An environment for multicolumn output∗†

Frank Mittelbach
Email: see top of the source file
Printed May 24, 2006

Abstract

This article describes the use and the implementation of the \texttt{multicols} environment. This environment allows switching between one and multicolumn format on the same page. Footnotes are handled correctly (for the most part), but will be placed at the bottom of the page and not under each column. \LaTeX{}’s float mechanism, however, is partly disabled in the current implementation. At the moment only page-wide floats (i.e., star-forms) can be used within the scope of the environment.

Preface to version 1.5 + 1.6

The 1.5 release contains two major changes: \texttt{multicols} will now support up to 10 columns and two more tuning possibilities have been added to the balancing routine. The balancing routine now checks the badness of the resulting columns and rejects solutions that are larger than a certain threshold. At the same time \texttt{multicols} has been upgraded to run under \LaTeX{}2ε.

Later changes to 1.5 include \texttt{\columnbreak} and \texttt{multicols*}.

For version 1.6 micro-spacing around the boxes produced by \texttt{multicols} has been improved to allow for baseline-grid typesetting.

1 Introduction

Switching between two column and one column layout is possible in \LaTeX{}, but every use of \texttt{\twocolumn} or \texttt{\onecolumn} starts a new page. Moreover, the last page of two column output isn’t balanced and this often results in an empty, or nearly empty, right column. When I started to write macros for \texttt{doc.sty} (see “The \texttt{doc}-Option”, \textit{TUGboat} volume 10 #2, pp. 245–273) I thought that it would be nice to place the index on the same page as the bibliography. And balancing the last page would not only look better, it also would save space; provided of course that it is also possible to start the next article on the same page. Rewriting the index environment was comparatively easy, but the next goal, designing an environment which takes care of footnotes, floats etc., was a harder task. It took me a whole weekend\footnote{I started with the algorithm given in the \TeX{}book on page 417. Without this help a weekend would not have been enough. (This remark was made in the documentation of the initial release, since then several hundreds more hours went into it improving the original code.)} to get together the few lines of code below and there is still a good chance that I missed something after all.

Try it and, hopefully, enjoy it; and \textit{please} direct bug reports and suggestions back to Mainz.

∗This file has version number v1.6g, last revised 2006/05/18.
†Note: This package is released under terms which affect its use in commercial applications. Please see the details at the top of the source file.
2 The User Interface

To use the environment one simply says
\begin{multicols}{(number)}
\begin{multicolumn}{1}{l}
(multicolumn text)
\end{multicols}
\end{multicols}

where \textit{(number)} is the required number of columns and \textit{(multicolumn text)} may contain arbitrary \LaTeX\ commands, except that floats and marginpars are not allowed in the current implementation\footnote{This is dictated by lack of time. To implement floats one has to reimplement the whole \LaTeX\ output routine.}. As its first action, the \texttt{multicols} environment measures the current page to determine whether there is enough room for some portion of multicolumn output. This is controlled by the \texttt{(dimen)} variable \texttt{\premulticols} which can be changed by the user with ordinary \LaTeX\ commands. If the space is less than \texttt{\premulticols}, a new page is started. Otherwise, a \texttt{\vskip} of \texttt{\multicolsep} is added.\footnote{Actually the added space may be less because we use \texttt{\addvspace} (see the \LaTeX\ manual for further information about this command).}

When the end of the \texttt{multicols} environment is encountered, an analogous mechanism is employed, but now we test whether there is a space larger than \texttt{\postmulticols} available. Again we add \texttt{\multicolsep} or start a new page.

It is often convenient to spread some text over all columns, just before the multicolumn output, without any page break in between. To achieve this the \texttt{multicols} environment has an optional second argument which can be used for this purpose. For example, the text you are now reading was started with
\begin{verbatim}
\begin{multicols}{3}
  \section{The User Interface} ...
\end{multicols}
\end{verbatim}

If such text is unusually long (or short) the value of \texttt{\premulticols} might need adjusting to prevent a bad page break. We therefore provide a third argument which can be used to overwrite the default value of \texttt{\premulticols} just for this occasion. So if you want to combine some longer single column text with a \texttt{multicols} environment you could write
\begin{verbatim}
\begin{multicols}{3}
  \section{Index}
  This index contains ...
  \[6cm]\]
  ...
\end{verbatim}

The space between columns is controlled by the length parameter \texttt{\columnseprule}. The width for the individual columns is automatically calculated from this parameter and the current \texttt{\linewidth}. In this article a value of \texttt{18.0pt} was used.

Separation of columns with vertical rules is achieved by setting the parameter \texttt{\columnseprule} to some positive value. In this article a value of \texttt{.4pt} was used.

Since narrow columns tend to need adjustments in interline spacing we also provide a \texttt{(skip)} parameter called \texttt{\multicolbaselineskip} which is added to the \texttt{\baselineskip} parameter inside the \texttt{multicols} environment. Please use this parameter with care or leave it alone; it is intended only for package file designers since even small changes might produce totally unexpected changes to your document.

2.1 Balancing columns

Besides the previously mentioned parameters, some others are provided to influence the layout of the columns generated.

Paragraphing in \LaTeX\ is controlled by several parameters. One of the most important is called \texttt{\tolerance}: this controls the allowed ‘looseness’ (i.e. the amount of blank space between words). Its default value is \texttt{200} (the \LaTeX\ \texttt{\fussy}) which is too small for narrow columns. On the other hand the \texttt{\sloppy} declaration (which sets \texttt{\tolerance} to \texttt{10000} = \texttt{\infty}) is too large, allowing really bad spacing.\footnote{Look at the next paragraph, it was set with the \texttt{\sloppy} declaration.}

We therefore use a \texttt{\multicoltolerance} parameter for the \texttt{\tolerance} value inside the \texttt{multicols} environment. Its default value is \texttt{9999} which is less than infinity but ‘bad’ enough for most paragraphs in a multicolumn environment. Changing its value should be done outside the \texttt{multicols} environment. Since \texttt{\tolerance} is set to \texttt{\multicoltolerance} at the beginning of every \texttt{multicols} environment one can locally overwrite this default by assigning \texttt{\tolerance} = \texttt{\multicoltolerance} (desired value). There also exists a \texttt{\multicolpretolerance} parameter holding the value for \texttt{\pretolerance} within a \texttt{multicols} environment. Both parameters are usually used only by package designers.

Generation of multicolumn output can be divided into two parts. In the first part we are collecting material for a page, shipping it out, collecting material for the next page, and so on.
As a second step, balancing will be done when the end of the \multicols environment is reached. In the first step \TeX might consider more material whilst finding the final columns than it actually use when shipping out the page. This might cause a problem if a footnote is encountered in the part of the input considered, but not used, on the current page. In this case the footnote might show up on the current page, while the footnotemark corresponding to this footnote might be set on the next one.\footnote{The reason behind this behavior is the asynchronous character of the \TeX page\_builder. However, this could be avoided by defining very complicated output routines which don’t use \TeX primitives like \verb|\insert| but do everything by hand. This is clearly beyond the scope of a weekend problem.} Therefore the \multicols environment gives a warning message\footnote{This message will be generated even if there are no footnotes in this part of the text.} whenever it is unable to use all the material considered so far.

If you don’t use footnotes too often the chances of something actually going wrong are very slim, but if this happens you can help \TeX by using a \texttt{\pagebreak} command in the final document. Another way to influence the behavior of \TeX in this respect is given by the counter variable ‘collectmore’. If you use the \texttt{\setcounter} declaration to set this counter to \langle \text{number} \rangle, \TeX will consider \langle \text{number} \rangle more (or less) lines before making its final decision. So a value of \texttt{-1} may solve all your problems at the cost of slightly less optimal columns.

In the second step (balancing columns) we have other bells and whistles. First of all you can say \texttt{\raggedcolumns} if you don’t want the bottom lines to be aligned. The default is \texttt{\flushcolumns}, so \TeX will normally try to make both the top and bottom baselines of all columns align.

Additionally you can set another counter, the ‘unbalance’ counter, to some positive \langle \text{number} \rangle. This will make all but the right-most column \langle \text{number} \rangle of lines longer than they would normally have been. ‘Lines’ in this context refer to normal text lines (i.e. one \texttt{\baselineskip}\space apart); thus, if your columns contain displays, for example, you may need a higher \langle \text{number} \rangle to shift something from one column into another.

Unlike ‘collectmore,’ the ‘unbalance’ counter is reset to zero at the end of the environment so it only applies to one \multicols environment.

The two methods may be combined but I suggest using these features only when fine tuning important publications.

Two more general tuning possibilities were added with version 1.5. \TeX allows to measure the badness of a column in terms of an integer value, where 0 means optimal and any higher value means a certain amount of extra white space. 10000 is considered to be infinitely bad (\TeX does not distinguish any further). In addition the special value 100000 means overfull (i.e., the column contains more text than could possibly fit into it).

The new release now measures every generated column and ignores solutions where at least one column has a badness being larger than the value of the counter \texttt{columnbadness}. The default value for this counter is 10000, thus \TeX will accept all solutions except those being overfull. By setting the counter to a smaller value you can force the algorithm to search for solutions that do not have columns with a lot of white space.

However, if the setting is too low, the algorithm may not find any acceptable solution at all and will then finally choose the extreme solution of placing all text into the first column.

Often, when columns are balanced, it is impossible to find a solution that distributes the text evenly over all columns. If that is the case the last column usually has less text than the others. In the earlier releases this text was stretched to produce a column with the same height as all others, sometimes resulting in really ugly looking columns.

In the new release this stretching is only done if the badness of the final column is not larger than the value of the counter \texttt{finalcolumnbadness}. The default setting is 9999, thus preventing the stretching for all columns that \TeX would consider infinitesimally bad. In that case the final column is allowed to run short which gives a much better result.

And there are two more parameters of some experimental nature, one called \texttt{\multicolovershoot} the other \texttt{\multicolundershoot}. They control the amount of space a column is allowed to be “too full” or “too short” without affecting the column badness. They are set to \texttt{0pt} and \texttt{2pt}, respectively.

### 2.2 Not balancing the columns

Although this package was written to solve the problem of balancing columns, I got repeated requests to provide a version where all white space is automatically placed in the last column or columns. Since version
2.3 Manually breaking columns

Another request often voiced was: “How to I tell \LaTeX{} that it should break the first column after this particular line?” The \texttt{\pagebreak} command (which works with the two-column option of \TeX) is of no use here since it would end the collection phase of \texttt{multicols} and thus all columns on that page. So with version 1.5u the \texttt{\columnbreak} command was added. If used within a paragraph it marks the end of the current line as the desired breakpoint. You can observe its effect on the previous page where three lines of text have been artifically forced into the second column (resulting in some white space between paragraphs in the first column).

2.4 Floats inside a \texttt{multicols} environment

Within the \texttt{multicols} environment the usual star float commands are available but their function is somewhat different as in the two-column mode of standard \TeX. Starrred floats, e.g., \texttt{figure*}, denote page wide floats that are handled in a similar fashion as normal floats outside the \texttt{multicols} environment. However, they will never show up on the page where they are encountered. In other words, one can influence their placement by specifying a combination of \texttt{t}, \texttt{b}, and/or \texttt{p} in their optional argument, but \texttt{h} doesn’t work because the first possible place is the top of the next page. One should also note, that this means that their placement behavior is determined by the values of \texttt{\topfraction}, etc. rather than by \texttt{\dbl}...

