lineno.sty v4.1 2004/10/19

A LaTeX package to attach line numbers to paragraphs

Stephan I. Böttcher
Uwe Lück

boettcher@physik.uni-kiel.de
ednotes.sty@web.de

Contents

1 Introductions

(New v4.00) Parts of former first section have been rendered separate subsections for package version v4.00. (/New v4.00)

1.1 Introduction to versions v < 4

This package provides line numbers on paragraphs. After TeX has broken a paragraph into lines there will be line numbers attached to them, with the possibility to make references through the LaTeX `\ref`, `\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 `\par`, `\@par` and `\@@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 TEXs 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 `\marginpar`s, but not too well with displaymath. `\footnote`s 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 v4.00 and v4.1 (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://www.homepages.ucl.ac.uk/~ucgadkw/edmac/`

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

Soon, some weaknesses of lineno.sty showed up, mainly since Christian's critical editions (using ednotes.sty) needed lots of `\linelabel`s and footnotes. (These weaknesses are due to weaknesses of LATEX's `\marginpar` mechanism that Stephan used for `\linelabel`.) So we changed some lineno.sty definitions in some extra files, which moreover offered new features. We sent these files to Stephan, hoping he would take the changes into lineno.sty. However, he was too short of time.

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

The improvements are as follows:

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

(ii) The number of `\linelabel`s in one paragraph is no longer limited to 18.

(iii) `\pagebreak`, `\nopagebreak`, `\vspace`, and the star and optional versions of `\\` work as one would expect (section 7).

(iv) A command is offered which chooses the first line number to be printed in the margin (subsection 5.4).

(v) (New v4.1) LaTeX tabular environments (optionally) get line numbers as well, and you can refer to them in the usual automatic way. (It may be considered a shortcoming that, precisely, *rows* are numbered, not lines. See subsection 8.2.)

(vi) We are moving towards referring to math items (subsection 8.1 and the hooks in subsection 4.1). (/New v4.1)

(Thanks to Stephan for making this possible!)

You may trace the earlier developments of these changes by requesting our files linenox0.sty, linenox1.sty, and lnopatch.sty. Most of our changes have been in linenox0.sty. Our linenox1.sty has extended linenox0.sty for one single purpose in a not very stable way. lnopatch.sty has done the first line number thing referred to in case (iv) up to now. (New v4.1) Case (v) earlier was provided by our edtab02.sty—now called 'edtable.sty'. (/New v4.1)

Ednotes moreover profits from Stephan's offer with regard to the documentation of our code which yielded these improvements formerly. This documentation now becomes printable, being part of the lineno.sty documentation.

Of course, Stephan's previous lineno.sty versions were a great and ingenious work and exhibit greatest TeXpertise. I never could have done this. I learnt a lot in studying the code when Christian pointed out strange output results and error messages, and there are still large portions of lineno.sty which I don't understand (consider only pagewise numbering of lines). Fortunately, Stephan has offered future help if needed.—My code for attaching line numbers to *tabular environments* (as mentioned above, now still in edtable.sty) developed from macros which Stephan and Christian experimented with in December 2002. Stephan built the basics. (However, I then became too proud to follow his advice only to use and modify longtable.sty.)

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

On the other hand, the sections on improvements of the implementation have been blown up very much and may be tiring and litte understandable for mere *users*. These users may profit from the present presentation just by jumping to sections 8 and 10. There is a user's guide ulineno.tex which may be even more helpful, but it has not been updated for a while.

1.3 Availability

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

1.4 Introductory code

This style option is written for $\LaTeX\,2_\varepsilon$, November 1994 or later, since we need the `\protected@write` macro.

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

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

(/New v4.00)

2 Put the line numbers to the lines

(New v4.00) This section contained the most basic package code previously. For various purposes of version 4..., much of these basics have been to be modified. Much of my (UL's) reasoning on these modifications has been to be reported. Sorry, the present section has been blown up awfully thus and contains ramifications that may be difficult to trace. We add some

`\subsection` commands in order to cope with the new situation. (/New v4.00)

2.1 Basic code of lineno.sty `\output`

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

(New v3.04) Longtable uses `\penalty`−30000. The lineno penalty range was shrunk to −188000 . . . − 32000. (/New v3.04)

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

(UL) Hm. It is never needed below that this is a counter. `\def\linenopenalty{-100000\relax}` would do. (I guess this consumes more memory, but it is more important to save counters than to save memory.) I was frightened by `-\linenopenalty` below, but indeed TEX interprets the string `--100000` as 100000. Has any user or extension package writer ever called `\linenopenalty=xxx`, or could I really change this?—The counter is somewhat faster than the macro. Together with the compatibility question this seems to support keeping the counter. (???) Note that Stephan chose `\mathchardef` below, so his choice above seems to have been deliberate. (/UL)

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

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

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

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

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

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

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

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

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

(i) Problems with footnotes formerly resulted from LaTeX's `\@reinserts` in `\@specialoutput` which Stephan's `\linelabel` called via the `\marginpar` mechanism.

(ii) LaTeX commands using `\vadjust` formerly didn't work as one would have hoped. The problem is as follows: Printing the line number results from a box that the output routine inserts at the place of the `\interlinepenalty`. `\vadjust` items appear *above* the `\interlinepenalty` (TeXbook p. 105). So `\pagebreak`, e.g., formerly sent the line number to the next page, while the penalty from `\nopagebreak` could not tie the following line, since it was screened off by the line number box.—Our trick is putting the `\vadjust` items into a list macro from which the output routine transfers them into the vertical list, below the line number box.

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

```
\mathchardef\@Mllbcodepen=11111
\mathchardef\@Mppvacodepen=11112
```

(/New v4.00)

```
\let\@LN@output\output
\newtoks\output
\output=\expandafter{\the\@LN@output}
```

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

```
\@LN@output={%
            \LineNoTest
            \if@tempswa
```

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

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

—and of waiting `\vadjust` items:

```
              \else
                \ifnum\outputpenalty=-\@Mppvacodepen
                  \PassVadjustList
                \else
```

Now we give control back to Stephan. (/New v4.00)

```
                   \LineNoHoldInsertsTest
                   \if@tempswa
                      \if@twocolumn\let\@makecol\@LN@makecol\fi
                      \the\output
                      \ifnum\holdinginserts=-3
                        \global\holdinginserts 3
                      \fi
                   \else
                      \global\holdinginserts-3
                      \unvbox\@cclv
                      \ifnum\outputpenalty=10000\else
                        \penalty\outputpenalty
                      \fi
                   \fi
```

(New v4.00) Two new `\fi`s for the `\linelabel` and `\vadjust` tests—

```
                 \fi
               \fi
```

