The poetry* package

Mark Wooding

28 May 1996

Contents

1 User guide

The poem package is designed to provide appropriate typesetting for all manner of 'sensible' poems, by which I mean not to exclude the works of such great poets as Spike Milligan, but more those who lay out their words to form pretty patterns: such works must be dealt with on an individual basis, I'm afraid.

An overview of the features provided wouldn't go amiss, I think.

- Poems are normally centred on the page based on the length of the longest line. This package handles this requirement, but allows poems to be left or right aligned if desired.
- Lines of poems are numbered, and may be labelled and referenced using the normal `\label` and `\ref` commands of LaTeX. Numbers are by default printed every 5 lines, on the right hand side, but this is fully configurable, as is the style of the numbers.
- Stanzas can be numbered, titled, either, neither or both. Stanza numbers can be labelled and referenced.

1.1 Typesetting simple poems

poem

You can typeset a poem using the poem environment. The lines of the poem are separated by \\ commands as usual. Use the \stanza* command to start new stanzas. Something like this would do the job:

TO DO

There should be a demo here

Lines of a poem will be broken if they get too long. However, a 'logical' line of a poem will never be broken between pages.[1] Continued lines are indented from the left margin by a fair distance, so that they don't get confused with the starts of new lines.

\poemline

You've probably noticed that the poem lines are numbered down the right hand side. This happens automatically, although you can turn it off if it's inappropriate. All the line numbers are generated by the command \poemline, which you can define however you like. Saying

```
\renewcommand{\poemline}{}
```

will cause nothing to be printed for the line numbers, turning them off.

TO DO

A command to disable numbering?

\title

You can use the \title command to typeset a title for your poem. The title is inserted right there and then, so watch out. It's conventional to put the title at the top of the poem, although this is art we're talking about, so who knows? Just say ⟨*title*⟩}.

\author

Similarly, the author of a poem can be credited with the \author command. Just put the author's name in the argument. Authors usually go at the bottom of poems.

\poemtitle
\poemauthor

The \title and \author commands are implemented internally by the commands \poemtitle and \poemauthor, which you can redefine if you like. You should probably have a look at the default definitions before you do this: they use some little features which haven't been described yet. Don't be intimidated, though: I'll get to them later!

1.2 Playing with stanzas

\stanza

The \stanza command is actually fairly complicated. It always starts a new stanza, leaving a gap if necessary after the previous line. Also, the stanza will be numbered, unless you use the \stanza* command. You can also give the stanza a title by saying \stanza[⟨*title*⟩] (or \stanza*... if you don't want the number). The title and number are printed above the new stanza.

\labelstanza

The stanza numbers are typeset by the command \labelstanza which you

*The poetry package is currently at version 1.00, dated 28 May 1996.

[1] This is an artifact of the way I've implemented the poems. I don't think it's a terribly nasty restriction.

can define however you like. To disable them entirely, say

```
\renewcommand{\labelstanza}{}
```

There are a collection of other style parameters for stanza titles. These are described below (if you're not interested in this sort of thing, skip to the next section).

`\tanza` is a LaTeX counter which contains the current stanza number.

`\thestanza` typesets the value of the `stanza` counter in normal text.

`\labelstanza` typesets the value of the `stanza` counter specially for use as a stanza title. (The default style uses small caps here, which is generally inappropriate in running text.)

`\stanzaname` is a command with one argument which typesets a stanza title string, as passed to the `\stanza` command (not including the number).

`\stanzacombine` is given two arguments: a title (built by `\labelstanza`) and a title (formatted by `\stanzaname`). It should format and space these two arguments. It *can't* change the font of this text – it's too late for that now. This command is only used when both a number and a stanza title are given.

`\stanzaspace` is called with no arguments. It should somehow separate the previous stanza (if any) from the new one. Look at the counter value to find out whether this is the first stanza, if it matters (e.g., you're drawing little rows of stars or something).

`\stanzatitle` is given one argument: a 'combined' title. It should typeset the title as a line in LR mode. Again, it's too late to play with fonts now.

All of the commands described above are given fairly simple definitions by default: you should be able to customise these without difficulty.

