The \texttt{vwcol} package

Will Robertson

2015/02/10    Vol.2

1 Introduction

This package provides an environment that allows paragraph text to be typeset into multiple columns of uneven width, with text that flows from one column to the next. The columns can not span over multiple pages.

Due to difficulties with the processing of such a thing, little else \textit{besides} text is allowed within (feel free to experiment, but you’re on your own). Here’s an example:

\begin{vwcol}[widths={0.3,0.2,0.5}]
\lipsum[1]
\end{vwcol}

2 Options

As shown above, at heart this package is quite simple. This section discusses the options that can be passed to the \texttt{vwcol} environment. The options are:

- \texttt{widths} The number and size of the columns.
- \texttt{sep} The width of the space between the columns.
- \texttt{sidesep} Whether to add space on the outside of the columns (equiv. to the following two options together).
- \texttt{presep} Whether to add space before the columns.
- \texttt{postsep} Whether to add space after the columns.
- \texttt{rule} The width of the rule.

\footnote{Requires the \texttt{lipsum} package to print the sample text.}
Whether to draw a rule on the outside of the columns (equiv. to the following two options together).

prerule Whether to draw a rule before the columns.

postrule Whether to draw a rule after the columns.

rulecolor The colour of the rule.\(^2\)

justify Paragraph justification within the columns.

indent Indentation size within the columns (if relevant).

Paragraph options \texttt{justify} and \texttt{indent} are covered in section §3 on page 5, and some advanced options are discussed in section §4 on page 6.

\vwcolsetup This macro may be used to set the default values for the options (described subsequently) of the \texttt{vwcol} environment.

\begin{vwcol}
\lipsum[1]
\end{vwcol}

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque viverra.tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Pellentesque eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Pellentesque eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Pellentesque eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc.

\texttt{vwcolsetup} \texttt{widths=\{0.3,0.2,0.5\},rule=2pt}

As shown in the example in section §1, when the width ratios sum to 100% then the multi-columns will span the entire line width regardless of the chosen separation between the columns. A set of widths may be any combination of ratios and lengths, but the total width should not exceed the linewidth available (a warning will be given if so).

\texttt{vwcolsetup} \texttt{widths} This option must always be present (either as a default value previously set in \texttt{vwcolsetup} or specified in the environment directly) and consists of any number of comma-separated lengths or ratios. Lengths set the column width to an explicit size, whereas a ratio (as above) sets the column width to a fraction of the available linewidth (leaving some space for some separation between the columns).

As shown in the example in section §1, when the width ratios sum to 100% then the multi-columns will span the entire line width regardless of the chosen separation between the columns. A set of widths may be any combination of ratios and lengths, but the total width should not exceed the linewidth available (a warning will be given if so).

\texttt{vwcolsetup} \texttt{sep} The separation between the columns can be chosen as either a length, a ratio of the linewidth, or the keyword \texttt{fill}. The default is \texttt{sep=0.05} (\textit{i.e.}, 5\% of the linewidth).

\footnote{Added in vo.2.}
The keyword \textit{fill} adds stretchable space between the columns so the multi-columns fill the entire linewidth (without altering the widths of the columns themselves):

\begin{vwcol}[widths={2cm,2cm,0.4},sep=fill]
\lipsum[1]
\end{vwcol}

If ratio column widths are used with a variable separation gap, then the separation gap is considered zero for the total width calculation. In this example, because the ratios for the column widths sum to 100\% there is no room left over for a separation gap:

\begin{vwcol}[widths={0.3,0.2,0.5},sep=fill]
\lipsum[1]
\end{vwcol}

These options control whether an extra separation is added before and/or after the multicolumns. \texttt{presep} (or \texttt{presep=true}) adds space before the columns (and \texttt{presep=false} suppresses it); \texttt{postsep} adds space after the columns; \texttt{sidesep} is a shorthand for activating both at once.


rule The width of the rule is configurable (again, either a length or a ratio of the line width) and does not affect the separation gap. Use rule=none or rule=0pt to suppress drawing the rule. The default is rule=0.4pt.

prerule These options control whether extra vertical rules are added before and/or after the columns. prerule places a rule before the columns; postrule after them. (Again, prerule=false (etc.) turns this feature off.) And siderule is a shorthand to activate both. Using these options implicitly activates the relevant presep and/or postsep options, because you can’t have the rule without the gap.