2.5 Warnings

Under certain circumstances the use of the \texttt{multicols} environment may result in some warnings from \TeX or \LaTeX. Here is a list of the important ones and the possible cause:

\begin{itemize}
  \item \texttt{Underfull }\texttt{\hbox (badness ...)}
  \end{itemize}

As the columns are often very narrow \TeX wasn’t able to find a good way to break the paragraph. Underfull denotes a loose line but as long the badness values is below 10000 the result is probably acceptable.

\begin{itemize}
  \item \texttt{Underfull }\texttt{\vbox ... while }\texttt{\output is active}
  \end{itemize}

If a column contains an character with an unusual depth, for example a ‘(’, in the bottom line then this message may show up. It usually has no significance as long as the value is not more than a few points.

\begin{itemize}
  \item \texttt{LaTeX Warning: I moved some lines to the next page}
  \end{itemize}

As mentioned above, \texttt{multicols} sometimes screws up the footnote numbering. As a precaution, whenever there is a footnote on a page that \texttt{multicols} had to leave a remainder for the following page this warning appears. Check the footnote numbering on this page. If it turns out that it is wrong you have to manually break the page using \texttt{\newpage} or \texttt{\pagebreak[..]}.

\begin{itemize}
  \item \texttt{Floats and marginpars not allowed inside 'multicols' environment!}
  \end{itemize}

This message appears if you try to use the \texttt{\marginpar} command or an unstared version of the \texttt{figure} or \texttt{table} environment. Such floats will disappear!

\begin{itemize}
  \item \texttt{Very deep columns! Grid alignment might be broken}
  \end{itemize}

This message can only appear if the option \texttt{grid} was chosen. In that case it will show up if a column has a very large depth so that \texttt{multicols} is unable to back up to its baseline. This is only relevant if one tries to produce a document where all text lines are aligned at an invisible grid, something that requires careful adjustment of many parameters and macros, e.g., heading definitions.

2.6 Tracing the output

To understand the reasoning behind the decisions \TeX makes when processing a \texttt{multicols} environment, a tracing mechanism is provided. If you set the counter \texttt{\texttt{\texttt{\texttt{\texttt{tracingmulticols}}}}} to a positive \texttt{\texttt{\texttt{\texttt{\texttt{(number)}}}}} you then will get some tracing information on the terminal and in the transcript file:

\begin{itemize}
  \item \texttt{\texttt{\texttt{\texttt{\texttt{\{number\}}}}}} = 1. \TeX will now tell you, whenever it enters or leaves a \texttt{multicols} environment, the number of columns it is working on and its decision about starting a new page before or after the environment.
  \item \texttt{\texttt{\texttt{\texttt{\texttt{\{number\}}}}}} = 2. In this case you also get information from the balancing routine: the heights tried for the left and right-most columns, information about shrinking if the \texttt{\texttt{\texttt{\texttt{\texttt{\raggedcolumns}}}}} declaration is in force and the value of the ‘unbalance’ counter if positive.
\end{itemize}
\( \langle \text{number} \rangle = 3 \). Setting \( \langle \text{number} \rangle \) to this value will additionally trace the mark handling algorithm. It will show what marks are found, what marks are considered, etc. To fully understand this information you will probably have to read carefully through the implementation.

\( \langle \text{number} \rangle \geq 4 \). Setting \( \langle \text{number} \rangle \) to such a high value will additionally place an \texttt{\textbackslash hrule} into your output, separating the part of text which had already been considered on the previous page from the rest. Clearly this setting should not be used for the final output. It will also activate even more debugging code for mark handling.

## 3 Prefaces to older versions

### 3.1 Preface to version 1.4

Besides fixing some bugs as mentioned in the \texttt{multicol.bug} file this new release enhances the \texttt{multicols} environment by allowing for balancing in arbitrary contexts. It is now, for example, possible to balance text within a \texttt{multicols} or a \texttt{minipage} as shown in 2 where a \texttt{multicols} environment within a \texttt{quote} environment was used. It is now even possible to nest \texttt{multicols} environments.

The only restriction to such inner \texttt{multicols} environments (nested, or within \TeX{}'s internal vertical mode) is that such variants will produce a box with the balanced material in it, so that they cannot be broken across pages or columns.

Additionally I rewrote the algorithm for balancing so that it will now produce slightly better results.

I updated the source documentation but like to apologize in advance for some ‘left over’ parts that slipped through the revision.

A note to people who like to improve the balancing algorithm of \texttt{multicols}: The balancing routine in now placed into a single macro which is called \texttt{\textbackslash balance@columns}. This means that one can easily try different balancing routines by rewriting this macro. The interface for it is explained in table 1. There are several improvements possible, one can think of integrating the \texttt{\textbackslash badness} function of \TeX{}3, define a faster algorithm for finding the right column height, etc. If somebody thinks he/she has an enhancement I would be pleased to learn about it. But please obey the copyright notice and don’t change \texttt{multicol.dtx} directly!

### 3.2 Preface to version 1.2

After the article about the \texttt{multicols} environment was published in \textit{TUGboat} 10#3, I got numerous requests for these macros. However, I also got a changed version of my style file, together with a letter asking me if I would include the changes to get better paragraphing results in the case of narrow lines. The main differences to my original style option were additional parameters (like \texttt{\textbackslash multicoladjdemerits} to be used for \texttt{\textbackslash adjdemerits}, etc.) which would influence the line breaking algorithm.

But actually resetting such parameters to zero or even worse to a negative value won’t give better line breaks inside the \texttt{multicols} environment. \TeX{}’s line breaking algorithm will only look at those possible line breaks which can be reached without a badness higher than the current value of \texttt{\textbackslash tolerance} (or \texttt{\textbackslash pretolerance} in the first pass). If this isn’t possible, then, as a last resort, \TeX{} will produce overfull boxes. All those (and only those) possible break points will be considered and finally the sequence which results in the fewest demerits will be chosen. This means that a value of \( -1000 \) for \texttt{\textbackslash adjdemerits} instructs \TeX{} to prefer visibly incompatible lines instead of producing better line breaks.

However, with \TeX{} 3.0 it is possible to get decent line breaks even in small columns by setting \texttt{\textbackslash emergencystretch} to an appropriate value. I implemented a version which is capable of running both in the old and the new \TeX{} (actually it will simply ignore the new feature if it is not available). The calculation of \texttt{\textbackslash emergencystretch} is probably incorrect. I made a few tests but of course one has have much more experience with the new possibilities to achieve the maximum quality.

Version 1.1a had a nice ‘feature’: the penalty for using the forbidden floats was their ultimate removal from \TeX{}’s \texttt{\@freelist} so that after a few \texttt{\marginpars} inside the \texttt{multicols} environment floats where disabled forever. (Thanks to Chris Rowley for pointing this out.)
The macro \balance@columns that contains the code for balancing gathered material is a macro without parameters. It assumes that the material for balancing is stored in the box \mult@box which is a \vbox. It also "knows" about all parameters set up by the \multicols environment, like \col@number, etc. It can also assume that \@colroom is the still available space on the current page.

When it finishes it must return the individual columns in boxes suitable for further processing with \page@sofar. This means that the left column should be stored in box register \mult@gfirstbox, the next in register \mult@firstbox + 2, ..., only the last one as an exception in register \mult@grightbox. Furthermore it has to set up two the macros \kept@firstmark and \kept@botmark to hold the values for the first and bottom mark as found in the individual columns. There are some helper functions defined in section 5.1 which may be used for this. Getting the marks right "by hand" is non-trivial and it may pay off to first take a look at the documentation and implementation of \balance@columns below before trying anew.

Table 1: Interface description for \balance@columns

\setemergencystretch: This is a hook for people who like to play around. It is supposed to set the \emergencystretch (dimen) register provided in the new \TeX 3.0. The first argument is the number of columns and the second one is the current \hsize. At the moment the default definition is 4pt × #1, i.e. the \hsize isn’t used at all. But maybe there are better formulae.

\set@floatcmds: This is the hook for the experts who like to implement a full float mechanism for the \multicols environment. The @ in the name should signal that this might not be easy.

Table 2: The new commands of multicol.sty version 1.2. Both commands might be removed if good solutions to these open problems are found. I hope that these commands will prevent that nearly identical style files derived from this one are floating around.

removed this misbehaviour and at the same time decided to allow at least floats spanning all columns, e.g., generated by the \figure* environment. You can see the new functionality in table 2 which was inserted at this very point. However single column floats are still forbidden and I don’t think I will have time to tackle this problem in the near future. As an advice for all who want to try: wait for \TeX 3.0. It has a few features which will make life much easier in multi-column surroundings. Nevertheless we are working here at the edge of \TeXs capabilities, really perfect solutions would need a different approach than it was done in \TeXs page builder.

The text below is nearly unchanged, I only added documentation at places where new code was added.

4 The Implementation

We are now switching to two-column output to show the abilities of this environment (and bad layout decisions).

4.1 The documentation driver file

The next bit of code contains the documentation driver file for \TeX, i.e., the file that will produce the documentation you are currently reading. It will be extracted from this file by the docstrip program. Since this is the first code in this file one can produce the documentation simply by running \TeX on the .dtx file.

\begin{verbatim}
\documentclass{ltxdoc}
\usepackage{multicol}[1999/05/25]
\end{verbatim}
4 \usepackage{doc}
First we set up the page layout suitable for this article.
5 \setlength{\textwidth}{39pc}
6 \setlength{\textheight}{54pc}
7 \setlength{\parindent}{1em}
8 \setlength{\parskip}{0pt plus 1pt}
9 \setlength{\oddsidemargin}{0pc}
10 \setlength{\marginparwidth}{0pc}
11 \setlength{\topmargin}{-2.5pc}
12 \setlength{\headsep}{20pt}
13 \setlength{\columnsep}{1.5pc}
We want a rule between columns.
14 \setlength\columnseprule{.4pt}
We also want to ensure that a new \texttt{multicols}
environment finds enough space at the bottom of the page.
15 \setlength\premulticols{6\baselineskip}
When balancing columns we disregard solutions that are too bad. Also, if the last column is too bad we
typeset it without stretch.
16 \setcounter{columnbadness}{7000}
17 \setcounter{finalcolumnbadness}{7000}
The index is supposed to come out in four columns. And we don't show macro names in the margin.
18 \setcounter{IndexColumns}{4}
19 \let\DescribeMacro\SpecialUsageIndex
20 \let\DescribeEnv\SpecialEnvIndex
21 \renewcommand\PrintMacroName[1]{}
22 \CodelineIndex
23 \%\DisableCrossrefs \% Partial index
24 \RecordChanges \% Change log
25 \renewcommand{\theCodelineNo}
26 \%\DisableCrossrefs % Partial index
27 \setwidth\MacIndent{\scriptsize\rm 00\ }
28 \begin{document}
29 \typeout{****************************************
30 \^^J* Expect some Under- and overfull boxes.
31 \^^J****************************************}
32 \DocInput{multicol.dtx}
33 \end{document}
34 \langle /driver 

4.2 Identification and option processing

We start by identifying the package. Since it makes use of features only available in \LaTeX\ 2\epsilon we ensure that this format is available. (Now this is done earlier in the file.)
37 \% \NeedsTeXFormat{LaTeX2e}
38 \% \ProvidesPackage{multicol}[..../..
39 \% \v... multicolumn formatting]

Next we declare options supported by \texttt{multicols}. Twocolumn mode and \texttt{multicols} do not work together so we warn about possible problems. However, since you can revert to \texttt{onecolumn} in which case \texttt{multicols} does work, we don’t make this an error.
41 \DeclareOption{twocolumn}{
42 \PackageWarning{multicol}{May not work
43 with the twocolumn option}}
44 \let\mc@gridwarn\maxdimen
45 \DeclareOption{grid}{\def\mc@gridwarn{\maxdepth}}
46 \ProcessOptions
48 \begin{document}
49 \typeout{****************************************
50 \^^J* Expect some Under- and overfull boxes.
51 \^^J*\% Partial index
52 \^^J****************************************}
53 \DocInput{multicol.dtx}
54 \end{document}
55 \langle /driver 