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

```
            \else
              \MakeLineNo
            \fi
            }
```

(New v4.00) Our new macros `\WriteLineNo` and `\PassVadjustList` will be dealt with in sections 4 and 7.1. (/New v4.00)

2.2 `\LineNoTest`

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

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

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

```
\def\LineNoTest{%
  \let\@@par\@@@par
  \ifnum\interlinepenalty<-\linenopenaltypar
     \advance\interlinepenalty-\linenopenalty
```

(UL) Following line renders previous line obsolete, doesn't it? (/UL)

```
     \my@nobreaktrue
     \fi
  \@tempswatrue
  \ifnum\outputpenalty>-\linenopenaltypar\else
     \ifnum\outputpenalty>-188000\relax
       \@tempswafalse
       \fi
     \fi
  }

\def\my@nobreaktrue{\let\if@nobreak\iftrue}
```

(UL) I would prefer `\@LN@nobreaktrue`.—I thought here were another case of the save stack problem explained in TEXbook, p. 301, namely through both local and global changing `\if@nobreak`. However, `\my@nobreak` is called during `\@LN@output` only, while `\@nobreaktrue` is called by LATEX's `\@startsection` only. The latter never happens during `\@LN@output`. So there is no local value of `\if@nobreak` on save stack when `\@nobreaktrue` acts, since `\the\@LN@output` (where `\@LN@output` is a new name for the original `\output`) is executed within a group (TEXbook p. 21). (/UL)

2.3 `\LineNoHoldInsertsTest`

(New v4.00) No change here! Just a separate subsection. (/New v4.00)

```
\def\LineNoHoldInsertsTest{%
  \ifnum\holdinginserts=3\relax
    \@tempswafalse
  \fi
  }
```

2.4 `\MakeLineNo`: Actually attach line number

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

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

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

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

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

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

TeX puts only nonzero interline penalties into the vertical list (TeXbook p. 105), while lineno.sty formerly replaced the signal interline penalty by something closing with an explicit penalty of the value that the interline penalty would have without lineno.sty. This is usually 0. Now, explicit vertical penalties can be very nasty with respect to `\nopagebreak`, e.g., a low (even positive) `\widowpenalty` may force a widow where you explicitly tried to forbid it by `\nopagebreak` (see explanation soon below). The `\nopagebreak` we create here would never work if all those zero penalties were present.—On the other hand, we cannot just omit Stephan's zero penalties, because TeX puts a penalty of 10000 after what lineno.sty inserts (TeXbook

p. 125). This penalty must be overridden to allow page breaks between ordinary lines. To revive `\nopagebreak`, we therefore replace those zero (or low) penalties by penalties that the user demanded by `\nopagebreak`.—This mechanism is not perfect and does not exactly restore the original LaTeX working of `\pagebreak` and `\nopagebreak`. Viz., if there are several vertical penalties after a line which were produced by closely sitting `\[no]pagebreak`s, without lineno.sty the lowest penalty would be effective (cf. TeXbook exercise 14.10). Our mechanism, by contrast, chooses the *last* user-set penalty of the line as the effective one. It would not be very difficult to come more close to the original mechanism, but until someone urges us we will cling to the present simple way. You may consider an advantage of the difference between our mechanism and the original one that the user here can actually override low penalties by `\nopagebreak`, which may be what a lay LaTeX user would expect.—Zero glue would do instead of zero penalty! This could make things easier. Maybe next time. (/UL, /New v4.00)

```
\def\MakeLineNo{%
   \boxmaxdepth\maxdimen\setbox\z@\vbox{\unvbox\@cclv}%
   \@tempdima\dp\z@ \unvbox\z@
   \sbox\@tempboxa{\hbox to\z@{\makeLineNumber}}%
```

(New v4.00) Previously,

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

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

(UL) I wondered about this. Why not just `\global\advance\c@linenumber\@ne`? See my reasoning in section 5. OK, I keep it. (/UL)

But then, our edtable.sty and its `longtable` option should use it as well. So use a shorthand supporting uniformity. You can even use it as a hook for choosing `\global\advance\c@linenumber\@ne` instead of our choice.

```
   \stepLineNumber
```

(/New v4.00)

```
   \dp\@tempboxa=\@tempdima\ht\@tempboxa=\z@
   \nointerlineskip\kern-\@tempdima\box\@tempboxa
```

(New v4.00) The line number has now been placed (it may be invisible depending on the modulo feature), so we can insert the `\vadjust` items. We cannot do this much later, because their right place is above the artificial interline penalty which Stephan's code will soon insert (cf. TEXbook p. 105). The next command is just `\relax` if no `\vadjust` items have been accumulated for the current line. Otherwise it is a list macro inserting the `\vadjust` items and finally resetting itself. (This is made in section 7.1 below.) If the final item is a penalty, it is stored so it can compete with other things about page breaking.

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

At this place,

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

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

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

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

```
      \xdef\@LN@parpgbrk{\penalty\number\count@\relax
        \gdef\noexpand\@LN@parpgbrk{\kern\z@}}%
```

That penalty will replace former `\kern\z@` in `\linenumberpar`, see section 3.—A few days earlier, I tried to send just a penalty value. However, the `\kern\z@` in `\linenumberpar` is crucial, as I then found out. See below.—The final penalty is repeated, but this does no harm. (It would not be very difficult to avoid the repeating, but it may even be less efficient.) It may be repeated due to the previous `\xdef`, but it may be repeated as well below in the present macro where artificial interline penalty is to be overridden.

```
    \fi
  \else
```

(/New v4.00)

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

(New v4.00)

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

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

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

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

```
   \fi
   }
```

(New v4.00)

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

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

3 Control line numbering

The line numbering is controlled via `\par`. LaTeX saved the TeX-primitive `\par` in `\@@par`. We push it one level further out, and redefine `\@@par` to insert the `\interlinepenalty` needed to trigger the line numbering. And we need to allow pagebreaks after a paragraph.

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

```
\let\@@@par\@@par
\newcount\linenoprevgraf
```

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

```
\def\linenumberpar{\ifvmode\@@@par\else\ifinner\@@@par\else
     \advance\interlinepenalty \linenopenalty
        \linenoprevgraf\prevgraf
        \global\holdinginserts3%
        \@@@par
        \ifnum\prevgraf>\linenoprevgraf
           \penalty-\linenopenaltypar
           \fi
```

(New v4.00)

```
%          \kern\z@
```

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

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

```
%         \penalty\@LN@parpgpen\relax
```

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

```
        \@LN@parpgbrk
```

After heading, `\kern\z@` resulting from previous line (see below) is followed by `\write` or `\penalty10000`, so causes no page break.