1.3 Starting new lines

`\\` New lines within a stanza are started with the `\\` command. This always starts a new line. The `\\*` command (which forbids a following page break) and the optional argument (which adds vertical space) are fully supported.

`\nl` However, there's also a command `\nl` which works like `\\` (it has a $*$-version and so on) except that it won't start a new line unless there's something already on the current one. This is useful in commands like `\poemauthor` which want to typeset their text on a new line without possibly leaving an ugly looking gap.

For example, the definition of `\poemauthor` is:

```
\providecommand{\poemauthor}[1]{%
  \nl*[\smallskipamount]%
  \nonumber%
  \hfill\normalfont\itshape#1%
  \\%
}
```

The important part to us is that `\nl*[\smallskipamount]` at the beginning. This starts a new line, making sure that there's no page break between it and the previous line, and adds a little extra space before the author's name. The `\nonumber` command just prevents this line from being numbered, since it's not actually part of the poem itself: numbering is dealt with in detail in the next section.

1.4 Line numbering

`\poemline` I skimmed over line numbering earlier, because it's a bit complex. I'll start with the default definition of the `\poemline` command, which will give me something specific to talk about. The command is used to generate the line number for the line which has *just finished*.

```
\providecommand{\poemline}{%
  \ifmultipleof{5}{\value{poemline}}%
    {\poemlineposition[r]{\scriptsize\thepoemline}}%
    {}%
  \refstepcounter{poemline}%
}
```

`\ifmultipleof` The `\ifmultipleof{5}{\value{poemline}}`... construction restricts the printed numbers to every fifth line (`\value{poemline}` is the value of the `poemline` counter). Saying `\ifmultipleof{`n`}{`x`}{`$\langle true\rangle$`}{`$\langle false\rangle$`}` will do $\langle true\rangle$ if x is a multiple of n; otherwise it does $\langle false\rangle$.

`\poemlineposition` The `\poemlineposition` command positions its text to the right or left of the poem, according to whether its optional argument is '`l`' or '`r`'.

So, the code up there just prints the poem line in small numbers on the right hand side of every fifth line of the poem. (Phew!) It then steps the counter so it'll be all right for cross-references in the next line down. Got that?

`\nonumber` Something a little simpler now: saying `\nonumber` in a line of poetry will suppress the line number on that line. The counter won't be stepped, and no number is printed. This is mainly useful in titles and other adornments in poems.

1.5 Other little extras

`xpoem` The poem environment doesn't actually do a lot by itself. If you look at its definition, you'll see that it just starts a standard LaTeX verse environment and then calls the xpoem environment to do the actual work. The idea is that you can then redefine poem to do whatever setting up you want and then use xpoem to do its typesetting magic. For example, the definitions

```
\newcommand{\poemend}{}
\renewenvironment{poem}[2]{%
  \begin{verse}%
  \renewcommand{\poemend}{\author{#2}}%
  \begin{xpoem}%
  \title{#1}%
}{%
  \poemend%
  \end{xpoem}%
  \end{verse}%
}
```

modifies the environment so that it takes two arguments, the title and the author, and sets them at the beginning and end of the poem respectively.

TEX hackers who know about such things could make a `poem` environment which 'obeys' line breaks in the input file by making active newlines do an `\nl` command. The possibilities are endless.

`\splitline` The `\splitline` command should be used at the start of a new line (it starts a new line all by itself otherwise). It shunts all the text of the line to the right so that it starts where the previous line finished.

TO DO

Come up with an example for this

2 Implementation

2.1 Various allocations

I need a shocking number of allocations for this package to work. I'll start with the counters, because they're probably the most reasonable.

`poem@count` keeps track of which poem this is, so I can look up the width in my magic list (I'll describe width handling later in detail). `poemline` is a user-level counter which keeps track of the current line number. `stanza` keeps track of the current stanza number.

The `\poemchunksize` counter (which is also faked as a LATEX counter) tells me how big a chunk should be. The final counter, `\poem@linesleft` tells me how many more lines I can do in this chunk.

