An environment for multicolumn output*

Frank Mittelbach
Email: see top of the source file

Printed June 8, 2023

This file is maintained by the \LaTeX{} Project team. Bug reports can be opened (category tools) at https://latex-project.org/bugs.html.

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 this implementation. At the moment only page-wide floats (i.e., star-forms) can be used within the scope of the environment.

Preface to version 1.8

The 1.8 release improves on the balancing approach. If due to a limited number of break points (e.g., due to large objects) the balanced columns exceed the available vertical space, then balancing is canceled and a normal page is produced first. Some overflow is allowed (controlled by the parameter \texttt{maxbalancingoverflow} which defaults to 12pt). This ensures that we only cut a normal page if we get enough material carried over to next page.

Also added was support for \texttt{enlargethispage}. This means it is now possible to request a page to be artificially enlarged or shortened. Note that if you enlarge pages by more than one line you may have to increase the \texttt{collectmore} counter value to ensure that enough material is being picked up.

This command was used on the second page of this manual to shorten it by one line, in order to get rid of a number of widow lines on the following pages.

There are also some small enhancements to the balancing algorithm including a ways to require a minimum number of rows in the result.

Finally, version 1.8 adds the command \texttt{docolaction} to help with more complicated actions that depend on the current column. This command expects 3 arguments: code that is executed if we are in the “first” column, code to execute if we end up in any “middle” column (if there are more than two) and finally code to execute if we are in the “last” column. Thus

\begin{verbatim}
\docolaction{first}
  {middle}{last}
\end{verbatim}

would typeset a different word depending the type of column this code is executed. Using it like this is probably pointless, but you can imagine applications like writing something into the nearest margin, etc.

As this feature needs at least two \LaTeX{} runs to produce correct results and as it adds to the processing complexity it is only made available if one add the option \texttt{colaction} when loading the package.

\footnotesize
\begin{flushleft}
*This file has version number v1.9f, last revised 2023/03/30.
†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.
\end{flushleft}
Preface to version 1.7 (right to left support)

The 1.7 release adds support for languages that are typeset right-to-left. For those languages the order of the columns on the page also need to be reversed—something that wasn’t supported before. The next paragraph demonstrates the result (as it is typeset as if we are writing in a left-to-right language—so read the rightmost column first). The change is initialized via \RLmulticolcolumns and returning to left-right (default) is done via \LRmulticolcolumns.

For example:

\renewcommand \footnoterule{\% \kern-3pt \hbox to \textwidth \{\hskip .6\textwidth \hrulefill \} \kern2.6pt}

Right-to-left typesetting will only reverse the column orders. Any other support needed will have to be provided by other means, e.g., using appropriate fonts and reversing the writing directions within the columns. As footnotes are typeset in full measure the footnote rule needs to be redefined as if they are below a single column, i.e., using \textwidth not \columnwidth.

Preface to version 1.5 + 1.6

The 1.5 release contains two major changes: 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 multicols has been upgraded to run under \LaTeX{} 2ε.

Later changes to 1.5 include \columnbreak and multicols*. For version 1.6 micro-spacing around the boxes produced by 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 \twocolumn or \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 \doc.sty (see “The doc-Option”, 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\(^1\) 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 please direct bug reports and suggestions back to Mainz.

2 The User Interface

To use the environment one simply says

\begin{multicols}{⟨number⟩}
⟨multicolumn text⟩
\end{multicols}

where ⟨number⟩ is the required number of columns and ⟨multicolumn text⟩ may contain arbitrary \LaTeX{} commands, except that floats and marginpars are not allowed in the current implementation\(^2\).

As its first action, the multicols environment measures the current page to determine whether there is enough room for some portion of multicolumn output.

\(^1\)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 improving the original code.)

\(^2\)This is dictated by lack of time. To implement floats one has to reimplement the whole \LaTeX{} output routine.
put. 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{\multicols} 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{\premulticols} available. Again we add \texttt{\multicols} 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}] ... \\
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 multicols environment you could write

\begin{verbatim}
\begin{multicols}{3}
  \[section{Index}
   This index contains ...] [6cm]
  ... \\
\end{verbatim}
\end{verbatim}

The space between columns is controlled by the length parameter \texttt{\columnsep}. 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{\multicolsrule} to some positive value. In this article a value of \texttt{4pt} was used.

The color of the rules separating the columns can be specified through \texttt{\multicolsrulecolor}. The default value is \texttt{normalcolor}.

Since narrow columns tend to need adjustments in interline spacing we also provide a \texttt{(skip)} parameter called \texttt{\multicolsbaselineskip} 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 \TeX{} 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 = \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{\multicols} 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{\multicols} at the beginning of every \texttt{multicols} environment one can locally overwrite this default by assigning \texttt{\tolerance} = \langle desired value \rangle. There also exists a \texttt{\multicols} 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 \texttt{multicols} environment is reached. In the first step \TeX{} might consider more material whilst finding the final column content than it actually uses 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{This foot-
ament gives a warning message when it is unable to use all the
aterial considered so far.

If you don’t use footnotes too often the chances of something actu-
ally going wrong are very slim, but if this happens you can help \TeX
by using a `\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
`\setcounter` declaration to set this counter to \langle number\rangle, \TeX
will consider \langle number\rangle more (or less) lines before making its fi-
nal decision. So a value of $-1$ 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 `\raggedcolumns` if you
don’t want the bottom lines to be aligned. The default is `\flushcolumns`, so \TeX\ will normally try to make both the
top and bottom baselines of all columns align.

If there is only a small amount of material available for bal-
ancing then you may end up with very few lines per column. In an
extreme case there may be only one line which looks distinctly
odd. In that case it might be better to have more material dis-
tributed to the earlier columns even if that means that later
columns are empty or partially empty. This is controlled through
the counter `minrows` (default 1). If set to a higher value then the
balancing will have at least that many rows in the first column
(and also all further columns until it runs out of material).

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

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 ver-
sion 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 ig-
nores solutions where at least one column has a badness be-
ing larger than the value of the counter `columnbadness`. The de-
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 ex-
treme 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 oth-
ers. 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 fi-
nalcolumnbadness. The default setting is 9999, thus preventing
the stretching for all columns that \TeX\ would consider infi-
nitely bad. In that case the fi-
nal column is allowed to run short
which gives a much better result.

And there are two more parameters of some experimental
ature, one called `multicolovershoot` the other
`multicolundershoot`. They
control the amount of space a col-
umn within the `multicols` environ-
ment is allowed to be “too full”
or “too short” without affecting the
column badness. They are
set to 0pt and 2pt, respectively.

Finally, when doing the bal-
ancing at the end, columns
may become higher than the
remaining available space. In
that case the algorithm aborts
and instead generates a normal
page. However, if the amount
is not too large, e.g., a line or
so, then it might be better to
keep everything on the same
page instead of starting a new
page with just one line after
balancing. So the parameter
`maxbalancingoverflow` gov-
erns this process: only when the
excess gets larger than its value
balancing is aborted.
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 v1.5q this now exists: if you use \multicols*(slightly desirable) to the usual environment the columns on the last page are not balanced. Of course, this environment only works on top-level, e.g., inside a box one has to balance to determine a column height in absence of a fixed value.

2.3 Manually breaking columns

Another request often voiced was: “How do I tell L\TeX{} that it should break the first column after this particular line?” The \pagebreak command (which works with the two-column option of L\TeX) is of no use here since it would end the collection phase of \multicols and thus all columns on that page. So with version 1.5u the \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 artificially forced into the second column (resulting in some white space between paragraphs in the first column).

From version 1.9 onwards \columnbreak accepts an optional argument (just like \pagebreak) in which you can specify the desirability to break the column at that point: supported values are 0 (slightly desirable) to 4 (forced). This version also adds \newcolumnt which forces a column break but runs the column short (comparable to \newpage).

2.4 Floats inside a multicols environment

Within the \multicols environment the usual star float commands are available but their function is somewhat different as in the twocolumn mode of standard L\TeX. Starred floats, e.g., figure*, denote page wide floats that are handled in a similar fashion as normal floats outside the \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 t, b, and/or p in their optional argument, but 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 \topfraction, etc. rather than by \dbl....

2.5 Support for right-to-left typesetting

In right-to-left typesetting the order of the columns on the page also need to be reversed, i.e., the first column has to appear on the far right and the last column on the left. This is supported through the commands \RLmulticolcolumns (switching to right-to-left typesetting) and \Lmulticolcolumns (switching to left-to-right typesetting) the latter being the default.

