `\nopagebreak`.—This mechanism is not perfect and does not exactly restore
 39 the original L^AT_EX working of `\pagebreak` and `\nopagebreak`. Viz., if there
 40 are several vertical penalties after a line which were produced by closely

1 sitting `\[no]pagebreaks`, without `lineno.sty` the lowest penalty would be
 2 effective (cf. `TeXbook` exercise 14.10). Our mechanism, by contrast, chooses
 3 the *last* user-set penalty of the line as the effective one. It would not be very
 4 difficult to come more close to the original mechanism, but until someone
 5 urges us we will cling to the present simple way. You may consider an ad-
 6 vantage of the difference between our mechanism and the original one that
 7 the user here can actually override low penalties by `\nopagebreak`, which
 8 may be what a lay `LATEX` user would expect. (/UL, /New v4.00)

```
55 \def\MakeLineNo{%
56   \@LN@maybe@normalLineNumber           % v4.31
57   \boxmaxdepth\maxdimen\setbox\z@\vbox{\unvbox\@cclv}%
58   \@tempdima\dp\z@ \unvbox\z@
59   \sbox\@tempboxa{\hb@xt@\z@{\makeLineNumber}}%
```

9 (New v4.00) Previously,

```
10 %   \stepcounter{linenumber}%
```

11 followed. (Of course, there was no comment mark; I put it there to make
 12 reading the actual code easy.)

13 (New v4.22: improved) Why not just

```
\global\advance\c@linenumber\@ne?
```

14 `\stepcounter` additionally resets “subordinate” counters, but which could
 15 these (usefully) be? Again, may be column counters with `edtable.sty`!

16 But then, our `edtable.sty` and its `longtable` option should use it as
 17 well. So use a shorthand supporting uniformity. You can even use it as
 18 a hook for choosing `\global\advance\c@linenumber\@ne` instead of our
 19 choice. (/New v4.22)

```
60 \stepLineNumber
```

20 (New v4.4) Now

```
61 \ht\@tempboxa\z@ \@LN@depthbox
```

21 appends the box containing the line number without changing `\prevdepth`—
 22 see end of section. Now is the time for inserting the ... (/New v4.4) `\vadjust`
 23 items. We cannot do this much later, because their right place is above the
 24 artificial interline penalty which Stephan’s code will soon insert (cf. `TeXbook`
 25 p. 105). The next command is just `\relax` if no `\vadjust` items have been
 26 accumulated for the current line. Otherwise it is a list macro inserting the
 27 `\vadjust` items and finally resetting itself. (This is made in section 8.1
 28 below.) If the final item is a penalty, it is stored so it can compete with
 29 other things about page breaking.

```
62 \LN@do@vadjusts
63 \count@\lastpenalty
```

1 At this place,

```
2 % \ifnum\outputpenalty=-\linenopenaltypar\else
```

3 originally followed. We need something *before* the `\else`:

```
64 \ifnum\outputpenalty=-\linenopenaltypar
65 \ifnum\count@=\z@ \else
```

4 So final `\pagebreak[0]` or `\nopagebreak[0]` has no effect—but this will
5 make a difference after headings only, where nobody should place such a
6 thing anyway.

```
66 \xdef\LN@parpgbrk{%
67 \penalty\the\count@
68 \global\let\noexpand\LN@parpgbrk
69 \noexpand\LN@screenoff@pen}% v4.4
```

7 That penalty will replace former `\kern\z@` in `\linenumberpar`, see subsec-
8 tion 3.1.—A few days earlier, I tried to send just a penalty value. However,
9 the `\kern\z@` in `\linenumberpar` is crucial, as I then found out. See below.—
10 The final penalty is repeated, but this does no harm. (It would not be very
11 difficult to avoid the repeating, but it may even be less efficient.) It may be
12 repeated due to the previous `\xdef`, but it may be repeated as well below in
13 the present macro where artificial interline penalty is to be overridden.

```
70 \fi
71 \else
```

14 (/New v4.00)

```
72 \@tempcnta\outputpenalty
73 \advance\@tempcnta -\linenopenalty
```

15 (New v4.00)

```
16 % \penalty\@tempcnta
```

17 followed previously. To give `\nopagebreak` a chance, we do

```
74 \penalty \ifnum\count@<\@tempcnta \@tempcnta \else \count@ \fi
```

1 instead.—In `linenox0.sty`, the `\else` thing once was omitted. Sergei
 2 Mariev’s complaint (thanks!) showed that it is vital (see comment before
 3 `\MakeLineNo`). The remaining `\fi` from previous package version closes the
 4 `\ifnum\outputpenalty...` (/New v4.00)

```
75 \fi
76 }
```

5 (New v4.00)

```
77 \newcommand\stepLineNumber{\stepcounter{linenumber}}
```

6 For reason, see use above. (/New v4.00)

7 (New v4.4) The depth preserving trick is drawn here from `\MakeLineNo`
 8 because it will be used again in section 3.1.

```
78 \def\LN@depthbox{%
79 \dp\@tempboxa=\@tempdima
80 \nointerlineskip \kern-\@tempdima \box\@tempboxa}
```

9 (/New v4.4)

10 3 Control line numbering

11 3.1 Inserting `\output` calls

12 The line numbering is controlled via `\par`. L^AT_EX saved the T_EX-primitive
 13 `\par` in `\@@par`. We push it one level further out, and redefine `\@@par` to
 14 insert the `\interlinepenalty` needed to trigger the line numbering. And
 15 we need to allow pagebreaks after a paragraph.

16 New (2.05beta): the `prevgraf` test. A paragraph that ends
 17 with a displayed equation, a `\noindent\par` or `wrapfig.sty` produce
 18 empty paragraphs. These should not get a spurious line number via
 19 `\linenopenaltypar`.

```
81 \let\@@@par\@@par
82 \newcount\linenoprevgraf
```

20 (UL) And needs `\linenoprevgraf` to be a counter? Perhaps there may
 21 be a paragraph having thousands of lines, so `\mathchardef` doesn’t suffice
 22 (really??). A macro ending on `\relax` might suffice, but would be somewhat
 23 slow. I think I will use `\mathchardef` next time. Or has any user used
 24 `\linenoprevgraf`? (/UL)

```

83 \def\linenumberpar{%
84   \ifvmode \@@@par \else
85     \ifinner \@@@par \else
86       \xdef\@LN@outer@holdins{\the\holdinginserts}% v4.2
87       \advance \interlinepenalty \linenopenalty
88       \linenoprevgraf \prevgraf
89       \global \holdinginserts \thr@@
90       \@@@par
91       \ifnum\prevgraf>\linenoprevgraf
92         \penalty-\linenopenaltypar
93       \fi

```

1 (New v4.00)

```

2 %           \kern\z@

```

3 was here previously. What for? According to T_EXbook p. 125, Stephan’s
4 interline penalty is changed into 10000. At the end of a paragraph, the
5 `\parskip` would follow that penalty of 10000, so there could be a page break
6 neither at the `\parskip` nor at the `\baselineskip` (T_EXbook p. 110)—so
7 there could never be a page break between two paragraphs. So something
8 must screen off the 10000 penalty. Indeed, the `\kern` is a place to break.
9 (Stephan once knew this: see ‘allow pagebreaks’ above.)

10 Formerly, I tried to replace `\kern\z@` by

```

11 %           \penalty\@LN@parpgpen\relax

```

12 —but this allows a page break after heading. So:

```

94           \@LN@parpgbrk

```

13 These and similar changes were formerly done by `linenox1.sty`. (/New
14 v4.00)

15 (New v4.4) A `\belowdisplayskip` may precede the previous when the
16 paragraph ends on a display-math; or there may be a `\topsep` from a list, etc.
17 `\addvspace` couldn’t take account for it with `\kern\z@` here. v4.32 therefore
18 moved the space down – with at least two bad consequences. Moreover, David
19 Josef Dev observes that `\kern\z@` may inappropriately yield column depth
20 0pt. For these reasons, we introduce `\@LN@screenoff@open` below. (/New
21 v4.4)

```

95           \global\holdinginserts\@LN@outer@holdins % v4.2
96           \advance\interlinepenalty -\linenopenalty
97   \fi      % from \ifinner ... \else
98 \fi}      % from \ifvmode ... \else

```

1 (New v4.00, v4.4) Initialize `\LN@parpgbrk`, accounting for earlier space
 2 and for appropriate `columndepth`. We use former `\MakeLineNo`'s depth-
 3 preverving trick `\LN@depthbox` again:

```

99 \def\LN@screenoff@open{%
100   \ifdim\lastskip=\z@
101     \tempdima\prevdepth \setbox\@tempboxa\null
102     \LN@depthbox \fi}
103
104 \global\let\LN@parpgbrk\LN@screenoff@open

```

4 (/New v4.4, v4.00)

5 3.2 Turning on/off

6 The basic commands to enable and disable line numbers. `\@par` and `\par`
 7 are only touched, when they are `\let` to `\@@@par/\linenumberpar`. The line
 8 number may be reset to 1 with the star-form, or set by an optional argument
 9 [*number*].

10 (New v4.00) We add `\ifLineNumbers` etc. since a number of our new ad-
 11 justments need to know whether `linenumbering` is active. This just provides a
 12 kind of shorthand for `\ifx\@par\linenumberpar`; moreover it is more sta-
 13 ble: who knows what may happen to `\@par`?—A caveat: `\ifLineNumbers`
 14 may be wrong. E.g., it may be `\iffalse` where it acts, while a `\linenumbers`
 15 a few lines below—in the same paragraph—brings about that the line where
 16 the `\ifLineNumbers` appears gets a marginal number. (New v4.3) Just
 17 noticed: Such tricks have been disallowed with v4.11, see subsections 4.2
 18 and 3.2.—Moreover, the switching between meanings of `\linelabel` for a
 19 possible error message as of v4.11 is removed. Speed is difficult to esteem
 20 and also depends on applications. Just use the most simple code you find.
 21 (/New v4.3)

```

105 \newif\ifLineNumbers \LineNumbersfalse

```

22 (/New v4.00)

```

106 \def\linenumbers{%
107   \LineNumberstrue % v4.00
108   \xdef\LN@outer@holdins{\the\holdinginserts}% v4.3

```

23 (New v4.3) The previous line is for `{\linenomath}` in a first numbered para-
 24 graph. (/New v4.3)

```

109   \let\@par\linenumberpar
110 %   \let\linelabel\LN@linelabel % v4.11, removed v4.3
111   \ifx\@par\@@par\let\@par\linenumberpar\fi
112   \ifx\par\@@par\let\par\linenumberpar\fi
113   \@LN@maybe@moduloresume      % v4.31
114   \@ifnextchar[{\resetlinenumber}%]
115     {\@ifstar{\resetlinenumber}{}}%
116   }
117
118 \def\nolinenumbers{%
119   \LineNumbersfalse             % v4.00
120   \let\@par\@@par
121 %   \let\linelabel\LN@LError    % v4.11, removed v4.3
122   \ifx\@par\linenumberpar\let\@par\@@par\fi
123   \ifx\par\linenumberpar\let\par\@@par\fi
124   }

```

1 (New v4.00) Moreover, it is useful to switch to `\nolinenumbers` in
 2 `\@arrayparboxrestore`. We postpone this to section 8.2 where we'll have
 3 an appending macro for doing this. (/New v4.00)

4 What happens with a display math? Since `\par` is not executed, when
 5 breaking the lines before a display, they will not get line numbers. Sorry,
 6 but I do not dare to change `\interlinepenalty` globally, nor do I want to
 7 redefine the display math environments here.

display math

8 See the subsection below, for a wrapper environment to make it work. But
 9 that requires to wrap each and every display in your \LaTeX source (see option
 10 `displaymath` in subsections 6.4 and 7.1 for some relief [UL]).