All the counters are assigned globally, or at least they should be.

```
\newcounter{poem@count}
\newcounter{poemline}
\newcount\poemchunksize
\let\c@poemchunksize\poemchunksize
\newcount\poem@linesleft
\poemchunksize=30
```

Now for some length registers. `\poem@width` contains the width of the poem as read from the `.aux` file; `\poem@thiswidth` contains the width of the longest line read so far. Both of these are updated as I go through the poem. The final value of `\poem@thiswidth` is written back to the list when all's finished.

`\poem@lastwidth` contains the width of the last line – it's used in handling `\splitline`s. `\poem@prevdepth` is used to fiddle `\prevdepth` when handling long lines.

All of these length parameters should be modified globally at all times.

```
\newdimen\poem@width
\newdimen\poem@thiswidth
\newdimen\poem@lastwidth
\newdimen\poem@prevdepth
```

The switch `\ifpoem@long` is used to decide whether we need to save the poem width in the aux file.

```
\newif\ifpoem@long
```

Lastly, a skip register. This is the glue on the left hand side of a poem. It should be `\@centering` to center the poem horizontally, or something rigid and nonzero to left-align.

```
\newskip\poemleftskip
\poemleftskip\@centering
```

2.2 Handling poem widths

Poems are horizontally centred, based on the width of their longest line. This can be done without too many problems using an `\halign`. However, this would require TEX to read in the whole poem before being able to lay out the first line; this is clearly impractical for something like *The Rime of the Ancient Mariner*.

The solution is fairly similar to that used by the longtable package. I'll divide a poem up into chunks, centring each chunk horizontally. I'll also keep track of the longest line so far, and make sure that it affects each chunk, so as to prevent the chunks looking odd. When all's finished, I'll write a list containing the widths of all the poems to the `.aux` file so that next time everything will look nice.

The list is held in just one macro, which contains entries of the form `[`⟨*poem-number*⟩`]{`⟨*width*⟩`}`. I build the new updated list in another macro as I go – this version will be written to the `.aux` file at the very end, to ensure that inserted or removed poems don't mess anything up permanently. It also avoids problems to do with poem widths decreasing, which gives longtable a bit of a headache.

These two macros are always assigned globally.

```
\def\poem@widths{}
\def\poem@savedwidths{}
```

`\poem@getwidth` The width of the current poem can be read using this macro. It assigns the width to the `\poem@width` register; it gets the value 0 pt if no value for this poem actually exists.

```
\def\poem@getwidth#1{%
  \def\@tempa##1[#1]##2##3\@@{##2}%
  \global\poem@width\expandafter\@tempa\poem@savedwidths[#1]\z@\@@%
  \relax%
}
```

`\poem@setwidth` I can also write the width of the current poem using this macro. It updates the new improved list with the value of `\poem@thiswidth`.

```
\def\poem@setwidth#1{%
  \def\@tempb##1[#1]\z@{##1}%
  \def\@tempa##1[#1]##2##3\@@{%
    \xdef\poem@widths{%
      ##1%
      [#1]{\the\poem@thiswidth}%
      \ifdim##2=\z@\else\expandafter\@tempb\fi##3%
    }%
  }%
  \expandafter\@tempa\poem@widths[#1]\z@\@@%
}
```

At the very end of the document, I want to write the poem widths to the `.aux` file. The following code will do the job nicely.

```
\AtEndDocument{%
  \if@filesw%
    \immediate\write\@auxout%
      {\gdef\noexpand\poem@savedwidths{\poem@widths}}%
  \fi%
}
```

2.3 Some little details

`\@maybe@unskip` This macro solves a little problem. In an alignment (and in other places) it's desirable to suppress trailing space. The usual method, to say `\unskip`, is a little hamfisted, because it removes perfectly reasonable aligning spaces like `\hfil`s. While as a package writer I can deal with this sort of thing by saying `\kern\z@` in appropriate places, it can annoy users who are trying to use `\hfill` to override alignment in funny places.

My current solution seems to be acceptable. I'll remove the natural width of the last glue item, so that it can still stretch and shrink if necessary. The implementation makes use of the fact that multiplying a ⟨*skip*⟩ by a ⟨*number*⟩ kills off the stretch.

```
\def\@maybe@unskip{\hskip-\@ne\lastskip\relax}
```

2.4 Line numbering

Poem lines are numbered in a fairly sensible and normal way. However, it's not normal to number every single line. The macro `\poemline` below will decide whether and how to number a line.