2.6 Warnings

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

Underfull \hbox{ (badness ...)}

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 as the badness value is below 10000 the result is probably acceptable.

Underfull \vbox ... while \output is active

If a column contains a 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.

\LaTeX Warning: I moved some lines to the next page

As mentioned above, \multicols sometimes screws up the footnote numbering. As a precaution, whenever there is a footnote on a page where \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 \newpage or \pagebreak[..].

Floats and marginpars not allowed inside ‘multicols’ environment!

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

Very deep columns! Grid alignment might be broken!

This message can only appear if the option grid was chosen. In that case it will show up if a column has a very large depth so that \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.

2.7 Tracing the output

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

\langle number \rangle = 1. \TeX will now tell you, whenever it enters or leaves a \multicols environment, the number of columns it is working on and its decision about starting a new page before or after the environment.

\langle number \rangle = 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 \raggedcolumns declaration is in force and the value of the 'unbalance' counter if positive.

\langle number \rangle = 3. Setting \langle 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 number \rangle \geq 4. Setting \langle number \rangle to such a high value will additionally place an \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

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

The only restriction to such inner \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 can not 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 \multicols: The balancing routine is now placed into a single macro which is called \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 \badness function of \TeX3, 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 \multicol.dtx directly!

3.2 Preface to version 1.2

After the article about the \multicols environment was published in \textsc{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 \multicoladjdemerits to be used for \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 \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 \tolerance (or \pretolerance in the first pass). If this isn’t pos-
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@firstbox, the next in register \mult@firstbox + 2, ..., only the last one as an exception in register \mult@grightbox. Furthermore it has to set up the two 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>
<thead>
<tr>
<th>Table 1: Interface description for \balance@columns</th>
</tr>
</thead>
</table>
| 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 \texttt{−1000} for \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 \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 \emergencystretch is probably incorrect. I made a few tests but of course one has to 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 \TeXs \@freelist so that after a few \marginpars inside the multicols environment floats where disabled forever. (Thanks to Chris Rowley for pointing this out.) I removed this misbehavior 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.

```latex
\documentclass{ltxdoc}
\usepackage{multicol}
\usepackage{docstrip}
\usepackage{balance}
\begin{document}
\包{balance}
\end{document}
```

We use the balancingshow option when loading multicols so that full tracing is produced. This has to be done before the doc package is loaded, since doc
\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.

otherwise requires multicols without any options.
3 \usepackage{multicol}{[1999/05/25]}
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 multicols environment finds enough space at the bottom of the page.
15 \setlength{\premulticols}{0\baselineskip}
When balancing columns we disregard solutions that are too bad. Also, if the last column is too bad we typeset it without stretch.

4.2 Identification and option processing

We start by identifying the package. Since it makes use of features only available in \LaTeX{} we ensure that this format is available. (Now this is done earlier in the file.)
37 ⟨\begin{package}
38 % \NeedsTeXFormat{LaTeX2e}
39 % \ProvidesPackage{multicol}[]{\ldots/\ldots/\ldots}
40 % \v\ldots\multicolumn formatting

Next we declare options supported by multicols. Two-column mode and multicols do not work together so we warn about possible problems. However, since you can revert to \texttt{\onecolumn} in which case multicols does work, we don’t make this an error.
41 \DeclareOption{twocolumn}

The index is supposed to come out in four columns. And we don’t show macro names in the margin.
16 \setcounter{columnbadness}{7000}
17 \setcounter{finalcolumnbadness}{7000}

Line numbers are very small for this article.
25 \renewcommand{\theCodelineNo}{\scriptsize\rm\arabic{CodelineNo}}
26 \settowidth{\MacroIndent}{\scriptsize\rm 00\ }

Tracing is done using a counter. However it is also possible to invoke the tracing using the options declared below.
42 \PackageWarning{multicol}{May not work with the twocolumn option}
The next option is intended for typesetting on a \baselineskip grid. Right now it doesn’t do anything other than warning if it thinks that the grid got lost.

Next option enables the \docolaction command. As this changes the .aux file content this is not automatically enabled.

4.3 Starting and Ending the \multicols Environment

\multicols

As mentioned before, the \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 \col@number register.

If the user forgot the argument, \LaTeX 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.

We have only enough box registers for twenty columns, so we need to check that the user hasn’t asked for more.

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 \multicols 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 \LaTeX stack overflow as this would generate a self-referencing definition.

Now we can safely look for the optional arguments.

As mentioned before, the \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 \col@number register.

If the user forgot the argument, \LaTeX 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.

We have only enough box registers for twenty columns, so we need to check that the user hasn’t asked for more.

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 \multicols 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 \LaTeX stack overflow as this would generate a self-referencing definition.

Now we can safely look for the optional arguments.

\multicols

If the user forgot the argument, \LaTeX 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.

We have only enough box registers for twenty columns, so we need to check that the user hasn’t asked for more.

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 \multicols 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 \LaTeX stack overflow as this would generate a self-referencing definition.

Now we can safely look for the optional arguments.

\multicols

If the user forgot the argument, \LaTeX 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.

We have only enough box registers for twenty columns, so we need to check that the user hasn’t asked for more.

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 \multicols 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 \LaTeX stack overflow as this would generate a self-referencing definition.

Now we can safely look for the optional arguments.

\multicols

If the user forgot the argument, \LaTeX 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.

We have only enough box registers for twenty columns, so we need to check that the user hasn’t asked for more.

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 \multicols 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 \LaTeX stack overflow as this would generate a self-referencing definition.

Now we can safely look for the optional arguments.

\multicols

If the user forgot the argument, \LaTeX 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.

We have only enough box registers for twenty columns, so we need to check that the user hasn’t asked for more.

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 \multicols 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 \LaTeX stack overflow as this would generate a self-referencing definition.

Now we can safely look for the optional arguments.

\multicols

If the user forgot the argument, \LaTeX 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.

We have only enough box registers for twenty columns, so we need to check that the user hasn’t asked for more.

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 \multicols 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 \LaTeX stack overflow as this would generate a self-referencing definition.

Now we can safely look for the optional arguments.

\multicols

If the user forgot the argument, \LaTeX 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.

We have only enough box registers for twenty columns, so we need to check that the user hasn’t asked for more.

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 \multicols 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 \LaTeX stack overflow as this would generate a self-referencing definition.

Now we can safely look for the optional arguments.

\multicols

If the user forgot the argument, \LaTeX 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.

We have only enough box registers for twenty columns, so we need to check that the user hasn’t asked for more.

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 \multicols 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 \LaTeX stack overflow as this would generate a self-referencing definition.

Now we can safely look for the optional arguments.

\multicols

If the user forgot the argument, \LaTeX 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.

We have only enough box registers for twenty columns, so we need to check that the user hasn’t asked for more.

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 \multicols 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 \LaTeX stack overflow as this would generate a self-referencing definition.

Now we can safely look for the optional arguments.
with \par since otherwise the test for “inner” mode wouldn’t show if we are in a box.

99 \par
100 \ifinner \boxedmulticolstrue

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.

101 \else
102 \ifnum \doublecol@number>\z@ \boxedmulticolstrue
103 \fi
104 \fi

Then we look to see if statistics are requested:

105 \mult@info\z@
106 \{Starting environment with
107 \the\col@number space columns%
In boxed mode we add some more info.

108 \if\boxedmulticol\MessageBreak (boxed mode)\fi
109 \}

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.

110 \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[]{...} where the argument is also implicitly surrounded by braces. 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.

111 \par\addvspace\multicolsep
112 \if\boxedmulticol

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 \multicols are rigid and thus the distance from the baseline of a previous text line to the first line in a \multicols 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 \TeX and is not supposed to be a real depth.

113 \ifdim \prevdepth = \-\@m\p@ \else
114 \ifdim \prevdepth = \-\@realdimen\p@
115 \endgraf

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.

116 \@tempcnta\prevdepth
117 \@tempcntb\baselineskip
118 \divide\@tempcnta\@tempcntb
119 \advance\@tempcnta\z@
120 \advance\@tempcnta\@tempcntb
121 \advance\@tempcnta\@tempcntb
122 \advance\@tempcnta\baselineskip
123 \@tempcnta\baselineskip
124 \fi

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

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

130 \fi

We finish by suppressing initial spaces.

131 \ignorespaces}

Here is the switch and the box for “boxed” \multicols code.