11 The next two commands are provided to turn on line numbering in
 12 a specific mode. Please note the difference: for pagewise numbering,
 13 `\linenumbers` comes first to inhibit it from seeing optional arguments, since
 14 re-/presetting the counter is useless.

```

125 \def\pagewiselinenumbers{\linenumbers\setpagewiselinenumbers}
126 \def\runninglinenumbers{\setrunninglinenumbers\linenumbers}

```

15 Finally, it is a \LaTeX style, so we provide for the use of environments, includ-
 16 ing the suppression of the following paragraph's indentation.

17 (UL) I am drawing the following private thoughts of Stephan's to publicity
 18 so that others may think about them—or to remind myself of them in an
 19 efficient way. (/UL)

```

1 % TO DO: add \par to \linenumbers, if called from an environment.
2 % To DO: add an \@endpe hack if \linenumbers are turned on
3 %       in horizontal mode. {\par\parskip\z@\noindent} or
4 %       something.

```

(UL) However, I rather think that `\linenumbers` and `\nolinenumbers` should execute a `\par` already. (Then the `\pars` in the following definitions should be removed.) (/UL)

```

127 \@namedef{linenumbers*}{\par\linenumbers*}
128 \@namedef{runninglinenumbers*}{\par\runninglinenumbers*}
129
130 \def\endlinenumbers{\par\@endpetrue}
131 \let\endrunninglinenumbers\endlinenumbers
132 \let\endpagewiselinenumbers\endlinenumbers
133 \expandafter\let\csname endlinenumbers*\endcsname\endlinenumbers
134 \expandafter\let\csname endrunninglinenumbers*\endcsname\endlinenumbers
135 \let\endnolinenumbers\endlinenumbers

```

3.3 Display math

Now we tackle the problem to get display math working. There are different options.

1. Precede every display math with a `\par`. Not too good.
2. Change `\interlinepenalty` and associates globally. Unstable.
3. Wrap each display math with a `{\linenomath}` environment.

We'll go for option 3. See if it works:

$$\textit{display math} \tag{1}$$

The star form `{\linenomath*}` should also number the lines of the display itself,

$$\begin{array}{l} \textit{multi} \qquad \qquad \textit{line} \end{array} \tag{2}$$

$$\begin{array}{l} \textit{display} \qquad \qquad \textit{math} \end{array} \tag{3}$$

$$\begin{array}{l} \textit{with} \\ \textit{array} \end{array} \tag{4}$$

including multiline displays.

First, here are two macros to turn on linenumbers on paragraphs preceding displays, with numbering the lines of the display itself, or without.

- 1 The `\ifx..` tests if line numbering is turned on. It does not harm to add
 2 these wrappers in sections that are not numbered. Nor does it harm to wrap
 3 a display twice, e.g, in case you have some `{equation}`s wrapped explicitly,
 4 and later you redefine `\equation` to do it automatically.
 5 (New v4.3) To avoid the spurious line number above a display in vmode,
 6 I insert `\ifhmode.` (/New v4.3)

```

136 \newcommand\linenomathNonnumbers{%
137   \ifLineNumbers
138     \ifnum\interlinepenalty>-\linenopenaltypar
139       \global\holdinginserts\thr@@
140       \advance\interlinepenalty \linenopenalty
141       \ifhmode % v4.3
142         \advance\predisplaypenalty \linenopenalty
143       \fi
144     \fi
145   \fi
146   \ignorespaces
147 }
148
149 \newcommand\linenomathWithnumbers{%
150   \ifLineNumbers
151     \ifnum\interlinepenalty>-\linenopenaltypar
152       \global\holdinginserts\thr@@
153       \advance\interlinepenalty \linenopenalty
154       \ifhmode % v4.3
155         \advance\predisplaypenalty \linenopenalty
156       \fi
157       \advance\postdisplaypenalty \linenopenalty
158       \advance\interdisplaylinepenalty \linenopenalty
159     \fi
160   \fi
161   \ignorespaces
162 }

```

- 7 The `{linenomath}` environment has two forms, with and without a star. The
 8 following two macros define the environment, where the starred/non-starred
 9 form does/doesn't number the lines of the display or vice versa.

```

163 \newcommand\linenumberdisplaymath{%
164   \def\linenomath{\linenomathWithnumbers}%
165   \@namedef{linenomath*}{\linenomathNonnumbers}%
166 }
167
168 \newcommand\nolinenumberdisplaymath{%
169   \def\linenomath{\linenomathNonnumbers}%
170   \@namedef{linenomath*}{\linenomathWithnumbers}%

```

```

171 }
172
173 \def\endlinenomath{%
174   \ifLineNumbers % v4.3
175   \global\holdinginserts\@LN@outer@holdins % v4.21
176   \fi
177   \global % v4.21 support for LaTeX2e earlier than 1996/07/26.
178   \@ignoretrue
179 }
180 \expandafter\let\csname endlinenomath*\endcsname\endlinenomath

```

- 1 The default is not to number the lines of a display. But the package option
- 2 `mathlines` may be used to switch that behavior.

```

181 \nolinenumberdisplaymath

```

3 4 Line number references

4 4.1 Internals

- 5 The only way to get a label to a line number in a paragraph is to ask the
- 6 output routine to mark it.

- 7 (New v4.00) The following two paragraphs don't hold any longer, see
- 8 below. (/New v4.00)

```

9 % We use the marginpar mechanism to hook to ~\output~ for a
10 % second time. Marginpars are floats with number $-1$, we
11 % fake marginpars with No $-2$. Originally, every negative
12 % numbered float was considered to be a marginpar.
13 %
14 % The float box number ~\@currbox~ is used to transfer the
15 % label name in a macro called ~\@LNL@~<box-number>.

```

- 16 A `\newlabel` is written to the aux-file. The reference is to `\theLineNumber`,
- 17 *not* `\thelinenumber`. This allows to hook in, as done below for pagewise
- 18 line numbering.

- 19 (New v3.03) The `\@LN@ExtraLabelItems` are added for a hook to keep
- 20 packages like `{hyperref}` happy. (/New v3.03)

- 21 (New v4.00) We fire the `\marginpar` mechanism, so we leave L^AT_EX's
- 22 `\@addmarginpar` untouched.

```

23 % \let\@LN@addmarginpar\@addmarginpar
24 % \def\@addmarginpar{%
25 %   \ifnum\count\@currbox>-2\relax
26 %     \expandafter\@LN@addmarginpar

```

```

1 % \else
2 %   \@cons\@freelist\@currbox
3 %   \protected@write\@auxout{}\@%
4 %     \string\newlabel
5 %       {\csname @LNL@\the\@currbox\endcsname}%
6 %       {\theLineNumber}\thepage\@LN@ExtraLabelItems}%
7 % \fi}

```

8 OK, we keep Stephan's \@LN@ExtraLabelItems: (/New v4.00)

```
182 \let\@LN@ExtraLabelItems\@empty
```

9 (New v4.00) We imitate the \marginpar mechanism without using the
10 \@freelist boxes. \linelabel will indeed place a signal penalty
11 (\@Mllbcodepen, new), and it will put a label into some list macro
12 \@LN@labellist. A new part of the output routine will take the labels
13 from the list and will write \newlabels to the .aux file.

14 The following is a version of L^AT_EX's \@xnext.

```
183 \def\@LN@xnext#1\@lt#2\@#3#4{\def#3{#1}\gdef#4{#2}}
```

15 This takes an item #1 from a list #4 into #3; to be used as
16 \expandafter\@LN@xnext#4\@#3#4. Our lists use \@lt after each item
17 for separating. Indeed, there will be another list macro which can appear as
18 argument #4, this will be used for moving \vadjust items (section 8.1). The
19 list for \linelabels is the following:

```
184 \global\let\@LN@labellist\@empty
```

20 The next is the new part of the output routine writing the \newlabel to the
21 .aux file. Since it is no real page output, the page is put back to top of the
22 main vertical list.

```

185 \def\WriteLineNo{%
186   \unvbox\@cclv
187   \expandafter \@LN@xnext \@LN@labellist \@
188     \@LN@label \@LN@labellist
189   \protected@write\@auxout{}\string\newlabel{\@LN@label}%
190     {\theLineNumber}\thepage\@LN@ExtraLabelItems}%
191 }

```

23 (/New v4.00)

1 4.2 The `\linelabel` command

2 To refer to a place in line `\ref{<foo>}` at page `\pageref{<foo>}` you place a
3 `\linelabel{<foo>}` at that place.

4 (New v4.11)

5 % If you use this command outside a `~\linenumbers~`
6 % paragraph, you will get references to some bogus
7 % line numbers, sorry. But we don't disable the command,
8 % because only the `~\par~` at the end of a paragraph may
9 % decide whether to print line numbers on this paragraph
10 % or not. A `~\linelabel~` may legally appear earlier than
11 % `~\linenumbers~`.

See if it
works:
This
paragraph
starts on
page 22,
line 4.

12 This trick is better not allowed—see subsections 4.2 and 3.2. (/New v4.11)

13 `\linelabel`

14 %, via a fake float number $\$-2\$, %%$ new mechanism v4.00

15 puts a `\penalty` into a `\vadjust`, which triggers the pagebuilder after
16 putting the current line to the main vertical list. A `\write` is placed
17 on the main vertical list, which prints a reference to the current value of
18 `\thelinenumber` and `\thepage` at the time of the `\shipout`.

19 A `\linelabel` is allowed only in outer horizontal mode. In outer ver-
20 tical mode we start a paragraph, and ignore trailing spaces (by fooling
21 `\@esphack`).

22 (New v4.00) We aim at relaxing the previous condition. We insert a hook
23 `\@LN@mathhook` and a shorthand `\@LN@postlabel` to support the `mathrefs`
24 option which allows `\linelabel` in math mode.

25 The next paragraph is no longer valid.

26 % The argument of `~\linelabel~` is put into a macro with a
27 % name derived from the number of the allocated float box.
28 % Much of the rest is dummy float setup.