`\ifmultipleof` This macro is called as `\ifmultipleof{`n`}{`x`}{`⟨*true*⟩`}{`⟨*false*⟩`}`. If the number x is a multiple of n, then the whole lot expands to ⟨*true*⟩; otherwise it expands to ⟨*false*⟩. The test here relies on TEX doing integer division (which it does).

```
\def\ifmultipleof#1#2{%
  \count@#2%
  \divide\count@#1%
  \multiply\count@#1%
  \relax%
  \ifnum#2=\count@%
    \expandafter\@firstoftwo%
  \else%
    \expandafter\@secondoftwo%
  \fi%
}
```

`\poemlineposition` This macro typesets its argument relative to the poem in some neat way. It's called as `\poemlineposition[`⟨*posn*⟩`]{`⟨*text*⟩`}`. The ⟨*posn*⟩ may be '`l`' or '`r`', where 'l' and 'r' mean left and right respectively.

This command only produces at all sensible results when typesetting poem line numbers.

```
\def\poemlineposition{\@ifnextchar[\poem@lp@i{\poem@lp@i[l]}}
```

Now there's some sorting out to do. If the number is to go on the right, then there's no problem: it can just be typeset as it is. Positioning on the left isn't too hard either – I just need to shift the number to the left by `\linewidth` plus a bit for niceness.

```
\def\poem@lp@i[#1]#2{%
  \if#1r%
    \hfil\kern8\p@#2%
  \else\if#1l%
    \llap{#2\kern8\p@\kern\linewidth}%
  \fi\fi%
}
```

`\poemline` The default definition of `\poemline` will put a line number in script size (so as not to appear too obvious) on every fifth line.

```
\providecommand{\poemline}{%
  \ifmultipleof{5}{\value{poemline}}%
    {\poemlineposition[r]{\scriptsize\thepoemline}}%
    {}%
  \refstepcounter{poemline}%
}
```

2.5 The main environment

`xpoem` The xpoem environment is where the nastiness really starts. Actually, the early bit is simple enough.

This environment has a funny name, so that users and style designers can define a usable 'poem' environment the way they want. Typically this will involve playing with some parameters, maybe setting up some active characters in a funny way, and probably adding a list environment to provide appropriate indentation on the left and right sides.

```
\def\xpoem{%
```

The first thing to do is to reset the line number counter.

```
  \global\c@poemline\z@%
```

Now for some hookery – the internal `\poem@printline` command will do the job of deciding whether to print a line number or not on the current line. Unless otherwise disabled, this will be equal to `\poemline`.

```
  \global\let\poem@printline\poemline%
```

The `\nonumber` command, which is also used by eqnarray,[2] suppresses numbering of the current line by changing `\poem@printline`. It will be reset by the next line end, so it only applies to a single line.

```
  \def\nonumber{\global\let\poem@printline\@empty}%
```

The `\title` and `\author` commands need redefining. I'll set these equal to some user-configurable commands below.

```
  \let\title\poemtitle%
  \let\author\poemauthor%
```

[2] Just a plug: check out the improved eqnarray environment in the mathenv package!

Do some nasty things to make lists work properly.

```
  \global\@inlabelfalse%
  \global\@newlistfalse%
```

Now it's time to start the alignment. I'll clear the `\everycr` tokens, and set up the `\\` command. I'll make `\par` expand to nothing exciting, so that blank lines in poems won't mess anything up, and set up the 'outside' meaning of `\nl`.

```
  \everycr{}%
  \let\\\poem@cr%
  \def\nl{\poem@nl}%
  \global\let\poem@nl\poem@donl%
  \let\par\@empty%
```

Now to set the widths of the poem. `\poem@width` is read from the `.aux` file from the *last* time the poem was typeset, and is used to set the width *this* time, while `\poem@thiswidth` is initially zero, and is set up as we go through *this* time, and will be used to set the actual poem width *next* time. Is that clear? No? Oh, well.

```
  \expandafter\poem@getwidth\expandafter{\the\c@poem@count}%
  \global\poem@thiswidth\z@%
  \global\poem@longfalse
```

