Glisterings: Here or there; Parallel texts; Abort the compilation

Peter Wilson

A stately rocke beset with Diamonds faire,
And pouldred round about with Rubles red,
Where Emeralds greene doo glister in the air,
With Mantill blew of Saphyres ouer spred.

The Ship of safegarde, BARNABE GOOGE

The aim of this column is 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.

In a civil war, a general must know — and I'm afraid it's a thing rather of instinct than of practice — he must know exactly when to move over to the other side.

Not a Drum was Heard: The War Memoirs of General Gland, (unpublished radio play, 1959) HENRY REED

1 Here or there

Paul Kaletta asked on `ctt` [slightly edited]:

I am writing a twoside document which means that even and odd pages have different margins. Unfortunately all images I include are aligned with the left side of the text on every page. Some of them are broader than the line width and protrude into the right margin, which is nice for odd pages, but looks weird for even ones.

I would love to align the images to the inner margin, so that they always protrude to the outer one. Is this possible?

Heiko Oberdiek gave a solution so that an image would not exceed the width of the text plus the marginpar area [3].

This has been a problem that has cropped up from time to time on `ctt`. More generally the problem is how to decide into which margin something should be put, and then put it there. The code below for the first problem is based on code that I wrote for my memoir class. This version requires the changepage package [8] for correctly deciding whether an odd or even page is being typeset.[1]

The `\pikmargin` workhorse macro, used for specifying a margin, takes one argument which must be one of: `left`, `right`, `outer`, or `inner`. The result is `\pkmarg` which is in the range 0–3 for the allowed arguments, otherwise it is −1. The code is rather tedious.

```
\usepackage{changepage}
\newcommand*{\pikmargin}[1]{\bgroup
  \def\targ{#1}\def\parg{left}%
  \ifx\targ\parg
    \gdef\pkmarg{0}%
  \else
    \def\parg{right}%
    \ifx\targ\parg
      \gdef\pkmarg{1}%
    \else
      \def\parg{outer}%
      \ifx\targ\parg
        \gdef\pkmarg{2}%
      \else
        \def\parg{inner}%
        \ifx\targ\parg
          \gdef\pkmarg{3}%
        \else
          \gdef\pkmarg{-1}%
        \fi
      \fi
    \fi
  \fi
  \egroup}
```

The `\settheside` workhorse macro takes one argument, the value of `\pkmarg` from `\pikmargin`, and sets `\ifputatright` TRUE or FALSE according to whether material should be put into the right or left margin. The basic algorithm is:

1. A negative argument is converted to 2 (outer).
2. For two columns always the nearest margin.
3. For one sided documents:
   0 (left) FALSE
   not 0 (all else) TRUE
4. For two sided documents:
   0 (left) FALSE
   1 (right) TRUE
   2 (outer) TRUE on an odd page and FALSE on an even page
   3 (inner) FALSE on an odd page and TRUE on an even page

The code is tedious, even more so than for the previous macro.

```
\newif\ifputatright
\makeatletter
\newcommand*{\settheside}[1]{%
  \def\m@rgcode{#1}%
  \ifnum #1<0\relax
    %% error! write message and set to ‘outer’
  \typeout{Error! arg is ‘#1’. Set to ‘outer’}
    \def\m@rgcode{2}%
  \fi
```

[1] Because of the asynchronous nature of TEX's page breaking algorithm simply checking the page number does not always lead to the correct result. The changepage macros are an integral part of memoir.

```
  \if@twocolumn
    \if@firstcolumn
      \putatrightfalse
    \else
      \putatrighttrue
    \fi
  \else
    \checkoddpage%  from the changepage package
    \if@twoside
      \ifcase\m@rgcode\relax
        \putatrightfalse
      \or%    1 = left
        \putatrighttrue
      \or%    2 = outer
        \ifoddpage
          \putatrighttrue
        \else
          \putatrightfalse
        \fi
      \or%   3 = inner
        \ifoddpage
          \putatrightfalse
        \else
          \putatrighttrue
        \fi
      \fi
    \else%   1-sided
      \ifnum\m@rgcode=0\relax
        \putatrightfalse
      \else
        \putatrighttrue
      \fi
    \fi
  \fi}
\makeatother
```

You can use the \pikmargin and \settheside macros directly but in case there might be more than one kind of material to be put into the margins it is better to be conservative and use them indirectly.

With the two workhorse macros in hand, here is code for letting overwide images extend a particular distance, \ximwidth, into the margin.