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

```
        \global\holdinginserts0%
     \advance\interlinepenalty -\linenopenalty
     \fi\fi
     }
```

(New v4.00) Initialize `\@LN@parpgbrk`:

```
\gdef\@LN@parpgbrk{\kern\z@}
```

(/New v4.00)

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

(New v4.00) We add `\ifLineNumbers` etc. since many of our new adjustments need to know whether linenumbering is active. This just provides a kind of shorthand for `\ifx\@@par\linenumberpar`; moreover it is more stable: who knows what may happen to `\@@par`?—A caveat: `\ifLineNumbers` may be wrong. E.g., it may be `\iffalse` where it acts, while a `\linenumbers` a few lines below—in the same paragraph—brings about that the line where the `\ifLineNumbers` appears gets a marginal number.

```
\newif\ifLineNumbers \LineNumbersfalse

\def\linenumbers{\LineNumberstrue
     \let\@@par\linenumberpar
```

```
% \def\linenumbers{\let\@@par\linenumberpar
```

(/New v4.00)

```
     \ifx\@par\@@@par\let\@par\linenumberpar\fi
     \ifx\par\@@@par\let\par\linenumberpar\fi
     \@ifnextchar[{\resetlinenumber}%]
                 {\@ifstar{\resetlinenumber}{}}%
     }
```

(New v4.00)

```
\def\nolinenumbers{\LineNumbersfalse
  \let\@@par\@@@par
```

```
% \def\nolinenumbers{\let\@@par\@@@par
```

(/New v4.00)

```
  \ifx\@par\linenumberpar\let\@par\@@@par\fi
  \ifx\par\linenumberpar\let\par\@@@par\fi
  }
```

(New v4.00) Moreover, it is useful to switch to `\nolinenumbers` in `\@arrayparboxrestore`. We postpone this to section 7.2 where we'll have an appending macro for doing this. (/New v4.00)

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

display math

See the subsection below, for a wrapper environment to make it work. But that requires to wrap each and every display in your LaTeX source.

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

```
\def\pagewiselinenumbers{\linenumbers\setpagewiselinenumbers}
\def\runninglinenumbers{\setrunninglinenumbers\linenumbers}
```

Finally, it is a LaTeX style, so we provide for the use of environments, including the suppression of the following paragraph's indentation.

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

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

```
\@namedef{linenumbers*}{\par\linenumbers*}
\@namedef{runninglinenumbers*}{\par\runninglinenumbers*}

\def\endlinenumbers{\par\@endpetrue}
\let\endrunninglinenumbers\endlinenumbers
\let\endpagewiselinenumbers\endlinenumbers
\expandafter\let\csname endlinenumbers*\endcsname\endlinenumbers
\expandafter\let\csname endrunninglinenumbers*\endcsname\endlinenumbers
\let\endnolinenumbers\endlinenumbers
```

3.1 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:

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

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

$$\begin{aligned} multi &\qquad line \qquad (2)\\ display &\qquad math \qquad (3)\\ \begin{array}{c} with\\ array \end{array} & \qquad\qquad (4) \end{aligned}$$

including multline displays.

First, here are two macros to turn on linenumbering on paragraphs preceeding displays, with numbering the lines of the display itself, or without. The `\ifx..` tests if line numbering is turned on. It does not harm to add these wrappers in sections that are not numbered. Nor does it harm to wrap a display twice, e.q, in case you have some `{equation}`s wrapped explicitely, and later you redefine `\equation` to do it automatically.

(UL) Newly, we could replace first lines by `\ifLineNumbers`. (/UL)

```
\newcommand\linenomathNonumbers{%
  \ifx\@@par\@@@par\else
    \ifnum\interlinepenalty>-\linenopenaltypar
      \global\holdinginserts3%
      \advance\interlinepenalty \linenopenalty
      \advance\predisplaypenalty \linenopenalty
    \fi
  \fi
  \ignorespaces
  }

\newcommand\linenomathWithnumbers{%
  \ifx\@@par\@@@par\else
    \ifnum\interlinepenalty>-\linenopenaltypar
      \global\holdinginserts3%
      \advance\interlinepenalty \linenopenalty
      \advance\predisplaypenalty \linenopenalty
      \advance\postdisplaypenalty \linenopenalty
      \advance\interdisplaylinepenalty \linenopenalty
    \fi
  \fi
  \ignorespaces
  }
```

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

```
\newcommand\linenumberdisplaymath{%
  \def\linenomath{\linenomathWithnumbers}%
  \@namedef{linenomath*}{\linenomathNonumbers}%
  }

\newcommand\nolinenumberdisplaymath{%
  \def\linenomath{\linenomathNonumbers}%
  \@namedef{linenomath*}{\linenomathWithnumbers}%
  }

\def\endlinenomath{%
   \global\holdinginserts0
   \@ignoretrue
}
\expandafter\let\csname endlinenomath*\endcsname\endlinenomath
```

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

```
\nolinenumberdisplaymath
```

4 Line number references

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

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

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

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

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

(New v4.00) We fire the `\marginpar` mechanism, so we leave LaTeX's `\@addmarginpar` untouched.

```
% \let\@LN@addmarginpar\@addmarginpar
% \def\@addmarginpar{%
%    \ifnum\count\@currbox>-2\relax
%      \expandafter\@LN@addmarginpar
%    \else
%      \@cons\@freelist\@currbox
%      \protected@write\@auxout{}{%
%          \string\newlabel
%             {\csname @LNL@\the\@currbox\endcsname}%
%             {{\theLineNumber}{\thepage}\@LN@ExtraLabelItems}}%
%    \fi}
```

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

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

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

The following is a version of LaTeX's `\@xnext`.

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

This takes an item `#1` from a list `#4` into `#3`; to be used as `\expandafter\@LN@xnext#4\@@#3#4`. Our lists use `\@lt` after each item for separating. Indeed, there will be another list macro which can appear as argument `#4`, this will be used for moving `\vadjust` items (section 7.1). The list for `\linelabel`s is the following:

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

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

```
\def\WriteLineNo{%
  \unvbox\@cclv
  \expandafter \@LN@xnext \@LN@labellist \@@
                          \@LN@label \@LN@labellist
  \protected@write\@auxout{}{\string\newlabel{\@LN@label}%
         {{\theLineNumber}{\thepage}\@LN@ExtraLabelItems}}%
}
```

(/New v4.00)

4.1 The `\linelabel` command

To refer to a place in line `\ref{`⟨*foo*⟩`}` at page `\pageref{`⟨*foo*⟩`}` you place a `\linelabel{`⟨*foo*⟩`}` at that place.