Now some hacking to position the poem horizontally. I need to inspect the current list margins, so as to make it look right. I'll set `\dimen@` to be the size of the right hand margin.

```
  \dimen@\hsize%
  \advance\dimen@-\@totalleftmargin%
  \advance\dimen@-\linewidth%
```

Now for some silly little things before I really get going. Leave some vertical space, and step the counter ready for the first line.

```
  \bigskip%
  \stepcounter{poemline}%
  \def\@currentlabel{\p@poemline\thepoemline}%
```

Other things may want to add their declarations here. I'll provide a hook.

```
  \poem@hook%
```

Now start the first poem chunk and give control to the user.

```
  \poem@startchunk%
}
```

That's the start of the environment done; what happens at the end? Well, some fairly simple things, actually.

```
\def\endxpoem{%
```

First of all, I forcibly truncate this chunk of poem.

```
  \nl%
  \poem@endchunk%
```

Now, if the poem is longer than the chunk size, I'll add it to the new width list. If it's shorter than the chunk size, there's no need to do this, since TEX will always work out the correct width 'in time'.

```
  \ifnum\c@poemline>\poemchunksize\poem@longtrue\fi%
  \ifpoem@long%
    \expandafter\poem@setwidth\expandafter{\the\c@poem@count}%
  \fi%
```

Now I'll step the poem counter, leave a little gap, and end the environment.

```
  \global\advance\c@poem@count\@ne%
  \bigskip%
}
```

`\poem@hook` The hook used above in `\poem` starts off empty. Macro packages can add to it later.

```
\def\poem@hook{}
```

`\poem@addtohook` Packages add to that hook by saying `\poem@addtohook{`⟨*declarations*⟩`}`. This is truly trivial.

```
\def\poem@addtohook#1{%
  \expandafter\def\expandafter\poem@hook\expandafter{\poem@hook#1}%
}
```

I'll take a break from the deep hacking for a while, and implement some style things. These commands should be redefined to alter the style of the poems. (I've tried hard to make them as simple as possible.)

`\poemtitle` Poem titles are large, bold, and centred. The `\nl` command starts a new row if necessary. I want to avoid a page break after the title, for obvious reasons.

```
\providecommand{\poemtitle}[1]{%
  \nl%
  \nonumber%
  \hfill\normalfont\large\bfseries#1\hfill%
  \\*[\bigskipamount]%
}
```

`\poemauthor` Authors are typeset in italics, right aligned.

```
\providecommand{\poemauthor}[1]{%
  \nl*[\smallskipamount]%
  \nonumber%
  \hfill\normalfont\itshape#1%
  \\%
}
```

2.6 Poem chunk handling

Poems are divided into chunks to save TEX's memory. Chunks are started like this:

`\poem@startchunk`

```
\def\poem@startchunk{%
```

Reset the 'lines left' counter. When this hits zero, I end the chunk and start another one.

```
  \global\poem@linesleft\poemchunksize%
```

Now for the alignment itself. The poem is centred by tabskip glue around its first column. There are an infinite number of zero-width columns off to the right, in which the line numbers are typeset (this avoids problems if users accidentally tab over to the next column).

The 'main' column is a bit odd. It reads the text into a box, which is global to preserve save stack space, and then calls a macro `\poem@doline` to typeset the text in the box correctly.

```
  \skip@\@totalleftmargin%
  \advance\skip@\poemleftskip%
  \tabskip\skip@%
  \halign to\hsize\bgroup%
    \global\let\poem@nl\poem@cr%
    \global\setbox\@ne\hbox{{\ignorespaces##\@maybe@unskip}}\poem@doline%
    \tabskip\@centering&&%
    \poem@rightcolumn\hbox{{##}}\tabskip\dimen@\cr%
}
```

`\poem@endchunk` This is really easy. I end the line, in case it hasn't been ended already (although it should have been), and end the alignment.

```
\def\poem@endchunk{%
  \crcr%
  \noalign{\global\dimen@i\prevdepth\nointerlineskip}%
  \omit\hb@xt@\poem@width{}\cr%
  \egroup%
  \prevdepth\dimen@i%
}
```

2.7 Typesetting poem lines

`\poem@doline` This is where most of the real mess lies. Given a line of doggerel in box 1, I must typeset it beautifully.

```
\def\poem@doline{%
```

In order to know whether I need to split the line, I must know how wide the line number is. (Judging from the books I've seen, lines are allowed to encroach on the space allocated to line numbers, as long as there isn't a number on this line. Maybe as a future extension, I could decide whether it might be better to suppress this line, and maybe force a number for the next one since it won't fit here.)