3 Paragraph settings

**justify** The justification to use; one of ragged (default), flush, raggedleft, or center. These settings are made using the ragged2e package, with the result that hyphenation is enabled even in the ragged settings (this is a good thing!); due to a limitation of \TeX’s `\parshape`, \LaTeX’s ordinary `\raggedright` setting cannot be used.


Note that the first column always begins with a \noindent. Let me know if you don’t like this idea.

### 4 Advanced (read: not very useful) options

**quiet** The `vwcol` package passes certain information about what it’s doing via errors in compilation, warnings in the console output, and info in the `.log` file. Loading `vwcol` with the [quiet] option ‘demotes’ the priority of these diagnostics: errors become warnings, warnings become info in the `.log` file, and info is suppressed entirely.

**lines** With the default [lines=auto], the `vwcol` environment tries to estimate how
much space is required but it will sometimes get it wrong. Pass an integer to
the lines option to specify exactly how many lines to use (which will also save
processing time), but if the value chosen is too small then text will be lost (and
an error given):

\begin{vwcollines}{0.35,0.25,0.4,lines=4}
\lipsum[1]
\end{vwcollines}
\begin{vwcollines}{0.35,0.25,0.4,lines=11}
\lipsum[1]
\end{vwcollines}


The rationale behind producing an error is that you really want to be alerted if
text in your input is not making it into the output document (cf. with trying to
insert a character that doesn’t exist in the current font).

\begin{minipage}{0.8\linewidth}
\begin{vwcollines}{0.3,0.7,indent=1.8em}
\lipsum[66]\lipsum[66]
\end{vwcollines}
\end{minipage}

\begin{minipage}{0.8\linewidth}
\begin{vwcollines}{0.3,0.7,indent=1.8em}
\lipsum[66]\lipsum[66]
\end{vwcollines}
\end{minipage}

5 \quad \textbf{Usage notes}

If you want the widths ratios to use a different width to denote 100\% (instead of\linewidth), put the whole thing in a minipage or \parbox:

\begin{minipage}{0.8\linewidth}
\begin{vwcollines}{0.3,0.7,indent=1.8em}
\lipsum[66]\lipsum[66]
\end{vwcollines}
\end{minipage}
(I might add an option to \texttt{vwcol} to allow this directly; \textit{E.g.}, \texttt{[totalwidth=0.8/linewith]. Let me know if you like the idea.})

The \texttt{vwcol} environment ends the previous paragraph at \texttt{\begin{vwcol}} and terminates the paragraph it is contained within at \texttt{\end{vwcol}}. This means you can’t place two \texttt{vwcol} environments next to each other, for example (or next to anything else, for that matter). If you want to be able to do this, again, put them in \texttt{minipage}s or \texttt{parbox}es:

```latex
\rule{0.1\linewidth-\fboxsep}{1ex}
\fbox{
\parbox{0.8\linewidth}{\begin{vwcol}[widths={0.3,0.7},indent=1.8em]
\lipsum[66]\lipsum[66]
\end{vwcol}}}
\rule{0.1\linewidth-\fboxsep}{1ex}
\begin{minipage}{\linewidth}
\end{minipage}
```

6 \textit{Acknowledgements}

Many thanks to Flavio Costa for testing an early version of this package and especially for proof-reading this documentation. In large part due to him this manual makes much more sense :)
7 Preamble

7.1 Packages

\ProvidesPackage{vwcol}
[2015/02/10 v0.2 Variable-width multicolumn text]

