# *lineno.sty v3.09 2003/01/14*

# A LaTeX package to attach Line numbers to paragraphs

Stephan I. Böttcher

boettcher@physik.uni-kiel.de

## Contents

# 1 Introduction

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 to well with displaymath. `\footnote`s are a problem, especially when they are split, but we may get there.

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.

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

```
\NeedsTeXFormat{LaTeX2e}[1994/11/04]
\ProvidesPackage{lineno}
  [2003/01/14 line numbers on paragraphs v3.09]
```

# 2 Put the line numbers to the lines

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 `\penaly`-30000. The lineno penalty range was shrunk to $-188000 \ldots -32000$. (/New v3.04)

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

```
\let\@LN@output\output
\newtoks\output
\output=\expandafter{\the\@LN@output}
\@LN@output={%
            \LineNoTest
            \if@tempswa
              \LineNoHoldInsertsTest
              \if@tempswa
                 \if@twocolumn\let\@makecol\@LN@makecol\fi
```

```
                \the\output
                \ifnum\holdinginserts=-3
                  \global\holdinginserts 3
                \fi
              \else
                \global\holdinserts-3
                \unvbox\@cclv
                \ifnum\outputpenalty=10000\else
                  \penalty\outputpenalty
                \fi
              \fi
            \else
              \MakeLineNo
            \fi
            }
```

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

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

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

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.

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

```
\def\MakeLineNo{\@tempdima\dp\@cclv \unvbox\@cclv
   \sbox\@tempboxa{\hbox to\z@{\makeLineNumber}}%
   \stepcounter{linenumber}%
   \dp\@tempboxa=\@tempdima\ht\@tempboxa=\z@
   \nointerlineskip\kern-\@tempdima\box\@tempboxa
   \ifnum\outputpenalty=-\linenopenaltypar\else
       \@tempcnta\outputpenalty
       \advance\@tempcnta -\linenopenalty
       \penalty\@tempcnta
   \fi
   }
```

# 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

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

```
        \kern\z@
        \global\holdinginserts0%
    \advance\interlinepenalty -\linenopenalty
    \fi\fi
    }
```

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*⟩`]`.

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

\def\nolinenumbers{\let\@@par\@@@par
  \ifx\@par\linenumberpar\let\@par\@@@par\fi
  \ifx\par\linenumberpar\let\par\@@@par\fi
  }
```

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 enviroment 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.

```
\@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,

$$multi \qquad line \tag{2}$$

$$display \qquad math \tag{3}$$

$$\begin{array}{c} with \\ array \end{array} \tag{4}$$

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.

```
\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.

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)

```
\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}

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

## 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 decides 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 9, line 14.

`\linelabel`, via a fake float number −2, 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`).

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.

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

# 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 customizes `\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.

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

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

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 9, line 14, and back here on page 15, line 10.

```
\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
3 print a column tag to the .aux file.

```
322 \let\@LN@orig@makecol\@makecol
323 \def\@LN@makecol{%
324    \@LN@orig@makecol
325    \setbox\@outputbox \vbox{%
326       \boxmaxdepth \@maxdepth
327       \protected@write\@auxout{}{%
328           \string\@LN@col{\if@firstcolumn1\else2\fi}%
329       }%
330       \box\@outputbox
331    }% \vbox
332 }
333
334 \def\@LN@col#1{\def\@LN@column{#1}}
335 \@LN@col{1}
```

## 5.4 Numbering modulo 5

Most users want to have only one in five lines numbered. `\LineNumber` is
6 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.

```
336 \newcount\c@linenumbermodulo
337
338 \def\themodulolinenumber{{\@tempcnta\c@linenumber
339   \divide\@tempcnta\c@linenumbermodulo
340   \multiply\@tempcnta\c@linenumbermodulo
341   \ifnum\@tempcnta=\c@linenumber\thelinenumber\fi
342   }}
```

9 The user command to set the modulo counter:

```
343 \newcommand\modulolinenumbers[1][0]{%
344  \let\LineNumber\themodulolinenumber
345  \ifnum#1>1\relax
346    \c@linenumbermodulo#1\relax
347  \else\ifnum#1=1\relax
348    \def\LineNumber{\thelinenumber}%
349  \fi\fi
350  }
351
352 \setcounter{linenumbermodulo}{5}
353 \modulolinenumbers[1]
```

# 6 Package options

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 presencs 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{{}{}{}}{}}

\ProcessOptions
```

## 6.1 Package Extensions

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

### 6.1.1 *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
```

### 6.1.2 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
\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
   \hbox to\z@{\makeLineNumber}\global\advance\c@linenumber\@ne
   \advance\c@internallinenumbers\m@ne
   \expandafter\makeinternalLinenumbers\fi
   }
```

### 6.1.3 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@linenumner\c@runninglinenumner
     \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 LaTeXs 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}
```

### 6.1.4 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}
```

### 6.1.5 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 The final touch

There is one deadcycle for each line number.

```
\advance\maxdeadcycles 100

\endinput
```

# 8 The user commands

The user command 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 page-breaks. 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.

`\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}
```

## 8.1 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.