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

See if it works: This paragraph starts on page 20, line 4.

`\linelabel`

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

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

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

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

The next paragraph is no longer valid.

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

(/New v4.00)

```
\def\linelabel#1{%
   \ifvmode
       \ifinner \else
          \leavevmode \@bsphack \@savsk\p@
       \fi
   \else
       \@bsphack
   \fi
   \ifhmode
     \ifinner
       \@parmoderr
     \else
```

(New v4.00)

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

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

(/New v4.00)

```
        \@esphack
```

(New v4.00) The `\@ignorefalse` was appropriate before because the `\@Esphack` in `\end@float` set `\@ignoretrue`. Cf. LaTeX's `\@xympar`. (/New v4.00)

```
      \fi
    \else
```

(New v4.00)

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

```
%     \@parmoderr
```

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

```
    \fi
    }
```

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

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

(/New v4.00)

5 The appearance of the line numbers

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

. . . here are the hooks:

```
\def\makeLineNumberLeft{\hss\linenumberfont\LineNumber\hskip\linenumbersep}

\def\makeLineNumberRight{\linenumberfont\hskip\linenumbersep\hskip\columnwidth
                         \hbox to\linenumberwidth{\hss\LineNumber}\hss}

\def\linenumberfont{\normalfont\tiny\sffamily}

\newdimen\linenumbersep
\newdimen\linenumberwidth

\linenumberwidth=10pt
\linenumbersep=10pt
```

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

```
\def\switchlinenumbers{\@ifstar
    {\let\makeLineNumberOdd\makeLineNumberRight
     \let\makeLineNumberEven\makeLineNumberLeft}%
    {\let\makeLineNumberOdd\makeLineNumberLeft
     \let\makeLineNumberEven\makeLineNumberRight}%
    }

\def\setmakelinenumbers#1{\@ifstar
  {\let\makeLineNumberRunning#1%
   \let\makeLineNumberOdd#1%
   \let\makeLineNumberEven#1}%
  {\ifx\c@linenumber\c@runninglinenumber
      \let\makeLineNumberRunning#1%
   \else
      \let\makeLineNumberOdd#1%
      \let\makeLineNumberEven#1%
   \fi}%
  }

\def\leftlinenumbers{\setmakelinenumbers\makeLineNumberLeft}
\def\rightlinenumbers{\setmakelinenumbers\makeLineNumberRight}

\leftlinenumbers*
```

`\LineNumber` is a hook which is used for the modulo stuff. It is the command 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:

- numbers `running` through (parts of) the text
- `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 LaTeX counter, with the attached facilities serving both.

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

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.

(UL) `\newcounter{linenumber}` was the only `\newcounter` in the whole package. What is (or: "was"!) its purpose (i.e., Stephan's reasoning)? (In fact, Stephan couldn't remember his thoughts on on similar questions—this is why I reason here as if I were a historian.) Firstly, there is the check whether the name has been introduced earlier—forget about this. Secondly, `\thelinenumber` is defined—we could do this on our own. Note that `\setcounter{linenumber}` and `\addtocounter{linenumber}` work even after `\newcount\c@linenumber`, without `\newcounter`. So the final (main) difference to `\newcount` is that `\stepcounter{linenumber}` resets LaTeX counters ⟨*foo*⟩ that have been declared by `\newcounter{`⟨*foo*⟩`}[linenumber]`. I wondered what this is needed for. It reminds me of "sublines" which are dealt with in John Lavagnino's and Dominik Wujastyk's EDMAC.—This is the main reason why I think it is really better to keep the `\stepcounter` facility and, so, `\newcounter`. Finally, I found another reason for keeping it in section 5.4. (/UL)

(New v4.00)

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

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)

(UL) I thought of incrementing `\c@linenumber` (which is the same as `\c@linenumber` when `\resetlinenumber` has an effect at all) *before* printing (see what precedes `\MakeLineNo` above) and of adding `\advance\c@runninglinenumber\m@ne` here correspondingly. Even if incrementing is kept as it was: Now that we have decided to use `\stepcounter{linenumber}` for incrementing—in order to support "subordinate" counters `\c@foo` that have been introduced by `\newcounter{foo}[linenumber]`—subordinate counters should be reset here as well. This could be done as follows.

```
% \newcommand\resetlinenumber[1][1]{%
%   \ifx\c@linenumber\c@runninglinenumber
%     \c@linenumber#1\relax
%     \advance\c@linenumber\m@ne
%     \stepcounter{linenumber}%
%   \else
%     \PackageError{lineno}%
%       {You can't reset line number in pagewise mode}%
%       {This should suffice.}%
%   \fi
% }
```

But be careful! Note that `\resetlinenumber` acts locally only, while `\stepcounter` acts globally!—Well, this is a problem due to the received `\resetlinenumber`! The received situation raises the danger of misusing save stack—TEXbook p. 301. `\c@linenumber` can hardly be incremented in another way than globally! This is a very serious reason to make `\resetlinenumber` act globally!—I should have added `\global` right now, but here I am afraid of a serious compatibility problem. Stephan urged me to avoid such problems this time. Moreover, note that section 10 says that the commands can be used "globally" as well as "locally within groups". Are we allowed to change this? We might introduce a star form which acts globally indeed. Or we just advise the user to precede the command with a `\global`. (/UL)

5.1 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.

```
\def\makeRunningLineNumber{\makeLineNumberRunning}

\def\setrunninglinenumbers{%
   \def\theLineNumber{\thelinenumber}%
   \let\c@linenumber\c@runninglinenumber
   \let\makeLineNumber\makeRunningLineNumber
   }

\setrunninglinenumbers\resetlinenumber
```

5.2 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.

```
\def\setpagewiselinenumbers{%
   \let\theLineNumber\thePagewiseLineNumber
   \let\c@linenumber\c@pagewiselinenumber
   \let\makeLineNumber\makePagewiseLineNumber
   }

\def\makePagewiseLineNumber{\logtheLineNumber\getLineNumber
  \ifoddNumberedPage
     \makeLineNumberOdd
  \else
     \makeLineNumberEven
  \fi
  }
```

Each numbered line gives a line to the aux file

`\@LN{`⟨*line*⟩`}{`⟨*page*⟩`}`

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`.

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

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

```
%     \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.). (UL) 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 `\cl@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 LaTeX's `\@outputpage`.—So the macro definition ends as follows.