4.3 Starting and Ending the multicols Environment

As mentioned before, the \texttt{multicols} environment has one mandatory argument (the number of columns) and up to two optional ones. We start by reading the number of columns into the \texttt{\col@number} register.
58 \def\multicols#1{\col@number#1\relax
59 If the user forgot the argument, \TeX{} will complain about a missing number at this point. The error recovery mechanism will then use zero, which isn’t a good choice in this case. So we should now test
whether everything is okay. The minimum is two columns at the moment.
\ifnum\col@number<\tw@
\PackageWarning{multicol}{Using \number\col@number columns doesn’t seem a good idea.}\MessageBreak
I therefore use two columns instead}
\col@number\tw@ \fi

We have only enough box registers for ten columns, so we need to check that the user hasn’t asked for more.
\ifnum\col@number>10
\PackageError{multicol}{Too many columns}{Current implementation doesn’t support more than 10 columns.}\MessageBreak
I therefore use 10 columns instead}
\col@number10 \fi

Within the environment we need a special version of the kernel \@footnotetext command since the original sets the \hsize to \columnwidth which is not correct in the multicol environment. Here \columnwidth refers to the width of the individual column and the footnote should be in \textwidth. Since \@footnotetext has a different definition inside a minipage environment we do not redefine it directly. Instead we locally set \columnwidth to \textwidth and call the original (current) definition stored in \orig@footnotetext. If the multicols environment is nested inside another multicols environment then the redefinition has already happened. So be better test for this situation. Otherwise, we will get a \TeX stack overflow as this would generate a self-referencing definition.
\ifx\@footnotetext\mult@footnotetext\else
\let\orig@footnotetext\@footnotetext
\let\@footnotetext\mult@footnotetext
\fi

Now we can safely look for the optional arguments.
\@ifnextchar\[
\mult@cols{\mult@cols[]}\]

The \mult@cols macro grabs the first optional argument (if any) and looks for the second one.
\def\mult@cols[#1]{\@ifnextchar[
This argument should be a \texttt{dimen} denoting the minimum free space needed on the current page to start the environment. If the user didn’t supply one, we use \premulticols as a default.

After removing all arguments from the input we are able to start with \mult@cols.
\def\mult@cols[#1[#2]]{}

First thing we do is to decide whether or not this is an unbounded multicols environment, i.e. one that may split across pages, or one that has to be typeset into a box. If we are in \TeX’s “inner” mode (e.g., inside a box already) then we have a boxed version of multicols therefore we set the \boxedmulticols switch to true. The multicols should start in vertical mode. If we are not already there we now force it with \par since otherwise the test for “inner” mode wouldn’t show if we are in a box.
\par
\ifinner \boxedmulticols\true\fi

Otherwise we check \doublecol@number. This counter is zero outside a multicols environment but positive inside (this happens a little later on). In the second case we need to process the current multicols also in “boxed mode” and so change the switch accordingly.
\else
\ifnum \doublecol@number>\z@ \@boxedmulticolstrue \fi
\fi

Then we look to see if statistics are requested:
\mult@info\z@{Starting environment with \the\col@number space columns}\MessageBreak
Then we measure the current page to see whether a useful portion of the multicolumn environment can be typeset. This routine might start a new page.
\enough@room{#2}

Now we output the first argument and produce vertical space above the columns. (Note that this argument corresponds to the first optional argument of the multicols environment.) For many releases this argument was typeset in a group to get a similar effect as \twocolumn[...]. However, this conflicts with local changes done by things like sectioning commands (which account for the majority of commands used in that argument) messing up vertical spacing etc. later in the document so that
from version v1.5q on this argument is again typeset at the outer level.

When the last line of a paragraph had a positive depth then this depth normally taken into account by the baselineskip calculation for the next line. However, the columns produced by a following \multicol are rigid and thus the distance from the baseline of a previous text line to the first line in a \multicol would differ depending on the depth of the previous line. To account for this we add a negative space unless the depth is -1000pt which signals something special to T\LaTeX and is not supposed to be a real depth.

The actual generation of this corrective space is a little bit more complicated as it doesn’t make sense to always back up to the previous baseline (in case an object with a very large depth was placed there, e.g., a centered tabular). So we only back up to the extend that we are within the \baselineskip grid. We know that the box produced by \multicols has \topskip at its top so that also needs to be taken into account.

We start a new grouping level to hide all subsequent changes (done in \prepare@multicols for example).

If we are in boxed mode we now open a box to typeset all material from the multicols body into it, otherwise we simply go ahead.

We may have to reset some parameters at this point, perhaps \parboxrestore would be the right action but I leave it for the moment.

We finish by suppressing initial spaces.

\begin{quote}
Here is the switch and the box for “boxed” multicols code.
\end{quote}

The \enough@room macro used above isn’t perfect but works reasonably well in this context. We measure the free space on the current page by subtracting \pagetotal from \pagegoal. This isn’t entirely correct since it doesn’t take the ‘shrinking’ (i.e. \pageshrink) into account. The ‘recent contribution list’ might be nonempty so we start with \par and an explicit \penalty. Actually, we use \addpenalty to ensure that a following \addvspace will ‘see’ the vertical space that might be present. The use of \addpenalty will have the effect that all items from the recent contributions will be moved to the main vertical list and the \pagetotal value will be updated correctly. However, the penalty will be placed in front of any dangling glue item with the result that the main vertical list may already be overfull even if T\LaTeX is not invoking the output routine.

Measuring makes only sense when we are not in “boxed mode” so the routine does nothing if the switch is true.

To empty the contribution list the first release contained a penalty zero but this had the result that \addvspace couldn’t detect preceding glue. So this was changed to \addpenalty. But this turned out to be not enough as \addpenalty will not add a penalty when \nobreak is true. Therefore we force this switch locally to false. As a result there may be a break between preceding text and the start of a multicol environment, but this seems acceptable since there is the optional argument for exactly this reason.

To be able to output the value we need to assign it to a register first since it might be a register (default) in which case we need to use \the or it might be a plain value in which case \the would be wrong.

See the documentation of \endmulticols for further details.
Now we test whether tracing information is required:

\mult@info\z@\MessageBreak
\Current page:\MessageBreak
height=\the\pagegoal: used \the\pagetotal
\space \rightarrow free=\the\page@free
\MessageBreak
needed \the\@tempskipa
\space(for #1)\%

Our last action is to force a page break if there isn’t enough room left.
\ifdim \page@free <#1\newpage \fi

When preparing for multicolumn output several things must be done.
\def\prepare@multicols{
We start saving the current \@totalleftmargin and then resetting the \parshape in case we are inside some list environment. The correct indentation for the multicols environment in such a case will be produced by moving the result to the right by \multicol@leftmargin later on. If we would use the value of of \@totalleftmargin directly then lists inside the multicols environment could cause a shift of the output.

\multicol@leftmargin\@totalleftmargin
\@totalleftmargin\z@
\parshape\z@

We also set the register \doublecol@number for later use. This register should contain \(2 \times \col@number\). This is also an indicator that we are within a multicols environment as mentioned above.

\doublecol@number\col@number
\multiply\doublecol@number\tw@\MessageBreak
\advance\doublecol@number\mult@rightbox\MessageBreak
\if@boxedmulticols
\let\l@kept@firstmark\kept@firstmark\MessageBreak
\let\l@kept@botmark\kept@botmark\MessageBreak
\global\let\kept@firstmark\@empty\MessageBreak
\global\let\kept@botmark\@empty\MessageBreak
\else
We add an empty box to the main vertical list to ensure that we catch any insertions (held over or inserted at the top of the page). Otherwise it might happen that the \eject is discarded without calling the output routine. Inside the output routine we remove this box again. Again this code applies only if we are on the main vertical list and not within a box. However, it is not enough to turn off interline spacing, we also have to clear \topskip before adding this box, since \topskip is always inserted before the first box on a page which would leave us with an extra space of \topskip if multicols start on a fresh sheet.
\nointerlineskip \{\topskip\z@\null\}
\output{\global\setbox\partial@page\vbox{\noweave}{
Now we have to make sure that we catch one special situation which may result in loss of text! If the user has a huge amount of vertical material within the first optional argument that is larger then \premulticols and we are near the bottom of the page then it can happen that not the \eject is triggering this special output routine but rather the overfull main vertical list. In that case we get another breakpoint through the \eject penalty. As a result this special output routine would be called twice and the contents of \partial@page, i.e. the material before the multicols environment gets lost. There are several solutions to avoid this problem, but for now we will simply detect this and inform the user that he/she has to enlarge the \premulticols by using a suitable value for the second argument.

\PackageError{multicol}{Error saving partial page}{The part of the page before the multicols environment was nearly full with the result that starting the environment will produce an overfull page. Some text may be lost! Please increase \premulticols either generally or for this environment by specifying a suitable value in the second optional argument to the multicols environment.}
\unvbox\partial@page \MessageBreak
\box\last@line \MessageBreak
\else
\unvbox\@cclv \MessageBreak
\global\setbox\last@line\lastbox
\fi
\MessageBreak
\endinput
can't use \topmark because this register will not always contain what its name promises because \LaTeX{} sometimes calls the output routine for float management.\footnote{During such a call the \botmark gets globally copied to \topmark by the \TeX{} program.} Therefore we use the second best solution by initializing it with \firstmark. In fact, for our purpose this doesn't matter as we use \kept@topmark only to initialize \firstmark and \botmark of a following page if we don't find any marks on the current one.

\begin{verbatim}
\global\let\kept@topmark=\firstmark \eject
\init@mult@footins
\set@mult@vsize\relax
\end@dblfloat
\setemergencystretch\col@number\hsize
\set@floatcmds
\end@dblfloat
\save@float@page
\global\let\kept@topmark\firstmark
\end@dblfloat
\setemergencystretch\col@number\hsize
\set@floatcmds
\end@dblfloat
\save@float@page
\global\let\kept@topmark\firstmark
\end@dblfloat
\setemergencystretch\col@number\hsize
\set@floatcmds
\end@dblfloat
\save@float@page
\global\let\kept@topmark\firstmark
\end@dblfloat
\setemergencystretch\col@number\hsize
\set@floatcmds
\end@dblfloat
\save@float@page
\global\let\kept@topmark\firstmark
\end@dblfloat
\setemergencystretch\col@number\hsize
\set@floatcmds
\end@dblfloat
\save@float@page
\global\let\kept@topmark\firstmark
\end@dblfloat
\setemergencystretch\col@number\hsize
\set@floatcmds
\end@dblfloat
\save@float@page
\begin{verbatim}
\setemergencystretch\col@number\hsize
\set@floatcmds
\end@dblfloat
\save@float@page
\global\let\kept@topmark\firstmark
\end@dblfloat
\setemergencystretch\col@number\hsize
\set@floatcmds
\end@dblfloat
\save@float@page
\global\let\kept@topmark\firstmark
\end@dblfloat
\setemergencystretch\col@number\hsize
\set@floatcmds
\end@dblfloat
\save@float@page
\global\let\kept@topmark=\firstmark \eject
\end{verbatim}
\end{verbatim}

The next thing to do is to assign a new value to \vsize. \LaTeX{} maintains the free room on the page (i.e. the page height without the space for already contributed floats) in the register \col@room. We must subtract the height of \partial@page to put the actual free room into this variable.

\begin{verbatim}
\advance\@colroom-\ht\partial@page
\end{verbatim}

Then we have to calculate the \vsize value to use during column assembly. \set@mult@vsize takes an argument which allows to make the setting local (\relax) or global (\global). The latter variant is used inside the output routine below. At this point here we have to make a local change to \vsize because we want to get the original value for \vsize restored in case this \multicols environment ends on the same page where it has started.

\begin{verbatim}
\set@mult@vsize\relax
\end{verbatim}

Now we switch to a new \output routine which will be used to put the gathered column material together.