132 \newif\ifboxedmulticol
133 \boxedmulticolstrue
134 \boxedmulticolfalse
135 \newbox\multibox

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’

\footnote{See the documentation of \endmulticols for further details.}
(i.e. \pageshrink) into account. Actually, we use \addpenalty to ensure that a following \advspace 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 \TeX is not invoking the output routine.

\begin{verbatim}
135 \def\enough@room#1{%
Measuring makes only sense when we are not in "boxed mode" so the routine does nothing if the switch is true.
136 \if@boxedmulticols\else
137 \par
To empty the contribution list the first release contained a penalty zero but this had the result that \advspace 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 multicols environment, but this seems acceptable since there is the optional argument for exactly this reason.
138 \bgroup@nobreakfalse\addpenalty\z@\egroup
139 \page@free \pagegoal
140 \advance \page@free \pagetotal
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.
141 \@tempskipa#1\relax
\end{verbatim}

Now we test whether tracing information is required:

\begin{verbatim}
142 \mult@info\z@% {Current page:\MessageBreak
143 \the\pagegoal: used \the\pagetotal
144 \space \rightarrow free=\the\page@free
145 \MessageBreak
146 needed \the\@tempskipa
147 \space (for #1)}% Our last action is to force a page break if there isn't enough room left.
148 \ifdim \page@free <\#1\newpage \fi
\end{verbatim}

When preparing for multicolumn output several things must be done.

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 \@totalleftmargin directly then lists inside the multicols environment could cause a shift of the output.

\begin{verbatim}
153 \multicol@leftmargin\@totalleftmargin
154 \@totalleftmargin\z@\parshape\z@
155 \@tempskipa#1\relax
\end{verbatim}

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

\begin{verbatim}
156 \doublecol@number\col@number
157 \multiply\doublecol@number\tw@
158 \advance\doublecol@number\mult@rightbox
159 \if@boxedmulticols
160 \let\l@kept@firstmark\kept@firstmark
161 \let\l@kept@botmark\kept@botmark
162 \global\let\kept@firstmark\@empty
163 \global\let\kept@botmark\@empty
164 \else
165 \nointerlineskip \null% 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.
166 \nointerlineskip \null%\{topskip\z@\null\}
167 \output%\global\setbox\partial@page\vbox
168 \%
\end{verbatim}

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.

Finally we need to record the marks that are present by using a suitable value for the second argument.

\begin{verbatim}
169 \check
170 \ifvoidpartial@page\else
171 \PackageError{multicol}{
172 {Error saving partial page}
173 \The part of the page before
174 the multicols environment was
175 nearly full with \^^J\the result
176 that starting the environment
177 will produce an overfull
178 page. Some \^^J\text may be lost!
179 Please increase \$\premulticols$
180 either generally or for this
181 \^^Jenvironment by specifying a
182 suitable value in the second
183 optional argument to \^^J\the
184 multicols environment.}
185 \unvboxpartial@page
186 \box\last@line
187 \fi
\end{verbatim}

Finally we need to record the marks that are present within the $\partial@page$ so that we can construct correct first and bottom marks later on. This is done by the following code.

\begin{verbatim}
192 \prep@keptmarks
\end{verbatim}

Finally we have to initialize $\kept@topmark$ which should ideally be initialized with the mark that is current on “top” of this page. Unfortunately we can’t use $\topmark$ because this register will not always contain what its name promises because \TeX 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}
193 \global\let\kept@topmark\firstmark
194 \eject
\end{verbatim}

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

\begin{verbatim}
195 \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}
196 \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}
197 \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 page-wide 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}
198 \init@mult@footins
\end{verbatim}

For the same reason (page-wide 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}
199 \reinsert@footnotes
\end{verbatim}

Inside \multicols a $\clearpage$ is fairly useless as we aren’t supporting floats. In fact, it can cause harm as it doesn’t know about the $\partial@page$ and may therefore result in making columns too long. So we change that to behave like $\newpage$ but also check if there are any deferred floats. If so, perhaps the user tried to place them through that $\clearpage$ (but that needs to be done before starting the \multicols environment.)

\begin{verbatim}
200 \def\clearpage{%
\end{verbatim}
All the code above was only necessary for the unrestricted \texttt{multicol} 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 \texttt{vsize}, etc.

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

\begin{verbatim}
\def\init@mult@footins{\multiply\count\footins\col@number}
\def\set@mult@vsize{\set@mult@vsize\hsize\linewidth\full@width\linewidth}
\end{verbatim}

Since nearly always the first pass will fail we ignore it completely telling \LaTeX{} 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 \texttt{pretolerance}.

\begin{verbatim}
\pretolerance\multicolpretolerance
\end{verbatim}

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

\begin{verbatim}
\setemergencystretch{\col@number\hsize}
\end{verbatim}

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

Additionally, we advance \texttt{\baselineskip} by \texttt{\multicolbaselineskip} to allow corrections for narrow columns.

The \texttt{hsize} of the columns is given by the formula:

\begin{verbatim}
\linewidth - (\columnwidth - 1) \times \columnsep
\columnwidth
\end{verbatim}

The formula above has changed from release to release. We now start with the current value of \texttt{\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:

\begin{verbatim}
\hsize\linewidth \advance\hsize\columnsep
\advance\hsize-\columnwidth \columnsep
\divide\hsize\columnwidth
\end{verbatim}

We also set \texttt{\linewidth} and \texttt{\columnwidth} to \texttt{\linewidth} in the past \texttt{\columnwidth} was left unchanged. This is inconsistent, but \texttt{\columnwidth} is used only by floats (which aren’t allowed in their current implementation) and by the \texttt{\footnote} macro. Since we want page-wide footnotes\footnote{I’m not sure that I really want page-wide 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 \LaTeX{} (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 \LaTeX{} thereafter.} this simple trick saved us from rewriting the \texttt{\footnote} macros. However, some applications referred to \texttt{\columnwidth} as the “width of the current column” to typeset displays (the \texttt{amsmath} package, for example) and to allow the use of such applications together with \texttt{multicol} this is now changed.

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

\begin{verbatim}
\full@width\linewidth
\linewidth\hsize
\columnwidth\hsize
\end{verbatim}

This macro is used to set the parameters associated with footnote floats. It can be redefined by applications that require different amount of spaces when typesetting footnotes.

\begin{verbatim}
\def\init@mult@footins{\}%
\multiply\count@footins\col@number
\multiply\skip@footins\col@number
\end{verbatim}

Since we have to set \texttt{\col@number} columns on one page, each with a height of \texttt{\@colroom}, we have to assign \texttt{\vsize = \col@number \times \@colroom} in order to collect enough material before entering the \texttt{\output} routine again. In fact we have to add another \texttt{(\col@number - 1) \times (\baselineskip - \topskip)} if you think about it.

\begin{verbatim}
\def\set@mult@vsize{\}
\end{verbatim}
But this might not be enough since we use \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 \global because we want to use this macro later inside the output routine too.

\begin{multicols}{\#1}\vspace{\baselineskip}\end{multicols}

Here is the dimen register we need for saving away the outer value of \@totalleftmargin.

\newdimen\multicol@leftmargin

\OMvi
In versions prior to 1.8r the balancing at the end of the environment was done by changing the output routine from \multicols* to \balance@column@out. As it turned out that this has a couple of issues when the last columns should not be balanced after all (for example because they contained several \columnbreak commands) we now stay with one output routine for the environment and only signal that we reached the end of the environment by marking it with a special penalty that we can check for later.

\mathchardef\OMvi=10006 % 10005 is \columnbreak

\endmulticols

When the end of the \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 \par command.

\def\endmulticols{\par} \if\boxedmulticols{par} \else \fi

In boxed mode we have to close the box in which we have gathered all material for the columns. But before we do this we need to remove any space at the end of the box as we don’t want to use this in balancing. Because of the \color@endgroup this can’t be done later in \balance@columns as the color command will hide it.

\remove@discardable@items \color@endgroup \egroup

Now we call \balance@columns the routine that balances material stored in the box \mult@box.

After balancing the result has to be returned by the command \page@sofar. But before we do this we reinser any marks found in box \mult@box.

\return@nonemptymark(first)\% \kept@firstmark
\return@nonemptymark(bot)\% \kept@botmark

When the boxed multicol is returned to the page it can happen that it doesn’t fit onto it and \LaTeX therefore breaks earlier. The problem in that case is that during the generation \hspace, etc. got changed and this setting is still in effect right now, and if this boxed multicol is within, say, \multicols* then its output routine gets very upset. We therefore delay returning the result by saving it in box for now until we have left the group below.