\RequirePackage{calc}
\RequirePackage{color}
\RequirePackage{environ}[2008/06/18]
\RequirePackage{keyval}
\RequirePackage{ragged2e}

7.2 Things we need

\newlength\vwcol@sep
\newlength\vwcol@rule
\newlength\vwcol@totalwidth
\newlength\vwcol@averagewidth
\newlength\vwcol@parindent
\newlength\vwcol@last
\newcount\vwcol@Ncols
\newcount\vwcol@Nlines
\newcount\vwcol@maxrecursion
\newbox\vwcol@box
\newbox\vwcol@plainbox
\newbox\vwcol@outputbox
\newif\if@vwcol@boxready
\newif\if@vwcol@prerule
\newif\if@vwcol@postsep

7.3 Conveniences

Start error and warning text on a new line coz I think it looks better that way:
\newcommand\vwcol@PackageError[2]{%
\PackageError{vwcol}{^^J\space#1}{#2}}
\newcommand\vwcol@PackageWarning[1]{% 
\PackageWarning{vwcol}{% 
~'J\space\space#1~'JThis warning occurred}}
\newcommand\vwcol@PackageInfo[1]{% 
\PackageWarning{vwcol}{% 
~'J\space\space#1~'JThis warning occurred}}

7.4 Package option

\DeclareOption{quiet}{% 
\renewcommand\vwcol@PackageError[2]{% 
\vwcol@PackageWarning{#1.}}% 
\let\vwcol@PackageInfo\@gobble} 
\ProcessOptions

8 Auxiliary macros

\vwcol@test@length \{#1\}: Rational number or length (\textit{i.e.}, with unit)
\{#2\}: Multiplier for the rational (\textit{e.g.}, \texttt{\linewidth})
This macro returns \@tempswa true if the input is a rational number (\textit{e.g.}, 0.1, 1, \textit{etc.}) or false if it is a length (\textit{e.g.}, 2pt, 3cm). \@tempdima contains the length corresponding to the rational number multiplier of \#2 or the length input, respectively.

\begin{verbatim}
\def\vwcol@test@length#1#2{\afterassignment\vwcol@test@@ 
\@tempdima=#1#2\@nil}
\def\vwcol@test@@#1\@nil{\ifx\@nil#1\@nil 
\@tempswatrue \else \@tempswafalse \fi}
\end{verbatim}

Based on a similar macro by David Kastrup: \url{http://groups.google.com/group/comp.text.tex/msg/9bd5349ea2416c95}
Actually, I don’t use \if@tempswa in this package (I use \@tempdima directly), but I’ve left the conditional in there in case someone else finds it useful.

9 Environment options

\vwcolsetup To set the defaults:
\def\vwcolsetup{\setkeys{vwcol}}

widths The number and size of each column.
\define@key{vwcol}{widths}{\def\vwcol@widths{#1}}
No defaults.

maxrecursion Number of iterations used to estimate the number of lines. I doubt if it will ever need to be changed from the default.
\define@key{vwcol}{maxrecursion}{\vwcol@maxrecursion=#1}
Default:
\vwcolsetup{maxrecursion=5}

rule The width of the intercolumn rule as a length or as a ratio of the total line width or as the keyword none.
\define@key{vwcol}{rule}{%
  \def\@tempa{#1}%
  \def\@tempb{none}%
  \ifx\@tempa\@tempb
    \vwcol@rule=0pt
  \else
    \vwcol@test@length{#1}{\linewidth}%
    \vwcol@rule=\@tempdima
  \fi}
Default:
\vwcolsetup{rule=0.4pt}

lines The number of lines of text in each column or the keyword auto.
\define@key{vwcol}{lines}{%
  \def\@tempa{#1}%
  \def\@tempb{auto}%
  \ifx\@tempa\@tempb
    \vwcol@Nlines=0
  \else
    \vwcol@Nlines=#1
  \fi}
The distance between each column (including space taken up by the rule, if any) as a length or as a ratio or as the keyword fill.