\begin{verbatim}
\output{%
\multi@column@out}
\end{verbatim}

Finally we handle the footnote insertions. We have to multiply the magnification factor and the extra skip by the number of columns since each footnote reduces the space for every column (remember that we have pagewide footnotes). If, on the other hand, footnotes are typeset at the very end of the document, our scheme still works since \count\footins is zero then, so it will not change. To allow even further customization the setting of the \footins parameters is done in a separate macro.

\begin{verbatim}
\output{%
\multi@column@out}
\end{verbatim}

The same reason (pagewide footnotes), the \dimen register controlling the maximum space used for footnotes isn’t changed. Having done this, we must reinsert all the footnotes which are already present (i.e. those encountered when the material saved in \partial@page was first processed). This will reduce the free space (i.e. \pagetotal) by the appropriate amount since we have changed the magnification factor, etc. above.

\begin{verbatim}
\reinsert@footnotes
\end{verbatim}

All the code above was only necessary for the unrestricted \multicols version, i.e. the one that allows page breaks. If we are within a box there is no point in setting up special output routines or \vsize, etc.

\begin{verbatim}
\fi
\end{verbatim}

But now we are coming to code that is necessary in all cases. We assign new values to \vbadness, \hbadness and \tolerance since it’s rather hard for \TeX{} to produce ‘good’ paragraphs within narrow columns.

\begin{verbatim}
\vbadness\@Mi \hbadness5000 \tolerance\multicoltolerance
\end{verbatim}

Since nearly always the first pass will fail we ignore it completely telling \TeX{} to hyphenate directly. In fact, we now use another register to keep the value for the multicol pre-tolerance, so that a designer may allow to use \pretolerance.

\begin{verbatim}
\pretolerance\multicoltolerance
\end{verbatim}

For use with the new \TeX{} we set \emergency@stretch to \col@number \times 4pt. However this is only a guess so at the moment this is done in a macro \set@emergency@stretch which gets the current \hsize and the number of columns as arguments. Therefore users are able to figure out their own formula.

\begin{verbatim}
\set@emergency@stretch\col@number\hsize
\end{verbatim}

Another hook to allow people adding their own extensions without making a new package is \set@float@cmds which handles any redefinitions of \LaTeX{}s internal float commands to work with the \multicols environment. At the moment it is only used to redefine \dblfloat and \end@dblfloat.

\begin{verbatim}
\set@float@cmds
\end{verbatim}

Additionally, we advance \baselineskip by \multicol@baselineskip to allow corrections for narrow columns.

\begin{verbatim}
\advance\baselineskip\multicol@baselineskip
\end{verbatim}

The \hsize of the columns is given by the formula:

\begin{verbatim}
\linewidth - (\col@number - 1) \times \columnsep
\col@number
\end{verbatim}

The formula above has changed from release to release. We now start with the current value of \linewidth so that the column width is properly calculated when we are inside a minipage or a list or some other environment. This will be achieved with:
We also set \linewidth and \columnwidth to \hsize. In the past \columnwidth was left unchanged. This is inconsistent, but \columnwidth is used only by floats (which aren’t allowed in their current implementation) and by the \footnote macro. Since we want pagewise footnotes\footnote{I’m not sure that I really want pagewise footnotes. But balancing of the last page can only be achieved with this approach or with a multi-path algorithm which is complicated and slow. But it’s a challenge to everybody to prove me wrong! Another possibility is to reimplement a small part of the \texttt{fire-up} procedure in \TeX{} (the program). I think that this is the best solution if you are interested in complex page makeup, but it has the disadvantage that the resulting program cannot be called \TeX{} thereafter.} this simple trick saved us from rewriting the \footnote macros. However, some applications refered to \columnwidth as the “width of the current column” to typeset displays (the \texttt{amsmath} package, for example) and to allow of such applications together with \texttt{multicol} this is now changed.

Before we change \linewidth to the new value we record its old value in some register called \full@width. This value is used later on when we package all columns together.

\begin{verbatim}
\def\set@mult@vsize#1{\vsize\@colroom\@tempdima\baselineskip\advance\@tempdima-\topskip\advance\vsize\@tempdima\vsize\col@number\vsize\advance\vsize-\@tempdima\def\init@mult@footins{%\multiply\count\footins\col@number\multiply\skip \footins\col@number\} Since we have to set \col@number columns on one page, each with a height of \@colroom, we have to assign \vsize = \col@number \times \@colroom in order to collect enough material before entering the \output routine again. In fact we have to add another \((\col@number - 1) \times (\baselineskip - \topskip)\) if you think about it.
\def\set@mult@vs@size#1{% \vs@size\@colroom\@tempdima\baselineskip\advance\@tempdima-\topskip\vsize\@colnumber\vsize\advance\vsize-\@tempdima\endverbatim}

But this might not be enough since we use \texttt{\vsplit} later to extract the columns from the gathered material. Therefore we add some ‘extra lines,’ one for each column plus a corrective action depending on the value of the ‘collectmore’ counter. The final value is assigned globally if \#1 is \texttt{\global} because we want to use this macro later inside the output routine too.

The dimen register we need for saving away the outer value of \texttt{@totalleftmargin}.

When the end of the \texttt{multicols} environment is sensed we have to balance the gathered material. Depending on whether or not we are inside a boxed multicol different things must happen. But first we end the current paragraph with a \texttt{\par} command.

\begin{verbatim}
def\endmulticols{\par\if@boxedmulticols\egroup\balance@columns\page@sofar\return@nonemptymark{first}\kept@firstmark\return@nonemptymark{bot}\kept@botmark\page@sofar\global\let\kept@firstmark\l@kept@firstmark\global\let\kept@botmark\l@kept@botmark\endverbatim}

This finishes the code for the “boxed” case.
If there was a \columnbreak on the very last line all material will have been moved to the \colbreak@box. Thus the the galley will be empty and no output routine gets called so that the text is lost. To avoid this problem (though unlikely) we check if the current galley is empty and the \colbreak@box contains text and if so return that to the galley. If the galley is non-empty any material in \colbreak@box is added in the output routine since it needs to be put in front.

\begin{verbatim}
\(+\text{colbreak}\)
\ifdim\pagegoal>\maxdimen
\else
\mult@info\tw@
break(s) for splitting}\
\unvbox\colbreak@box\fi
\fi
\end{verbatim}

If we are in an unrestricted \multicols environment we end the current paragraph above with \par but this isn’t sufficient since \TeXs page_builder will not totally empty the contribution list.\footnote{This once caused a puzzling bug where some of the material was balanced twice, resulting in some overprints. The reason was the \vsize} Therefore we must also add an explicit \penalty. Now the contribution list will be emptied and, if its material doesn’t all fit onto the current page then the output routine will be called before we change it. At this point we need to use \penalty not \addpenalty to ensure that a) the recent contributions are emptied and b) that the very last item on the main vertical list is a valid break point so that \TeX breaks the page in case it is overfull.

\begin{verbatim}
\penalty\z@\prevdepth\z@

The processed material might consist of a last line with a decender in which case the \vsize will be non-zero. However, this material is getting reformatted now so that this value is likely to be wrong. We therefore normalize the situation by pretending that the depth is zero and arrange later that the box containing the assembled columns has in fact this property.

\begin{verbatim}
\vsize
\end{verbatim}

Now it’s safe to change the output routine in order to balance the columns.

\begin{verbatim}
\output{\balance@columns@out}\eject
\end{verbatim}

If the \multicols environment body was completely empty or if a multi-page \multicols just ends at a page boundary we have the unusual case that the \vsize will have no effect (since the main vertical list is empty)—thus no output routine is called at all. As a result the material preceding the \multicols (stored in \partialpage will get lost if we don’t take of this by hand.

\begin{verbatim}
\if vbox\partialpage
\un vbox\partialpage\fi
\end{verbatim}

After the output routine has acted we restore the kept marks to their initial value.

\begin{verbatim}
\global\let\kept@firstmark\empty
\global\let\kept@botmark\empty
\end{verbatim}

To fix an obscure bug which is the result of the current definition of the \begin ... \end macros, we check that we are still (logically speaking) in the \multicols environment. If, for example, we forget to close some environment inside the \multicols environment, the following \endgroup would be incorrectly considered to be the closing of this environment.

\begin{verbatim}
\@checkend{multicols}\
\endgroup
\end{verbatim}

We also set the ‘\unbalance’ counter to its default. This is done globally since \LaTeX counters are always changed this way.\footnote{Actually, we are still in a group started by the \begin macro, so \global must be used anyway.}

\begin{verbatim}
\global\c@unbalance\z@
\end{verbatim}

Now it’s time to return any footnotes if we are in unrestricted mode:

\begin{verbatim}
\if@boxedmulticols\else
\reinsert@footnotes\fi
\end{verbatim}

We also take a look at the amount of free space on the current page to see if it’s time for a page break. The vertical space added thereafter will vanish if \enough@room starts a new page.

But there is one catch. If the \end{multicols} is at the top of which can happen if there is a break point just before it (such as end ending environment) which was chosen. In that case we would do the next page using the internal \vsize for multicol collection which is a desaster. So we better catch this error.

\begin{verbatim}
\if vbox\vsize\else
\global\vsize\fi
\end{verbatim}

The output routine above will take care of the \vsize and reinsert the balanced columns, etc. But it can’t reinsert the \footnotes because we first have to restore the \footins parameter since we are returning to one column mode. This will be done in the next line of code; we simply close the group started in \multicols.
case. Fortunately we can detect it by looking at `\pagegoal`.

\begin{verbatim}
\ifdim \pagegoal=\maxdimen
 seen.\else
 \enough@room\postmulticols\fi
\fi
\addvspace\multicolsep
\fi
\end{verbatim}

If statistics are required we finally report that we have finished everything.

\begin{verbatim}
\mult@info\z@
\end{verbatim}

Let us end this section by allocating all the registers used so far.

\begin{verbatim}
\newcount\c@unbalance
\newcount\c@collectmore
\newcount\doublecol@number
\newcount\multicoltolerance
\newcount\multicolpretolerance
\newdimen\full@width
\newdimen\page@free
\newdimen\premulticols
\newdimen\postmulticols
\newskip\multicolsep
\newskip\multicolbaselineskip
\newbox\partial@page
\newbox\last@line
\end{verbatim}

And here are their default values:

\begin{verbatim}
c@unbalance = 0
c@collectmore = 0
\end{verbatim}

To allow checking whether some macro is used within the \texttt{multicol}s environment the counter \texttt{\col@number} gets a default of 1 outside the the environment.

\begin{verbatim}
\col@number = 1
\multicoltolerance = 9999
\multicolpretolerance = -1
\premulticols = 50pt
\postmulticols= 20pt
\multicolsep = 12pt plus 4pt minus 3pt
\multicolbaselineskip=0pt
\end{verbatim}

\section*{4.4 The output routines}

We first start with some simple macros. When typesetting the page we save the columns either in the box registers 0, 2, 4, \ldots (locally) or 1, 3, 5, \ldots (globally). This is \texttt{Plain} \TeX{} policy to avoid an overflow of the save stack.