29 (/New v4.00)

30 (New v4.11)

31 % `\def\linelabel#1{%`

32 I forgot `\linenumbers` today, costed me hours or so.

192 `\def\@LN@LError{\PackageError{lineno}{%`
193 `\string\linelabel\space without \string\linenumbers}{%`
194 `Just see documentation. (New feature v4.11)}\@gobble}`

1 (New v4.3) Here some things have changed for v4.3. The previous #1
 2 has been replaced by `\@gobble`. Ensuing, the `\linelabel` error mes-
 3 sage is re-implemented. I find it difficult to compare efficiency of slight
 4 alternatives—so choose an easy one. Explicit switching in `\linenumbers`
 5 and `\nolinenumbers` is an additional command that may better be avoided.

```
195 \newcommand\linelabel{%
196   \ifLineNumbers \expandafter \@LN@linelabel
197   \else          \expandafter \@LN@LError   \fi}
198
199 \gdef\@LN@linelabel#1{%
```

6 `\gdef` for hyperref “symbolically”. (/New v4.11)

```
200 \ifx\protect\@typeset@protect
```

7 ← And a `\linelabel` should never be replicated in a mark or a TOC entry.
 8 (/New v4.3)

```
201 \ifvmode
202   \ifinner \else
203     \leavevmode \@bsphack \@savsk\p@
204   \fi
205 \else
206   \@bsphack
207 \fi
208 \ifhmode
209   \ifinner
210   \@parmoderr
211 \else
```

9 (New v4.00)

```
212   \@LN@postlabel{#1}%

10 %   \@floatpenalty -\@Mii
11 %   \@next\@currbox\@freelist
12 %   {\global\count\@currbox-2%
13 %   \expandafter\gdef\csname @LNL@the\@currbox\endcsname{#1}}%
14 %   {\@floatpenalty\z@ \@fltovf \def\@currbox{\@tempboxa}}%
15 %   \begingroup
16 %   \setbox\@currbox \color@vbox \vbox \bgroup \end@float
17 %   \endgroup
18 %   \@ignorefalse \@esphack
```

19 (/New v4.00)

213 `\@esphack`

1 (New v4.00) The `\@ignorefalse` was appropriate before because the
 2 `\@Esphack` in `\end@float` set `\@ignoretrue`. Cf. L^AT_EX's `\@xympar`. (/New
 3 v4.00)

214 `\fi`
 215 `\else`

4 (New v4.00)

216 `\@LN@mathhook{#1}%`

5 % `\@parmoderr`

6 Instead of complaining, you may just do your job. (/New v4.00)

217 `\fi`
 218 `\fi`
 219 `}`

7 (New v4.00) The shorthand just does what happened with `linenox0.sty`
 8 before `ednmath0.sty` (New v4.1: now `mathrefs` option) appeared, and the
 9 hook is initialized to serve the same purpose. So errors come just where
 10 Stephan had built them in, and this is just the L^AT_EX `\marginpar` behaviour.

220 `\def\@LN@postlabel#1{\g@addto@macro\@LN@labellist{#1\@t}%`
 221 `\vadjust{\penalty-\@Mllbcodepen}}`
 222 `\def\@LN@mathhook#1{\@parmoderr}`

11 (/New v4.00)

5 The appearance of the line numbers

13 5.1 Basic code

The line numbers are set as `\tiny\sffamily\arabic{linenumber}`, 10pt
 left of the text. With options to place it right of the text, or . . .

16 . . . here are the hooks:

```

223 \def\makeLineNumberLeft{%
224   \hss\linenumberfont\LineNumber\hskip\linenumbersep}
225
226 \def\makeLineNumberRight{%
227   \linenumberfont\hskip\linenumbersep\hskip\columnwidth
228   \hb@xt@\linenumberwidth{\hss\LineNumber}\hss}
229
230 \def\linenumberfont{\normalfont\tiny\sffamily}
231
232 \newdimen\linenumbersep
233 \newdimen\linenumberwidth
234
235 \linenumberwidth=10pt
236 \linenumbersep=10pt

```

- 1 Margin switching requires pagewise numbering mode, but choosing the left or right margin for the numbers always works.

```

237 \def\switchlinenumbers{\@ifstar
238   {\let\makeLineNumberOdd\makeLineNumberRight
239    \let\makeLineNumberEven\makeLineNumberLeft}%
240   {\let\makeLineNumberOdd\makeLineNumberLeft
241    \let\makeLineNumberEven\makeLineNumberRight}%
242   }
243
244 \def\setmakelinenumbers#1{\@ifstar
245   {\let\makeLineNumberRunning#1%
246    \let\makeLineNumberOdd#1%
247    \let\makeLineNumberEven#1}%
248   {\ifx\c@linenumber\c@runninglinenumber
249    \let\makeLineNumberRunning#1%
250    \else
251    \let\makeLineNumberOdd#1%
252    \let\makeLineNumberEven#1%
253    \fi}%
254   }
255
256 \def\leftlinenumbers{\setmakelinenumbers\makeLineNumberLeft}
257 \def\rightlinenumbers{\setmakelinenumbers\makeLineNumberRight}
258
259 \leftlinenumbers*

```

\LineNumber is a hook which is used for the modulo stuff. It is the command
4 to use for the line number, when you customize \makeLineNumber. Use
\thelinenumber to change the outfit of the digits.

We will implement two modes of operation:

- 7 • numbers running through (parts of) the text

- 1 • pagewise numbers starting over with one on top of each page.

Both modes have their own count register, but only one is allocated as a L^AT_EX counter, with the attached facilities serving both.

```
260 \newcounter{linenumber}
261 \newcount\c@pagewiselinenumber
262 \let\c@runninglinenumber\c@linenumber
```

- 4 Only the running mode counter may be reset, or preset, for individual paragraphs. The pagewise counter must give a unique anonymous number for each line.

7 (New v4.3) `\newcounter{linenumber}` was the only `\newcounter` in the whole package, and formerly I was near using `\newcount` instead. Yet `\newcounter` may be quite useful for `\includeonly`. It also supports resetting “subcounters”, but what could these be? Well, `edtable` might introduce a subcounter for columns. (Note that L^AT_EX’s setting commands would work with `\newcount\c@linenumber` already, apart from this. And perhaps sometimes `\refstepcounter{linenumber}` wouldn’t work—cf. my discussion of `\stepcounter` in subsection 2.4, similarly `\refstep...` would be quite useless. Even the usual redefinitions of `\thelinenumber` would work. It is nice, on the other hand, that `\thelinenumber` is predefined here. L^AT_EX’s initialization of the value perhaps just serves making clear L^AT_EX counters should always be changed globally.—Shortened and improved the discussion here.)

19 (/New v4.3)

(New v4.22) `\c@linenumber` usually is—globally—incremented by `\stepcounter` (at present), so resetting it locally would raise the save stack problem of T_EXbook p. 301, moreover it would be is useless, there is no hope of keeping the values local (but see subsection 7.2). So I insert `\global:`

22 (/New v4.22)

```
263 \newcommand*\resetlinenumber[1][\@ne]{%
264   \global                               % v4.22
265   \c@runninglinenumber#1\relax}
```

- 25 (New v4.00)

```
% \newcommand\resetlinenumber[1][1]{\c@runninglinenumber#1}
```

Added `\relax`, being quite sure that this does no harm and is quite important, as with `\setcounter` etc. I consider this a bug fix (although perhaps no user has ever had a problem with this). (/New v4.00)

(v4.22: I had made much fuss about resetting subordinate counters here—
31 removed, somewhat postponed.)

1 5.2 Running line numbers

Running mode is easy, `\LineNumber` and `\theLineNumber` produce `\thelinenumber`, which defaults to `\arabic{linenumber}`, using the `\c@runninglinenumber` counter. This is the default mode of operation.

```
266 \def\makeRunningLineNumber{\makeLineNumberRunning}
267
268 \def\setrunninglinenumbers{%
269   \def\theLineNumber{\thelinenumber}%
270   \let\c@linenumber\c@runninglinenumber
271   \let\makeLineNumber\makeRunningLineNumber
272 }
273
274 \setrunninglinenumbers\resetlinenumber
```

5.3 Pagewise line numbers

Difficult, if you think about it. The number has to be printed when there is no means to know on which page it will end up, except through the aux-file. My solution is really expensive, but quite robust.

With version v2.00 the hashsize requirements are reduced, because we do not need one controlsequence for each line any more. But this costs some computation time to find out on which page we are.

`\makeLineNumber` gets a hook to log the line and page number to the aux-file. Another hook tries to find out what the page offset is, and subtracts it from the counter `\c@linenumber`. Additionally, the switch `\ifoddNumberedPage` is set true for odd numbered pages, false otherwise.

```
275 \def\setpagewiselinenumbers{%
276   \let\theLineNumber\thePagewiseLineNumber
277   \let\c@linenumber\c@pagewiselinenumber
278   \let\makeLineNumber\makePagewiseLineNumber
279 }
280
281 \def\makePagewiseLineNumber{\logtheLineNumber\getLineNumber
282   \ifoddNumberedPage
283     \makeLineNumberOdd
284   \else
285     \makeLineNumberEven
286   \fi
287 }
```

16 Each numbered line gives a line to the aux file

```
\@LN{<line>}{<page>}
```

1 very similar to the `\newlabel` business, except that we need an arabic representation of the page number, not what there might else be in `\thepage`.

```
288 \def\logtheLineNumber{\protected@write\auxout-}{%
```

(New v4.00) (UL) As Daniel Doherty observed, the earlier line

```
4 % \string\@LN{\the\c@linenumber}\noexpand\the\c@page}}
```

here may lead into an infinite loop when the user resets the page number (think of `\pagenumbering`, e.g.). Stephan and I briefly discussed the matter and decided to introduce a “physical”-page counter to which `\logtheLineNumber` refers. It was Stephan’s idea to use `\c@page` for reliably augmenting the “physical”-page counter. However, this relies on the output routine once doing `\stepcounter{page}`. Before Stephan’s suggestion, I had thought of appending the stepping to L^AT_EX’s `\@outputpage`.—So the macro definition ends as follows.

```
289 \string\@LN{\the\c@linenumber}{%
```

13 (New v4.2) The ‘truepage’ counter must start with `\c@` so it works with `\include`, and the `\@addtoreset` below is needed for the same purpose.

```
290 \noexpand\the\c@LN@truepage}}}
```

```
291
```

```
292 \newcount\c@LN@truepage
```

```
293 \g@addto@macro\c@page{\global\advance\c@LN@truepage\@ne}
```

```
294 \@addtoreset{LN@truepage}{\ckpt}
```

(/New v4.2) I had thought of offering more features of a L^AT_EX counter. However, the user should better *not* have access to this counter. `\c@page` should suffice as a pagewise master counter.—To be sure, along the present lines the user *can* manipulate `\c@LN@truepage` by `\stepcounter{page}`. E.g., she might do this in order to manually insert a photograph. Well, seems not to harm.

