Glisterings

Peter Wilson

Our stars must glister with new fire, or be
To daie extinct;

The Two Noble Kinsmen, John Fletcher (and William Shakespeare?)

The aim of this column has been (see last sections) to provide odd hints or small pieces of code that might help in solving a problem or two while hopefully not making things worse through any errors of mine.

And scribbled lines like fallen hopes
On backs of tattered envelopes.

Instead of a Poet, Francis Hope

1 Reading lines

The `\input` macro reads a complete file into TeX as an atomic action. This was not what Lars Madsen needed when he posted to `ctt` wanting to be able to read a file that consisted of blocks of lines of text, where a block was ended by a blank line, and then do something with the last non-blank of the block(s). The impetus for the following was Dan Luecking's posting [6], which was one of several responses.

The basis of a solution to Lars' problem is the TeX construct

`\read` ⟨*stream*⟩ `to \mymacro`

which reads one line from the file associated with ⟨*stream*⟩ and defines `\mymacro` to be the contents of that line.

Let's start with a file of the kind that Lars is concerned with. Using the `filecontents` environment, putting the following in the preamble will, if it does not already exist, create the file `glines16.txt` which will start with four TeX comment lines written by `filecontents`, stating how and when the file was created; if the `filecontents*` environment is used instead then the initial four comment lines are not output, just the body of the environment as given [7].

```
\begin{filecontents}{glines16.txt}
This is the file glines16.txt
containing some text lines.

They come in blocks
with blank lines between.


This is the third block
consisting of
three lines.

\end{filecontents}
```

We need to set up a `\read` stream and associate it with a file to be read, making sure that the file does exist, along the lines of:

```
\newread\instream \openin\instream= qwr!?.tex
\ifeof\instream
  \message{No file 'qwr!?.tex'!^^J}
  \textbf{File 'qwr!?.tex' not found!}
\else
  \message{File 'qwr!?.tex' exists.^^J}
  \textbf{File 'qwr!?.tex' exists.}
  % do something with qwr!?.tex
\fi
\closein\instream
```

File 'qwr!?.tex' not found!

Dan's statement was that:
If you

    \def\ispar{par}

and then

    \read <handle> to \myline

you will find that

    \ifx\myline\ispar

will be true for a blank line and also true for a `\read` *taken after that last line of a file (when* `\ifeof` *is also true).*

Putting all this together, the next piece of code produces the result shown afterwards.

```
\newcommand*{\ispar}{\par}
\newcommand*{\processfile}[1]{%
  \openin\instream=#1\relax
  \ifeof\instream
    \message{No file '#1'!^^J}%
    \textbf{File '#1' not found!}%
  \else
    \message{File '#1' exists.^^J}%
    \textbf{File '#1' exists!}%
    \par\noindent
    \loop
      \let\lastline\aline
      \read\instream to \aline
    \ifeof\instream\else
    \ifx\aline\empty (commentline) \\ \else
      \ifx\aline\ispar
        \ifx\lastline\ispar
          (blankline) \\
        \else
          (lastline) \lastline (followed by)\\
          (blankline) \\
        \fi
      \else
        (aline) \aline\\
      \fi
    \fi
    \repeat
  \fi
  \closein\instream}
\processfile{glines16.txt}
```

```
    File ‘glines16.txt’ exists.
(commentline)
(commentline)
(commentline)
(commentline)
(aline) This is the file glines16.txt
(aline) containing some text lines.
(lastline) containing some text lines. (followed by)
(blankline)
(aline) They come in blocks
(aline) with blank lines between.
(lastline) with blank lines between. (followed by)
(blankline)
(blankline)
(aline) This is the third block
(aline) consisting of
(aline) three lines.
(lastline) three lines. (followed by)
(blankline)
```

When to the sessions of sweet silent thought
I summon up remembrance of things past,
I sigh the lack of many a thing I sought,
And with old woes new wail my dear times’ waste.

Sonnet 30, William Shakespeare

2 Paragraph endings

In earlier columns I described several aspects related to the typesetting of paragraphs [9, 10] and here are some additions to those.

2.1 Singletons

Andrei Alexandrescu wrote to `ctt` [1] that:
My publisher has the rule that a single word on a line should not end a paragraph, as long as reflowing wouldn’t make things really ugly otherwise. So I defined this macro:

```
\newcommand\lastwords[2]{%
  #1\leavevmode\penalty500\ \mbox{#2}}
```

and used it like this:

```
Lorem ipsum yadda \lastwords{amet}{dolor}.
```

The macro forces the last word never to be hyphenated, and imposes a penalty of 500 for inserting a line break between the first-to-last and the last word. My understanding is that 500 is the same penalty as that of a hyphen (by default).

Things work pretty well, but it turns out quite a lot of paragraphs need `\lastwords` *— a whole 188 for a 500 page book ...*

Is there an automated means to enact the rule above?

Suggestions ranged from ignoring the rule, to using existing code to make the last line at least ⟨*some length*⟩ long (see [9]), to code based on a further suggestion by Andrei and using `\everypar` that, subject to many caveats, implements the requirement.