```
  \string\@LN{\the\c@linenumber}{%
    \noexpand\number\n@LN@truepage}}}

\newcount\n@LN@truepage
\g@addto@macro\cl@page{\global\advance\n@LN@truepage\@ne}
```

I had thought of offering more features of a LaTeX 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 `\n@LN@truepage` by `\stepcounter{page}`. E.g., she might do this twice in order to manually insert a photography. Well, the physical-page counter will skip some values then, but this will not disable pagewise line numbering.

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

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.

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

```
\def\LastNumberedPage{first}
\def\LN@Pfirst{\nextLN\relax}
```

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

```
\let\lastLN\relax  % compare to last line on this page
\let\firstLN\relax % compare to first line on this page
\let\pageLN\relax  % get the page number, compute the linenumber
\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.

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

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 ⟨*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 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.

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

```
\def\@LN#1#2{{\expandafter\@@LN
                 \csname LN@P#2C\@LN@column\expandafter\endcsname
                 \csname LN@PO#2\endcsname
                 {#1}{#2}}}

\def\@@LN#1#2#3#4{\ifx#1\relax
    \ifx#2\relax\gdef#2{#3}\fi
    \expandafter\@@@LN\csname LN@P\LastNumberedPage\endcsname#1
    \xdef#1{\lastLN{#3}\firstLN{#3}\pageLN{#4}{\@LN@column}{#2}\nextLN\relax}%
  \else
    \def\lastLN##1{\noexpand\lastLN{#3}}%
    \xdef#1{#1}%
  \fi
  \xdef\LastNumberedPage{#4C\@LN@column}}
```

The previous page macro gets its pointer to the current one, replacing the `\relax` with the cs-token `\LN@P`⟨*page*⟩.

```
\def\@@@LN#1#2{{\def\nextLN##1{\noexpand\nextLN\noexpand#2}%
                \xdef#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 cache, which is `\let` to the last page's macro. But for now it is initialized to expand `\LN@first`, where the poiner to the first numbered page has been stored in.

```
\def\NumberedPageCache{\LN@Pfirst}
```

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

```
\def\testLastNumberedPage#1{\ifnum#1<\c@linenumber
      \let\firstLN\@gobble
  \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 `\gotNumberedPage` with four arguments: the first line number on this column, the page number, the column number, and the first line on the page.

```
\def\testFirstNumberedPage#1{\ifnum#1>\c@linenumber
     \def\nextLN##1{\testNextNumberedPage\LN@Pfirst}%
  \else
      \let\nextLN\@gobble
      \def\pageLN{\gotNumberedPage{#1}}%
  \fi}
```

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

```
\long\def \@gobblethree #1#2#3{}

\def\testNumberedPage{%
  \let\lastLN\testLastNumberedPage
  \let\firstLN\testFirstNumberedPage
  \let\pageLN\@gobblethree
  \let\nextLN\testNextNumberedPage
  \NumberedPageCache
  }
```

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 TEX again, but for the rest of this run, the cache macro will just return four zeros. This saves a 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.

If there is another page, we iterate the search.

```
\def\testNextNumberedPage#1{\ifx#1\relax
     \global\def\NumberedPageCache{\gotNumberedPage0000}%
     \PackageWarningNoLine{lineno}%
                    {Linenumber reference failed,
      \MessageBreak  rerun to get it right}%
   \else
     \global\let\NumberedPageCache#1%
   \fi
   \testNumberedPage
   }
```

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 20, line 4, and back here on page 28, line 14.

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

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.

```
\newif\ifoddNumberedPage
\newif\ifcolumnwiselinenumbers
\columnwiselinenumbersfalse

\def\gotNumberedPage#1#2#3#4{\oddNumberedPagefalse
  \ifodd \if@twocolumn #3\else #2\fi\relax\oddNumberedPagetrue\fi
  \advance\c@linenumber 1\relax
  \ifcolumnwiselinenumbers
     \subtractlinenumberoffset{#1}%
  \else
     \subtractlinenumberoffset{#4}%
  \fi
  }
```

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

```
\def\runningpagewiselinenumbers{%
  \let\subtractlinenumberoffset\@gobble
  }

\def\realpagewiselinenumbers{%
  \def\subtractlinenumberoffset##1{\advance\c@linenumber-##1\relax}%
  }

\realpagewiselinenumbers
```

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:

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

And here is what happens when the label is refered to:

```
\def\getpagewiselinenumber#1{{%
  \c@linenumber #1\relax\testNumberedPage
  \thelinenumber
  }}
```

A summary of all per line expenses:

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

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.

5.3 Twocolumn mode (New v3.06)

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

```
\let\@LN@orig@makecol\@makecol
\def\@LN@makecol{%
   \@LN@orig@makecol
   \setbox\@outputbox \vbox{%
      \boxmaxdepth \@maxdepth
      \protected@write\@auxout{}{%
          \string\@LN@col{\if@firstcolumn1\else2\fi}%
      }%
      \box\@outputbox
   }% \vbox
}

\def\@LN@col#1{\def\@LN@column{#1}}
\@LN@col{1}
```

5.4 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 linenumber should be printed first. Support of this is now introduced here.—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)

```
\newcount\c@linenumbermodulo
```

(UL) On my question why, e.g., `\chardef` would not have sufficed, Stephan couldn't remember exactly; guessed that he wanted to offer LaTeX counter facilities. However, the typical ones don't come this way. So I'm quite sure that I will change this next time.

However, I observed at least two times that users gave a very high value to `\c@linenumbermodulo` in order to suppress printing of the line number. One of these users preferred an own way of handling line numbers, just wanted to use `\linelabel` and Ednotes features. Should we support this? I rather would like to advise them to `\let\makeLineNumber\relax`. (/UL)

(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
%   \divide\@tempcnta\c@linenumbermodulo
%   \multiply\@tempcnta\c@linenumbermodulo
%   \ifnum\@tempcnta=\c@linenumber\thelinenumber\fi
%   }}
```

(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 line. It could be incremented in the same way as `\n@truepage` is incremented via `\cl@page`! This is another point in favour of `{linenumber}` being a LaTeX counter! When this new counter equals `\c@linenumbermodulo`, it is 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 TeX counters still is limited.

For the same purpose, moreover, attaching the line number box could be intercepted earlier (in `\MakeLineNo`), without changing `\LineNumber`. However, 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 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 `\begingroup`/`\endgroup` here.