Anyway, I'll do this the easy way. I'll work out the width of the line number, and subtract it from the basic line width.

```
  \dimen@\linewidth%
  \global\setbox\@labels\hbox{\poem@printline}%
  \advance\dimen@-\wd\@labels%
```

If the width of the doggerel is wider than `\dimen@`, I must split the text over more than one line, or at least I must try to. (TEX may be able to squeeze the text onto one line by shrinking the glue, so I've got to watch out for this possibility.)

```
  \ifdim\wd\@ne>\dimen@%
```

I'll now put the text in a vbox, so I can play with it. The parshape is set up so that the first line misses the line number (if there is one), while subsequent lines are indented, but take up the full available width of the page. The text is not indented (just to make sure).

The messing with `\leftskip` and the initial kern provides the indentation, and saves a little arithmetic. There is a more plausible historical reason for it too.

```
    \global\setbox\@ne\vtop{%
      \parshape\tw@ \z@\dimen@ \z@\linewidth%
      \leftskip3em%
      \noindent%
      \kern-3em%
      \unhbox\@ne%
      \@@par%
    }%
```

Since table cells are set in LR mode, the baselineskip glue will be set all wrong underneath this line. I also need to set `\poem@lastwidth` correctly. I'll copy the box to another box, and pick off the bottom line so I can peek inside.

I'll set `\poem@prevdepth` from the depth of the box (this will be set properly at the end of the line). I'll also rip that box apart, remove the `\parfillskip` glue, and rebox it in an attempt to calculate `\poem@lastwidth`. This isn't perfect, since the line might actually be shrinking instead of stretching. This is unlikely, though.

```
    \global\setbox\thr@@\vbox{%
      \unvcopy\@ne%
      \global\setbox\thr@@\lastbox%
      \global\poem@prevdepth\dp\thr@@%
      \global\setbox\thr@@\hbox{\unhbox\thr@@\unskip}%
      \global\poem@lastwidth\wd\thr@@%
    }%
```

Now that's done, I can output the box. I'll clear box 3, which I vandalised above. I also know that the line was too long, so I can set the poem widths to `\linewidth` with impunity.

```
    \box\@ne%
    \global\setbox\thr@@\box\voidb@x%
    \global\poem@width\linewidth%
    \global\poem@thiswidth\linewidth%
  \else%
```

If it fits, I can update the widths if necessary, set `\poem@lastwidth`, and spew out the text. Finally, I'll set `\poem@prevdepth` to a sentinel value meaning 'don't change'.

```
    \ifdim\wd\@ne>\poem@width\global\poem@width\wd\@ne\fi%
    \ifdim\wd\@ne>\poem@thiswidth\global\poem@thiswidth\wd\@ne\fi%
    \global\poem@lastwidth\wd\@ne%
    \unhbox\@ne\hfil%
    \global\poem@prevdepth\maxdimen%
  \fi%
}
```

2.8 Starting a new line

There are two different routes to starting new lines. The \\ command always starts a new line. The command \nl will work out if the current line hasn't been started yet, and behaves appropriately.

\poem@cr The \poem@cr macro implements the \\ command and the \nl command once a new line has been started.