The above usage of `\g@addto@macro` and `\c@page` may be not as stable as Stephan intended. His proposal used `\xdef` directly. But he used `\c@page` as well, and who knows . . . And as to `\g@addto@macro`, I have introduced it for list macros anyway. (/UL) (/New v4.00)

25 From the aux-file we get one macro `\LN@P⟨page⟩` for each page with line numbers on it. This macro calls four other macros with one argument each. These macros are dynamically defined to do tests and actions, to find out on which page the current line number is located.

28 We need sort of a pointer to the first page with line numbers, initialized to point to nothing:

```

295 \def\LastNumberedPage{first}
296 \def\LN@Pfirst{\nextLN\relax}

```

1 The four dynamic macros are initialized to reproduce themselves in an `\xdef`

```

297 \let\lastLN\relax % compare to last line on this page
298 \let\firstLN\relax % compare to first line on this page
299 \let\pageLN\relax % get the page number, compute the linenumber
300 \let\nextLN\relax % move to the next page

```

During the end-document run through the aux-files, we disable `\@LN`. I may put in a check here later, to give a rerun recommendation.

```

301 \AtEndDocument{\let\@LN\@gobbletwo}

```

4 Now, this is the tricky part. First of all, the whole definition of `\@LN` is grouped, to avoid accumulation on the save stack. Somehow `\csname<cs>\endcsname` pushes an entry, which stays after an `\xdef` to that

7 `<cs>`.

If `\LN@P<page>` is undefined, initialize it with the current page and line number, with the *pointer-to-the-next-page* pointing to nothing. And the

10 macro for the previous page will be redefined to point to the current one.

If the macro for the current page already exists, just redefine the *last-line-number* entry.

13 Finally, save the current page number, to get the pointer to the following page later.

```

302 \def\@LN#1#2{\expandafter\@@LN
303             \csname LN@P#2C\@LN@column\expandafter\endcsname
304             \csname LN@P#2\endcsname
305             {#1}{#2}}
306
307 \def\@@LN#1#2#3#4{\ifx#1\relax
308   \ifx#2\relax\gdef#2{#3}\fi
309   \expandafter\@@LN\csname LN@P\LastNumberedPage\endcsname#1%
310   \xdef#1{\lastLN{#3}\firstLN{#3}%
311           \pageLN{#4}{\@LN@column}{#2}\nextLN\relax}%
312 \else
313   \def\lastLN##1{\noexpand\lastLN{#3}}%
314   \xdef#1{#1}%
315 \fi
316 \xdef\LastNumberedPage{#4C\@LN@column}}

```

The previous page macro gets its pointer to the current one, replacing the

16 `\relax` with the cs-token `\LN@P<page>`.

```

317 \def\@@LN#1#2{{\def\nextLN##1{\noexpand\nextLN\noexpand#2}%
318         \xdef#1{#1}}}

```

- 1 Now, to print a line number, we need to find the page, where it resides. This will most probably be the page where the last one came from, or maybe the next page. However, it can be a completely different one. We maintain a
- 4 cache, which is `\let` to the last page's macro. But for now it is initialized to expand `\LN@first`, where the pointer to the first numbered page has been stored in.

```

319 \def\NumberedPageCache{\LN@Pfirst}

```

- 7 To find out on which page the current `\c@linenumber` is, we define the four dynamic macros to do something useful and execute the current cache macro. `\lastLN` is run first, testing if the line number in question may be on a later
- 10 page. If so, disable `\firstLN`, and go on to the next page via `\nextLN`.

```

320 \def\testLastNumberedPage#1{\ifnum#1<\c@linenumber
321     \let\firstLN@gobble
322     \fi}

```

- Else, if `\firstLN` finds out that we need an earlier page, we start over from the beginning. Else, `\nextLN` will be disabled, and `\pageLN` will run
- 13 `\gotNumberedPage` with four arguments: the first line number on this column, the page number, the column number, and the first line on the page.

```

323 \def\testFirstNumberedPage#1{\ifnum#1>\c@linenumber
324     \def\nextLN##1{\testNextNumberedPage\LN@Pfirst}%
325     \else
326         \let\nextLN@gobble
327         \def\pageLN{\gotNumberedPage{#1}}%
328     \fi}

```

- We start with `\pageLN` disabled and `\nextLN` defined to continue the search
- 16 with the next page.

```

329 \long\def \@gobblethree #1#2#3{}
330
331 \def\testNumberedPage{%
332     \let\lastLN\testLastNumberedPage
333     \let\firstLN\testFirstNumberedPage
334     \let\pageLN@gobblethree
335     \let\nextLN\testNextNumberedPage
336     \NumberedPageCache
337 }

```

1 When we switch to another page, we first have to make sure that it is there.
 If we are done with the last page, we probably need to run T_EX again, but for
 the rest of this run, the cache macro will just return four zeros. This saves a
 4 lot of time, for example if you have half of an aux-file from an aborted run, in
 the next run the whole page-list would be searched in vain again and again
 for the second half of the document.

7 If there is another page, we iterate the search.

```
338 \def\testNextNumberedPage#1{\ifx#1\relax
339   \global\def\NumberedPageCache{\gotNumberedPage0000}%
340   \PackageWarningNoLine{lineno}%
341     {Linenummer reference failed,
342     \MessageBreak rerun to get it right}%
343   \else
344     \global\let\NumberedPageCache#1%
345   \fi
346   \testNumberedPage
347 }
```

To separate the official hooks from the internals there is this equivalence, to
 hook in later for whatever purpose:

Let's see if
 it finds the
 label on
 page 22,
 line 4, and
 back here
 on page
 31, line 8.

```
348 \let\getLineNumber\testNumberedPage
```

10 So, now we got the page where the number is on. We establish if we are on
 an odd or even page, and calculate the final line number to be printed.

```
349 \newif\ifoddNumberedPage
350 \newif\ifcolumnwiselinenumbers
351 \columnwiselinenumbersfalse
352
353 \def\gotNumberedPage#1#2#3#4{\oddNumberedPagefalse
354   \ifodd \if@twocolumn #3\else #2\fi\relax\oddNumberedPagetrue\fi
355   \advance\c@linenumber\@ne
356   \ifcolumnwiselinenumbers
357     \subtractlinenumberoffset{#1}%
358   \else
359     \subtractlinenumberoffset{#4}%
360   \fi
361 }
```

13 You might want to run the pagewise mode with running line numbers, or
 you might not. It's your choice:

```
362 \def\runningpagewiselinenumbers{%
363   \let\subtractlinenumberoffset\gobble
364 }
```

```

365
366 \def\realpagewiselinenumbers{%
367   \def\subtractlinenumberoffset##1{\advance\c@linenumber-##1\relax}%
368   }
369
370 \realpagewiselinenumbers

```

- 1 For line number references, we need a protected call to the whole procedure, with the requested line number stored in the `\c@linenumber` counter. This is what gets printed to the aux-file to make a label:

```

371 \def\thePagewiseLineNumber{\protect
372   \getpagewiselinenumber{\the\c@linenumber}}%

```

- 4 And here is what happens when the label is referred to:

```

373 \def\getpagewiselinenumber#1{%
374   \c@linenumber #1\relax\testNumberedPage
375   \thelinenumber
376   }}

```

A summary of all per line expenses:

7 **CPU:** The `\output` routine is called for each line, and the page-search is done.

DISK: One line of output to the aux-file for each numbered line

10 **MEM:** One macro per page. Great improvement over v1.02, which had one control sequence per line in addition. It blew the hash table after some five thousand lines.

5.4 Twocolumn mode (New v3.06)

- 13 Twocolumn mode requires another patch to the `\output` routine, in order to print a column tag to the .aux file.

```

377 \AtBeginDocument{% v4.2, revtex4.cls (e.g.).
378 % <- TODO v4.4+: Or better in \LineNoLaTeXOutput!?
379   \let\LN@orig@makecol\@makecol}
380 \def\LN@makecol{%
381   \LN@orig@makecol
382   \setbox\@outputbox \vbox{%
383     \boxmaxdepth \@maxdepth
384     \protected@write\@auxout}{%
385       \string\LN@col{\if@firstcolumn1\else2\fi}}%

```

```

386     }%
387     \box\@outputbox
388   }% \vbox
389 } %% TODO cf. revtexln.sty.
390
391 \def\@LN@col{\def\@LN@column} % v4.22, removed #1.
392 \@LN@col{1}

```

1 5.5 Numbering modulo m , starting at f

Most users want to have only one in five lines numbered. `\LineNumber` is supposed to produce the outfit of the line number attached to the line, while `\thelinenumber` is used also for references, which should appear even if they are not multiples of five.

(New v4.00) Moreover, some users want to control which line number should be printed first. Support of this is now introduced here—see `\firstlinenumber` below.—`numline.sty` by Michael Jaegermann and James Fortune offers controlling which *final* line numbers should not be printed. What is it good for? We ignore this here until some user demands it.—Peter Wilson’s `ledmac.sty` offers much different choices of line numbers to be printed, due to Wayne Sullivan. (/New v4.00)

(New v4.22) `\c@linenumbermodulo` is rendered a fake counter, as discussed since v4.00. So it can no longer be set by `\setcounter`. `\modulolinenumbers` serves this purpose. Well, does anybody want to do what worked with `\addtocounter`? (Then please tell me.)—At least, `\value` still works. For the same purpose I rename the fake ‘firstlinenumber’ counter `\n@...` to `\c@...`. (/New v4.22)

```

19 % \newcount\c@linenumbermodulo % removed for v4.22

```

(New v4.00)

`\themodulolinenumber` waits for being declared `\LineNumber` by `\modulolinenumbers`. (This has been so before, no change.) Here is how it looked before:

```

% \def\themodulolinenumber{\@tempcnta\c@linenumber
25 %   \divide\@tempcnta\c@linenumbermodulo
%   \multiply\@tempcnta\c@linenumbermodulo
%   \ifnum\@tempcnta=\c@linenumber\thelinenumber\fi
28 % }

```

(UL) This was somewhat slow. This arithmetic happens at every line. This time I tend to declare an extra line counter (as opposed to my usual recommendations to use counters as rarely as possible) which is stepped every

1 line. It could be incremented in the same way as `\c@LN@truepage` is incre-
 mented via `\c1@page!` This is another point in favour of `{\linenumber}` being
 a L^AT_EX counter! When this new counter equals `\c@linenumbermodulo`, it is
 4 reset, and `\thelinenumber` is executed.—It gets much slower by my support
 of controlling the first line number below. I should improve this.—On the
 other hand, time expense means very little nowadays, while the number of
 7 T_EX counters still is limited.