\pikimagemargin is for selecting the margin for a wide image. The margin code is stored as \pkimg.

```
% \usepackage{graphicx} need this package
\newcommand*{\pikimagemargin}[1]{%
  \pikmargin{#1}%
  \ifnum \pkmarg<0\relax
    %% error! write message and set to ‘outer’
    %% or perhaps to something more appropriate
  \typeout{Error! arg is ‘#1’. Set to ‘outer’}
    \def\pkimg{2}%
  \else
    \let\pkimg\pkmarg
  \fi}
```

The next bit of code sets the maximum width for an image.

```
\newdimen\ximwidth% extra width
\newdimen\maximwidth% max total width
\makeatletter
\newcommand*{\maxiw}{%   MAX Image Width
  \ifdim\Gin@nat@width>\maximwidth
    \maximwidth
  \else
    \Gin@nat@width
  \fi}
\makeatother
```

An external image is included by calling \MaxImage which is a wrapper around the regular graphicx package \includegraphics macro and takes the same arguments, except for the optional width argument which is supplied internally.

```
\newcommand*{\MaxImage}[2][]{%
  \par\noindent
  \settheside{\pkimg}%
  \ifputatright
  \else
    \hspace{0pt minus \ximwidth}% move left
  \fi
  \includegraphics[{#1,width=\maxiw}]{#2}%
  \ifputatright
    \hspace{0pt minus \ximwidth}%
  \fi
  \par}
```

The general user scheme is:

```
%% set the dimensions
\setlength{\maximwidth}{\textwidth}
\setlength{\ximwidth}{\marginparwidth}
\addtolength{\maximwidth}{\ximwidth}
%% specify the margin (say the outer)
\pikimagemargin{outer}
...
%% image may be in a figure, but need not be
\begin{figure}
\centering
\MaxImage[height=\textheight,
          keepaspectratio]{myimage}
\caption{...}
\end{figure}
```

> To do just the opposite is also a form of imitation.
>
> *Aphorismen*, Georg Christoph Lichtenberg

2 Parallel texts

2.1 Opposites

On occasion somebody wants to set two documents in parallel on facing pages. This is typically in the form of an original in one language on even numbered pages and a translation in another language on the facing odd numbered pages. The ledpar package [10] is designed for this purpose, enabling individual line numbering and multiple footnotes on the parallel pages. But sometimes this may be overkill. Stephen Hicks [2] presented a method in response to a query on `texhax`, where it didn't matter if one of the texts was much longer than the other (if necessary the shorter text being 'completed' with blank pages). He explained his basic algorithm as:

1. *Load both documents into separate boxes (i.e., galleys)*
   ```
   \setbox\left@box\vbox\bgroup
       \input left\egroup
   \setbox\right@box\vbox\bgroup
       \input right\egroup
   ```
   This might lead to difficulties if anything in the documents have, say, `\eject` *or anything else weird re: page handling, or it might just work if the whatsits behave well inside boxes.*
2. *Alternately* `\vsplit` *off* `\textheight` *from each box and* `\unvbox` *it into the current page, followed by a* `\clearpage`*.*

Stephen's code for implementing this was as follows, except that I have made a minor change described later, and exercised some editorial privilege.

```
\documentclass{report}% or other class
...
\makeatletter
\newbox\left@box \newbox\right@box
\newenvironment{leftpage}{%
  \global\setbox\left@box\vbox\bgroup}%
  {\egroup}
\newenvironment{rightpage}{%
  \global\setbox\right@box\vbox\bgroup}%
  {\egroup}
\def\alternate{%\cleardoublepage
  \cleartostart
  \let\@next\@alternate
  \ifdim\ht\left@box=\z@\ifdim\ht\right@box=\z@
    \let\@next\relax\fi\fi
  \@next}
\def\@unvsplit#1{\ifdim\ht#1=\z@\vbox{}\else
  \setbox\z@\vsplit#1 to\textheight\unvbox\z@
  \fi}
\def\@alternate{\@unvsplit\left@box\eject
  \@unvsplit\right@box\eject\alternate}
\makeatother
...
\begin{document} ...
\begin{leftpage}
\input{lefttext}
\end{leftpage}
\begin{rightpage}
\input{righttext}
\end{rightpage}
\alternate
... \end{document}
```

As Stephen said, there are limits to what can be successfully included in the parallel texts. For example, footnotes may throw things out of kilter and page headings can get out of synch if they are changed inside either of the texts by, say, including some `\section`s.