Therefore we define a \texttt{\process@cols} macro to help us in using these registers in the output routines below. It has two arguments: the first one is a number; the second one is the processing information. It loops starting with \texttt{$\count@=#1$} (\texttt{$\count@$} is a scratch register defined in \texttt{Plain} \TeX{}), processes argument \texttt{$#2$}, adds two to \texttt{$\count@$}, processes argument \texttt{$#2$} again, etc. until \texttt{$\count@$} is higher than \texttt{\doublecol@number}. It might be easier to understand it through an example, so we define it now and explain its usage afterwards.

\begin{verbatim}
def\process@cols#1#2{\count@#1\relax
\loop
\langle∗debug\rangle
\typeout{Looking at box \the\count@}
\langle/\debug\rangle
\#2,\count@ \ifnum\count@<\doublecol@number
\repeat}
\end{verbatim}

We now define \texttt{\page@sofar} to give an example of the \texttt{\process@cols} macro. \texttt{\page@sofar} should output everything prepared by the balancing routine \texttt{\balance@columns}.

\begin{verbatim}
def\page@sofar{%
\balance@columns
prepares its output in the even numbered scratch box registers. Now we output
the columns gathered assuming that they are saved in the box registers 2 (left column), 4 (second column), \ldots However, the last column (i.e. the rightmost) should be saved in box register 0.\footnote{You will see the reason for this numbering when we look at the output routines \texttt{\multi@column@out} and \texttt{\balance@columns@out}.} First we ensure that the columns have equal width. We use \texttt{\process@cols} for this purpose, starting with \texttt{$\count@$ = \mult@rightbox}. Therefore \texttt{$\count@$} loops through \texttt{\mult@rightbox}, \texttt{\mult@rightbox} + 2, \ldots (to \texttt{\doublecol@number}).
\def\process@cols\mult@rightbox
\end{verbatim}

We have to check if the box in question is void, because the operation \texttt{\wd\langle number\rangle} on a void box will not change its dimension (sigh).

\begin{verbatim}
{\ifvoid\count@
\repeat}
\end{verbatim}

\section*{4.4 The output routines}

We first start with some simple macros. When typesetting the page we save the columns either in the box registers 0, 2, 4, \ldots (locally) or 1, 3, 5, \ldots (globally). This is \texttt{Plain} \TeX{} policy to avoid an overflow of the save stack.

Therefore we define a \texttt{\process@cols} macro to help us in using these registers in the output routines below. It has two arguments: the first one is a number; the second one is the processing information. It loops starting with \texttt{\count@=#1} (\texttt{\count@} is a scratch register defined in \texttt{Plain} \TeX{}), processes argument \texttt{\#2}, adds two to \texttt{\count@}, processes argument \texttt{\#2} again, etc. until \texttt{\count@} is higher than \texttt{\doublecol@number}. It might be easier to understand it through an example, so we define it now and explain its usage afterwards.

\begin{verbatim}
def\process@cols#1#2{\count@#1\relax
\loop
\langle∗debug\rangle
\typeout{Looking at box \the\count@}
\langle/\debug\rangle
\#2,\count@ \ifnum\count@<\doublecol@number
\repeat}
\end{verbatim}

We now define \texttt{\page@sofar} to give an example of the \texttt{\process@cols} macro. \texttt{\page@sofar} should output everything prepared by the balancing routine \texttt{\balance@columns}.

\begin{verbatim}
def\page@sofar{%
\balance@columns
prepares its output in the even numbered scratch box registers. Now we output
the columns gathered assuming that they are saved in the box registers 2 (left column), 4 (second column), \ldots However, the last column (i.e. the rightmost) should be saved in box register 0.\footnote{You will see the reason for this numbering when we look at the output routines \texttt{\multi@column@out} and \texttt{\balance@columns@out}.} First we ensure that the columns have equal width. We use \texttt{\process@cols} for this purpose, starting with \texttt{\count@ = \mult@rightbox}. Therefore \texttt{\count@} loops through \texttt{\mult@rightbox}, \texttt{\mult@rightbox} + 2, \ldots (to \texttt{\doublecol@number}).
\def\process@cols\mult@rightbox
\end{verbatim}

We have to check if the box in question is void, because the operation \texttt{\wd\langle number\rangle} on a void box will not change its dimension (sigh).

\begin{verbatim}
{\ifvoid\count@
\repeat}
\end{verbatim}
Now we give some tracing information.
\count@ \col@number \advance \count@ \m@ne \
l@mult@info \z@
\{Column spec: \the \full@width \space = indent 
+ columns + sep =\MessageBreak
\the \multicol@leftmargin \space 
+ \the \col@number \space 
x \the \hsize \space 
+ \the \count@ \space 
x \the \columnsep \}
\ifvmode \else \errmessage {Multicol Error} \fi
At this point we should always be in vertical mode.
\ifvmode \else \vermessage {Multicol Error} \fi
Now we put all columns together in an \hbox of width \full@width (shifting it by \multicol@leftmargin to the right so that it will be placed correctly if we are within a list environment)
The box containing the columns has a large height and thus will always result in using \lineskip if the normal \baselineskip calculations are used. We therefore better cancel that process.
\nointerlineskip
As mentioned earlier we want to have the reference point of the box we put on the page being at the baseline of the last line of the columns but we also want to ensure that the box has no depth so that any following skip is automatically starting from that baseline. We achieve this by recording the depths of all columns and then finally backing up by the maximum. (perhaps a simpler method would be to assemble the box in a register and set the depth of that box to zero (not checked).

We need a global scratch register for this; using standard \TeX conventions we choose \dimen2 and initialize it with the depth of the character “p” since that is one of the depths that compete for the maximum.
\setbox \z@ \hbox {p} \global \dimen \tw@ \dp \z@
\moveright \multicol@leftmargin \hbox to \full@width { 
and separating the columns with a rule if desired.
\process@cols \mult@gfirstbox 
If the depth of the current box is larger than the maximum found so far in \dimen2 we update that register.
\ifdim \dp \count@ > \dimen \tw@ \else \fi 
\global \dimen \tw@ \dp \count@ 
\box \count@ 
\hss \normalcolor \rule
\@width \columnseprule \hss \}
As you will have noticed, we started with box register \mult@gfirstbox (i.e. the left column). So this time \count@ looped through 2, 4,... (plus the appropriate offset). Finally we add box 0 and close the \hbox. Again we may have to update \dimen \tw@.
\ifdim \dp \mult@grightbox > \dimen \tw@ \else \fi 
\global \dimen \tw@ \dp \mult@grightbox \fi 
\box \mult@grightbox 
The depths of the columns depend on their last lines. To ensure that we will always get a similar look as far as the rules are concerned we force the depth at least the depth of a letter ‘p’.
\rlap {\phantom p} \}
Now after typesetting the box we back up to its baseline by using the value stored in \dimen2.
\kern -\dimen \tw@ 
However, in case of of the columns was unusually deep \TeX may have tried some corrective actions in which case backing up by the abve value will not bring us back to the baseline. A good indication for this is a depth of \maxdepth though it is not an absolute proof. If the option grid is used \mc@gridwarn will expand to this, otherwise to \maxdimen in which case this warning will not show up.
\ifdim \dimen \tw@ = \mc@gridwarn \PackageWarning {multicol} {Very deep columns! MessageBreak 
\fi
Before we tackle the bigger output routines we define just one more macro which will help us to find our way through the mysteries later. \reinsert@footnotes will do what its name indicates: it reinserts the footnotes present in \footinbox so that they will be reprocessed by \TeX’s \page_builder.
Instead of actually reinserting the footnotes we insert an empty footnote. This will trigger insertion mechanism as well and since the old footnotes are still in their box and we are on a fresh page \skipfootins should be correctly taken into account.
\def \reinsert@footnotes { \ifvoid \footins \else 
\insert \footins \fi } 
Now we can’t postpone the difficulties any longer. The \multicol@out routine will be called in two situations. Either the page is full (i.e. we have collected enough material to generate all the
required columns) or a float or marginpar (or a \clearpage is sensed. In the latter case the \outputpenalty is less than \-10000, otherwise the penalty which triggered the output routine is higher. Therefore it's easy to distinguish both cases: we simply test this register.

\begin{verbatim}
def\multi@column@out{\% \ifnum\outputpenalty <-\@M \speci@ls \else \speci@ls \fi}

Let us now consider the normal case. We have to \vsplit the columns from the accumulated material in box 255. Therefore we first assign appropriate values to \splittopskip and \splitmaxdepth.

\begin{verbatim}
\splittopskip\topskip \splitmaxdepth\maxdepth
\end{verbatim}

Then we calculate the current column height (in \dimen). Note that the height of \particalpage is already subtracted from \@colroom so we can use its value as a starter.

\begin{verbatim}
\dimen0\@colroom
\end{verbatim}

But we must also subtract the space occupied by footnotes on the current page. Note that we first have to reset the skip register to its normal value. Again, the actual action is carried out in a utility macro, so that other applications can modify it.

\begin{verbatim}
\divide\skip\footins\col@number \ifvoid\footins \else \leave@mult@footins \fi
\end{verbatim}

Now we are able to \vsplit off all but the last column. Recall that these columns should be saved in the box registers 2, 4,… (plus offset).

\begin{verbatim}
\process@cols\mult@firstbox\% \setbox\count@ \vsplit\cclv to\dimen@ \set@keptmarks \ifshr\nking \setbox\count@\vbox to\dimen@{\unvbox\count@\vfill} \fi \fi
\end{verbatim}

Then the last column follows.

\begin{verbatim}
\setbox\mult@rightbox \vsplit\cclv to\dimen@ \set@keptmarks \ifshr\nking \setbox\mult@rightbox\vbox to\dimen@{\unvbox\mult@rightbox\vfill} \fi
\end{verbatim}

Having done this we hope that box 255 is emptied. If not, we reinsert its contents.

\begin{verbatim}
\ifvoid\@cclv \unvbox\@cclv \penalty\outputpenalty
\end{verbatim}

In this case a footnote that happens to fall into the leftover bit will be typeset on the wrong page. Therefore we warn the user if the current page contains footnotes. The older versions of multicols produced this warning regardless of whether or not footnotes were present, resulting in many unnecessary warnings.

\begin{verbatim}
\ifvoid\footins \else \PackageWarning{multicol} \MessageBreak Footnotes on page \thepage space might be wrong \fi
\end{verbatim}

If the `tracingmulticols` counter is 4 or higher we also add a rule.

\begin{verbatim}
\ifnum\c@tracingmulticols>\thr@@ \hrule\allowbreak \fi
\end{verbatim}

To get a correct marks for the current page we have to (locally redefine \firstmark and \botmark. If \kept@firstmark is non-empty then \kept@botmark must be non-empty too so we can use their values. Otherwise we use the value of \kept@topmark which was first initialized when we gathered the \particalpage and later on was updated to the \botmark for the preceding page.

\begin{verbatim}
\ifx\@empty\kept@firstmark \setbox\count@ \vbox to\dimen@ \{\unvbox\count@\vfill\} \fi
\end{verbatim}

Then the last column follows.

\begin{verbatim}
\setbox\mult@rightbox \vsplit\cclv to\dimen@ \set@keptmarks \ifshr\nking \setbox\mult@rightbox\vbox to\dimen@{\unvbox\mult@rightbox\vfill} \fi
\end{verbatim}

Having done this we hope that box 255 is emptied. If not, we reinsert its contents.

\begin{verbatim}
\ifvoid\@cclv \unvbox\@cclv \penalty\outputpenalty
\end{verbatim}

In this case a footnote that happens to fall into the leftover bit will be typeset on the wrong page. Therefore we warn the user if the current page contains footnotes. The older versions of multicols produced this warning regardless of whether or not footnotes were present, resulting in many unnecessary warnings.

\begin{verbatim}
\ifvoid\footins \else \PackageWarning{multicol} \MessageBreak Footnotes on page \thepage space might be wrong \fi
\end{verbatim}

If the `tracingmulticols` counter is 4 or higher we also add a rule.

\begin{verbatim}
\ifnum\c@tracingmulticols>\thr@@ \hrule\allowbreak \fi
\end{verbatim}

To get a correct marks for the current page we have to (locally redefine \firstmark and \botmark. If \kept@firstmark is non-empty then \kept@botmark must be non-empty too so we can use their values. Otherwise we use the value of \kept@topmark which was first initialized when we gathered the \particalpage and later on was updated to the \botmark for the preceding page.
We also initialize \topmark with \kept@topmark. This will make this mark okay for all middle pages of the multicols environment.