For the same purpose, moreover, attaching the line number box could be
 intercepted earlier (in `\MakeLineNo`), without changing `\LineNumber`. How-
 10 ever, this may be bad for the latter’s announcement as a wizard interface in
 section 10. (/UL)

Here is the new code. It is very near to my `lnopatch.sty` code which
 13 introduced the first line number feature before.—I add starting with a `\relax`
 which is so often recommended—without understanding this really. At least,
 it will not harm.—Former group braces appear as `\beginngroup/\endgroup`
 16 here.

```

393 \def\themodulolinenumber{\relax
394   \ifnum\c@linenumber<\c@firstlinenumber \else
395     \beginngroup
396       \@tempcnta\c@linenumber
397       \advance\@tempcnta-\c@firstlinenumber
398       \divide\@tempcnta\c@linenumbermodulo
399       \multiply\@tempcnta\c@linenumbermodulo
400       \advance\@tempcnta\c@firstlinenumber
401       \ifnum\@tempcnta=\c@linenumber \thelinenumber \fi
402     \endngroup
403   \fi
404 }

```

(/New v4.00)

The user command to set the modulo counter: (New v4.31) ... a star
 19 variant is introduced to implement Hillel Chayim Yisraeli’s idea to print
 the first line number after an interruption of the edited text by some ed-
 itor’s text, regardless of the modulo. If it is 1, it is printed only with
 22 `\firstlinenumber{1}`. I.e., you use `\modulolinenumbers*` for the new
 feature, without the star you get the simpler behaviour that we have had
 so far. And you can switch back from the refined behaviour to the sim-
 25 ple one by using `\modulolinenumbers` without the star.—This enhance-
 ment is accompanied by a new package option `modulo*` which just executes
`\modulolinenumbers*` (subsection 6.4).—‘With `\firstlinenumber{1}`’ ex-
 28 actly means: ‘1’ is printed if and only if the last `\firstlinenumber` before or
 in the paragraph that follows the “interruption” has argument ‘1’ (or some-
 thing *expanding* to ‘1’, or (to) something that T_EX “reads” as 1, e.g.: a T_EX

1 count register storing 1).—At present, this behaviour may be unsatisfactory with pagewise line-numbering ... I'll make an experimental extra package if someone complains ...

```

405 \newcommand\modulolinenumbers{%
406   \@ifstar
407     {\def\LN@maybe@moduloresume{%
408       \global\let\LN@maybe@normalLineNumber
409         \LN@normalLineNumber}%
410       \LN@modulolinenos}%
411     {\let\LN@maybe@moduloresume\relax \LN@modulolinenos}%
412 }
413
414 \global\let\LN@maybe@normalLineNumber\relax
415 \let\LN@maybe@moduloresume\relax
416 \gdef\LN@normalLineNumber{%
417   \ifnum\c@linenumber=\c@firstlinenumber \else
418     \ifnum\c@linenumber>\@ne
419       \def\LineNumber{\thelinenumber}%
420     \fi
421   \fi

```

4 `\def` instead of `\let` enables taking account of a redefinition of `\thelinenumber` in a present numbering environment (e.g.).

```

422 \global\let\LN@maybe@normalLineNumber\relax}

```

7 Instead of changing `\LineNumber` directly by `LN@moduloresume`, these tricks enable `\modulolinenumbers*` to act as locally as I can make it. I don't know how to avoid that the output routine switches back to the normal modulo behaviour by a global change. (An `\aftergroup` may fail in admittedly improbable cases.)

```

423 \newcommand*\LN@modulolinenos[1][\z@]{%

```

The definition of this macro is that of the former `\modulolinenumbers`.
(/New v4.31)

```

424 \let\LineNumber\thelinenumber
425 \ifnum#1>\@ne
426   \chardef                               % v4.22, note below
427     \c@linenumbermodulo#1\relax
428 \else\ifnum#1=\@ne
13 %   \def\LineNumber{\thelinenumber}%

```

1 (New v4.00) I am putting something here to enable `\firstlinenumber` with `\c@linenumbermodulo = 1`. With `lnopatch.sty`, a trick was offered for this purpose. It is now obsolete.

```
429 \def\LineNumber{\@LN@ifgreat\thelinenumber}%
```

4 (/New v4.00)

```
430 \fi\fi
```

```
431 }
```

(New v4.00) The default of `\@LN@ifgreat` is

```
432 \let\@LN@ifgreat\relax
```

The previous changes as soon as `\firstlinenumber` is used:

```
433 \newcommand*\firstlinenumber[1]{%
```

```
434 \chardef\c@firstlinenumber#1\relax
```

7 No counter, little values allowed only—OK?—(UL) The change is local—OK? The good thing is that `\global\firstlinenumber{\langle number \rangle}` works. Moreover, `\modulolinenumbers` acts locally as well. (/UL)

10 (New v4.31)

```
435 \let\@LN@ifgreat\@LN@ifgreat@critical}
```

```
436
```

```
437 \def\@LN@ifgreat@critical{%
```

```
438 \ifnum\c@linenumber<\c@firstlinenumber
```

```
439 \expandafter \@gobble
```

```
440 \fi}%
```

(/New v4.31)

13 The default value of `\c@firstlinenumber` is 0. This is best for what one would expect from modulo printing.

```
441 \let\c@firstlinenumber=\z@
```

For usage and effects of `\modulolinenumbers` and `\firstlinenumbers`, please consult section 10. Two details on `\firstlinenumbers` here:

16 (i) `\firstlinenumber` acts on a paragraph if and only if (a) the paragraph is broken into lines “in line-numbering mode” (after `\linenumbers`, e.g.);
 19 (b) it is the last occurrence of a `\firstlinenumbers` before or in the paragraph. (The practical applications of this that I can imagine don’t seem appealing to me.) Cf. the explanation above of how `\modulolinenumbers` and `\firstlinenumbers` interact—for this and for (ii), which is concerned
 22 with possible arguments for `\firstlinenumbers`.

Note that the line numbers of the present section demonstrate the two devices. (/New v4.00)

```

442 \chardef\c@linenumbermodulo=5      % v4.2; ugly?
443 \modulolinenumbers[1]

```

1 (New v4.22) The new implementation through `\chardef` decreases the functionality and raises certain compatibility problems. I face this without fear. The maximum modulo value is now 255. I expect that this suffices for
 4 usual applications. However, some users have “abused” `lineno.sty` to get `ednotes.sty` features without line numbers, so have set the modulo to a value beyond the total number of lines in their edition. This ought to be
 7 replaced by `\let\makeLineNumber\relax`. (/New v4.22)

6 Package options

(New v4.1) The last heading formerly was the heading of what is now sub-
 10 section 6.4. The options declared there were said to execute user commands only. This was wrong already concerning `displaymath` and `hyperref`. At least, however, these options were no or almost no occasion to skip definitions
 13 or allocations. This is different with the options that we now insert.

6.1 Extended referencing to line numbers. (v4.2)

This subsection explains and declares package option `addpageno`.

16 If a line to whose number you refer by `\ref` is not on the present page, it may be useful to add the number of the page on which the line occurs—and perhaps it should not be added otherwise. In general, you could use
 19 the Standard L^AT_EX package `varioref` for this. However, the latter usually produces verbose output like ‘on the preceding page’— unless customized—, while in critical editions, e.g., one may prefer just adding the page number
 22 and some mark on the left of the line number, irrespectively of how far the page is apart etc. To support this, package option `addpageno` provides a command `\vpagelineref` to be used in place of `\ref`. This produces, e.g.,
 25 ‘34.15’ when referring to line 15 on page 34 while the present page is not 34. You can customize the outcome, see the package file `vplref.sty` where the code and further details are. You may conceive of `\vpagelineref` as a
 28 certain customization of `varioref`’s `\vref`.

This implies that option `addpageno` requires the files `vplref.sty` and `varioref.sty`. `addpageno` automatically loads both of them. Yet you can
 31 also load `varioref.sty` on your own to use its package options.

Of course, you might better introduce a shorter command name for `\vpagelineref` for your work, while we cannot predict here what shorthand
 34 will fit your work. E.g., `\newcommand{\lref}{\vpagelineref}`.

1 If you really want to add the page number in *any* case, use, e.g., some `\myref` instead of `\ref`, after

```
newcommand*\myref{\pageref{#1}.\ref{#1}}
```

or what you like. You don't need the `addpageno` option in this case.

4 `addpageno` is due to a suggestion by Sergei Mariev.

```
444 \DeclareOption{addpageno}{%
445 \AtEndOfPackage{\RequirePackage{vplref}[2005/04/25]}}
```

6.2 `\linelabel` in math mode

We have made some first steps towards allowing `\linelabel` in math mode.

7 Because our code for this is presently experimental, we leave it to the user to decide for the experiment by calling option `mathrefs`. We are in a hurry now and thus leave the code, explanations, and discussion in the separate package `ednmath0.sty`. Maybe we later find the time to improve the code and move the relevant content of `ednmath0.sty` to here. The optimal situation would be to define `\linelabel` from the start so it works in math mode, omitting the `mathrefs` option.

13 Actually, this package even provides adjustments for analogously allowing `ednotes.sty` commands in math mode. Loading the package is postponed to `\AtBeginDocument` when we know whether these adjustments are needed.

```
446 \DeclareOption{mathrefs}{\AtBeginDocument
447 {\RequirePackage{ednmath0}[2004/08/20]}}
```

6.3 Arrays, tabular environments (Revised v4.11)

This subsection explains and declares package options `edtable`, `longtable`, and `nolongtablepatch`.

19 The standard \LaTeX tabular environments come as single boxes, so the `lineno.sty` versions before v4.00 treated them as (parts of) single lines, printing (at most) one line number beside each and stepping the line number counter once only. Moreover, `\linelabels` got lost. Of course, tables are usually so high that you will want to treat each row like a line. (Christian Tapp even desires that the lines of table entries belonging to a single row are treated like ordinary lines.) Footnotes get lost in such environments as well, which was bad for `ednotes.sty`.

28 We provide adjustments to count lines, print their numbers etc. as desired at least for *some* \LaTeX tabular environments. (Like with other details,

1 “some” is to some extent explained in `edtable.sty`.) We do this similarly as
 with option `mathrefs` before. We leave code and explanations in the separate
 package `edtable.sty`. (For wizards: this package provides adjustments for
 4 `ednotes.sty` as well. However, in the present case we don’t try to avoid them
 unless `ednotes.sty` is loaded.) Package option `edtable` defines—by load-
 ing `edtable.sty`—an environment `{edtable}` which is able to change some
 7 L^AT_EX tabular environments with the desired effects. (v4.11: `edtable.sty`
 v1.3 counts L^AT_EX’s `{array}` [etc.] as a “tabular environment” as well.)

The `{edtable}` environment doesn’t help with `longtable.sty`, however.
 10 To make up for this, `{longtable}` is adjusted in a different way—and this
 happens only when another `lineno.sty` option `longtable` is called. In this
 case, option `edtable` needn’t be called explicitly: option `longtable` works
 13 as if `edtable` had been called.