Default:
\vwcolsetup{lines=auto}

sep  The distance between each column (including space taken up by the rule, if any) as a length or as a ratio or as the keyword fill.
\define@key{vwcol}{sep}{%
  \def\@tempa{#1}%
  \def\@tempb{fill}%
  \ifx\@tempa\@tempb
  \vwcol@sep=1sp
  \else
  \vwcol@test@length{#1}{\linewidth}%
  \vwcol@sep=\@tempdima
  \fi}

Default:
\vwcolsetup{sep=0.05}

presep  Whether to include a gap before the first column.
\define@key{vwcol}{presep}[true]{%
  \def\@tempa{#1}%
  \def\@tempb{true}%
  \ifx\@tempa\@tempb
  \@vwcol@preseptrue
  \else
  \def\@tempb{false}%
  \ifx\@tempa\@tempb
  \@vwcol@preseppfalse
  \else
  \vwcol@PackageWarning{%
    '#1' not a valid option for option 'presep';
    'true' or 'false' only.}%
  \fi
  \fi}

Default:
\vwcolsetup{presep=false}

postsep  Whether to include a gap after the last column.
\define@key{vwcol}{postsep}[true]{%
  \def\@tempa{#1}%
  \def\@tempb{true}%
  \ifx\@tempa\@tempb
  \@vwcol@postseptrue
  \else
  \def\@tempb{false}%
  \ifx\@tempa\@tempb
  \@vwcol@postseppfalse
  \else
  \vwcol@PackageWarning{%
    '#1' not a valid option for option 'postsep';
    'true' or 'false' only.}%
  \fi
  \fi}
else \def\@tempb{false} 
 ifx\@tempa\@tempb
 \@vwcol@postsepfalse 
 else \vwcol@PackageWarning{\%
 '##1' not a valid option for option 'postsep';
 'true' or 'false' only.\%
 \fi \fi}

Default: \vwcolsetup{postsep=false}

sidesep  Shorthand for setting both presep and postsep at once.
\define@key{vwcol}{sidesep}[true]{%
 \def\@tempa{#1}\%
 \def\@tempb{true}\%
 ifx\@tempa\@tempb
 \@vwcol@preseptrue \@vwcol@postseptrue
 else
 \def\@tempb{false}\%
 ifx\@tempa\@tempb
 \@vwcol@presepfalse \@vwcol@postsepfalse
 else \vwcol@PackageWarning{\%
 '##1' not a valid option for option 'sidesep';
 'true' or 'false' only.\%
 \fi \fi}

prerule  Whether to place a rule before the first column (implies presep).
\define@key{vwcol}{prerule}[true]{%
 \def\@tempa{#1}\%
 \def\@tempb{true}\%
 ifx\@tempa\@tempb
 \@vwcol@preseptrue \@vwcol@preruletrue
 else
 \def\@tempb{false}\%
 ifx\@tempa\@tempb
 \@vwcol@presepfalse \@vwcol@prerulefalse
 else

\vwcol@PackageWarning{%
 '#1' not a valid option for option 'prerule';
 'true' or 'false' only.}
\fi
\fi}

Default:
\vwcolsetup{prerule=false}

**postrule**  Whether to place a rule after the last column (implies postsep).
\define@key{vwcol}{postrule}{true}{%
\def\@tempa{#1}%
\def\@tempb{true}%
\ifx\@tempa\@tempb
\@vwcol@postseptrue
\@vwcol@postruletrue
\else
\def\@tempb{false}%
\ifx\@tempa\@tempb
\@vwcol@postrulefalse
\else
\vwcol@PackageWarning{%
 '#1' not a valid option for option 'postrule';
 'true' or 'false' only.}
\fi
\fi}