\let\topmark\kept@topmark
\message{Use kept top mark:\meaning\kept@topmark}
\message{Use kept first mark:\meaning\kept@firstmark}
\message{Use kept bot mark:\meaning\kept@botmark}
\message{Produce first mark:\meaning\firstmark}
\message{Produce bot mark:\meaning\botmark}
\@gobbletwo}
⟨/marktrace⟩
\message{(Re)Init top mark:\meaning\kept@topmark}
\@gobbletwo}
⟨/marktrace⟩

With a little more effort we could have done better. If we had, for example, recorded the shrinkage of the material in \partial@page it would be now possible to try higher values for \dimen@ (i.e. the column height) to overcome the problem with the nonempty box 255. But this would make the code even more complex so I skipped it in the current implementation.

Now we use \LaTeX’s standard output mechanism. Admittedly this is a funny way to do it.

\setbox\@cclv\vbox{\unvbox\partial@page}
\message{Colroom: \the\@colht = \the\@colroom}\@gobble}

This macro is used to subtract the amount of space occupied by footnotes for the current space from the space available for the current column. The space current column is stored in \dimen@. See above for the description of the default action.
\def\leave@mult@footins{\advance\dimen@-\skip\footins\advance\dimen@-\ht\footins}

We left out two macros: \process@deferreds and \speci@ls.
\def\speci@ls{%
\ifnum\outputpenalty <-\@Mi
\leave@mult@footins\
\advance\dimen@-\ht\footins
\else
\leave@mult@footins
\fi}

If the document ends in the middle of a multicols environment, e.g., if the user forgot the \end{multicols}, \TeX adds a very negative penalty to the end of the galley which is intended to signal the output routine that it is time to prepare for shipping out everything remaining. Since inside multicols the output routine of \LaTeX is disabled sometimes we better check for this case: if we find a very negative penalty we produce an error.
message and run the default output routine for this case.

\ifnum \outputpenalty<-\@MM
\PackageError{multicol}{Document end inside multicol\environment}{@ehd
\@specialoutput
\else
For the next block of code see comments in section 7.2.
\fi
\ifnum \outputpenalty = -\@MV
\mult@info\ne{Forced column break seen}\
\global \advance \vsize - \pagetotal
\global \setbox \colbreak@box
\vbox{\ifvoid \colbreak@box
\else
\unvbox \colbreak@box
\penalty - \@Mv
\fi
\unvbox \@cclv}
\reinsert@footnotes
\else
\fi
\fi
\ifnum \outputpenalty = -10001
\PackageWarningNoLine{multicol}{Floats and marginpars not allowed inside 'multicols' environment!}
\unvbox \@cclv \reinsert@footnotes
\else
\fi
\endgroup
\def \process@deferreds{\@floatplacement
\@tryfcolumn \@deferlist
\if@fcolmade\else
\begingroup
\let \@tempb \@deferlist
\let \@elt \@scolelt
\gdef \@deferlist{}\let \@elt \@scolelt
\@tempb \endgroup
\fi}
\newif{\shr@nking}
\def \raggedcolumns{\@bsphack \shr@nkingtrue \@esphack}
\def \flushcolumns{\@bsphack \shr@nkingfalse \@esphack}
\def \balance@columns@out{\@doclearpage \fi}
\PackageWarningNoLine{multicol}{Floats and marginpars not allowed inside 'multicols' environment!}
\@doclearpage
\ifx \@reserved@a \@empty
\PackageWarningNoLine{multicol}{Floats and marginpars not allowed inside 'multicols' environment!}
\fi
\@doclearpage
\texttt{\@doclearpage}
\process@deferreds is a simplified version of \LaTeX’s \texttt{@startpage}. We first call the macro \texttt{@floatplacement} to save the current user parameters in internal registers. Then we start a new group and save the \texttt{\@deferlist} temporarily in the macro \texttt{\@tempb}.
\def \process@deferreds{\@floatplacement
\@tryfcolumn \@deferlist
\if@fcolmade\else
\begingroup
\let \@tempb \@deferlist
\let \@elt \@scolelt
\gdef \@deferlist{}\let \@elt \@scolelt
\@tempb \endgroup
\fi}
\newif{\shr@nking}
\def \raggedcolumns{\@bsphack \shr@nkingtrue \@esphack}
\def \flushcolumns{\@bsphack \shr@nkingfalse \@esphack}
\def \balance@columns@out{\@doclearpage \fi}
\PackageWarningNoLine{multicol}{Floats and marginpars not allowed inside 'multicols' environment!}
\@doclearpage
\ifx \@reserved@a \@empty
\PackageWarningNoLine{multicol}{Floats and marginpars not allowed inside 'multicols' environment!}
\fi
\@doclearpage
\texttt{\@doclearpage}
\process@deferreds is a simplified version of \LaTeX’s \texttt{@startpage}. We first call the macro \texttt{@floatplacement} to save the current user parameters in internal registers. Then we start a new group and save the \texttt{\@deferlist} temporarily in the macro \texttt{\@tempb}.
\def \process@deferreds{\@floatplacement
\@tryfcolumn \@deferlist
\if@fcolmade\else
\begingroup
\let \@tempb \@deferlist
\let \@elt \@scolelt
\gdef \@deferlist{}\let \@elt \@scolelt
\@tempb \endgroup
\fi}
\newif{\shr@nking}
\def \raggedcolumns{\@bsphack \shr@nkingtrue \@esphack}
\def \flushcolumns{\@bsphack \shr@nkingfalse \@esphack}
\def \balance@columns@out{\@doclearpage \fi}
\PackageWarningNoLine{multicol}{Floats and marginpars not allowed inside 'multicols' environment!}
\@doclearpage
\ifx \@reserved@a \@empty
\PackageWarningNoLine{multicol}{Floats and marginpars not allowed inside 'multicols' environment!}
\fi
\@doclearpage
\texttt{\@doclearpage}
\process@deferreds is a simplified version of \LaTeX’s \texttt{@startpage}. We first call the macro \texttt{@floatplacement} to save the current user parameters in internal registers. Then we start a new group and save the \texttt{\@deferlist} temporarily in the macro \texttt{\@tempb}.
\def \process@deferreds{\@floatplacement
\@tryfcolumn \@deferlist
\if@fcolmade\else
\begingroup
\let \@tempb \@deferlist
\let \@elt \@scolelt
\gdef \@deferlist{}\let \@elt \@scolelt
\@tempb \endgroup
\fi}
\newif{\shr@nking}
\def \raggedcolumns{\@bsphack \shr@nkingtrue \@esphack}
\def \flushcolumns{\@bsphack \shr@nkingfalse \@esphack}
\def \balance@columns@out{\@doclearpage \fi}
\PackageWarningNoLine{multicol}{Floats and marginpars not allowed inside 'multicols' environment!}
\@doclearpage
\ifx \@reserved@a \@empty
\PackageWarningNoLine{multicol}{Floats and marginpars not allowed inside 'multicols' environment!}
\fi
\@doclearpage
\texttt{\@doclearpage}
\process@deferreds is a simplified version of \LaTeX’s \texttt{@startpage}. We first call the macro \texttt{@floatplacement} to save the current user parameters in internal registers. Then we start a new group and save the \texttt{\@deferlist} temporarily in the macro \texttt{\@tempb}.
\def \process@deferreds{\@floatplacement
\@tryfcolumn \@deferlist
\if@fcolmade\else
\begingroup
\let \@tempb \@deferlist
\let \@elt \@scolelt
\gdef \@deferlist{}\let \@elt \@scolelt
\@tempb \endgroup
\fi}
\newif{\shr@nking}
\def \raggedcolumns{\@bsphack \shr@nkingtrue \@esphack}
\def \flushcolumns{\@bsphack \shr@nkingfalse \@esphack}
\def \balance@columns@out{\@doclearpage \fi}
\PackageWarningNoLine{multicol}{Floats and marginpars not allowed inside 'multicols' environment!}
\@doclearpage
\ifx \@reserved@a \@empty
\PackageWarningNoLine{multicol}{Floats and marginpars not allowed inside 'multicols' environment!}
\fi
\@doclearpage
\texttt{\@doclearpage}
\process@deferreds is a simplified version of \LaTeX’s \texttt{@startpage}. We first call the macro \texttt{@floatplacement} to save the current user parameters in internal registers. Then we start a new group and save the \texttt{\@deferlist} temporarily in the macro \texttt{\@tempb}.
\def \process@deferreds{\@floatplacement
\@tryfcolumn \@deferlist
\if@fcolmade\else
\begingroup
\let \@tempb \@deferlist
\let \@elt \@scolelt
\gdef \@deferlist{}\let \@elt \@scolelt
\@tempb \endgroup
\fi}
\newif{\shr@nking}
\def \raggedcolumns{\@bsphack \shr@nkingtrue \@esphack}
\def \flushcolumns{\@bsphack \shr@nkingfalse \@esphack}
\def \balance@columns@out{\@doclearpage \fi}
\PackageWarningNoLine{multicol}{Floats and marginpars not allowed inside 'multicols' environment!}
\@doclearpage
\ifx \@reserved@a \@empty
\PackageWarningNoLine{multicol}{Floats and marginpars not allowed inside 'multicols' environment!}
\fi
\@doclearpage
\texttt{\@doclearpage}
\process@deferreds is a simplified version of \LaTeX’s \texttt{@startpage}. We first call the macro \texttt{@floatplacement} to save the current user parameters in internal registers. Then we start a new group and save the \texttt{\@deferlist} temporarily in the macro \texttt{\@tempb}.
\def \process@deferreds{\@floatplacement
\@tryfcolumn \@deferlist
\if@fcolmade\else
\begingroup
\let \@tempb \@deferlist
\let \@elt \@scolelt
\gdef \@deferlist{}\let \@elt \@scolelt
\@tempb \endgroup
\fi}
\newif{\shr@nking}
\def \raggedcolumns{\@bsphack \shr@nkingtrue \@esphack}
\def \flushcolumns{\@bsphack \shr@nkingfalse \@esphack}
\def \balance@columns@out{\@doclearpage \fi}
\PackageWarningNoLine{multicol}{Floats and marginpars not allowed inside 'multicols' environment!}
\@doclearpage
\ifx \@reserved@a \@empty
\PackageWarningNoLine{multicol}{Floats and marginpars not allowed inside 'multicols' environment!}
\fi
\@doclearpage
\texttt{\@doclearpage}
Then follow values assignments to get the \vsplitting right. We use the natural part of \topskip as the natural part for \splittopskip and allow for a bit of undershoot and overshoot by adding some stretch and shrink.

The next step is a bit tricky: when \TeX assembles material in a box, the first line isn’t preceded by interline glue, i.e. there is no parameter like \boxtopskip in \TeX. This means that the baseline of the first line in our box is at some unpredictable point depending on the height of the largest character in this line. But of course we want all columns to align properly at the baselines of their first lines. For this reason we have opened \mult\box with a \penalty -10000. This will now allow us to split off from \mult\box a tiny bit (in fact nothing since the first possible break-point is the first item in the box). The result is that \splittopskip is inserted at the top of \mult\box which is exactly what we like to achieve.

Next we try to find a suitable starting point for the calculation of the column height. It should be less than the height finally chosen, but large enough to reach this final value in only a few iterations. The formula which is now implemented will try to start with the nearest value which is a multiple of \baselineskip. The coding is slightly tricky in \TeX and there are perhaps better ways.

The code above sets \tempdim to the length of a column if we simply divide the whole box into equal pieces. To get to the next lower multiple of \baselineskip we convert this dimen to a number (the number of scaled points) then divide this by \baselineskip (also in scaled points) and then multiply this result with \baselineskip assigning the result to \dimen. This makes \dimen ≤ to \tempdim.