Now, we are convinced that vertical spacing around `{longtable}` works
 wrongly—see L^AT_EX bugs database tools/3180 and 3485, or see explanations
 16 in the package `ltabptch.sty` (which is to be obtained from CTAN folder
`/macros/latex/ltabptch`). Our conviction is so strong that the `longtable`
 option loads—after `longtable.sty`—the patch package `ltabptch.sty`. If
 19 the user doesn’t want this (maybe preferring her own arrangement with the
 vertical spacing), she can forbid it by calling `nolongtablepatch`.

The following code just collects some choices, which are then executed
 22 in section 6.7. We use an `\if...` without `\newif` since `\if...true` and
`\if...false` would occur at most two times and only within the present
 package. (`\AtEndOfClass{\RequirePackage{edtable}}` could be used in-
 25 stead, I just overlooked this. Now I don’t change it because it allows to
 change the version requirement at one place only.)

```
448 \let\if@LN@edtable\iffalse
449
450 \DeclareOption{edtable}{\let\if@LN@edtable\iftrue}
451
452 \DeclareOption{longtable}{\let\if@LN@edtable\iftrue
453   \PassOptionsToPackage{longtable}{edtable}}
454
455 \DeclareOption{nolongtablepatch}{%
456   \PassOptionsToPackage{nolongtablepatch}{edtable}}
```

(/New v4.1)

28 6.4 Switch among settings

There is a bunch of package options that execute user commands only.

- 1 Options `left` (`right`) put the line numbers on the left (right) margin.
This works in all modes. `left` is the default.

```
457 \DeclareOption{left}{\leftlinenumbers*}
458
459 \DeclareOption{right}{\rightlinenumbers*}
```

- 4 Option `switch` (`switch*`) puts the line numbers on the outer (inner) margin
of the text. This requires running the pagewise mode, but we turn off the
page offset subtraction, getting sort of running numbers again. The `pagewise`
option may restore true pagewise mode later.

```
460 \DeclareOption{switch}{\setpagewiselinenumbers
461                      \switchlinenumbers
462                      \runningpagewiselinenumbers}
463
464 \DeclareOption{switch*}{\setpagewiselinenumbers
465                       \switchlinenumbers*%
466                       \runningpagewiselinenumbers}
```

- 7 In twocolumn mode, we can switch the line numbers to the outer margin,
and/or start with number 1 in each column. Margin switching is covered by
the `switch` options.

```
467 \DeclareOption{columnwise}{\setpagewiselinenumbers
468                             \columnwiselinenumberstrue
469                             \realpagewiselinenumbers}
```

- 10 The options `pagewise` and `running` select the major linenummer mechanism.
`running` line numbers refer to a real counter value, which can be reset for
any paragraph, even getting multiple paragraphs on one page starting with
13 line number one. `pagewise` line numbers get a unique hidden number within
the document, but with the opportunity to establish the page on which they
finally come to rest. This allows the subtraction of the page offset, getting
16 the numbers starting with 1 on top of each page, and margin switching in
twoside formats becomes possible. The default mode is `running`.

- The order of declaration of the options is important here `pagewise` must
19 come after `switch`, to override running pagewise mode. `running` comes last,
to reset the running line number mode, e.g, after selecting margin switch
mode for `pagewise` running. Once more, if you specify all three of the options
22 [`switch,pagewise,running`], the result is almost nothing, but if you later
say `\pagewiselinenumbers`, you get margin switching, with real pagewise
line numbers.

```

470 \DeclareOption{pagewise}{\setpagewiselinenumbers
471                               \realpagewiselinenumbers}
472
473 \DeclareOption{running}{\setrunninglinenumbers}

```

- 1 The option `modulo` causes only those linenumbers to be printed which are multiples of five.

```

474 \DeclareOption{modulo}{\modulolinenumbers\relax}

```

- Option `modulo*` modifies `modulo` in working like `\modulolinenumbers*`—see section 10.

```

475 \DeclareOption{modulo*}{\modulolinenumbers*\relax}

```

- The package option `mathlines` switches the behavior of the `{\linenomath}` environment with its star-form. Without this option, the `{\linenomath}` environment does not number the lines of the display, while the star-form does. With this option, its just the opposite.

```

476 \DeclareOption{mathlines}{\linenumberdisplaymath}

```

- `displaymath` now calls for wrappers of the standard L^AT_EX display math environment. This was previously done by `mlineno.sty`.

- (New v4.3) Option ‘`displaymath`’ becomes default according to Erik Luijten’s suggestion. I was finally convinced of this as soon as I discovered how to avoid a spurious line number above `\begin{linenomath}` (subsection 3.3). `\endlinenomath` provides `\ignorespaces`, so what could go wrong now?

```

477 \DeclareOption{displaymath}{\PackageWarningNoLine{lineno}{%
478                               Option [displaymath] is obsolete -- default now!}}

```

- 16 (/New v4.3)

6.5 Compatibility with hyperref

- The `hyperref` package, via `nameref`, requires three more groups in the second argment of a `\newlabel`. Well, why shouldn’t it get them? (New v3.07) The presence of the `nameref` package is now detected automatically `\AtBeginDocument`. (/New v3.07) (Fixed in v3.09) We try to be smart, and test `\AtBeginDocument` if the `nameref` package is loaded, but `hyperref` postpones the loading of `nameref` too, so this is all in vain.

1 (New v4.3) But we can also test at the first `\linelabel`. Regarding the error-message for misplaced `\linelabel` from v4.11: previously, `\linenumbers` rendered `\linelabel` the genuine version of `\linelabel` from
 4 the start on. This doesn't work now, since `\@LN@linelabel` may change its meaning after the first `\linenumbers` and before a next one (if there is some).
 (/New v4.3)

```
479 \DeclareOption{hyperref}{\PackageWarningNoLine{lineno}{%
480     Option [hyperref] is obsolete.
481     \MessageBreak The hyperref package is detected automatically.}}
482
483 \AtBeginDocument{%
484     \@ifpackageloaded{nameref}{%
```

7 (New v4.3) “Global” is merely “symbolic” `\AtBeginDoc...`. If `nameref` is not detected here, the next `\@LN@linelabel` will do almost the same, then globally indeed.

```
485     \gdef\@LN@ExtraLabelItems{}{}{}%
486 }{%
487     \global\let\@LN@@linelabel\@LN@linelabel
488     \gdef\@LN@linelabel{%
```

10 `\@ifpackageloaded` is “preamble only”, its—very internal—preamble definition is replicated here:

```
489     \expandafter
490     \ifx\csname ver@nameref.sty\endcsname\relax \else
491         \gdef\@LN@ExtraLabelItems{}{}{}%
492     \fi
```

Now aim at the “usual” behaviour:

```
493     \global\let\@LN@linelabel\@LN@@linelabel
494     \global\let\@LN@@linelabel\relax
495     \@LN@linelabel
496 }%
497 }%
498 }
```

13 (/New v4.3)
 (New v4.1)

1 6.6 A note on calling so many options

The number of package options may stimulate worrying about how to *enter* all the options that one would like to use—they may not fit into one line.

- 4 Fortunately, you can safely break code lines after the commas separating the option names in the `\usepackage` command (no comment marks needed).

6.7 Execute options

- 7 We stop declaring options and execute the ones that are called by the user. (/New v4.1)

```
499 \ProcessOptions
```

- (New v4.1) Now we know whether `edtable.sty` is wanted and (if it is) with which options it is to be called.

```
500 \if@LN@edtable \RequirePackage{edtable}[2005/03/07] \fi
```

(/New v4.1)

7 Former package extensions

- 13 The extensions in this section were previously supplied in separate `.sty` files.

7.1 *displaymath*

- (New v4.3) From now on, you no longer need to type the `{linenomath}` environment with the `\[`, `{equation}`, and `{eqnarray}` environments—and you no longer need to use the former package option `displaymath` for this feature. (/New v4.3)

- 19 The standard L^AT_EX display math environments are wrapped in a `{linenomath}` environment.

- (New 3.05) The `[fleqn]` option of the standard L^AT_EX classes defines the display math environments such that line numbers appear just fine. Thus, we need not do any tricks when `[fleqn]` is loaded, as indicated by presents of the `\mathindent` register. (/New 3.05)

- 25 (New 3.05a) for `{eqnarray}`s we rather keep the old trick. (/New 3.05a)

- (New 3.08) Wrap `\[` and `\]` into `{linenomath}`, instead of `{displaymath}`. Also save the definition of `\equation`, instead of replicating the current L^AT_EX definition. (/New 3.08)

```

501 \ifundefined{mathindent}{
502
503 \let\LN@displaymath\[%
504 \let\LN@enddisplaymath\]%
505 \renewcommand\[{ \begin{linenomath}\LN@displaymath}%
506 \renewcommand\}{\LN@enddisplaymath\end{linenomath}}%

507 \let\LN@equation\equation
508 \let\LN@endequation\endequation
509 \renewenvironment{equation}%
510     {\linenomath\LN@equation}%
511     {\LN@endequation\endlinenomath}%
512
513 }{}% \ifundefined{mathindent} -- 3rd arg v4.2, was \par!
514
515 \let\LN@eqnarray\eqnarray
516 \let\LN@endeqnarray\endeqnarray
517 \renewenvironment{eqnarray}%
518     {\linenomath\LN@eqnarray}%
519     {\LN@endeqnarray\endlinenomath}%

```

- 1 (UL) Indeed. The L^AT_EX macros are saved for unnumbered mode, which is detected by `\linenomath`. (/UL)

7.2 Line numbers in internal vertical mode

- 4 The command `\internallinenumbers` adds line numbers in internal vertical mode, but with limitations: we assume fixed baseline skip.