Default:
\vwcolsetup{postrule=false}

**siderule**  Shorthand for setting prerule and postrule simultaneously.
\define@key{vwcol}{siderule}{true}{%
\def\@tempa{#1}%
\def\@tempb{true}%
\ifx\@tempa\@tempb
\@vwcol@preseptrue
\@vwcol@postseptrue
\@vwcol@preruletrue
\@vwcol@postruletrue
\else
\def\@tempb{false}%
\ifx\@tempa\@tempb
\@vwcol@prerulefalse
\@vwcol@postrulefalse
\else
\vwcol@PackageWarning{%
 '#1' not a valid option for option 'siderule';
 'true' or 'false' only.}
\fi
\fi}

Default:
\vwcolsetup{siderule=false}
justify  The justification to use; one of flush/ragged/raggedleft/center.
\define@key{vwcol}{justify}{{% 
def\@tempa{#1}\
def\@tempb{ragged}\ifx\@tempa\@tempb \let\vwcol@justify\RaggedRight \else \def\@tempb{flush}\ifx\@tempa\@tempb \let\vwcol@justify\justifying \else \def\@tempb{raggedleft}\ifx\@tempa\@tempb \let\vwcol@justify\RaggedLeft \else \def\@tempb{center}\ifx\@tempa\@tempb \let\vwcol@justify\Centering \else \vwcol@PackageWarning{% '#1' not a valid option for option 'justify'; one of 'flush'/'ragged'/'raggedleft'/'center' only.}% 
\fi \fi \fi \fi} 

Default:
\vwcolsetup{justify=ragged}

indent  The paragraph indent to use with flush or ragged justification.
\define@key{vwcol}{indent}{{\setlength\vwcol@parindent{#1}}} 

Default:
\vwcolsetup{indent=1.5em}

rulecolor  The colour of each rule.
\define@key{vwcol}{rulecolor}{{\def\vwcol@rulecol{#1}}} 
\vwcolsetup{rulecolor=black}
No defaults.

10  \texttt{vwcol} environment definition

\texttt{vwcol} Always start a new par.

\begin{verbatim}
\NewEnviron{vwcol}{1}{\%
\par\noindent
Initialisation:
\@vwcol@boxreadyfalse
\vwcolsetup{#1}%

Ensure the space at the top of each column is uniform:
\splittopskip=\ht\ strutbox

Setup widths (this counts the columns and calculates the average and total widths of the columns):
\expandafter\vwcol@process@widths\expandafter{\vwcol@widths}%

Set up the paragraph parameters:
\vwcol@para@setup

From the width of the columns, the total width of the environment can be calculated. First, if \texttt{sep=fill} then the whole linewidth will be used:
\ifdim\vwcol@sep=1sp
\vwcol@totalwidth=\linewidth
\else
\vwcol@totalwidth=\numexpr\vwcol@totalwidth+(\vwcol@Ncols-1)*\vwcol@sep\relax sp
\fi

\texttt{else}
\vwcol@totalwidth=\numexpr\vwcol@totalwidth+(\vwcol@Ncols-1)*\vwcol@sep
\relax sp

Add on extra space due to the optional pre- and post-separation gaps and rules. Note that while rules between columns do not contribute to the total width of the columns (they subtract from the empty space in the gaps between the columns, which explains why the correction is needed in the \texttt{presep/postsep length processing}), pre- or post-rules \texttt{do}.
\if@vwcol@presep
\advance\vwcol@totalwidth\dimexpr\vwcol@sep-\vwcol@rule\relax\fi
\if@vwcol@postsep
\advance\vwcol@totalwidth\dimexpr\vwcol@sep-\vwcol@rule\relax\fi
\end{verbatim}
Finally, warn the author if their columns are going to be too large:

\ifdim\vwcol@totalwidth > \linewidth 
\vwcol@PackageWarning{%
Total width of columns plus their separations
is greater than the linewidth
(by \the\vwcol@totalwidth\space - \the\linewidth\space = \the\dimexpr\vwcol@totalwidth - \linewidth\space - \linewidth\space)}%
\fi
\ifnum\vwcol@Nlines=0%