```
\def\themodulolinenumber{\relax
  \ifnum\c@linenumber<\n@firstlinenumber
  \else
    \begingroup
```

```
      \@tempcnta\c@linenumber
      \advance\@tempcnta-\n@firstlinenumber
      \divide\@tempcnta\c@linenumbermodulo
      \multiply\@tempcnta\c@linenumbermodulo
      \advance\@tempcnta\n@firstlinenumber
      \ifnum\@tempcnta=\c@linenumber \thelinenumber \fi
    \endgroup
  \fi
}
```

(/New v4.00)

The user command to set the modulo counter:

```
\newcommand\modulolinenumbers[1][0]{%
  \let\LineNumber\themodulolinenumber
  \ifnum#1>1\relax
    \c@linenumbermodulo#1\relax
  \else\ifnum#1=1\relax
```

```
%    \def\LineNumber{\thelinenumber}%
```

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

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

(/New v4.00)

```
  \fi\fi
  }
```

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

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

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

```
\newcommand*\firstlinenumber[1]{%
  \chardef\n@firstlinenumber#1\relax
```

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

```
  \def\@LN@ifgreat{%
    \ifnum\c@linenumber<\n@firstlinenumber
      \expandafter \@gobble
    \fi
  }%
}
```

The default is 0. This is best for what one would expect from modulo printing.

```
\let\n@firstlinenumber=\z@
```

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

```
\setcounter{linenumbermodulo}{5}
\modulolinenumbers[1]
```

6 Former package extensions

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

6.1 *displaymath*

The standard LaTeX display math environments are wrapped in a `{linenomath}` environment.

(New 3.05) The `[fleqn]` option of the standard LaTeX 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)

(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 LaTeX definition. (/New 3.08)

```
\ifx\do@mlineno\@empty
 \@ifundefined{mathindent}{

  \let\LN@displaymath\[
  \let\LN@enddisplaymath\]
  \renewcommand\[{\begin{linenomath}\LN@displaymath}
  \renewcommand\]{\LN@enddisplaymath\end{linenomath}}

  \let\LN@equation\equation
  \let\LN@endequation\endequation
  \renewenvironment{equation}
     {\linenomath\LN@equation}
     {\LN@endequation\endlinenomath}

  }% \@ifundefined{mathindent}

  \let\LN@eqnarray\eqnarray
  \let\LN@endeqnarray\endeqnarray
  \renewenvironment{eqnarray}
     {\linenomath\LN@eqnarray}
     {\LN@endeqnarray\endlinenomath}

\fi
```

(UL) Indeed. The LaTeX macros are saved for unnumbered mode, which is detected by `\linenomath`. (/UL)

6.2 Line numbers in internal vertical mode

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

```
\def\internallinenumbers{\setrunninglinenumbers
     \let\@@par\internallinenumberpar
     \ifx\@par\@@@par\let\@par\internallinenumberpar\fi
     \ifx\par\@@@par\let\par\internallinenumberpar\fi
     \ifx\@par\linenumberpar\let\@par\internallinenumberpar\fi
     \ifx\par\linenumberpar\let\par\internallinenumberpar\fi
     \@ifnextchar[{\resetlinenumber}%]
                 {\@ifstar{\let\c@linenumber\c@internallinenumber
                           \c@linenumber\@ne}{}}%
     }

\let\endinternallinenumbers\endlinenumbers
\@namedef{internallinenumbers*}{\internallinenumbers*}
\expandafter\let\csname endinternallinenumbers*\endcsname\endlinenumbers

\newcount\c@internallinenumber
```

(UL) This counter appears in `\internallinenumbers` only. It seems to have been meant to be a version of `\c@linenumber` which is changed only *locally*—see `{internallinenumbers*}`, where the initialization is local. However, Stephan incremented it *globally* then, see below. Now, even this global incrementing would not increment the `\c@linenumber` version *outside* `{internallinenumbers}`—another reason to consider `\c@internallinenumber` an “internal version” of `\c@linenumber`. And another reason not to increment it globally, see below.—A drawback of this kind of “internal” seems to be: `{internallinenumbers}` cannot be used to “continue line counting in internal vertical mode temporarily”, exactly because, e.g., `\c@runninglinenumber` has the same value after `\end{internallinenumbers}` as it had at `\begin{internallinenumbers}`. (/UL)

```
\newcount\c@internallinenumbers

\def\internallinenumberpar{\ifvmode\@@@par\else\ifinner\@@@par\else\@@@par
     \begingroup
        \c@internallinenumbers\prevgraf
        \setbox\@tempboxa\hbox{\vbox{\makeinternalLinenumbers}}%
        \dp\@tempboxa\prevdepth
        \ht\@tempboxa\z@
        \nobreak\vskip-\prevdepth
        \nointerlineskip\box\@tempboxa
     \endgroup
     \fi\fi
     }

\def\makeinternalLinenumbers{\ifnum\c@internallinenumbers>0\relax
```

(New v4.00)

```
%    \hbox to\z@{\makeLineNumber}\global\advance\c@linenumber\@ne
```

followed here previously. Why no `\stepcounter`? OK, with unit `\baselineskip` there is no space for “sublines” anyway.—More serious: `\c@linenumber` is `\c@internallinenumber` here, which in `{internallinenumbers*}`has been initialized to be 1—locally! Save stack problem again. We could use `\global` *depending* on whether the star version is used or not. However, the “external” line counter is not affected anyway, and the `\global` is not needed internally. So just drop it. I have no idea how a compatibility problem could arise.

```
   \hbox to\z@{\makeLineNumber}\advance\c@linenumber\@ne
```

(/New v4.00)

```
   \advance\c@internallinenumbers\m@ne
   \expandafter\makeinternalLinenumbers\fi
   }
```

6.3 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 knows as `rlineno.sty`.

To refer to a pagewise line number with offset:

`\linerefp[`⟨*OFFSET*⟩`]{`⟨*LABEL*⟩`}`

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:

`\lineref[`⟨*OFFSET*⟩`]{`⟨*LABEL*⟩`}`

```
\newcommand\lineref{%
  \ifx\c@linenumber\c@runninglinenumber
     \expandafter\linerefr
  \else
     \expandafter\linerefp
  \fi
}

\newcommand\linerefp[2][\z@]{{%
   \let\@thelinenumber\thelinenumber
   \edef\thelinenumber{\advance\c@linenumber#1\relax\noexpand\@thelinenumber}%
   \ref{#2}%
}}
```

This goes deep into LaTeX's internals.