\global\setbox\mc@boxedresult\vbox{\page@sofar \global\let\kept@firstmark\l@kept@firstmark \global\let\kept@botmark\l@kept@botmark} \langle∗\marktrace⟩

This finishes the code for the “boxed” case.

\else

If there was a \columnbreak on the very last line all material will have been moved to the \colbreak@box. Thus 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.

\ifdim\pagegoal=\maxdimen\ifvoid\colbreak@box\else \mult@info\@ne{Re-adding forced break(s) for splitting}\unvbox\colbreak@box\fi \fi

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 \texttt{\vspace} which was placed at the end of the contribution list. Then the \texttt{page@builder} was called (an explicit \texttt{\vspace} will empty the contribution list), but the line with the \texttt{\vspace} didn’t fit onto the current page. It was then reconsidered after the output routine had ended, causing a second break after one line.} Therefore we must also add an explicit \texttt{\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 \texttt{\penalty} not \texttt{\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}
273 \penalty\z@
Now it’s safe to call the output routine in order to balance the columns. We do this by calling it with a special penalty.
274 \penalty-\@Mvi
If the \texttt{multicols} environment body was completely empty or if a multi-page \texttt{multicols} just ends at a page boundary we have the unusual case that the \texttt{\vspace} 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 \texttt{multicols} (stored in \texttt{\partial@page}) will get lost if we don’t put this back by hand.
275 \ifbox\partial@page
276 \un vbox\partial@page\fi
After the output routine has acted we restore the kept marks to their initial value.
277 \global\let\kept@firstmark@empty
278 \global\let\kept@botmark@empty
279 (\marktrace)
280 \mult@info\tw@
281 \{Make kept marks empty\}%
282 (\marktrace)
283 \fi
The output routine above will take care of the \texttt{\vsize} and reinsert the balanced columns, etc. But it can’t reinsert the \texttt{\footnotes} because we first have to restore the \texttt{\footnotes} 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 \texttt{\multicols}.

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

\texttt{\penalty-\@Mvi}\footnote{Actually, we are still in a group started by the \texttt{\begin} macro, so \texttt{\global} must be used anyway.}

We also set the ‘\texttt{unbalance}’ counter to its default. This is done globally since L\TeX\ counters are always changed this way.\footnote{This is done globally since L\TeX\ counters are always changed this way.}

\begin{verbatim}
284 \checkend{multicols}\
285 \endgroup
286 \global\c@unbalance\z@
Now it’s time to return any footnotes if we are in unrestricted mode. In boxed mode footnotes are kept inside, but in that case we have to first return the saved box to the page and then write another column status into the \texttt{.aux} file to support \texttt{docolaction} in case we have nested environments.
287 \if@boxedmulticols
288 \un vbox\mc@boxedresult
289 \mc@colstatus@write
290 \else
291 \reinsert@footnotes
\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 \texttt{\enough@room} starts a new page.

But there is one catch. If the \texttt{\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 \texttt{\vsize} for multicols collection which is a disaster. So we better catch this case. Fortunately we can detect it by looking at \texttt{\pagegoal}.

\begin{verbatim}
292 \ifdim \pagegoal=\maxdimen
293 \global\vsize\@colroom
294 \else
295 \enough@room\colroom
296 \fi
297 \fi
298 \addvspace\multicolsep
\end{verbatim}

There is one more thing to do: the balanced result of the environment is supposed to have a \texttt{\prevdepth} of zero as we backed up by its real \texttt{\prevdepth} within \texttt{page@sofar}. However if the balancing happened in the output routine then \TeX\ reverts to the \texttt{\prevdepth} that was current before the OR once the OR has finished. In short \texttt{\prevdepth} is something you can’t set globally it is always local to the current list being built. Thus we need to set it back to zero here to avoid incorrect spacing.

\begin{verbatim}
299 \prevdepth\z@
\end{verbatim}
If statistics are required we finally report that we have finished everything.

\newcount\c@unbalance
\newcount\c@collectmore
\newcount\col@number
\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
\newbox\mc@boxedresult

\c@unbalance = 0
\c@collectmore = 0
\col@number = 1
\multicoltolerance = 9999
\multicolpretolerance = -1
\premulticols = 50pt
\postmulticols = 20pt
\multicolsep = 12pt plus 4pt minus 3pt
\multicolbaselineskip = 0pt

\paragraph{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, … (locally) or 1, 3, 5, … (globally). This is \TeX policy to avoid an overflow of the save stack.

\process@cols

Therefore we define a \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 \count@ = \mult@rightbox (\count@ is a scratch register defined in \TeX), processes argument #2, adds two to \count@, processes argument #2 again, etc. until \count@ is higher than \doublecol@number. It might be easier to understand it through an example, so we define it now and explain its usage afterwards.

\def\process@cols#1#2{% 
\count@=\mult@rightbox 
\loop
\typeout{Looking at box \the\count@}
\ifnum\count@=\doublecol@number
\repeat
\count@ \col@number \advance\count@-1
\mult@info2
\{Column spec: \the\full@width \space = indent

\page@sofar

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

\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), … However, the last column (i.e. the rightmost) should be saved in box register 0.$^{12}$ First we ensure that the columns have equal width. We use \process@cols for this purpose, starting with \count@ = \mult@rightbox. Therefore \count@ loops through \mult@rightbox, \mult@rightbox + 2, … (to \doublecol@number).

\def\process@cols#1#2{% 
\ifvoid\count@
\setbox\count@ = \hbox to\hsize{}
\else
\wd\count@ = \hsize
\fi
\count@ \col@number \advance\count@-1
\ifnum\count@=\doublecol@number
\repeat

\mult@info2
\{Column spec: \the\full@width \space = indent

\newdimen\postmulticols
\newdimen\postmulticols = 12pt plus 4pt minus 3pt
\newdimen\multicolbaselineskip
\newdimen\multicolbaselineskip = 0pt

$^{12}$You will see the reason for this numbering when we look at the output routines \multi@column@out and \balance@columns@out.
+ columns + sep = Message Break
\the\multicolo@leftmargin \space
+ \the\col@number \space
x \the\hspace \space
+ \the\count@ \space
x \the\columnsep
\}%

At this point we should always be in vertical mode.
\ifvmode \else \errmessage{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) and separating the columns with a rule if desired.

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 “a” since that is one of the depths that compete for the maximum.

\setbox\z@ = \hbox{\multicolmindepthstring} \global \dimen\tw@ = \dp\z@
\moveright\multicol@leftmargin
\hbox to\full@width{%

If the document is written in a language that is typeset right-to-left then, of course, the multicol columns should be also typeset right-to-left. To support this we call \mc@align@columns which with execute different code depending on the typesetting direction.

\mc@align@columns

The depths of the columns depend on their last lines. To ensure that we will always get \similarity@footnotes as far as the rules are concerned we force the depth to be at least the depth of a letter ‘p’ or more exactly \multicolmindepthstring (which is what we set \dimen2 to above).

\rlap{\phantom \multicolmindepthstring}\%

The processed material might consist of a last line with a descender in which case the \prevdepth 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. However, if the \page@sofar is being called inside the OR then setting \prevdepth here has no long-lasting effect, we therefore have to repeat this once we return to the main vertical list. Here we set it only for those cases where the command is used within a list and then followed by something else.

\prevdepth = 0

Now after typesetting the box we back up to its baseline by using the value stored in \dimen2 (which will hold the largest depth found on any column).

\kern -\dimen\tw@
However, in case one of the columns was unusually deep \TeX may have tried some corrective actions in which case backing up by the saved 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! Message Break}
\Grid alignment might be broken\}%
\fi

\multicolmindepthstring

The default minimum depth of each column corresponds to the depth of a ‘p’ in the current font. This makes sense for Latin-based languages and was hard-wired initially, but for Asian languages it is better to use a zero depth (and alternatively one might want to use the depth of a strut or a parentheses). So we now offer a way to adjust this while maintaining backward compatibility. Use \renewcommand to alter it.

\def\multicolmindepthstring{p}

By default the vertical rule between columns will be in \normalcolor.

\def\columnseprulecolor{\normalcolor}
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.

\vfilmaxdepth

This curious definition is used as the space at the bottom of a column if we implement \raggedcolumns. Normally one only appends \vfill in that case but this is actually wrong for columns that are more or less full: by adding a glue at the bottom such a column doesn’t have any depth any more but without it the material would be allowed a depth of \@maxdepth. So we allow shrinking by that amount. This only makes a difference if the box would otherwise become overfull and shrinking never exceeds the specified value, so we should be fine.