First, I need to pick out the optional arguments. All the standard hacking for doing newlines in alignments appears here. If you want detailed commentary, look somewhere else – this is humdrum stuff now.

```
\def\poem@cr{%
  \relax%
  \global\let\poem@nl\poem@donl%
  \iffalse{\fi\ifnum0=`}\fi%
  \@ifstar{\poem@cr@i\@M}{\poem@cr@i\z@}%
}
\def\poem@cr@i#1{\@ifnextchar[{\poem@cr@ii{#1}}{\poem@cr@ii{#1}[\z@]}}
```

That's the standard hacking over. Here's the tricky bit.

```
\def\poem@cr@ii#1[#2]{%
  \ifnum0=`{}\fi%
```

First of all, I must clear the command which raises an error in the right hand column. Then I'll enter the column and insert the line number (which was stored in \@labels for safekeeping).

```
  \global\let\poem@rightcolumn\relax%
  &\relax%
  \llap{\unhbox\@labels}%
```

Now I'll reset the various hooks and things ready for the next like.

```
  \global\let\poem@printline\poemline%
  \global\let\poem@rightcolumn\poem@@rightcolumn%
```

Now to decide whether to start a new chunk. I'll decrement the counter, and if it reaches zero, I'll end that chunk and start a new one.

```
  \global\advance\poem@linesleft\m@ne%
  \ifnum\poem@linesleft=\z@%
    \poem@endchunk%
    \expandafter\poem@startchunk%
  \else%
    \expandafter\cr%
  \fi%
```

Finally, if I had a split line, I must change the \prevdepth setting to keep everyone happy.

```
  \noalign{%
    \addpenalty{#1}%
    \vskip#2%
    \ifdim\poem@prevdepth=\maxdimen\else\prevdepth\poem@prevdepth\fi%
  }%
}
```

\poem@donl The \poem@nl macro implements \nl during those 'in-between' times outside of a line of doggerel. This is actually spectacularly easy.

```
\def\poem@donl{%
  \noalign{\ifnum0=`}\fi%
  \@ifstar{\poem@donl@i{\addpenalty\@M}}{\poem@donl@i{}}%
}
\def\poem@donl@i#1{%
  \@ifnextchar[{\poem@donl@ii{#1}}{\poem@donl@ii{#1}[\z@]}%
}
\def\poem@donl@ii#1[#2]{%
  #1%
  \addvspace{#2}%
  \ifnum0=`{\fi}%
}
```

2.9 Other things

Well, that's all that I actually need to supply; everything else can be added over the top.

\splitline Some books appear to split lines, starting the second where the first ends. This is easy to handle with the \splitline command.

```
\def\splitline{\nl\nonumber\kern\poem@lastwidth\ }
```

\stanza New stanzas are started using the \stanza command, oddly enough. There's a problem, though: to number, or not to number? Following the example of LaTeX's sectioning commands, I'll not number if there's a following $*$. I don't really think that this is the right thing to do, since unnumbered stanzas are much more common than numbered ones. This is actually a real pain.

Anyway, if I'm going to handle numbered stanzas, I'll need a counter.

```
\newcounter{stanza}
```

Whatever happens, I'll start by adding in some vertical space above the stanza. Then I'll see if there's a following $*$. If so, step the counter and typeset the number; otherwise do nothing. However, there's a snaglet here: \@ifstar will do assignments and things, and start the next row of the alignment prematurely. I'll do the work in a \noalign to avoid problems. (Yuk.)

```
\def\stanza{%
  \nl%
  \noalign{\ifnum0=`}\fi%
  \@ifstar{%
    \stanza@i{}%
  }{%
    \stanza@i{\global\advance\c@stanza\@ne\labelstanza}%
  }%
}
```

OK. Now I have to see if there's an optional argument. I'm still safely inside that \noalign, remember.

```
\def\stanza@i#1{\@ifnextchar[{\stanza@ii{#1}}{\stanza@ii{#1}[]}}
```

I can now read the argument, and decide what actually needs to be done.

```
\def\stanza@ii#1[#2]{%
```

I want to be able to allow `\label`s inside the optional argument. However, I also want to be able to see whether the number and/or title is 'empty', bearing in mind that the title may contain just a `\label`, which shouldn't alter the spacing; which means really that I ought to put them into boxes and measure them. But this stops `\refstepcounter`'s setting of `\@currentlabel` (in the 'number' box) being noticed by the possible `\label` command in the other box. I *could* say something like

```
\refstepcounter{stanza}
\addtocounter{stanza}{-1}
```

which will do what I want, but defining `\@currentlabel` by hand is considerably easier, and more efficient.

```
\def\@currentlabel{\p@stanza\thestanza}%
\sbox\z@{#1}%
\sbox\tw@{\stanzaname{#2}}%
```

There are essentially four possibilities:

- There's nothing to typeset at all. This is easy: don't typeset anything.
- There's a number, but no title.
- There's a title, but no number.
- There's both a title *and* a number.