If the lines are not explicitly selected then they must be estimated. Typeset the text into a single box of the average column width (while ignoring overfull/underfull boxes):

\@tempcnta=\hbadness 
\hbadness=\maxdimen 
\setbox\vwcol@plainbox\hbox{\parbox{\vwcol@averagewidth}{\vwcol@justify\BODY}}%
\hbadness=\@tempcnta

Now the estimate of the number of lines per column, \( L \), can be calculated. Start by assuming that the ‘area’ of the material in the single block will be about the same when split into columns of un-equal width, \( w_i \). (By ‘area’ we actually mean the number of lines in a block multiplied by the number of lines \( N \).) If \( T \) is the total number of lines of the single block typeset above (which is calculated by dividing the height of the block by the baselineskip), this gives

\[ T \times w_a \approx L \times w_1 + L \times w_2 + \cdots = L \times \sum_{i=1}^{N} w_i. \]

The width of the single block is defined above to be the average of the column widths:

\[ w_a = \text{ave}(w_i) = \frac{\sum_{i=1}^{N} w_i}{N}. \]

These two expressions are easily combined to give

\[ L = \frac{T \times \text{ave}(w_i)}{\sum_{i=1}^{N} w_i} = \frac{T}{N}. \]

In words, the number of lines per column is simply to simply the number of lines in the single block divided by the number of columns.
However, differences may arise due to rounding (due to \TeX's integer arithmetic, the floor of the resultant value is always calculated\footnote{I think.}) and hyphenation/justification variations between the two cases.

Due to these differences, we start with the calculated number of lines and increment in a loop if necessary to ensure all of the material does actually fit. It's unlikely that the number of lines estimated will be greater than the number of lines required due to the effect of the ‘flooring’ of the calculations.

Here we could keep looping for as long as necessary, but in case of weird input we put a hard limit on the number of iterations. Stop after the line number has been incremented five times (by default) because surely the calculation couldn’t have been that far wrong.

If the \texttt{lines} was chosen explicitly then just run with it, giving an error if the lines were too small. I can imagine an \texttt{approxlines} option that varies the number of
lines over a range of say, 5 lines up and down then chooses the best one, but I can’t be bothered implementing that right now.

```
\else
  \hbox to \vwcol@totalwidth{\vwcol@{\BODY}}%
  \unless\if\vwcol@boxready
    \vwcol@PackageError{%
      \par
      \par

      \vwcol@totalwidth{\vwcol@{\BODY}}%
      \unless\if\vwcol@boxready
        \vwcol@PackageError{%
          \par
          \par

          \vwcol@totalwidth{\vwcol@{\BODY}}%
      \fi
    \fi
    \fi
\fi
\fi
\vfill

That's it!
```

```
\vwcol@para@setup Set up the paragraph options.
\def\vwcol@para@setup{%
  \parindent override if justfies is ragged or flush:
  \if\vwcol@justify\RaggedRight
    \parindent=\vwcol@parindent
  \else
    \parindent=\vwcol@parindent
  \fi

  The algorithm, unfortunately, doesn’t work with non-zero \parindent:
  \def\parindent=0pt
```

```
\vwcol@process@widths This macros takes the \textwidth input and calculates the number of columns and the total and average widths of the columns.
\def\vwcol@process@widths#1{%
  \count\vwcol@Ncols=#1\do{\advance\vwcol@Ncols 1}%

  Count the number of columns: (this must be done in a loop before the main one so that \vwcol@Ncols is known first)
  \for\ii:=1\do{\advance\vwcol@Ncols 1}%
```
Based on the colsep and rule width, calculate allowable space. For stretchable column gaps, the separation gap counts as zero but the rules still take up some space:

\ifdim\vwcol@sep=1sp
\@tempdimb=\numexpr\linewidth-(\vwcol@Ncols-1)*\vwcol@rule
\relax sp
\else
\ifdim\vwcol@rule > \vwcol@sep
\vwcol@sep=\vwcol@rule
\vwcol@PackageWarning{\textquote{sep} must be greater than or equal to \textquote{rule}}%
\fi
\else
\@tempdimb=\numexpr\linewidth-(\vwcol@Ncols-1)*\vwcol@sep
\relax sp
\fi
\fi

And for fixed-width column gaps: (chuck in the warning here about sep≥rule coz it's convenient)

\else
\ifdim\vwcol@rule > \vwcol@sep
\vwcol@sep=\vwcol@rule
\vwcol@PackageWarning{\textquote{sep} must be greater than or equal to \textquote{rule}}%
\fi
\else
\@tempdimb=\numexpr\linewidth-(\vwcol@Ncols-1)*\vwcol@sep
\relax sp
\fi

Remember that the rules do not take up any space of their own between the columns, so they subtract from the white space of the separation gap; this must be mirrored when additional space is included before or after the columns:

\if\vwcol@presep
\advance\@tempdimb\dimexpr(-\vwcol@sep+\vwcol@rule)/2\relax
\fi
\if\vwcol@postsep
\advance\@tempdimb\dimexpr(-\vwcol@sep+\vwcol@rule)/2\relax
\fi
\fi

The prerule and postrule both contribute to the total width, unlike the rules between the columns:

\if\vwcol@prerule\advance\@tempdimb-\vwcol@rule\fi
\if\vwcol@postrule\advance\@tempdimb-\vwcol@rule\fi

\@tempdimb now contains the maximum width that the columns can span before the environment is wider the \linewidth, after the rules and gaps are added in too. Use this as the reference length to calculate the lengths of the columns that have widths specified as ratios.

Now iterate to do stuff:

\@for\@ii:=#1\do{%

If the column width is a plain rational number (like 0.4) then set the columnwidth to be that fraction of the allowable width.

\vwcol@test@length\@ii\@tempdimb
Keep a running total of the total width being used:
\advance\vwcol@totalwidth@tempdima

Save the column widths for later in the \parshape processing:
\expandafter\expandafter\expandafter\def\expandafter\expandafter\expandafter{\expandafter\expandafter\expandafter\vwcol@setup@parlines}{%  \expandafter\vwcol@addlines\expandafter{\the@tempdima}}}%

End the loop. Finally, calculate the average width of the columns:
\vwcol@averagewidth=\dimexpr\vwcol@totalwidth/\vwcol@Ncols \relax}

\vwcol@setup@parlines This is the macro used to locally store the setup for the \parshape line specification: (see a few lines back for the \expandafter fun of getting stuff into it)
\def\vwcol@setup@parlines{\let\vwcol@parlines@empty}

\vwcol@addlines Adds paragraph specifications to \vwcol@parlines for a single column in the \parshape. For \textit{N} columns there will be \textit{N} calls to this macro inside \vwcol@setup@parlines, which gets expanded at the beginning of every paragraph to create the required \parshape specification.

\@tempcntb starts at 0 at the beginning of each paragraph and counts the number of lines over all the columns. \vwcol@last is the total number of lines that have so far been put into the columns. \vwcol@parlines is initialised at the beginning of each paragraph.

Each time \vwcol@addlines is executed, \@tempcnta iterates through each line in that column. Once the total line count reaches the number of lines that have been typeset, \vwcol@parlines starts filling up with \parshape lines for the next paragraph.
\def\vwcol@addlines#1{%  \@tempcnta=0  \loop\ifnum\@tempcnta<\vwcol@Nlines  \advance\@tempcntb 1  \ifnum\@tempcntb>\vwcol@last  \xdef\vwcol@parlines{\vwcol@parlines 0cm #1 }%  \fi  \advance\@tempcnta 1  \repeat