(v4.22) v3.10 provided a global (`\global\advance`) as well as a local version (star-form, using `\c@internallinenumbers`). `\resetlinenumbers` acted locally and was here used with the global version—save stack danger, T_EXbook p. 301—in v4.00 I disabled the global version therefore. Now I find that it is better to keep a global version, and the now global `\resetlinenumbers` is perfect for this. The global version allows continuing the “internal” numbers in the ensuing “external” text, and—unless reset by brackets argument—continuing the above series of line numbers. As with v3.10, the local version always starts with line number one. A new `\@LN@iglobal` steps `\globally` in the global version, otherwise it is `\relax`. (I also remove all my stupid discussions as of v4.00. And I use `\newcommand`.) (v4.22)

```

520 \let\@LN@iglobal\global % v4.22
521
522 \newcommand\internallinenumbers{\setrunninglinenumbers
523     \let\@@par\internallinenumberspar

```

```

524 \ifx\@par\@@@par\let\@par\internallinnumberpar\fi
525 \ifx\par\@@@par\let\par\internallinnumberpar\fi
526 \ifx\@par\linnumberpar\let\@par\internallinnumberpar\fi
527 \ifx\par\linnumberpar\let\par\internallinnumberpar\fi
528 \@ifnextchar[{\resetlinenumber}%]
529     {\@ifstar{\let\c@linenumber\c@internallinnumber
530         \let\@LN@iglobal\relax % v4.22
531         \c@linenumber\@ne}{}}%
532 }
533
534 \let\endinternallinnumbers\endlinenumbers
535 \@namedef{internallinnumbers*}{\internallinnumbers*}
536 \expandafter\let\csname endinternallinnumbers*\endcsname\endlinenumbers
537
538 \newcount\c@internallinnumber
539 \newcount\c@internallinnumbers
540
541 \newcommand\internallinnumberpar{%
542     \ifvmode\@@@par\else\ifinner\@@@par\else\@@@par
543     \begingroup
544         \c@internallinnumbers\prevgraf
545         \setbox\@tempboxa\hbox{\vbox{\makeinternallinnumbers}}%
546         \dp\@tempboxa\prevdepth
547         \ht\@tempboxa\z@
548         \nobreak\vskip-\prevdepth
549         \nointerlineskip\box\@tempboxa
550     \endgroup
551     \fi\fi
552 }
553
554 \newcommand\makeinternallinnumbers{%
555     \ifnum\c@internallinnumbers>\z@ % v4.2
556     \hb@xt@\z@{\makeLineNumber}%
557     \@LN@iglobal % v4.22
558     \advance\c@linenumber\@ne
559     \advance\c@internallinnumbers\m@ne
560     \expandafter\makeinternallinnumbers\fi
561 }
562 % TODO v4.4+: star: line numbers right!? cf. lncapt.sty

```

1 7.3 Line number references with offset

This extension defines macros to refer to line numbers with an offset, e.g., to refer to a line which cannot be labeled directly (display math). This was formerly known as `rlineno.sty`.

To refer to a pagewise line number with offset:

```
\linerefp[OFFSET]{LABEL}
```

- 1 To refer to a running line number with offset:

```
\linerefr[⟨OFFSET⟩]{⟨LABEL⟩}
```

To refer to a line number labeled in the same mode as currently selected:

- 4 `\lineref[⟨OFFSET⟩]{⟨LABEL⟩}`

```
563 \newcommand\lineref{%
564   \ifx\c@linenumber\c@runninglinenumber
565     \expandafter\linerefr
566   \else
567     \expandafter\linerefp
568   \fi
569 }
570
571 \newcommand*\linerefp[2][\z@]{%
572   \let\@thelinenumber\thelinenumber
573   \edef\thelinenumber{\advance\c@linenumber#1\relax
574                       \noexpand\@thelinenumber}%
575   \ref{#2}%
576 }}
```

This goes deep into L^AT_EX's internals.

```
577 \newcommand*\linerefr[2][\z@]{%
578   \def\@@linerefadd{\advance\c@linenumber#1}%
579   \expandafter\@setref\csname r@#2\endcsname
580   \@linerefadd{#2}%
581 }}
582
583 \newcommand*\@linerefadd[2]{\c@linenumber=#1\@@linerefadd\relax
584                               \thelinenumber}
```

7.4 Numbered quotation environments

- 7 The `{numquote}` and `{numquotation}` environments are like `{quote}` and `{quotation}`, except there will be line numbers.

An optional argument gives the number to count from. A star `*` (inside or outside the closing `}`) prevent the reset of the line numbers. Default is to count from one.

(v4.22: A local version using `\c@internallinenumber` might be useful, see subsection 7.2.)

```
585 \newcommand\quotelinenumbers
586   {\@ifstar\linenumbers{\@ifnextchar[\linenumbers{\linenumbers*}}}
587
```

```

588 \newdimen\quotelinumbersep
589 \quotelinumbersep=\linumbersep
590 \let\quotelinumberfont\linumberfont
591
592 \newcommand\numquotelist
593   {\leftlinenumbers
594     \linumbersep\quotelinumbersep
595     \let\linumberfont\quotelinumberfont
596     \addtolength{\linumbersep}{-\@totalleftmargin}%
597     \quotelinumbers
598   }
599
600 \newenvironment{numquote}   {\quote\numquotelist}\endquote}
601 \newenvironment{numquotation} {\quotation\numquotelist}\endquotation}
602 \newenvironment{numquote*}   {\quote\numquotelist*}\endquote}
603 \newenvironment{numquotation*} {\quotation\numquotelist*}\endquotation}

```

1 7.5 Frame around a paragraph

The `{bframe}` environment draws a frame around some text, across page breaks, if necessary.

- 4 This works only for plain text paragraphs, without special height lines. All lines must be `\baselineskip` apart, no display math.

```

604 \newenvironment{bframe}
605   {\par
606     \@tempdima\textwidth
607     \advance\@tempdima 2\bframesep
608     \setbox\bframebox\hb@xt@\textwidth{%
609       \hskip-\bframesep
610       \vrule\@width\bframerule\@height\baselineskip\@depth\bframesep
611       \advance\@tempdima-2\bframerule
612       \hskip\@tempdima
613       \vrule\@width\bframerule\@height\baselineskip\@depth\bframesep
614       \hskip-\bframesep
615     }%
616     \hbox{\hskip-\bframesep
617       \vrule\@width\@tempdima\@height\bframerule\@depth\z@}%
618     \nointerlineskip
619     \copy\bframebox
620     \nobreak
621     \kern-\baselineskip
622     \runninglinenumbers
623     \def\makeLineNumber{\copy\bframebox\hss}%
624   }
625   {\par
626     \kern-\prevdepth
627     \kern\bframesep

```

```

628 \nointerlineskip
629 \@tempdima\textwidth
630 \advance\@tempdima 2\bframesep
631 \hbox{\hskip-\bframesep
632       \vrule\@width\@tempdima\@height\bframerule\@depth\z@}%
633 }
634
635 \newdimen\bframerule
636 \bframerule=\fboxrule
637
638 \newdimen\bframesep
639 \bframesep=\fboxsep
640
641 \newbox\bframebox

```

1 8 Move \vadjust items (New v4.00)

This section completes reviving `\pagebreak`, `\nopagebreak`, `\vspace`, and the star and optional form of `\`. This was started in section 2.1 and resumed in section 2.4 and subsection 3.1. The problem was explained in section 2.1: `\vadjust` items come out at a bad position, and the \LaTeX commands named before work with `\vadjust` indeed. Our solution was sketched there as well.

7 According to the caveat in subsection 3.2 concerning `\ifLineNumbers`, the \LaTeX commands enumerated may go wrong if you switch line numbering inside or at the end of a paragraph.

10 8.1 Redefining \vadjust

`\vadjust` will temporarily be changed into the following command.

```

642 \def\PostponeVadjust#1{%
643   \global\let\vadjust\@LN@\vadjust

```

This undoes a `\global\let\vadjust\PostponeVadjust` which will start each of the refined \LaTeX commands. The `\globals` are most probably superfluous. They might be useful should one `\vadjust` appear in a group starting after the change of `\vadjust` into `\PostponeVadjust`. (UL) Even the undoing may be superfluous, cf. discussion in section 8.2 below. (UL)

```

644   \vadjust{\penalty-\@Mppvcodepen}%
645   \g@addto@macro\@LN@\vadjustlist{#1\@lt}%
646 }
647 \let\@LN@\vadjust\vadjust
648 \global\let\@LN@\vadjustlist\@empty
649 \global\let\@LN@do@\vadjusts\relax

```

1 These \globals are just to remind that all the changes of the strings after \let should be \global (T_EXbook p. 301). \@LN@vadjustlist collects the \vadjust items of a paragraph. \PassVadjustList tears one
 4 \vadjust item for the current line out of \@LN@vadjustlist and puts it into \@LN@do@vadjusts. The latter is encountered each line in \MakeLineNo (section 2.4), while those L^AT_EX \vadjust commands will come rather rarely.
 7 So I decided that \@LN@do@vadjust is \relax until a \vadjust item is waiting. In the latter case, \@LN@do@vadjusts is turned into a list macro which resets itself to \relax when the other contents have been placed in the vertical list.—\PassVadjustList is invoked by the output routine (section 2.1),
 10 so the \box255 must be put back.

```

650 \def\PassVadjustList{%
651   \unvbox\@cclv
652   \expandafter \@LN@xnext \@LN@vadjustlist \@@
653           \@tempa \@LN@vadjustlist
654   \ifx\@LN@do@vadjusts\relax
655     \gdef\@LN@do@vadjusts{\global\let\@LN@do@vadjusts\relax}%
656   \fi
657   \expandafter \g@addto@macro \expandafter \@LN@do@vadjusts
658     \expandafter {\@tempa}%
659 }

```

8.2 Redefining the L^AT_EX commands

13 Now we change \pagebreak etc. so that they use \PostponeVadjust in place of \vadjust. We try to do this as independently as possible of the implementation of the L^AT_EX commands to be redefined. Therefore,
 16 we don't just copy macro definition code from any single implementation (say, latest L^AT_EX) and insert our changes, but attach a conditional \global\let\vadjust\PostponeVadjust to their left ends in a way which
 19 should work rather independantly of their actual code. However, \vadjust should be the primitive again after execution of the command. So the \global\let... may be used only if it's guaranteed that a \vadjust is
 22 near.—(UL) Sure? In line numbering mode, probably each \vadjust coming from a L^AT_EX command should be \PostponeVadjust. \marginpars and floats seem to be the only cases which are not explicitly dealt with in
 25 the present section. This would be a way to avoid \@LN@nobreaktrue! Of course, the \vadjusts that the present package uses then must be replaced by \@LN@@vadjust.—Maybe next time. (/UL)

28 The next command and something else will be added to the L^AT_EX commands we are concerned with here.

```

660 \DeclareRobustCommand\@LN@changeadjust{%
661   \ifvmode\else\ifinner\else
662     \global\let\adjust\PostponeVadjust
663   \fi\fi
664 }

```

1 (UL) What about math mode? Math display? Warn? (/UL)
 \@tempa will now become a two place macro which adds first argu-
 4 ond argument. As long as we need it, we can't use the star form of
 \DeclareRobustCommand or the like, because AMS-~~La~~TeX uses \@tempa for
 \@ifstar. (New v4.41) And for the same reason, that \CheckCommand* had
 7 to be raised! (/New v4.41)

```

665 \CheckCommand*\@parboxrestore{\@arrayparboxrestore\let\\\@normalcr}
666
667 \def\@tempa#1#2{%
668   \expandafter \def \expandafter#2\expandafter{\expandafter
669     \ifLineNumbers\expandafter#1\expandafter\fi#2}%
670 }

```

(UL) This \ifLineNumber can be fooled by \linenumbers ahead etc. It
 might be better to place a signal penalty in any case and let the output
 10 routine decide what to do. (/UL)