\multicol@out

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 \@M, otherwise the penalty which triggered the output routine is higher. Therefore it’s easy to distinguish both cases: we simply test this register.

\dimen@\@colroom

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.

\ifvbox \@kludgeins
\divide \dimen@ -\ht\@kludgeins
\ifdim \wd\@kludgeins>\z@ \shrinkingtrue \fi
\fi

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).

\set@keptmarks

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.

\We also need to restrict \boxmaxdepth so that re-boxing is not generating boxes with arbitrary depth.

\boxmaxdepth \@maxdepth

Then we calculate the current column height (in \@dimen0). Note that the height of \partialpage is already subtracted from \@colroom so we can use its value as a starter.

\@dimen0 \@colroom

And there is one more adjustment that we have to make: if the user has issued a \enlargethispage command then the height the \@kludgeins box will be the negation of the size by which the page should be enlarged. If the star form of this command has been used then we also need to shrink the resulting column.

That local change will be reverted at the end of the output routine. So for the next page the original state will be reestablished. However, in theory there is a possibility to sneak in a whole multicols environment into the running header definition. If that happens then it will also be affected by this change—too bad I think.

\ifvoid \@kludgeins
\advance \dimen@ -\ht\@klugeins
\fi

The star form of \enlargethispage makes the width of the box greater than zero (sneaky isn’t it?).

\ifdim \wd\@kludgeins>\z@ \shrinkingtrue \fi

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).

\set@keptmarks

If \raggedcolumns is in force we add a \vfill at the bottom by unboxing the split box. But we need to unbox anyway to ensure that at the end of the box we do not have unwanted space. This can sneak in
in certain situations, for example, if two lists follow each other and we break between them. While such space is usually zero it still has an effect because it hides depth of the last line in the column and that will result in incorrect placement.

Then the last column follows.

\setbox\mult@rightbox
\vsplit\@cclv to\dimen@
\set@keptmarks
\setbox\mult@rightbox\vbox to\dimen@
\ifshr@nking\vfilmaxdepth\fi}%

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

\ifvoid\@cclv \else
\unvbox\@cclv
\ifnum\outputpenalty=\@M
\else
\penalty\outputpenalty
\fi\fi

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 multicol produced this warning regardless of whether or not footnotes were present, resulting in many unnecessary warnings.

\ifvoid\footins\else
\PackageWarning{multicol}{I moved some lines to the next page. Footnotes on page \thepage space might be wrong}%
\fi

If the ‘tracingmulticolor’ counter is 4 or higher we also add a rule.

\ifnum c@tracingmulticolors>\thr@@
\hrule\allowbreak \fi\fi

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 \partical@page 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.

\mult@info\tw@{Use kept top mark: \meaning\kept@topmark
Use kept first mark: \meaning\kept@firstmark
Use kept bot mark: \meaning\kept@botmark
Produce first mark: \meaning\firstmark
Produce bot mark: \meaning\botmark
@gobbletwo}%

With a little more effort we could have done better. If we had, for example, recorded the shrinkage of the material in \partical@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.\footnote{This will produce a lot of overhead since both output routines are held in memory. The correct solution would be to redesign the whole output routine used in \LaTeX.} Admittedly this is a funny way to do it.

Within the OR \boxmaxdepth needs to be unrestricted so we set it back now as it was changed above.

\boxmaxdepth\maxdimen
\setbox\@cclv\vbox{\unvbox\partial@page \page@sofar}\
\@makecol\@outputpage
\the\jobname
\message{Using \jobname\...\jobname\jobname\jobname
\jobname\jobname\jobname}
After the page is shipped out we have to prepare the kept marks for the following page. \texttt{\kept@firstmark} and \texttt{\kept@botmark} reinitialized by setting them to \texttt{@empty}. The value of \texttt{\botmark} is then assigned to \texttt{\kept@topmark}.

\begin{verbatim}
464 \global\let\kept@topmark\botmark
465 \global\let\kept@firstmark\@empty
466 \global\let\kept@botmark\@empty
\end{verbatim}

Now we reset \texttt{\@colroom} to \texttt{\@colht} which is \LaTeX{}'s saved value of \texttt{\textheight}. We also have to reset the recorded position of the last \texttt{\marginpar} as well as the recorded size of in-text floats as we are now on a new page.

\begin{verbatim}
473 \global\@colroom\@colht
474 \global \@mparbottom \z@ 
475 \global \@textfloatsheight \z@
\end{verbatim}

Then we process deferred floats waiting for their chance to be placed on the next page.

\begin{verbatim}
476 \process@deferreds
477 \@whilesw\if@fcolmade\fi{\@outputpage
478 \global\@colroom\@colht
479 \process@deferreds}%
\end{verbatim}

If the user is interested in statistics we inform him about the amount of space reserved for floats.

\begin{verbatim}
480 \mult@info\@ne
481 \{\textit{Colroom:}\MessageBreak\the\@colht\space
482 \textit{after float space removed}
483 = \the\@colroom \&\gobble\}%
\end{verbatim}

Having done all this we must prepare to tackle the next page. Therefore we assign a new value to \texttt{\vsizem}. New, because \texttt{\partial@page} is now empty and \texttt{\@colroom} might be reduced by the space reserved for floats.

\begin{verbatim}
485 \set@mult@vsize \global
\end{verbatim}

The \texttt{\footins} skip register will be adjusted when the output group is closed.

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

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 \texttt{\dimen0}. See above for the description of the default action.

\begin{verbatim}
487 \def\leave@mult@footins{%
488 \advance\dimen0\textskip\footins
489 \advance\dimen0\ht\footins
490 }
\end{verbatim}

We left out two macros: \texttt{\process@deferreds} and \texttt{\specials}.

\begin{verbatim}
491 \def\specials{%
492 \ifnum\outputpenalty<\@Mi
\end{verbatim}

If the document ends in the middle of a \texttt{\multicols} environment, e.g., if the user forgot the \texttt{\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 \texttt{\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.

\begin{verbatim}
493 \ifnum \outputpenalty<\@MM
494 \PackageError{multicol}{Document end inside multicols environment}@@ehd
495 \@specialoutput
496 \else
497 \fi}
\end{verbatim}

As this is the place of a forced break we now remove vertical white space just in front of it (or some of it at least) as it is quite likely that the break is not exactly in the right place, e.g., after a display environment (if \LaTeX{} would break here by its own it would break before the space following the display).

Thus we rebox box 255 once (using \texttt{\maxdepth} and calling \texttt{\remove@discardable@items} inside). The depth of 255 will then give us the depth the box would have had if it would have been a natural break. We then unbox 255 to get it into the \texttt{\colbreak@box} and then back up by this depth. This will position the bottom of the box at its natural baseline which is useful for balancing later on.

\begin{verbatim}
509 \boxmaxdepth\@maxdepth
510 \setbox\@cclv\vbox{%
511 \unvbox\@cclv
512 \remove@discardable@items}
513 \dimen0\dp\@cclv
514 \unvbox\@cclv
515 \kern-\dimen0
\end{verbatim}
Another special case is reaching the end of the \texttt{multicols} environment which is signaled by \texttt{-@Mvi}.  

If we are at this point then we have to run the balancing code (which was previously its own output routine).  

First we pretend that we had a normal forced breakpoint and then call \texttt{\balance@columns@out}.  

If we encounter a float or a marginpar in the current implementation we simply warn the user that this is not allowed. Then we reinsert the page and its footnotes.

If the penalty is \texttt{-10001} it will come from a \texttt{\clearpage} and we will execute \texttt{\doclearpage} to get rid of any deferred floats.  

\texttt{\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{\temp}.  

Our next action is to (globally) empty \texttt{\deferlist} and assign a new meaning to \texttt{\elt}. Here \texttt{\scolelt} is a macro that looks at the boxes in a list to decide whether they should be placed on the next page (i.e. on \texttt{\toplist} or \texttt{\botlist}) or should wait for further processing.

Now we call \texttt{\temp} which has the form \texttt{\elt (box register)\@elt (box register)\ldots}  

So \texttt{\elt} (i.e. \texttt{\scolelt}) will distribute the boxes to the three lists.

The \texttt{\raggedcolumns} and \texttt{\flushcolumns} declarations are defined with the help of a new \texttt{\if...} macro.

The actual definitions are simple: we just switch to \texttt{true} or \texttt{false} depending on the desired action. To avoid extra spaces in the output we enclose these changes in \texttt{\@bsphack...\@esphack}.