Peter Flynn [2] suggested the macro

```
\def\E #1 #2.{ \mbox{#1}~\mbox{#2}.}
```

which would be used like:

```
 Lorem ipsum yadda \E amet dolor.
```

He observed that it was faster to type and easier to edit in than `\lastword` but noted that it might not handle arguments with embedded commands, spaces, curly braces, math, etc.

Dan Luecking [5] came up with corrections to Andrei’s second suggestion, together with an extension to handle single-word paragraphs.

```
\usepackage{ifthen}
% handle (one word) paragraph, pass others on
\def\controlorphanword #1 #2\par{%
  \ifthenelse{\equal{#2}{}}
    {#1\par}% one word para
    {\controlorphanwordtwo #1 #2\par}}
% to handle multi-word paragraph
\def\controlorphanwordtwo #1 #2 #3\par{%
  \ifthenelse{\equal{#3}{}}
    {#1\leavevmode\penalty500\ \mbox{#2}\par}
    {#1 \controlorphanwordtwo #2 #3\par}}
```

Dan noted that these macros will not handle words separated by ‘`\space`’ or ‘`\`’, nor will it work with `\obeyspaces` in effect.[1] He also commented that the process seemed very inefficient.[2] The code should be called by using `\everypar` like:

```
\begin{document}
Normal paragraph. The macro
\cs{cs}\texttt{\{arg\}} will print \cs{arg}.

Another one, and introducing \cs{everypar}.

\everypar{\controlorphanword}
Almost every place I have ever read about
\cs{everypar} (or redefinition of
\cs{par} or changes to paragraph
parameters) there is this or similar caveat:

  (La)TeX may have strange ideas what is
  counted as a paragraph. Use at your
  own risk, or turn it off in complicated
  circumstances.

Turn off orphan word control by putting
\everypar{}
here.
```

[1] For instance, using `\verb` when `\controlorphanword` is in effect will cause LaTeX to hiccup violently.

[2] The macros would be called for each word in a paragraph.

```
Turn it back on:

\everypar{\controlorphanword}
Sentence.
```

There was a general consensus among the respondents that the publisher's requirement was not particularly sensible, one going so far as to call it 'crazy'.

(Although not a solution to the problem as stated, too-short last lines can be mostly avoided in an entirely different way: `\parfillskip=.75\hsize plus.06\hsize minus.75\hsize`, with the numbers tweaked as desired, and with the usual caveats about packages resetting this primitive, etc.)

2.2 All is not what it seems

On rare occasions it may be desirable to either fake the end of a paragraph or to insert an invisible end of paragraph.

Faking the end is simple:

```
\newcommand*{\fakepar}{\\[\parskip]%
  \hspace*{\parindent}}
```

and it can be used as:

```
\ldots the end of a sentence.\fakepar
A new sentence looking as though it starts
a new paragraph\ldots
```

which will be typeset as:

. . . the end of a sentence.
A new sentence looking as though it starts a new paragraph. . .

Sometimes it is useful to nudge TEX into breaking a page, which it is inclined to do at the end of a paragraph while keeping the appearance of unbroken text. From *The TEXbook* [4, Ex. 14.15] and [12] the `\parnopar` macro accomplishes this:

```
\newcommand*{\parnopar}{{%
  \parfillskip=0pt\par\parskip=0pt\noindent}}
```

TEX typesets paragraph by paragraph, initially taking no account of any page break. Only after the text has been set in lines does TEX consider if there should be a page break within the paragraph. If you need something different about the setting on the two pages, then the original paragraph must be split at the page break.

One application is when using the changepage package [11] to temporarily change the width or location of the textblock (e.g., like the quote environment). If you are trying to extend the textwidth into, say, the outer margin, which in two-sided documents is the left margin on even pages and the right margin on odd pages and there is a page break in the shifted text then the results are not what you hoped for. This can be manually fixed using `\parnopar`, and splitting the adjustment into two.

```
\usepackage{changepage}
...
% move text 4em into outer margin
\begin{adjustwidth*}{0em}{-4em}
... first part of paragraph with the natural
page break at this point\parnopar
\end{adjustwidth*}%
\begin{adjustwidth*}{0em}{-4em}
but the sentence continues on the
following page ...
\end{adjustwidth*}
```

2.3 Paraddendum

Selon Stan posted to texhax, asking [8]:
Is there a way to fill the last line of a paragraph with leaders that extend a fixed width beyond the edge of the paragraph, with right-aligned numbers on the right? I am trying . . .

Paul Isambert [3] replied with code that I have cast into the following form:

```
\def\parend#1{%
  \leaders\hbox{\,.\,}\hfill #1\par}
Here’s a sentence.\parend{1}
Here’s a sentence \\
on two lines.\parend{291}
```

```
      Here’s a sentence. . . . . . . . . . . . . . . . . . . . 1
      Here’s a sentence
on two lines. . . . . . . . . . . . . . . . . . . . . . . . 291
```

The shades of night were falling fast,
  The rain was falling faster
When through an Alpine village passed
  An Alpine village pastor;
A youth who bore mid snow and ice
  With nary a sign of fluster
A banner with a strange device —
  'Glisterings glister with lustre'.