We use the occasion to switch off linenumbers where they don't work
 anyway and where we don't want them, especially in footnotes:

```

671 \@tempa\nolinenumbers\@arrayparboxrestore

```

13 We hope this suffices ... let's check one thing at least: [(New v4.41) see
 \CheckCommand above (/New v4.41)]

Now for the main theme of the section. The next lines assume that
 16 \vspace, \pagebreak, and \nopagebreak use \adjust whenever they occur
 outside vertical mode; moreover, that they don't directly read an argument.
 Indeed \pagebreak and \nopagebreak first call something which tests for a
 19 left bracket ahead, while \vspace first tests for a star.

```

672 \@tempa\@LN@changeadjust\vspace
673 \@tempa\@LN@changeadjust\pagebreak
674 \@tempa\@LN@changeadjust\nopagebreak

```

\\, however, uses \adjust only in star or optional form. We relax independen-
 cy of implementation in assuming that \@normalcr is the fragile version
 22 of \\ (and we use \@ifstar!). (Using a copy of \\ would be safer, but an
 ugly repetition of \protect.)

```

675 \DeclareRobustCommand\{\%
676   \ifLineNumbers
677     \expandafter \@LN@cr
678   \else
679     \expandafter \@normalcr
680   \fi
681 }
682 \def\@LN@cr{%
683   \@ifstar{\@LN@changeadjust\@normalcr*}%
684     {\@ifnextchar[{\@LN@changeadjust\@normalcr}\@normalcr}%
685 }

```

1 Moreover we hope that `\newline` never leads to a `\adjust`, although names
of some commands invoked by `\` contain `newline`. At last, this seems to
have been OK since 1989 or even earlier.

4 Let's have a few tests. Testing `\pagebreak` and `\nopagebreak` would
5 be too expensive here, but—oops!—we have just experienced a successful
6 `\vspace*{.5\baselineskip}`. A `* [.5\baselineskip]`

7 may look even more drastical, but this time we are happy about it. Note
8 that the line numbers have moved with the lines. Without our changes, one
line number would have “anticipated” the move of the next line, just as you
9 can observe it now. (/New v4.00)

8.3 Reminder on obsolescence

11

(New v4.1) We have completed inclusion of the earlier extension packages
`linenox0.sty`, `linenox1.sty`, and `lnopatch.sty`. If one of them is loaded,
though, we produce an error message before something weird happens. We
avoid `\newif` because the switchings occur so rarely.

12
13
14
15

```

686 \AtBeginDocument{%
687   \let\ifLN@obsolete\iffalse
688   \@ifpackageloaded{linenox0}{\let\ifLN@obsolete\iftrue}\relax
689   \@ifpackageloaded{linenox1}{\let\ifLN@obsolete\iftrue}\relax
690   \@ifpackageloaded{lnopatch}{\let\ifLN@obsolete\iftrue}\relax
691   \ifLN@obsolete
692     \PackageError{lineno}{Obsolete extension package(s)}{%
693       With lineno.sty version 4.00 or later,\MessageBreak
694       linenox0/linenox1/lnopatch.sty must no longer be loaded.}%
695   \fi
696 }

```

1 9 The final touch

2 There is one deadcycle for each line number.

```
697 \advance\maxdeadcycles 100
698
699 \endinput
```

3 10 The user commands

4 The user commands to turn on and off line numbering are

5 `\linenumbers`

6 Turn on line numbering in the current mode.

7 `\linenumbers*`

8 and reset the line number to 1.

9 `\linenumbers[number]`

10 and start with *number*.

11 `\nolinenumbers`

12 Turn off line numbering.

13 `\runninglinenumbers*[number]`

14 Turn on *running* line numbers, with the same optional arguments as
 15 `\linenumbers`. The numbers are running through the text over page-
 16 breaks. When you turn numbering off and on again, the numbers will
 17 continue, except, of course, if you ask to reset or preset the counter.

18 `\pagewiselinenumbers`

19 Turn on *pagewise* line numbers. The lines on each page are numbered
 20 beginning with one at the first *pagewise* numbered line.

21 `\resetlinenumber[number]`

22 Reset [Set] the line number to 1 [*number*].

23 `\setrunninglinenumbers`

24 Switch to *running* line number mode. Do *not* turn it on or off.

<code>\setpagewiselinenumbers</code>	1
Switch to <i>pagewise</i> line number mode. Do <i>not</i> turn it on or off.	2
<code>\switchlinenumbers*</code>	3
Causes margin switching in <i>pagewise</i> modes. With the star, put the line numbers on the inner margin.	4 5
<code>\leftlinenumbers*</code>	6
<code>\rightlinenumbers*</code>	7
Set the line numbers in the left/right margin. With the star this works for both modes of operation, without the star only for the currently selected mode.	8 9 10
<code>\runningpagewiselinenumbers</code>	11
When using the <i>pagewise</i> line number mode, do not subtract the page offset. This results in running line numbers again, but with the possibility to switch margins. Be careful when doing line number referencing, this mode status must be the same while setting the paragraph and during references.	12 13 14 15 16
<code>\realpagewiselinenumbers</code>	17
Reverses the effect of <code>\runningpagewiselinenumbers</code> .	18
<code>\modulolinenumbers[⟨<i>number</i>⟩]</code>	19
Give a number only to lines which are multiples of [⟨ <i>number</i> ⟩]. If ⟨ <i>number</i> ⟩ is not specified, the current value in the counter <code>linenumbermodulo</code> is retained. ⟨ <i>number</i> ⟩=1 turns this off without changing <code>linenumbermodulo</code> . The counter is initialized to 5.	20 21 22 23
<code>\modulolinenumbers*[⟨<i>number</i>⟩]</code>	24
Like <code>\modulolinenumbers</code> , the only difference being that the first line number after a <code>\linenumbers</code> (or <code>\runninglinenumbers</code> , <code>\pagewiselinenumbers</code> , <code>\quotelinenumbers</code>) is printed regardless of the modulo—yet ‘1’ is printed only after (or ...) <code>\firstlinenumber{1}</code> . This also applies to the first line of a <code>{linenumbers}</code> or respective environment. See subsection 5.5 for another explanation. The behaviour may be unsatisfactory with <i>pagewise</i> line-numbering.	25 26 27 28 29 30 31 32

1 `\firstlinenumber`

2 `\firstlinenumber{⟨filino⟩}` brings about that (after it) line num-
 3 bers less than `⟨filino⟩` do *not* appear in the margin. Moreover,
 4 with `\modulolinenumbers[⟨number⟩]`, just the line numbers which
 5 are `⟨filino⟩` plus a multiple of `⟨number⟩` are printed.—If you had
 6 `\firstlinenumber{⟨pos⟩}` with some `⟨pos⟩ > 0` and want to switch
 7 to printing multiples of, e.g., 4, you best do `\modulolinenumbers[4]`
 8 and `\firstlinenumber{0}`. (See subsection 5.5 for technical details.)

9 `\linenumberdisplaymath`

10 Number the lines of a display math in a `{linenomath}` environment,
 11 but do not in a `{linenomath*}` environment. This is used by the
 12 package option `[mathlines]`.

13 `\nolinenumberdisplaymath`

14 Do not Number the lines of a display math in a `{linenomath}` envi-
 15 ronment, but do in a `{linenomath*}` environment. This is the default.

16 `\linelabel`

17 Set a `\linelabel{⟨foo⟩}` to the line number where this commands is
 18 in. Refer to it with the L^AT_EX referencing commands `\ref{⟨foo⟩}` and
 19 `\pageref{⟨foo⟩}`.

20 The commands can be used globally, locally within groups or as environ-
 21 ments. It is important to know that they take action only when the `\par`
 22 is executed. The `\end{⟨mode⟩linenumbers}` commands provide a `\par`. Ex-
 23 amples:

24 `{\linenumbers ⟨text⟩ \par}`

25 `\begin{linenumbers}`

26 `⟨text⟩`

27 `\end{linenumbers}`

28

29 `⟨paragraph⟩ {\linenumbers\par}`

30

31 `\linenumbers`

32 `⟨text⟩ \par`

33 `\nolinenumbers`

34

35 `\linenumbers`

36 `⟨paragraph⟩ {\nolinenumbers\par}`

37

(New v4.00) However, the examples containing *paragraph* show what you should *not* do, at least if you use `\pagebreak`, `\nopagebreak`, `\vspace`, `*` or `\{[space]`—cf. section 8.

The same care should be applied to the “wizard” devices `\ifLineNumbers` (subsection 3.2) and `\PostponeVadjust` (section 8.1). (/New v4.00)

(New v4.11) Oh, and the commands and environments of section s:Xt are missing. Sorry, I am in a hurry now. May be next time.—And the environments `{linenomath}` and `{linenomath*}` should get an own paragraph. In short, each math display, equation, or `{eqnarray}` should be “wrapped” in one of `{linenomath}` and `{linenomath*}`.

10.1 Customization hooks

There are several hooks to customize the appearance of the line numbers, and some low level hooks for special effects.

`\thelinenumber`

This macro should give the representation of the line number in the \LaTeX -counter `linenumber`. The default is provided by \LaTeX :

```
\arabic{linenumber}
```

`\makeLineNumberLeft`

This macro is used to attach a line number to the left of the text page. This macro should fill an `\hbox` to `0pt` which will be placed at the left margin of the page, with the reference point aligned to the line to which it should give a number. Please use the macro `\LineNumber` to refer to the line number.

The default definition is

```
\hss\linenumberfont\LineNumber\hskip\linenumbersep
```

`\makeLineNumberRight`

Like `\makeLineNumberLeft`, but for line numbers on the right margin.

The default definition is

```
\linenumberfont\hskip\linenumbersep\hskip\textwidth
\hbox to\linenumberwidth{\hss\LineNumber}\hss
```

`\linenumberfont`

This macro is initialized to

```
\normalfont\tiny\sffamily
```

1 `\linenumbersep`

2 This dimension register sets the separation of the linenumber to the
3 text. Default value is `10pt`.

4 `\linenumberwidth`

5 This dimension register sets the width of the line number box on the
6 right margin. The distance of the right edge of the text to the right
7 edge of the line number is `\linenumbersep + \linenumberwidth`. The
8 default value is `10pt`.

9 `\theLineNumber` (for wizards)

10 This macro is called for printing a `\newlabel` entry to the aux-file.
11 Its definition depends on the mode. For running line numbers it's just
12 `\thelinenumber`, while in pagewise mode, the page offset subtraction
13 is done in here.

14 `\makeLineNumber` (for wizards)

15 This macro produces the line numbers. The definition depends
16 on the mode. In the running line numbers mode it just expands
17 `\makeLineNumberLeft`.

18 `\LineNumber` (for wizards)

19 This macro is called by `\makeLineNumber` to typeset the line num-
20 ber. This hook is changed by the modulo mechanism and by
21 `\firstlinenumber`.