The tricky bit is the last possibility, since I don't know how the two will be separated. Oh, well: I'll just have to use a load of user macros.

As a first attempt, I'll put the thing to typeset into box 0. This is fairly simple. If there's a title, then I check if there's a number too: if so, I'll combine them both into box 0; otherwise I can just copy the box over. If there's anything to typeset at this point, it'll be in box 0. However, I'm currently in a `\noalign`, and that introduces a level of grouping. So I'll then move the box into box 1, which is global.

```
\ifdim\wd\tw@>\z@%
  \ifdim\wd\z@>\z@%
    \global\setbox\@ne\hbox{\stanzacombine{\unhbox\z@}{\unhbox\tw@}}%
  \else%
    \global\setbox\@ne\box\tw@%
  \fi%
\else%
  \global\setbox\@ne\hbox{\unhbox\z@\unhbox\tw@}%
\fi%
```

That's all the messy processing done. Now I can just typeset the title.

```
\ifnum0=`{\fi}%
\stanzaspace%
\ifdim\wd\@ne>\z@%
  \nonumber%
  \stanzatitle{\unhbox\@ne}%
\else
\fi%
```

That's it! I'm done.

```
236 }
```

The `stanza` counter must be reset at the beginning of the poem.

```
237 \poem@addtohook{\global\c@stanza\z@}
```

Now for some formatting defaults. This is easy stuff.

`\thestanza` Obviously, this is the default way to typeset a stanza number.

```
238 \renewcommand{\thestanza}{\Roman{stanza}}
```

`\labelstanza` This macro is responsible for giving the stanza number to be typeset in the title line.

```
239 \providecommand{\labelstanza}{\textsc{\roman{stanza}}}
```

`\stanzaname` This is responsible for typesetting the stanza's name. This is easy.

```
240 \providecommand{\stanzaname}[1]{\textsc{#1}}
```

`\stanzacombine` This is how to combine stanza numbers and names. I'll just leave a space.

```
241 \providecommand{\stanzacombine}[2]{#1\quad#2}
```

`\stanzaspace` Separate the previous stanza from a new one. This isn't done in `\stanzatitle` because there may not be a title.

```
242 \providecommand{\stanzaspace}{\nl[\medskipamount]}
```

`\stanzatitle` Finally, this is the typesetting of the stanza title in its entirety.

```
243 \providecommand{\stanzatitle}[1]{%
244   \hfill#1\hfill\\%
245 }
```

Mark Wooding, 28 May 1996

Appendix

A The GNU General Public License

The following is the text of the GNU General Public License, under the terms of which this software is distributed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions For Copying, Distribution and Modification

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   (a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   (b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   (c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

   Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   (a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   (b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   (c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

   The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

   If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do

not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

No Warranty

11. Because the program is licensed free of charge, there is no warranty for the program, to the extent permitted by applicable law. Except when otherwise stated in writing the copyright holders and/or other parties provide the program "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the program is with you. Should the program prove defective, you assume the cost of all necessary servicing, repair or correction.

12. In no event unless required by applicable law or agreed to in writing will any copyright holder, or any other party who may modify and/or redistribute the program as permitted above, be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the program (including but not limited to loss of data or data being rendered inaccurate or losses sustained by you or third parties or a failure of the program to operate with any other programs), even if such holder or other party has been advised of the possibility of such damages.

End of Terms and Conditions

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of

warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> one line to give the program's name and a brief idea of what it does.
> Copyright (C) yyyy name of author
>
> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

> Gnomovision version 69, Copyright (C) yyyy name of author
> Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
> This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w` and `show c`; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright interest in the program 'Gnomovision' (which makes passes at compilers) written by James Hacker.
>
> signature of Ty Coon, 1 April 1989
> Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Q

R

S

T

U

V

X