```
\newcommand\linerefr[2][\z@]{{%
   \def\@@linerefadd{\advance\c@linenumber#1}%
   \expandafter\@setref\csname r@#2\endcsname
   \@linerefadd{#2}%
}}

\newcommand\@linerefadd[2]{\c@linenumber=#1\@@linerefadd\relax
                              \thelinenumber}
```

(UL) Insert 'LN' in internal command names. (/UL)

6.4 Numbered quotation environments

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.

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

\newdimen\quotelinenumbersep
\quotelinenumbersep=\linenumbersep
\let\quotelinenumberfont\linenumberfont

\newcommand\numquotelist
   {\leftlinenumbers
    \linenumbersep\quotelinenumbersep
    \let\linenumberfont\quotelinenumberfont
    \addtolength{\linenumbersep}{-\@totalleftmargin}%
    \quotelinenumbers
   }

\newenvironment{numquote}     {\quote\numquotelist}{\endquote}
\newenvironment{numquotation} {\quotation\numquotelist}{\endquotation}
\newenvironment{numquote*}    {\quote\numquotelist*}{\endquote}
\newenvironment{numquotation*}{\quotation\numquotelist*}{\endquotation}
```

6.5 Frame around a paragraph

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

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

```
\newenvironment{bframe}
  {\par
   \@tempdima\textwidth
   \advance\@tempdima 2\bframesep
   \setbox\bframebox\hbox to\textwidth{%
      \hskip-\bframesep
      \vrule\@width\bframerule\@height\baselineskip\@depth\bframesep
      \advance\@tempdima-2\bframerule
      \hskip\@tempdima
      \vrule\@width\bframerule\@height\baselineskip\@depth\bframesep
      \hskip-\bframesep
   }%
   \hbox{\hskip-\bframesep
         \vrule\@width\@tempdima\@height\bframerule\@depth\z@}%
   \nointerlineskip
   \copy\bframebox
   \nobreak
   \kern-\baselineskip
   \runninglinenumbers
   \def\makeLineNumber{\copy\bframebox\hss}%
  }
  {\par
   \kern-\prevdepth
   \kern\bframesep
   \nointerlineskip
   \@tempdima\textwidth
   \advance\@tempdima 2\bframesep
   \hbox{\hskip-\bframesep
         \vrule\@width\@tempdima\@height\bframerule\@depth\z@}%
  }

\newdimen\bframerule
\bframerule=\fboxrule

\newdimen\bframesep
\bframesep=\fboxsep

\newbox\bframebox
```

7 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 sections 2.4 and 3. 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.

According to the caveat in section 3 concerning `\ifLineNumbers`, the LaTeX commands enumerated may go wrong if you switch line numbering inside or at the end of a paragraph.

7.1 Redefining `\vadjust`

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

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

This undoes a `\global\let\vadjust\PostponeVadjust` which will start each of the refined LaTeX commands. The `\global`s 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 7.2 below. (UL)

```
  \vadjust{\penalty-\@Mppvacodepen}%
  \g@addto@macro\@LN@vadjustlist{#1\@lt}%
}
\let\@LN@@vadjust\vadjust
\global\let\@LN@vadjustlist\@empty
\global\let\@LN@do@vadjusts\relax
```

These `\global`s are just to remind that all the changes of the strings after `\let` should be `\global` (TeXbook p. 301). `\@LN@vadjustlist` collects the `\vadjust` items of a paragraph. `\PassVadjustList` tears one `\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 LaTeX `\vadjust` commands will come rather rarely. 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), so the `\box255` must be put back.

```
\def\PassVadjustList{%
  \unvbox\@cclv
  \expandafter \@LN@xnext \@LN@vadjustlist \@@
                            \@tempa \@LN@vadjustlist
  \ifx\@LN@do@vadjusts\relax
    \gdef\@LN@do@vadjusts{\global\let\@LN@do@vadjusts\relax}%
  \fi
  \expandafter \g@addto@macro \expandafter \@LN@do@vadjusts
    \expandafter {\@tempa}%
}
```

7.2 Redefining the LaTeX commands

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 LaTeX commands to be redefined. Therefore, we don't just copy macro definition code from any single implementation (say, latest LaTeX) and insert our changes, but attach a conditional `\global\let\vadjust\PostponeVadjust` to their left ends in a way which 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 near.—(UL) Sure? In line numbering mode, probably each `\vadjust` coming from a LaTeX command should be `\PostponeVadjust`. `\marginpars` and floats seem to be the only cases which are not explicitly dealt with in the present section. This would be a way to avoid `\my@nobreaktrue`! Of course, the `\vadjusts` that the present package uses then must be replaced by `\@LN@@vadjust`.—Maybe next time. (/UL)

The next command and something else will be added to the LaTeX commands we are concerned with here.

```
\DeclareRobustCommand\@LN@changevadjust{%
  \ifvmode\else\ifinner\else
    \global\let\vadjust\PostponeVadjust
  \fi\fi
}
```

(UL) What about math mode? Math display? Warn? (/UL)

`\@tempa` will now become a two place macro which adds first argument (single token), enclosed by `\ifLineNumbers`...`\fi` to the left of second argument. As long as we need it, we can't use the star form of `\DeclareRobustCommand` or the like, because AMS-LaTeX uses `\@tempa` for `\@ifstar`.

```
\def\@tempa#1#2{%
  \expandafter \def \expandafter#2\expandafter{\expandafter
    \ifLineNumbers\expandafter#1\expandafter\fi#2}
}
```

(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 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:

```
\@tempa\nolinenumbers\@arrayparboxrestore
```

We hope this suffices ... let's check one thing at least:

```
\CheckCommand*\@parboxrestore{\@arrayparboxrestore\let\\\@normalcr}
```

Now for the main theme of the section. The next lines assume that `\vspace`, `\pagebreak`, and `\nopagebreak` use `\vadjust` 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 left bracket ahead, while `\vspace` first tests for a star.

```
\@tempa\@LN@changevadjust\vspace
\@tempa\@LN@changevadjust\pagebreak
\@tempa\@LN@changevadjust\nopagebreak
```

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

```
\DeclareRobustCommand\\{%
  \ifLineNumbers
    \expandafter \@LN@cr
  \else
    \expandafter \@normalcr
  \fi
}
\def\@LN@cr{%
  \@ifstar{\@LN@changevadjust\@normalcr*}%
          {\@ifnextchar[{\@LN@changevadjust\@normalcr}\@normalcr}%
}
```

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

Let's have a few tests. Testing `\pagebreak` and `\nopagebreak` would be too expensive here, but—oops!—we have just experienced a successful `\vspace*{.5\baselineskip}`. A `\\*[.5\baselineskip]` may look even more drastical, but this time we are happy about it. Note 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 can observe it now. (/New v4.00)

7.3 Reminder on obsoleteness

(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.

```
\AtBeginDocument{%
  \let\if@LN@obsolete\iffalse
  \@ifpackageloaded{linenox0}{\let\if@LN@obsolete\iftrue}\relax
  \@ifpackageloaded{linenox1}{\let\if@LN@obsolete\iftrue}\relax
  \@ifpackageloaded{lnopatch}{\let\if@LN@obsolete\iftrue}\relax
  \if@LN@obsolete
    \PackageError{lineno}{Obsolete extension package(s)}{%
    With lineno.sty version 4.00 or later,\MessageBreak
    linenox0/linenox1/lnopatch.sty must no longer be loaded.}%
  \fi
}
```

8 Package options

(New v4.1) The last heading formerly was the heading of what is now subsection 8.3. 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 or allocations. This is different with the options that we now insert.