Now for the last part of the show: the column balancing output routine. Since this code is called with an explicit penalty (\texttt{\eject}) there is no need to check for something special (eg floats). We start by balancing the material gathered.

For this we need to put the contents of box 255 into \texttt{\mult@box}.  

For the next block of code see also comments in section 7.2. All forced breaks except the last are inside \texttt{\colbreak@box} so all we have to do is to concatenate this box with box \texttt{\cclv} and put a penalty in between. Here we test if \texttt{\colbreak@box} is void so that the message is only generated if we really add forced breaks and the penalty.
The last column again is a forced break, so here we discard white space as well as that is normally unwanted.

If during balancing the columns got too long the flag \texttt{\iftoo\bad} is set to true.

In that case we put the material back in box 255 so that we can cut a normal page. The curious set of \texttt{\vskips} we add is necessary to cancel out the \texttt{\splittopskip} that got added for balancing.

We also have to re-add the end of environment penalty since after this page we may want balance the remaining material.

We then call the standard multicols output routine which will produce a normal page for us (remember we are still within the OR so \texttt{\multicolsep} etc. the code in \texttt{\multicolor@out} is actually not doing anything—perhaps this should be cleaned up at some point). This also means that if there was an \texttt{\enlargethispage} present it will apply to this page as \texttt{\multicolor@out} will look at the status of \texttt{\@kludgeins}.

Because balancing made the columns too long we are sure that there will be some material remaining which was put back onto the main vertical list by \texttt{\multicolor@out}. This will also put the explicit \texttt{\eject} penalty back so the current \texttt{\balance@columns@out} output routine will be called again (so we better do not add another penalty or else the OR will be called twice and we may get scrambled results).

If the balancing went ok, we are in the position to apply \texttt{\page@sofar}. But first we have to set \texttt{\vsizes} to a value suitable for one column output.

We also have to look at \texttt{\@kludgeins} and generate a new \texttt{\insert} in case there was one present due to an \texttt{\enlargethispage} command.

Then we \texttt{\unvbox} the \texttt{\partialpage} (which may be void if we are not processing the first page of this multicols environment.

Then we return the first and bottom mark and the gathered material to the main vertical list.

We need to add a penalty at this point which allows to break at this point since calling the output routine may have removed the only permissible break point thereby “glueing” any following skip to the balanced box. In case there are any weird settings for \texttt{\multicolsep} etc. this could produce funny results.

As we already know, reinserting of footnotes will be done in the macro \texttt{\endmulticols}.

This macro now does the actual balancing.

We start by setting the kept marks by updating them with any marks from this box. This has to be done before we add a penalty of \texttt{\penalty\z@} to the top of the box, otherwise only an empty box will be considered.

We then continue by resetting trying to remove any discardable stuff at the end of \texttt{\multicolor@box}. This is rather experimental. We also add a forced break point at the very beginning, so that we can split the box to height zero later on, thereby adding a known \texttt{\splittopskip} glue at the beginning.

Then follow values assignments to get the \texttt{\vsplitting} right. We use the natural part of \texttt{\topskip} as the natural part for \texttt{\splittopskip}
and allow for a bit of undershoot and overshoot by
adding some stretch and shrink.
\@tempdima\topskip
\@tempdima\splittopskip\@tempdima
\@plus\multicolundershoot
\@minus\multicolovershoot
\splitmaxdepth\@maxdepth

We also have to set \boxmaxdepth which normally
allows to build boxes with arbitrary depth, but as
we are building text columns we really want to re-
strict the depth. This is necessary as we sometimes
rebox the boxes generated by \vsplit and then the
restriction posed by \splitmaxdepth gets lost.
\boxmaxdepth\@maxdepth

The next step is a bit tricky: when \TeX assem-
bles 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 charac-
ter 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 \splittop@skip is inserted
at the top of \mult@box which is exactly what we
like to achieve.
\setbox\@tempboxa\vsplit\mult@box to\z@

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 ...
\@tempdima\ht\mult@box
\advance\@tempdima\dp\mult@box
\divide\@tempdima\col@number

The code above sets \@tempdima 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 num-
ber (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@ \leq to
\@tempdima.
\count@\@tempdima
\divide\count@\baselineskip
\dimen@\count@\baselineskip

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 sub-
tract one \baselineskip.
\advance\dimen@\topskip
\ifdim \dimen@ >\@tempdima
\advance\dimen@-\baselineskip
\fi

As a further restriction we want to see a minimum
number of rows in the balanced result based on the
setting of the counter \minrows. If the starting value
is lower we adjust.
\@tempdima\dimexpr \topskip +\c@minrows\baselineskip
-\baselineskip\relax
\ifnum\dimen@<\@tempdima
\mult@info\@ne
{Start value \the\dimen@ \space ->
\the\@tempdima \space (corrected for minrows)}%
\dimen@\@tempdima
\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\@ne
{Balance columns on@line:
\ifnum\c@unbalance=\z@ else
\the\c@unbalance \space ->
\the\mult@info \space (off balance=\number\c@unbalance)\fi
\@gobbletwo}%

But we don’t allow nonsense values for a start.
\ifnum\dimen@<\topskip
\mult@info\@ne
{Start value \the\dimen@ \space ->
\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 sup-
press 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@M

23
We also allow for overfull boxes while we trying to split the columns. They can easily happen if we have objects with unusual depth.

\[\v fuzz \maxdimen\]

The variable \texttt{last@try} will hold the dimension used in the previous trial splitting. We initialize it with a negative value.

\[\texttt{last@try}-\p@\]

\texttt{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 \texttt{process@cols} for this purpose, starting with \texttt{mult@grightbox}. Note the extra braces around this macro call. They are needed since Plain \TeX's \texttt{loop}...\texttt{repeat} mechanism cannot be nested on the same level of grouping.

\[\texttt{\process@cols\mult@grightbox}\]
\[\texttt{\global\setbox\count@\vsplit\mult@grightbox to\dimen@}\]

After splitting we need to ensure that there isn't any space at the bottom, so we rebox once more.

\[\texttt{\global\setbox\count@\vbox to\dimen@\{\unvbox\count@\}}\]

After every split we check the badness of the resulting column, normally the amount of extra white in the column.

\[\texttt{\{badness}\}\]
\[\ifnum\c@tracingmulticols>\@ne \@tempcnta\count@ \advance\@tempcnta-\mult@grightbox \divide\@tempcnta \tw@ \message{\textasciitilde\textasciitilde\textasciitilde Column \number\@tempcnta\space badness: \the\badness\space}\fi\]

If this badness is larger than the allowed column badness we reject this solution by setting \texttt{too@bad} to \texttt{true}.

\[\ifnum\badness>\c@columnbadness\ifnum\c@tracingmulticols>\@ne \message{too bad (>\the\c@columnbadness)}\fi\too@badtrue\fi\]

\[\{\textasciitilde\textasciitilde\textasciitilde\badness\}\]

There is one subtle point here: while all other constructed boxes have a depth that is determined by \texttt{\splitmaxdepth} and/or \texttt{\boxmaxdepth} the last box will get a natural depth disregarding the original setting and the value of \texttt{\splitmaxdepth} or \texttt{\boxmaxdepth}. This means that we may end up with a very large depth in box \texttt{\mult@grightbox} which would make the result of the testing incorrect. So we change the value by unboxing the box into itself.

\[\texttt{\global\setbox\mult@grightbox\vbox{\unvbox\mult@grightbox}}\]

We also save a copy \texttt{\mult@gfirstbox} at its “natural” size for later use.

\[\texttt{\setbox\mult@nat@firstbox\vbox{\unvcopy\mult@gfirstbox}}\]

After \texttt{\process@cols} has done its job we have the following situation:

\[\texttt{box \mult@rightbox ← − all material box \mult@firstbox ← − first column box \mult@firstbox + 2 ← − second column} \]
\[\texttt{⋯} \]
\[\texttt{box \mult@grightbox ← last column}\]

We report the height of the first column, in brackets the natural size is given.

\[\ifnum\c@tracingmulticols>\@ne \@tempcnta\count@ \message{\textasciitilde\textasciitilde\textasciitilde First column = \the\dimen@ space \the\ht\mult@nat@firstbox}\fi\]

If \texttt{\raggedcolumns} is in force older releases of this file also shrunk the first column to its natural height
at this point. This was done so that the first column doesn’t run short compared to later columns but it is actually producing incorrect results (overprinting of text) in boundary cases, so since version v1.5q `\raggedcolumns` means allows for all columns to run slightly short.

```latex
\ifsh@nking
  \global\setbox\mult@gfirstbox
  \copy\mult@nat@firstbox
\fi
```