Next step is to correct our result by taking into account the difference between \topskip and \baselineskip. We start by adding \topskip; if
this makes the result too large then we have to subtract one \baselineskip.
\advance\dimen@\topskip
\ifdim\dimen@>\@tempdima
\advance\dimen@-\baselineskip
\fi
At the user's request we start with a higher value (or lower, but this usually only increases the number of tries).
\advance\dimen@\c@unbalance\baselineskip
We type out statistics if we were asked to do so.
\mult@info\one
{Balance columns on line:}
\tempcnta\count@
\the\tempcnta
\the\topskip\space (corrected)\%
\dimen@\topskip
\fi
Now we try to find the final column height. We start by setting \vbadness to infinity (i.e. 10000) to suppress underfull box reports while we are trying to find an acceptable solution. We do not need to do it in a group since at the end of the output routine everything will be restored. The setting of the final columns will nearly always produce underfull boxes with badness 10000 so there is no point in warning the user about it.
\vbadness\z@
We also allow for nonsense values for a start.
\ifnum\dimen@<\topskip
\mult@info\one
{Start value \the\dimen@ -> \the\topskip (corrected)}\%
\dimen@\topskip
\fi
\vfuzz \col@number\baselineskip
The variable \last@try will hold the dimension used in the previous trial splitting. We initialize it with a negative value.
\last@try\p0
\loop
In order not to clutter up \TeX's valuable main memory with things that are no longer needed, we empty all globally used box registers. This is necessary if we return to this point after an unsuccessful trial. We use \process@cols for this purpose, starting with \mult@grightbox. Note the extra braces around this macro call. They are needed since Plain \TeX's \loop...\repeat mechanism cannot be nested on the same level of grouping.
\process@cols\mult@grightbox
\process@cols\mult@firstbox
\global\setbox\count@\box\voidb@
\global\setbox\mult@grightbox\copy\mult@box
We start with the assumption that the trial will be successful. If we end up with a solution that is too bad we set too@bad to true.
⟨∗badness⟩
\global\too@badfalse
⟨/badness⟩
Using \vsplit we extract the other columns from box register \mult@grightbox. This leaves box register \mult@box untouched so that we can start over again if this trial was unsuccessful.
\process@cols\mult@firstbox
\global\setbox\count@\vsplit\mult@grightbox to\dimen@
After every split we check the badness of the resulting column, normally the amount of extra white in the column.
\ifnum\c@tracingmulticols\@ne
\tempcnta\count@
\divide\tempcnta\tw@
\message{Column \number\tempcnta badness: \the\badness}\
\fi
If this badness is larger than the allowed column badness we reject this solution by setting too@bad to true.
⟨∗badness⟩
\ifnum\badness>\c@columnbadness
\message{too bad (>\the\c@columnbadness)}\
\global\too@badtrue
\fi
⟨/badness⟩
There is one subtle point here: while all other constructed boxes have a depth that is determined by \splitmaxdepth the last box will get a natural depth disregarding the original setting and the value of \splitmaxdepth or \boxmaxdepth. This means that we may end up with a very large depth in box \mult@grightbox which would make the result of the testing incorrect. So we change the value by unboxing the box into itself.
We check whether our trial was successful. The test used is very simple: we merely compare the first and the last column. Thus the intermediate columns may be longer than the first if \raggedcolumns is used. If the right-most column is longer than the first then we start over with a larger value for \dimen@.

If the height of the last box is too large we mark this trial as unsuccessful.

To ensure that there isn’t a forced break in the last column we try to split off a box of size \maxdimen from \mult\rightbox (or rather from a copy of it). This should result in a void box after the split, unless there was a forced break somewhere within the column in which case the material after the break would have stayed in the box.

Thus if \tempboxa is void we have a valid solution. In this case we take a closer look at the last column to decide if this column should be made as long as all other columns or if it should be allowed to be shorter. For this we first have to rebox the column into a box of the appropriate height. If tracing is enabled we then display the badness for this box.

We then compare this badness with the allowed badness for the final column. If it does not exceed this value we use the box, otherwise we rebox it once more and add some glue at the bottom.

We now compare the badness with the allowed badness for the final column. If it does not exceed this value we use the box, otherwise we add some glue at the bottom.

With \TeX version 3.141 it is now possible to use L\TeX's \newlinechar in the \message command, but people with older \TeX versions will now get "\"J instead of a new line on the screen.
If \@tempboxa above was not void our trial was unsuccessful and we report this fact and try again.

If the natural height of the first box is smaller than the current trial size but is larger than the previous trial size it is likely that we have missed a potentially better solution. (This could have happened if for some reason our first trial size was too high.) In that case we dismiss this trial and restart using the natural height for the next trial.

Finally the switch too@bad is tested. If it was made true either earlier on or due to a rightmost column being too large we try again with a slightly larger value for \dimen@.

4.5 The box allocations

Early releases of these macros used the first box registers 0, 2, 4,... for global boxes and 1, 3, 5,... for the corresponding local boxes. (You might still find some traces of this setup in the documentation, sigh.) This produced a problem at the moment we had more than 5 columns because then officially allocated boxes were overwritten by the algorithm. The new release now uses private box registers.
New macros and hacks for version 1.2

If we don’t use \TeX 3.0 \emergencystretch is undefined so in this case we simply add it as an unused \texttt{(dimen)} register.

\newif\@tempa
\@ifundefined{emergencystretch}{}{
ewdimen\emergencystretch}

My tests showed that the following formula worked pretty well. Nevertheless the \texttt{\setemergencystretch} macro also gets \texttt{\hsize} as second argument to enable the user to try different formulae.

\def\setemergencystretch#1#2{%\emergencystretch 4pt \multiply\emergencystretch#1}

Even if this should be used as a hook we use a @ in the name since it is more for experts.

\def\set@floatcmds{%\let\@dblfloat\@dbflt \def\end@dblfloat{\par \vskip\z@ \@minipagefalse \egroup \color@endbox \@largefloatcheck \outer@nobreak \this is cheap (deferring the floats until after the current page) but any other solution would go deep into \LaTeX’s output routine and I don’t like to work on it until I know which parts of the output routine have to be reimplemented anyway for \LaTeX3. \ifnum\@floatpenalty<\z@ \@cons\@deferlist\@currbox \fi \ifnum\@floatpenalty=-\@Mii \@Esphack \fi}}

Maintaining the mark registers

This section contains the routines that set the marks so that they will be handled correctly. They have been introduced with version 1.4.

First thing we do is to reserve three macro names to hold the replacement text for \TeX’s primitives \texttt{\firstmark}, \texttt{\botmark} and \texttt{\topmark}. We initialize the first two to be empty and \texttt{\kept@topmark} to contain two empty pair of braces. This is necessary since \texttt{\kept@topmark} is supposed to contain the last mark from a preceding page and in \LaTeX any “real” mark must contain two parts representing left and right mark information.

\def\kept@topmark{{}{}}
\let\kept@firstmark\@empty
\let\kept@botmark\@empty

Sometimes we want to return the value of a “kept” mark into a \texttt{\mark} node on the main vertical list. This is done by the function \texttt{\return@nonemptymark}. As the name suggests it only acts if the replacement text of the kept mark is non-empty. This is done to avoid adding an empty mark when no mark was actually present. If we would nevertheless add such a mark it would be regarded as a valid \texttt{\firstmark} later on.

\def\return@nonemptymark#1#2{%\ifx#2\@empty\else\fi\mult@info\tw@{Returned #1 mark:\MessageBreak \meaning#2}%%\fi\langle/\marktrace\rangle}

Since the contents of the mark may be arbitrary \LaTeX code we better make sure that it doesn’t get expanded any further. (Some expansion have been done already during the execution of \texttt{\markright} or \texttt{\maketitle} we would get stripped out for production.

\(*\marktrace\*)
\mult@info\tw@{(Returned #1 mark:\MessageBreak \meaning#2)}\%\nobreak
\%\fi\langle/\marktrace\rangle

\footnote{Due to the current definition of \texttt{\markright} etc. it wouldn’t help to define the \texttt{\protect} command to prohibit expansion as any \texttt{\protect} has already vanished due to earlier expansions.}
We therefore use the usual mechanism of a toks register to prohibit expansion.\footnote{We don't want any breakpoint between such a returned mark and the following material (which is usually just the box where the mark came from).}

```
\toks@\expandafter{#2}\
\mark{\the\toks@}\
```

If we have some material in a box register we may want to get the first and the last mark out of this box. This can be done with \get@keptmarks which takes one argument: the box register number or its nick name defined by \newbox.

```
\def\get@keptmarks#1{%
  \begingroup
  \vbadness\@M
  \setbox#1\copy#1%
  \setbox#1\vsplit#1\to\maxdimen
  \set@keptmarks
  \endgroup

  \typeout{Mark box #1 before: ht \the\ht#1, dp \the\dp#1}\
  \mult@info\tw@
    \{Set kept first mark:\MessageBreak\meaning\kept@firstmark%\@gobbletwo}%
%
  \fi
  \mult@info\tw@
    \{Set kept bot mark:\MessageBreak\meaning\kept@botmark%\@gobbletwo}%
%
  \if\kept@firstmark\@empty\else
    \global\let\kept@botmark\@tempa
    \mult@info\tw@
      \{Set kept bot mark:\MessageBreak\meaning\kept@botmark%\@gobbletwo}%
  \fi}
```

The \prep@keptmarks function is used to initialize the kept marks from the contents of \partial@page, i.e. the box that holds everything from the top of the current page prior to starting the multicols environment. However, such a box is only available if we are not producing a boxed multicols.

```
\def\prep@keptmarks{%
  \global\let\kept@botmark\@tempa
  \expandafter\def\expandafter\@tempa\expandafter{%
    \meaning\kept@botmark%\@gobbletwo}%
%
  \if\kept@firstmark\@empty\else
    \global\let\kept@botmark\@tempa
    \mult@info\tw@
      \{Set kept bot mark:\MessageBreak\meaning\kept@botmark%\@gobbletwo}%
  \fi
}
```

For debugging we take a look at the current dimensions of the box since in earlier versions of the code I made some mistakes in this area.

```
\typeout{Mark box #1 \space after: ht \the\ht#1, dp \the\dp#1}\
```

We always try to set the bottom mark to the \splitbotmark but of course only when there has been a \splitbotmark at all. Again, we assume that an empty \splitbotmark means that the split off box part didn’t contain any marks at all.

```
\expandafter\def\expandafter\@tempa\expandafter{%
  \meaning\kept@botmark%\@gobbletwo}%
```

The \set@keptmarks is responsible for setting \kept@firstmark and \kept@botmark, by checking the current values for \splitfirstmark and \splitbotmark.

```
\def\set@keptmarks{%
  \if\kept@firstmark\@empty
    \expandafter\def\expandafter\@tempa\expandafter{%
      \meaning\kept@botmark%\@gobbletwo}%
  \fi
}
```

We now put the contents of \splitfirstmark into \kept@firstmark. In the case that there wasn’t any mark at all \kept@firstmark will not change by that operation.

```
\if\kept@firstmark\@empty\else
  \expandafter\def\expandafter\@tempa\expandafter{%
    \meaning\kept@botmark%\@gobbletwo}%
  \fi
```

When debugging we show the assignment but only when something actually happened.

```
\if\kept@firstmark\@empty\else
  \expandafter\def\expandafter\@tempa\expandafter{%
    \meaning\kept@botmark%\@gobbletwo}%
  \fi
```

We don't want any breakpoint between such a returned mark and the following material (which is usually just the box where the mark came from).
\def\remove@discardable@items{\langle∗debug\rangle
def\@tempa{s=\the\lastskip,\n p=\the\lastpenalty,\n k=\the\lastkern}\n\typeout\@tempa\n\langle/\debug\rangle\n\unskip\unpenalty\unkern\n\langle∗debug\rangle\n\edef\@tempa{s=\the\lastskip,\n p=\the\lastpenalty,\n k=\the\lastkern}\n\typeout\@tempa\n\langle/\debug\rangle\n\unskip\unpenalty\unkern\n\langle∗debug\rangle\n\edef\@tempa{s=\the\lastskip,\n p=\the\lastpenalty,\n k=\the\lastkern}\n\typeout\@tempa\n\langle/\debug\rangle\n\unskip\unpenalty\unkern\n\langle∗debug\rangle\n\edef\@tempa{s=\the\lastskip,\n p=\the\lastpenalty,\n k=\the\lastkern}\n\typeout\@tempa\n\langle/\debug\rangle\n\unskip\unpenalty\unkern\n\langle∗badness\rangle\n\newif\iftoo@bad\n\newcount\c@columnbadness\n\c@columnbadness=10000\n\newcount\c@finalcolumnbadness\n\c@finalcolumnbadness=9999\n\newdimen\last@try\n\newdimen\multicolovershoot\n\newdimen\multicolundershoot\n\multicolovershoot=0pt\n\multicolundershoot=2pt\n\newbox\mult@nat@firstbox\n\def\mult@info#1#2{\ifnum\c@tracingmulticols>#1\n\GenericWarning{(multicol)\@spaces\@spaces}{}\n{\textwidth}{\@footnotetext} done by a class will correctly survive and second if \textwidth for footnotes rather than using the original definition. Starting with version v1.5r this is now done in a way that the original definition is still used, except that locally is set to . This solves two problems: first redefinitions of the \multicols command further up for the exact code.