\vwcol@ This is the macro for splitting the text into variable-width columns.
\newcommand\vwcol[1]{%
Setting the paragraphs  First set the text into a special box that varies width at the appropriate places so when it is split into equal segments they can be arranged into variable-width columns.
350  \setbox\vwcol@box\vbox{%
The trick is to keep a running counter of lines that we’ve gone through by inspecting every paragraph after it is typeset:
351  \def\par{\endgraf\advance\vwcol@last\the\prevgraf}%
(see \vwcol@addlines for a more detailed explanation):
352  \everypar{%
353      \@tempcntb=0
354      \vwcol@setup@parlines
355      \parshape=\numexpr \vwcol@Nlines*\vwcol@Ncols-\vwcol@last \relax
356      \vwcol@parlines}%
Insert a \strut at the top to ensure we chop off the first column at the same height as all the others:
357  \noindent\strut#1}%

Splitting the columns  First insert a pre-sep and -rule, if appropriate:
358  \if@vwcol@presep
359      \if@vwcol@prerule
360          \begingroup
361          \color{\vwcol@rulecol}
362          \vrule width \vwcol@rule
363          \endgroup
364      \fi
365  \hskip\dimexpr (\vwcol@sep-\vwcol@rule)/2 \relax
366  \fi

Iterate over the total number of columns:
367  \@tempcnta=0
368  \loop\ifnum\@tempcnta < \vwcol@Ncols
369      \advance\@tempcnta 1
370  \unless\ifnum\@tempcnta=1

Skip the separations and rules in the first case:
371  \ifnum\vwcol@sep=1sp
372      \hfill
373      \begingroup
374          \color{\vwcol@rulecol}
375          \vrule width \vwcol@rule
376      \endgroup
377      \hfill
378  \fi

Sep and rule between the columns if [sep=fill]:
379  \ifdim\vwcol@sep=1sp
380      \hfill
381      \begingroup
382          \color{\vwcol@rulecol}
383          \vrule width \vwcol@rule
384      \endgroup
385      \hfill
386  \fi

22
\else

Sep and rule between the columns if sep is a length:
\tempdima = dimexpr (vwcol@sep - vwcol@rule)/2 \relax
\hskip \tempdima
\begingroup
\color{vwcol@rulecol}
\vrule width vwcol@rule
\endgroup
\hskip \tempdima
\fi
\fi

Split off and place the text column, then loop:
\vsplit vwcol@box to \numexpr(vwcol@Nlines-1)*\baselineskip+\ht\strutbox \relax sp \repeat

Finally place the post-sep and -rule, if appropriate:
\if@vwcol@postsep
\hskip dimexpr (vwcol@sep-vwcol@rule)/2 \relax
\if@vwcol@postrule
\begingroup
\color{vwcol@rulecol}
\vrule width vwcol@rule
\endgroup
\fi
\fi
\fi

If \vwcol@box is void then we’ve used up all the material. This fact is passed on so we can re-run the algorithm with a different number of lines (or give a warning) if the material was truncated.
\ifvoid\vwcol@box
\global \vwcol@boxreadytrue
\fi}
11 Problem with \raggedright and \parshape

Check it out; when you make a \parshape with more lines specified than necessary, the linebreak of the first line is totally wrong:

\raggedright
\newlength\tmp\tmp=241.84842pt
\def\oneline{2.5em \tmp}
\def\fivelines{\oneline\oneline\oneline\oneline\oneline}
\textbf{Wrong}:\parshape 10 \fivelines\fivelines\lipsum[66]
\textbf{Right}:\parshape 7 \fivelines\oneline\oneline\lipsum[66]

Wrong:

Right:

This is why this package uses \ragged2e's \RaggedRight instead of \LaTeX's \raggedright.

In actual fact, when isolated into plain \LaTeX code this reveals a legitimate bug in \LaTeX's line-breaking algorithm. Unfortunately discovered too late in \LaTeX's life to be awarded a cheque.