Then we give information about the last column.\footnote{With \TeX\ version 3.141 it is now possible to use \LaTeX\’s `\newlinechar` in the `\message` command, but people with older \TeX\ versions will now get `\^^J` instead of a new line on the screen.}

```latex
\ifnum\c@tracingmulticols>\@ne
  \message{<> last column = \the\ht\mult@grightbox}
\fi
```

Some tracing code that we don’t compile into the production version unless asked for. It will produce huge listings of the boxes involved in balancing in the transcript file.

```latex
\ifnum\c@tracingmulticols>4
  \showoutput
  \batchmode
  \process@cols\mult@grightbox
  \showbox\count@}
\errorstopmode
\fi
```

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@`.

```latex
\ifdim\ht\mult@grightbox >\dimen@
  \message{Final badness: \the\badness}
\fi
```

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

```latex
\too@badtrue
\ifnum\c@tracingmulticols>\@ne
  \typeout{Rejected: last column too large!}
\fi
```

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@grightbox` (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.

```latex
\global\setbox\mult@grightbox
\vbox to\dimen@{
  \unvbox\mult@grightbox\vfil}
```

If `\@tempboxa` above was not void our trial was unsuccessful and we report this fact and try again.

```latex
\too@badtrue
\ifnum\c@tracingmulticols>\@ne
  \typeout{Rejected: unprocessed forced breaks}
\fi
```

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.

```latex
\ifnum\badness>\c@finalcolumnbadness
  \global\setbox\mult@grightbox
  \vbox to\dimen@
  \showbox\count@}
\errorstopmode
```

If we have unprocessed forced breaks we normally reiterate with a larger column size to fit them in eventually. However, if there are simply too many of them (e.g., 3 forced breaks but only 2 columns to balance) then this will never succeed and we would continue growing the columns until we hit the largest possible column size. So in addition we check how big the column size is compared to available room and if we exceed this by `\maxbalancingoverflow` we give up and instead of balancing cut another normal page. To be indicate this case we set `forcedbreak@leftover` to true.

```latex
\global\setbox\mult@grightbox\vbox to\dimen@
  \maxdimen
\vfil}
```

```latex
\too@badtrue
\ifnum\c@tracingmulticols>\@ne
  \typeout{Rejected: unprocessed forced breaks}
\fi
```

```latex
\too@badtrue
\ifnum\c@tracingmulticols>\@ne
  \typeout{Rejected: unprocessed forced breaks}
\fi
```

```latex
\too@badtrue
\ifnum\c@tracingmulticols>\@ne
  \typeout{Rejected: unprocessed forced breaks}
\fi
```

```latex
\too@badtrue
\ifnum\c@tracingmulticols>\@ne
  \typeout{Rejected: unprocessed forced breaks}
\fi
```

```latex
\too@badtrue
\ifnum\c@tracingmulticols>\@ne
  \typeout{Rejected: unprocessed forced breaks}
\fi
```
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@`. If we come out of the loop with the switch `forcedbreak@leftover` set to true then balancing has failed and we should cut a normal page. We indicate this below with `\too@badtrue` when putting the final column together we want overfull information:

If the resulting box is overfull there was too much material to fit into the available space. The question though is how much? If it wasn’t more than `\maxbalancingoverflow` we accept it still to avoid getting very little material for the next page (which we would then have difficulties to balance).

Fail the balancing attempt:

\footnote{Should get cleaned up as we now have two different routes to reach this part of the processing.}

\footnote{This might be wrong, since the shrinkability that accounts for the amount of material might be present only in some columns. But it is better to try then to give up directly.}
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.

There was in fact a bug in the new implementation because at one point \LaTeX started to use the extended registers and so jumped from below 255 to above omitting the boxes allocated for inserts and the output page box.

So nowadays we really have to check if we get the full sequence of boxes allocated without holes (i.e., \(2 \times \text{max cols} + 1\)) and if not alter the allocation registers to start allocating after 255. This is all done quite low-level by looking directly at the values of the allocation counters.

5 New macros and hacks for version 1.2

If we don’t use \LaTeX 3.0 the \texttt{emergencystretch} register is defined so in this case we simply add it as an unused \texttt{dimen} register.

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.

Even if this should be used as a hook we use a \@ in the name since it is more for experts.
We have to add the float to the \@deferlist because we assume that outside the multicols environment we are in one column mode. This is not entirely correct, I already used the multicols environment inside of \LaTeX's \texttt{twocolumn} declaration but it will do for most applications.

5.1 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.

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

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.

\begin{verbatim}
\def\return@nonemptymark#1#2{\ifx#2\@empty
\else
\fi}
\end{verbatim}

For debugging purposes we take a look at the value of the kept mark that we are about to return. This code will get stripped out for production.

\begin{verbatim}
\def\*{\marktrace}
\begin{verbatim}
\mult@info\tw@
\{Returned #1 mark:Message\break
\meaning#2\}\%
\nobreak
\%
\fi
\end{verbatim}
\end{verbatim}

Since the contents of the mark may be arbitrary \TeX 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{\markboth}.) We therefore use the usual mechanism of a toks register to prohibit expansion.\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.}

\begin{verbatim}
\def\get@keptmarks#1{\begingroup
\vbadness\@M
\setbox#1\copy#1\%
\setbox#1\vsplit#1to\maxdimen\%
Now we open a new group and locally copy the box to itself. As a result any operation, i.e. \texttt{\vsplit}, will only have a local effect. Without this trick the box content would get lost up to the level where the last assignment to the box register was done.

\begin{verbatim}
\begingroup
\vbadness\@M
\setbox#1\copy#1\%
\setbox#1\vsplit#1to\maxdimen\%
\end{verbatim}

Therefore we can now set the kept marks which is a global operation and afterwards close the group. This will restore the original box contents.
For debugging we take again a look at the box dimension which shouldn’t have changed.

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

\set@keptmarks The macro \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 is empty we assume that it isn’t set. This is strictly speaking not correct as we lose the ability to have marks that are explicit-\n\expandafter\gdef\expandafter\kept@firstmark\expandafter{%\splitfirstmark} When debugging we show the assignment but only when something actually happened.\protect\mult@info\tw@{Set kept first mark:\MessageBreak\meaning\kept@firstmark\@gobbletwo}\fi\prep@keptmarks 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{% \if@boxedmulticols \else \get@keptmarks\partial@page \fi}

\remove@discardable@items There are situations when we may have some space at the end of a column and this macro here will attempt to get rid of it. The typical \LaTeX{} sequence is a series of self-canceling glues so if we remove them recursively we are usually fine.

Special care is needed with handling \vspace* as that corresponds to \penalty10000, \vskip <skip>, followed by \vskip 0pt. If we see this sequence going backwards in the vertical list we assume that this is a “desired” space. We therefore stop the recursion and reinsert the spaces.

As the multicols code sometimes add an explicit penalty at the end of a column we first attempt to remove it in case it is there.\skip0=0pt\edef\the@zero@skip{\the\skip0}\def\remove@discardable@items{% \unpenalty Save a previous skip (if there) and then remove it, we can’t really tell the difference between no skip an\edef\@tempa{\the\lastskip}\unskip If it was a zero skip (or none) we save the next previous skip (if any).\ifx\@tempa\the@zero@skip\edef\@tempb{\the\lastskip}\else\unskip\fi If it was a zero skip (or none) we save the next previous skip (if any).\ifx\@tempa\the@zero@skip\edef\@tempb{\the\lastskip}\else\unskip\fi

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{%\splitbotmark} \ifx\@tempa\empty\else\unskip\fi\mult@info\tw@{Set kept bot mark:\MessageBreak\meaning\kept@botmark\@gobbletwo}\fi\gobbletwo\fi\if\@boxedmulticols\else\get@keptmarks\partial@page\fi\fi\ifnum \lastpenalty=10000\unskip\fi\\typeset{p=\lastpenalty}%\ifnum \lastpenalty=10000\unskip\fi\else

Otherwise we remove this “real” skip. Then we look if it was preceded by a penalty of 10000 (i.e., a \nobreak)\unskip\\typeset{p=\lastpenalty}\@gobbletwo\fi\ifnum \lastpenalty=10000\unskip\fi\else\unskip\fi
If so this was a \vspace* or something equivalent to it. Therefore we reintroduce the skips and stop.

Otherwise we recurse.