The technical change I made was replacing the macro `\cleardoublepage` with the new one named `\cleartostart`. This is called just before the left–right printing starts. With `\cleardoublepage` the left text starts on an odd page and continues on odd pages while the right text then starts on the following even page. It seems more logical to me that the left text should start on an even numbered page, this being the left of a two page spread. The standard `\clearpage` moves to the next page, which may be odd or even, while the `\cleardoublepage` moves to the next odd page. The `\cleartostart` macro, which is based on `\cleartoevenpage` from the memoir class [9], moves to the next even page.

```
\newcommand*{\cleartostart}{\clearpage
  \ifodd\c@page\hbox{}\newpage\fi}
```

2.2 Equals

Thomas Thurman, who described himself as a poet and programmer, posted to `ctt` saying [6]:

I have a particular typesetting task, described below. Can you tell me whether it's possible in TeX without major upheaval? (Pointers as to how it's possible are welcomed, but at the moment I want to check that it's possible at all.)

I have two source documents P and Q. P consists (as you might expect) of a series of words separated by spaces and punctuation. Q consists of exactly the same number of entirely different words, but separated by the same punctuation. The words may not necessarily be the same length, but there will be the same number of them.

So P might run "I am (of course) shocked! and appalled!" and Q might run "We drink (in summer) lemonade! and Pimms!"

What I want to do is to turn P and Q into a TeX document that either:

- consists of two columns per page, the left from P and the right from Q, but on each line the number of words in each column is the same. (So if there are five words from the P column on the first line, there are five words from the Q column on the first line.)

or

- consists of pages alternately from P and Q, but for each line the number of words on that line is equal to the number of words on the same line on the facing page.

Either is a good solution. (Both would be wonderful.)

Of course if P has a run of long words then the matching Q line will contain a lot of whitespace. This is quite all right.

This resulted in a conversation between Bruno Le Floch and Jean-François Burnol ending with essentially the following code from Jean-François [1] (I have edited it slightly to better fit the two-column format). I can't explain how it works any better than what you see.

```
\makeatletter
% ======== Some helper macros
\let\xpf\expandafter
\def\addtobuff#1#2{\xpf\def\xpf#1%
                \xpf{#1 #2}}
\long\def\ifneitherempty#1#2{%
  \xpf\ifx\xpf a\detokenize{#1}a%
    \xpf\@gobble
  \else
    \xpf\ifx\xpf a\detokenize{#2}a%
      \xpf\xpf\xpf\@gobble
    \else
      \xpf\xpf\xpf\@firstofone
    \fi
  \fi}

% ======== Splitting into paragraphs

\long\def\longsbs #1#2{%
   \longsbs@aux #1\par\Q #2\par\Q}

\long\def%
\longsbs@aux #1\par#2\Q #3\par#4\Q{%
  \sidebyside{#1}{#3}% do one paragraph
  \bigskip % space between paragraphs
  % If either is empty, we're done
  % else do "\sidebyside"
  \ifneitherempty{#2}{#4}%
  {\longsbs@aux #2\Q #4\Q
  }}

% ======== Splitting at each space

\def\sbs@parse #1 #2 \Q #3 #4 \Q{%
  \sbs@step{#1}{#3}%
  % if either text is empty,
  % we are (almost) done
  % else continue
  \ifneitherempty{#2}{#4}%
  {\sbs@parse #2 \Q #4 \Q}}

% ======= Checking the size of each line
% ======= and printing it when it's ready

\newif\ifsbs@break
\def\sbs@step#1#2{%
    \setbox1=\hbox{\sbs@buffi{} #1}%
    \setbox2=\hbox{\sbs@buffii{} #2}%
    \ifdim\wd1>.4\hsize\sbs@breaktrue\else
    \ifdim\wd2>.4\hsize\sbs@breaktrue\else
    \sbs@breakfalse\fi\fi
    \ifsbs@break\sbs@writeline%
      \def\sbs@buffi{#1}%
      \def\sbs@buffii{#2}%
    \else
      \addtobuff\sbs@buffi{#1}%
      \addtobuff\sbs@buffii{#2}%
    \fi}

\def\sbs@writeline{%
    \hbox to \hsize{\hss%
         \hbox to .4\hsize{\pr@buffi}%
         \hskip.1\hsize%
         \hbox to .4\hsize{\pr@buffii}%
     \hss}}

% ========= Master function

\def\sidebyside#1#2{%
    \def\sbs@buffi{\noindent}%
    \def\sbs@buffii{\noindent}%
    \sbs@parse #1 \Q #2 \Q
    \sbs@writeline% flush the last line
}
```