7 Further extensions

This section does contain code for extensions added to this package over time. Not all of them may be active, some might sit dormant and wait for being activated in some later release.

7.1 Not balancing the columns

This is fairly trivial to implement. we just have to disable the balancing output routine and replace it by the one that ships out the other pages.

The code for this environment was suggested by Matthias Clasen.
warn that we balance anyway.

\PackageWarning{multicol}{multicols* inside a box does not make sense.\MessageBreak Going to balance anyway}%
\else
\let\balance@columns@out\multi@column@out
\fi
\begin{multicols}

When ending the environment we simply end the inner \texttt{multicols} environment, except that we better also stick in some stretchable vertical glue so that the last column still containing text is not vertically stretched out.

\@namedef{endmulticols*}{\vfill\end{multicols}}
\end{multicols}

\begin{multicols}

\@namedef{endmulticols*}{\vfill\end{multicols}}
\end{multicols}

7.2 Manual column breaking

The problem with manual page breaks within \texttt{multicols} is the fact that during collection of material for all columns a page-forcing penalty (i.e. -10000 or higher) would stop the collecting pass which is not quite what is desired. On the other hand, using a penalty like -9999 would mean that there would be occasions where the \texttt{\vsplit}ing operations within \texttt{multicols} would ignore that penalty and still choose a different break point.

For this reason the current implementation uses a completely different approach. In a nutshell it extends the \LaTeX output routine handling by introducing an additional penalty flag (i.e., a penalty which is forcing but higher than -10000 so that the output routine can look at this value and thus knows why it has been called).

Inside the output routine we test for this value and if it appears we do two things: save the galley up to this point in a special box for later use and reduce the \texttt{\vsze} by the height of the material seen. This way the forcing penalty is now hidden in that box and we can restart the collection process for the remaining columns. (This is done in \texttt{\speci@ls} above.)

In the output routines that do the \texttt{\vsplit}ing either for balancing or for a full page we simply combine box 255 with the saved box thus getting a single box for splitting which now contains forcing breaks in the right positions.
Index

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

Symbols
\@footnotetext \ldots 835 \endmulticols \ldots 214 \mult@cols \ldots 81 \textbullet
\@footnotetext* \ldots 848 \mult@firstbox \ldots 687 \multi@column@out \ldots 814 \textbullet
\@footnotetext \ldots 120 \mult@footnotetext \ldots 78, 835 \partial@page \ldots 271 \textbullet
\@footnotetext \ldots 72 \mult@gfirstbox \ldots 687 \postrultural@page \ldots 271 \textbullet
\@footnotetext \ldots 494 \mult@grightbox \ldots 687 \pre@multicols \ldots 271 \textbullet
\@footnotetext \ldots 271 \mult@info \ldots 828 \preparation@multicols \ldots 779 \textbullet
\@collectmore \ldots 721 \mult@leftmargin \ldots 835 \prepare@multicols \ldots 137 \textbullet
\@columnbadness \ldots 815 \multicolbaselineskip \ldots 344 \preparation@multicols \ldots 295 \textbullet
\@finalcolumnbadness \ldots 815 \multicol@leftmargin \ldots 783 \process@cols \ldots 184 \textbullet
\@unbalance \ldots 272 \multicol@leftmargin \ldots 828 \process@deferreds \ldots 724 \textbullet
\@number \ldots 271 \multicol@leftmargin \ldots 271 \process@float@cmds \ldots 78 \textbullet
\@break@box \ldots 868 \multicol@leftmargin \ldots 184 \process@float@cmds \ldots 83 \textbullet
\@column@out \ldots 851 \multicol@leftmargin \ldots 346 \return@nonemptymark \ldots 721 \textbullet
\@column@out \ldots 2 \multicol@leftmargin \ldots 687 \set@firstmark \ldots 271 \textbullet
\@column@rule \ldots 2 \multicolor \ldots 344 \set@float@cmds \ldots 706 \textbullet
\@emergency@stretch \ldots 701 \multicolor@sep \ldots 271 \set@keptmarks \ldots 754 \textbullet
\@emergency@stretch \ldots 445 \multicolor@sep \ldots 687 \set@multi@vsize \ldots 344 \textbullet
\@emergency@stretch \ldots 2 \multicolor@sep \ldots 271 \set@spec@il@is \ldots 201 \textbullet

Change History

v1.0c \enou\gh@room: Penalty 0 added to empty the contribution list. \ldots 9
v1.0d General: All lines shortened to 72 or less. \ldots 1
v1.0e General: Redefinition of description env. to use \description@margin=5pt in documentation. \ldots 1
\prepare@multicols: \textwidth changed to \linewidth. \ldots 11
Setting of \columnwidth removed. \ldots 12
So this file will work with the ‘twocolumn’ command. \ldots 11
v1.0f General: Changed \z@ to 0pt in redefinition of description. \ldots 1
v1.1a General: \multicols@sep changed to \multicol@sep. \ldots 1
\flush@columns: \flushed@columns renamed to \flushcolumns. \ldots 18
v1.1b \balance@columns: Group around main loop re-moved. \ldots 20
\prepare@multicols: \pretolerance -1 because it nearly never succeeds. \ldots 11
\set@float@cmds added. \ldots 11
\setspec@il@is@stretch added. \ldots 11
\vb@d@ness 10001 now. \ldots 11
\set@float@cmds: Macro added. \ldots 23
\setspec@il@is@stretch: Macro added. \ldots 23
\spec@il@is: Float boxes freed. \ldots 18
v1.3a \balance@columns: Changed \vtop to \vbox. \ldots 22
v1.3b \endmulticols: Do \penalty with \addpenalty 13
\enough@room: Do \penalty with \addpenalty 9
\multicols: Minimum of two columns \ldots 8
v1.3c \balance@columns: \global@advance left over from older code \ldots 21
Limit column height to \@colroom \ldots 22
\endmulticols: Check closing env. \ldots 13
\multi@column@out: \unbox avoided. \ldots 16
Check if footnotes are actually present before issuing a warning. \ldots 16
Unnecessary code removed \ldots 17

27
General: Added support for multicol in inner mode
\balance@columns: Changed to proper endlinechar in message
\mult@col: Forgotten braces added
\mult@column@out: \botmark set to \splitbotmark
\prepare@multicols: Checking for text losses.
\kept@botmark: reinit \topmark locally zeroed.
\vsize: Setting of \emergencystretch on top removed.
\multicols: Maximum of 5 columns (temp)
\mult@col: \@totalleftmargin now in \prepare@multicols
\page@sofar: use \multicol@leftmargin instead of \@totalleftmargin
\prepare@multicols: saved \@totalleftmargin
\endmulticols: Check \partial@page being emptied
\return@nonemptymark: Make marks robust
\prepare@multicols: \topskip locally zeroed.
\mult@column@out: Use different \vsize setting
\prepare@multicols: Code moved to \set@mult@vsize
Use different \vsize setting
\set@mult@vsize: Macro added.
\balance@columns: Allow columns to come out a bit long or short
\multicol@out: New box mechanism
\prepare@multicols: Add offset to \doublecolnumber
\multicol@out: New box mechanism
\page@sofar: New box mechanism
\prepare@multicols: Use \init@mult@footins
\multicol@out: New badness mechanism
\balance@columns@out: added penalty at output routine exit
\endmulticols: Again use \penalty
\mult@column@out: Support \clearpage
\specials: Support \clearpage
\mult@column@out: reinit \topmark
\enough@room: Assign arg to skip register to be able to output value
\set@floatcmds: Updated since floats have changed
\balance@columns: Get kept marks first
\page@sofar: Check for void boxes
\set@floatcmds: Updated since floats have changed again
v1.5l
General: Try hard to explain unresolved reference that happens if \OnlyDescription is
used ................................. 6
\set@floatcmds: Added \@largefloatcheck . . 23
v1.5n
General: Applied improvement of documentation, kindly done by Robin Fairbairns. . . . 1
\set@floatcmds
\set@floatcmds: Added \@largefloatcheck. . 23
\set@floatcmds: Added \@largefloatcheck. . 23
v1.5o
\@footnotetext: Redefinition added pr/2664. . 25
\prepare@multicols: Setting of \columnwidth
added again pr/2664. ........................ 12
\multicols: Redefinition of \@footnotetext
only within env pr/2689. ........................ 8
\balance@columns@out: Support \columnbreak 18
\colbreak@box: Macro added ........................ 26
\columnbreak: Macro added ........................ 26
\multi@column@out: Support \columnbreak . . 16
\speci@ls: Support \columnbreak .............. 18
v1.5v
\balance@columns: Added tracing statements
for trial unsuccessful .......................... 21
Check last column if it contains forced break
and reject trial if that is the case .......... 21
\balance@columns@out: Added debug state-
ments for column break support ............ 18
\multicols: Make \@footnotetext long to allow
multi-paragraph footnotes. ........................ 8
\endmulticols: Detect and fix problem if a mul-
tics ends at the top of a page ............. 14
\balance@columns: Limit column height only in
unrestricted mode (pr/3212) ............... 22
\page@sofar: Ensure that column rule has al-
ways \normalcolor .......................... 15
v1.5w
\balance@columns@out: Added debug statements
for column break support .................. 16
\speci@ls: Added debug statements for column
break support ............................. 18
\c@finalcolumnbadness: Change wrong default
for \multicolovershoot to zero (pr/3465). 25
\mult@cols: Add a kern to cancel potential
depth of previous line ..................... 9
\page@sofar: Suppress interline glue at this
point ................................ 15
v1.6a
General: New option grid ........................ 7
\endmulticols: Adjusting \prevdepth ........ 13
\mult@cols: Adjust spacing ................... 9
\page@sofar: Now adjusting \prevdepth .... 15
Preparing for adjusting \prevdepth ....... 15
v1.6b
\page@sofar: Different info display .......... 15
v1.6c
\set@mult@vsize: Collect one line per column
more .................................. 12
\endmulticols: Catch problem with
\columnbreak in last line ................. 13
v1.6d
\multicols: Avoid self-referencing definition of
\@footnotetext (pr/3618) ............ 8
\balance@columns: /colbreak guard in the
wrong position ............................ 22
need to use \mult@grightbox in the loop .. 21
v1.6e
\set@floatcmds: Added \minipagefalse 23