The Shades of Night, A.E. Housman & Peter Wilson

3 In conclusion

Some years ago, at Barbara Beeton's suggestion, I agreed to take over Jeremy Gibbons' *Hey — It works!* column which was published between 1993 and 2000 in, firstly, *TEX and TUG News*, and then later in *TUGboat*. He provided many useful tips for solving LATEX typesetting problems. Between 2000 and 2011 I managed to write some 15 columns, titled *Glisterings* as in 'All that glisters is not gold' carrying on Jeremy's work but then found that my circumstances had changed and I could no longer produce a column on a regular basis. Also, the `comp.text.tex` newsgroup from which I got most of my inspiration seemed to be fading away, being replaced by `tex.stackexchange.com` which appealed to the younger generation but not to a GOM[3] like me where many questions were directed towards problems with tikz graphics, the beamer package and 'How do I produce this'.

I wrote a final 16th column trying to wrap everything up, but the wrapping ended up being so extensive that it would have taken up most of a *TUGboat* issue, so Karl decided that it would be best to split it up into several pieces and publish these over the coming years.[4]

> You load sixteen tons and what do you get?
> Another day older and deeper in debt.
> Say brother, don' you call me 'cause I can't go
> I owe my soul to the company store.
>
> *Sixteen Tons*, MERLE TRAVIS

4 Sixteen

For what I thought would be that final 16th *Glisterings* column I wrote the following, which perhaps might still be a suitable closing.

Sixteen is a rather remarkable number in that it can be expressed in many striking ways.

- In binary sixteen is: 10000
- In octal sixteen is: 20
- In decimal sixteen is: 16
- In hexadecimal sixteen is: 10

In decimal notation, which is the one most people are familiar with, there are quite a few ways in which sixteen can be represented. Among the more eye-catching ones are:

Powers

- $4^2 = 16$
- $2^4 = 16$
- $2^{2^2} = 16$

Additions

- sum of the first $\sqrt{16}$ odd numbers: $1 + 3 + 5 + 7 = 16$
- sum of adjacent numbers: $1 + 2 + 3 + 4 + 3 + 2 + 1 = 16$ which can also be expressed as: $1 + 4 + 6 + 4 + 1 = 16$

Among other properties sixteen is the smallest number with exactly 5 divisors — 1, 2, 4, 8 and 16. It is also the only number that is expressible as both m^n and n^m, with $m \neq n$.

[3] Grumpy Old Man

[4] I don't think that either of us thought that 'coming' would turn out to be 'next six'. [Editor's note: So true.]

5 Acknowledgements

Glisterings would not have been possible without the support and input of many others. In particular I thank Jeremy Gibbons for his *Hey — It works!* and Barbara Beeton and Karl Berry for their enthusiasm and editorial improvements to the column. There are many others who also contributed, often unknowingly, by asking questions on the various TeX related mailing lists and to those who answered. With my grateful thanks to all of you.

References

[1] Andrei Alexandrescu. Single word on a line at end of paragraph. `comp.text.tex`, 26 April 2010.

[2] Peter Flynn. Re: Single word on a line at end of paragraph. `comp.text.tex`, 1 May 2010.

[3] Paul Isambert. Re: [texhax] leaders protruding a fixed width from end of paragraph? `texhax` mailing list, 17 March 2011.

[4] Donald E. Knuth. *The TeXbook*. Addison-Wesley, 1984. ISBN 0-201-13448-9.

[5] Dan Luecking. Re: Single word on a line at end of paragraph. `comp.text.tex`, 28 April 2010.

[6] Dan Luecking. Re: package for processing text from external files. `comp.text.tex`, 23 May 2011.

[7] Scott Pakin. The filecontents package, 2009. `ctan.org/pkg/filecontents`.

[8] Selon Stan. [texhax] leaders protruding a fixed width from end of paragraph? `texhax` mailing list, 17 March 2011.

[9] Peter Wilson. Glisterings: Paragraphs regular, paragraphs particular, paragraphs Russian. *TUGboat*, 28(2):229–232, 2007. `tug.org/TUGboat/tb28-2/tb89glister.pdf`.

[10] Peter Wilson. Glisterings: More on paragraphs regular, LaTeX's defining triumvirate, TeX's dictator. *TUGboat*, 29(2):324–327, 2008. `tug.org/TUGboat/tb29-2/tb92glister.pdf`.

[11] Peter Wilson. The changepage package, 2009. `ctan.org/pkg/changepage`.

[12] Peter Wilson. The memoir class for configurable typesetting, 2016. `ctan.org/pkg/memoir`.

◇ Peter Wilson
12 Sovereign Close
Kenilworth, CV8 1SQ, UK
`herries dot press (at) earthlink dot net`