I have added the following code so that the user can specify if the left and right texts are to be set flush left (`[l]`), centered (the default) or flush right (`[r]`).

```
\newcommand*{\setsbsleft}[1][c]{%
  \def\pr@buffi{\hfill\sbs@buffi\hfill}%
\def\@tempa{#1}\def\@tempb{l}
  \ifx\@tempb\@tempa
    \def\pr@buffi{\sbs@buffi\hfill}%
  \else
    \def\@tempb{r}%
    \ifx\@tempb\@tempa
      \def\pr@buffi{\hfill\sbs@buffi}%
    \fi
  \fi}
\newcommand*{\setsbsright}[1][c]{%
  \def\pr@buffii{\hfill\sbs@buffii\hfill}%
\def\@tempa{#1}\def\@tempb{l}
  \ifx\@tempb\@tempa
    \def\pr@buffii{\sbs@buffii\hfill}%
  \else
    \def\@tempb{r}%
    \ifx\@tempb\@tempa
      \def\pr@buffii{\hfill\sbs@buffii}%
    \fi
  \fi}

%% center the texts
\setsbsleft
\setsbsright
\makeatother
```

The following is a short example of using the `\longsbs` macro which, unfortunately, may have difficulties if either of its arguments includes any macros. In this case the texts are set flush right and flush left.

```
\setsbsleft[r]
\setsbsright[l]
\longsbs {%
    I am (of course) ...

    Can you tell ...
}{%
    We drink (in summer) ...

    P consists ...
}
```

I am (of course) shocked! and appalled! I have a particular typesetting task, described herein.	We drink (in summer) lemonade! and Pimms! I have two source documents P and Q.
Can you tell me whether it's possible in TeX . . . at all.	P consists (as you might expect) of a series . . . same punctuation.

Eternity's a terrible thought. I mean, where's it all going to end?

Rosencrantz and Guildenstern are Dead, Tom Stoppard

3 Abort the compilation

Rasmus Villemoes wrote to `ctt` [7]:

I have a document which is only meant to be typeset using pdflatex. It is rather large, and the first pdf-only stuff doesn't occur until quite late. So if one accidentally compiles with latex it takes a couple of minutes before the error is discovered. I would therefore like to insert some code shortly after `\documentclass` *which aborts the compilation with an error message unless running under pdflatex.*

Both Lars Madsen and Heiko Oberdiek replied and the following code is a merge and extension of their responses. The definition of `\abort` is from Heiko and following a comment by Lars I included using the ifxetex package [5] in addition to the originally suggested ifpdf package [4] as both `pdflatex` and `xelatex` generate pdf output.

```
\documentclass[...]{...}
\usepackage{ifpdf}
\usepackage{ifxetex}
\newcommand*{\abort}{}
\ifpdf\else
  \ifxetex\else
    \typeout{You must be in PDF mode.
      Use pdflatex (or xelatex) instead.}
    \def\abort{\csname @@end\endcsname}
%   or \def\abort{\stop}
  \fi
\fi
\abort
...
\begin{document}
...
```

If desired, it would be simple to recast this as a package (a `.sty` file), which is what Lars exemplified in his response.

References

[1] Jean-François Burnol. Re: Arranging parallel texts. Post to `comp.text.tex` newsgroup, 24 February 2011.

[2] Stephen Hicks. Re: [texhax] multiple documents within a document. Post to `texhax` mailing list, 30 March 2010.

[3] Heiko Oberdiek. Re: How to make all images protrude into the outer border. Post to `comp.text.tex` newsgroup, 3 January 2010.

[4] Heiko Oberdiek. The ifpdf package, April 2012. `http://mirror.ctan.org/macros/latex/contrib/oberdiek`.

[5] Will Robertson. The `ifxetex` package, 2009. `http://mirror.ctan.org/macros/generic/ifxetex`.

[6] Thomas Thurman. Arranging parallel texts. Post to `comp.text.tex` newsgroup, 22 February 2011.

[7] Rasmus Villemoes. Aborting unless running pdflatex. Post to `comp.text.tex` newsgroup, 2 August 2010.

[8] Peter Wilson. The changepage package, 2009. `http://mirror.ctan.org/macros/latex/contrib/changepage/`.

[9] Peter Wilson. The memoir class for configurable typesetting, 2013. `http://mirror.ctan.org/macros/latex/contrib/memoir`.

[10] Peter Wilson. Parallel typesetting for critical editions: The ledpar package, 2014. `http://mirror.ctan.org/macros/latex/contrib/ledmac`.

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