8.1 `\linelabel` in math mode

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

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.

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

8.2 `\linelabel` in tabular environments

We provide adjustments to make `\linelabel` work in some LaTeX tabular environments. We do this similarly as with with option `mathrefs` before. We leave code and explanations in the separate package edtable.sty. This package provides adjustments for 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 loading edtable.sty—an environment `{edtable}` which is able to change some LaTeX tabular environments with the desired effects.

This method doesn't work with longtable.sty, however. 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 as if `edtable` had been called.

Now, we are convinced that vertical spacing around `{longtable}` works wrongly—see LaTeX bugs database tools/3180 and 3485, or see explanations 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 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 in section 8.5. 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 instead, I just overlooked this. Now I don't change it because it allows to change the version requirement at one place only.)

```
\let\if@LN@edtable\iffalse

\DeclareOption{edtable}{\let\if@LN@edtable\iftrue}

\DeclareOption{longtable}{\let\if@LN@edtable\iftrue
  \PassOptionsToPackage{longtable}{edtable}}

\DeclareOption{nolongtablepatch}{%
  \PassOptionsToPackage{nolongtablepatch}{edtable}}
```

(/New v4.1)

8.3 Switch among settings

There is a bunch of package options, all of them executing only user commands (see below).

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

```
\DeclareOption{left}{\leftlinenumbers*}
```

```
\DeclareOption{right}{\rightlinenumbers*}
```

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.

```
\DeclareOption{switch}{\setpagewiselinenumbers
                       \switchlinenumbers
                       \runningpagewiselinenumbers}

\DeclareOption{switch*}{\setpagewiselinenumbers
                        \switchlinenumbers*%
                        \runningpagewiselinenumbers}
```

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.

```
\DeclareOption{columnwise}{\setpagewiselinenumbers
                           \columnwiselinenumberstrue
                           \realpagewiselinenumbers}
```

The options `pagewise` and `running` select the major linenumber 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 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 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 come after `switch`, to overide 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 `[switch,pagewise,running]`, the result is almost nothing, but if you later say `\pagewiselinenumbers`, you get margin switching, with real pagewise line numbers.

```
\DeclareOption{pagewise}{\setpagewiselinenumbers
                         \realpagewiselinenumbers}

\DeclareOption{running}{\setrunninglinenumbers}
```

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

```
\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.

```
\DeclareOption{mathlines}{\linenumberdisplaymath}
```

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

```
\let\do@mlineno\relax
\DeclareOption{displaymath}{\let\do@mlineno\@empty}
```

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.

```
\DeclareOption{hyperref}{\PackageWarningNoLine{lineno}{%
                Option [hyperref] is obsolete.
  \MessageBreak The hyperref package is detected automatically.}}

\AtBeginDocument{%
  \@ifpackageloaded{nameref}{%
    \def\@LN@ExtraLabelItems{{}{}{}}}{}}
```

(New v4.1)

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

8.5 Execute options

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

```
\ProcessOptions
```

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

```
\if@LN@edtable \RequirePackage{edtable}[2004/10/12] \fi
```

(/New v4.1)

9 The final touch

There is one deadcycle for each line number.

```
\advance\maxdeadcycles 100

\endinput
```

10 The user commands

The user commands to turn on and off line numbering are

`\linenumbers`
: Turn on line numbering in the current mode.

`\linenumbers*`
: and reset the line number to 1.

`\linenumbers[`⟨*number*⟩`]`
: and start with ⟨*number*⟩.

`\nolinenumbers`
: Turn off line numbering.

`\runninglinenumbers*[`⟨*number*⟩`]`
: Turn on `running` line numbers, with the same optional arguments as `\linenumbers`. The numbers are running through the text over pagebreaks. When you turn numbering off and on again, the numbers will continue, except, of cause, if you ask to reset or preset the counter.

`\pagewiselinenumbers`

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

`\resetlinenumber[⟨number⟩]`

Reset `[Set]` the line number to 1 `[`⟨*number*⟩`]`.

`\setrunninglinenumbers`

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

`\setpagewiselinenumbers`

Switch to `pagewise` line number mode. Do *not* turn it on or off.

`\switchlinenumbers*`

Causes margin switching in pagewise modes. With the star, put the line numbers on the inner margin.

`\leftlinenumbers*`

`\rightlinenumbers*`

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.

`\runningpagewiselinenumbers`

When using the pagewise 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.

`\realpagewiselinenumbers`

Reverses the effect of `\runningpagewiselinenumbers`.

`\modulolinenumbers[⟨number⟩]`

Give a number only to lines which are multiples of `[`⟨*number*⟩`]`. If ⟨*number*⟩ is not specified, the current value in the counter `linenumbermodulo` is retained. ⟨*number*⟩=1 turns this off without changing `linenumbermodulo`. The counter is initialized to 5.

`\firstlinenumber`

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

`\linenumberdisplaymath`

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

`\nolinenumberdisplaymath`

> Do not Number the lines of a display math in a `{linenomath}` environment, but do in a `{linenomath*}` environment. This is the default.

`\linelabel`

> Set a `\linelabel{`⟨*foo*⟩`}` to the line number where this commands is in. Refer to it with the LaTeX referencing commands `\ref{`⟨*foo*⟩`}` and `\pageref{`⟨*foo*⟩`}`.

The commands can be used globally, locally within groups or as environments. It is important to know that they take action only when the `\par` is executed. The `\end{`⟨*mode*⟩`linenumbers}` commands provide a `\par`. Examples:

```
{\linenumbers ⟨text⟩ \par}

\begin{linenumbers}
⟨text⟩
\end{linenumbers}

⟨paragraph⟩ {\linenumbers\par}

\linenumbers
⟨text⟩ \par
\nolinenumbers

\linenumbers
⟨paragraph⟩ {\nolinenumbers\par}
```

(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 7.

The same care should be applied to the "wizard" devices `\ifLineNumbers` (section 3) and `\PostponeVadjust` (section 7.1). (/New v4.00)

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
```

`\linenumbersep`

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

`\linenumberwidth`

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

`\theLineNumber` (for wizards)

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

`\makeLineNumber` (for wizards)

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

`\LineNumber` (for wizards)

This macro is called by `\makeLineNumber` to typeset the line number. This hook is changed by the modulo mechanism and by `\firstlinenumber`.