\skip\@tempb \vskip\@tempa \relax
\else
\remove@discardable@items
\fi
\fi
\else
If the first skip was a non-zero skip we recurse as well.
\remove@discardable@items
\fi
\}
⟨∗
badness⟩
\newif\iftoo@bad
\def\too@badtrue{\global\let\iftoo@bad\iftrue}
\def\too@badfalse{\global\let\iftoo@bad\iffalse}
\newif\ifforcedbreak@leftover
\c@minrows
\newcount\c@minrows
\c@minrows=1
\newcount\c@columnbadness
\c@columnbadness=10000
\newcount\c@finalcolumnbadness
\c@finalcolumnbadness=9999
\newdimen\last@try
\multicolovershoot
\multicolundershoot
\multicolovershoot=0pt
\multicolundershoot=2pt
\newbox\mult@nat@firstbox
⟨/badness⟩
\mult@info
A helper for producing info messages
\def\mult@info#1#2{%\ifnum\c@tracingmulticols>#1%
\GenericWarning{(multicol)\@spaces\@spaces}{Package multicol: #2}%
\fi}
\fi
\mult@footnotetext
If we store the current column width in \columnwidth we have to redefine the internal \@footnotetext macro to use \textwidth for the width of the 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 \columnwidth is set to \textwidth.

This solves two problems: first redefinitions of \footnotetext done by a class will correctly survive and second if multicols is used inside a minipage environment the special definition of \@footnotetext in that environment will be picked up and not the one for the main galley (the latter would result in all footnotes getting lost in that case).

See the definition of the \multicols command further up for the exact code.

7 Fixing the \columnwidth

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.

\multicols* The code for this environment was suggested by Matthias Clasen.
\langle\nobalance\rangle
\namedef{multicols*}{%
7.2 Manual column breaking

The problem with manual page breaks within multicol 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 \vspliting operations within multicol 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 \TeX 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 \vsize 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 \specifs above.)

In the output routines that do the \vspliting 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.

\columnbreak is modeled after \pagebreak except that we generate a penalty -10005.

We have to ensure that it is only used within a \multicol environment since if that penalty would be seen by the unmodified \TeX output routine strange things would happen.

Increasingly lower penalty based on argument value. This is like \pagebreak but we use other penalty values are the \TeX defaults aree rather pointless for pagination.
This is modeled after \newpage but for column breaks.

\newcommand{\newcolumn}{%
  \ifnum\col@number<\tw@
    \PackageError{multicol}{}{This command can only be used within a multicols or multicols* environment.}%
  \else
    \ifvmode
      \nobreak\vfill\kern\z@\penalty -\@Mv\relax
    \else
      \@bsphack\vadjust{\nobreak\vfill\kern\z@\penalty -\@Mv\relax}\@esphack
    \fi
  \fi
}
\colbreak@box Need a box to collect the galley up to the column break.
\newbox{\colbreak@box}
\LR@column@boxes is called when we are assembling the columns for left to right typesetting. When we start we are inside an \hbox of full width. Left to right typesetting is fairly easy, we basically output each column box intermixed with vertical rules and proper spacing. As this happens inside a box of a defined width the rules and the columns automatically get into the right positions.
\LR@column@boxes is called when we are assembling the columns for left to right typesetting. When we start we are inside an \hbox of full width. Left to right typesetting is fairly easy, we basically output each column box intermixed with vertical rules and proper spacing. As this happens inside a box of a defined width the rules and the columns automatically get into the right positions.

\def{\LR@column@boxes}{{%
  \process@cols\mult@firstbox{%
    \hskip\hsize
    \hss{\columnseprulecolor}\vrule\@width\columnseprule}\hss%
  }%
  \hskip\hsize
  At this point in the code our typesetting reference point is at the right end of the rightmost column (or rather where that column should appear).
  We are now typesetting all columns by first backing up by their width (which is \hsize) then typesetting the box and then backing up again, but this time further, i.e., also across the column separation. That will then enable us to typeset the next column using the same approach until we are done with all but the final column.
\def{\LR@column@boxes}{{%
  \process@cols\mult@firstbox{%
    \hskip\hsize
    \hss{\columnseprulecolor}\vrule\@width\columnseprule\hss%
  }%
  \hskip\hsize
  At this point in the code our typesetting reference point is at the right end of the rightmost column (or rather where that column should appear).
  We are now typesetting all columns by first backing up by their width (which is \hsize) then typesetting the box and then backing up again, but this time further, i.e., also across the column separation. That will then enable us to typeset the next column using the same approach until we are done with all but the final column.
\def{\LR@column@boxes}{{%
  \process@cols\mult@firstbox{%
    \hskip\hsize
    \hss{\columnseprulecolor}\vrule\@width\columnseprule\hss%
  }%
  \hskip\hsize
  At this point in the code our typesetting reference point is at the right end of the rightmost column (or rather where that column should appear).
  We are now typesetting all columns by first backing up by their width (which is \hsize) then typesetting the box and then backing up again, but this time further, i.e., also across the column separation. That will then enable us to typeset the next column using the same approach until we are done with all but the final column.
\def{\LR@column@boxes}{{%
  \process@cols\mult@firstbox{%
    \hskip\hsize
    \hss{\columnseprulecolor}\vrule\@width\columnseprule\hss%
  }%
  \hskip\hsize
  At this point in the code our typesetting reference point is at the right end of the rightmost column (or rather where that column should appear).
  We are now typesetting all columns by first backing up by their width (which is \hsize) then typesetting the box and then backing up again, but this time further, i.e., also across the column separation. That will then enable us to typeset the next column using the same approach until we are done with all but the final column.
\def{\LR@column@boxes}{{%
  \process@cols\mult@firstbox{%
    \hskip\hsize
    \hss{\columnseprulecolor}\vrule\@width\columnseprule\hss%
  }%
  \hskip\hsize
  At this point in the code our typesetting reference point is at the right end of the rightmost column (or rather where that column should appear).
  We are now typesetting all columns by first backing up by their width (which is \hsize) then typesetting the box and then backing up again, but this time further, i.e., also across the column separation. That will then enable us to typeset the next column using the same approach until we are done with all but the final column.
The approach for the final column is similar only that we do not have to back up over any column gap.

However we do have to move the reference point to its right place: to make the rules appear at the expected places, we should get the typesetting position to the far right again. As we at the moment at the far left we skip to the far right like this:

\hskip\full@width}

Macros to switch between left-right and right-left typesetting. In LR typesetting the \RL@column@boxes is used to combine the columns. When typesetting right to left the \RL@column@boxes is used instead.

\newcommand\RLmulticolcolumns{
\let\mc@align@columns\RL@column@boxes
\newcommand\LRmulticolcolumns{
\let\mc@align@columns\LR@column@boxes
The default is left-to-right:

7.4 Supporting \docolaction

Whenever we want to do something that depends on the current column we execute \docolaction. This command takes one optional and three mandatory arguments. The mandatory ones denote what to do if this is a “left”, “middle”, or “right” column and the optional one is simply there to say what to do if we don’t know (default is to use the “left” column action in that case).
If column is unknown we use the default action (so that arg can take the value 1, 2, 3).
\ifcase \#1\or \#2\or\#3\or\#4\fi
\or
Otherwise we know (or think we know) that this is a first, middle, or last column:
\#2\% % 1 First col
\or
\#3\% % 2 any middle col
\or
\#4\% % 3 last col
else
\ERROR wrong default given
\fi
\if@docolactionstar \else \docolactioncheck \fi
}

Here is the if used above:
\newif\if@docolactionstar

Because of extra data writing to the aux file the aux file will now contain something like the following after the document is processed the first time:
\relax
\mc@col@status{1}
\mc@set@col@status{lcol-1}{0}
\mc@col@status{2}
\mc@set@col@status{lcol-2}{0}
\mc@col@status{3}
\mc@set@col@status{lcol-3}{0}
\mc@col@status{4}
\mc@col@status{2}
\mc@col@status{3}
\mc@set@col@status{lcol-4}{0}

The \mc@col@status line denotes the column type and has been written out just before corresponding the column box was placed onto the page. The \mc@set@col@status lines have been written out as part of shipping the column boxes out, e.g., \mc@set@col@status{lcol-1}{0} was therefore somewhere within the first column as it appears between \mc@col@status{1} and \mc@col@status{2}. The second argument in that line is the value used in the previous run (or zero if there was no previous run. We can use this to determine if a rerun is necessary.

Thus with this knowledge we can set things up to get the labels working.
\mc@col@status

When the aux file is read in \mc@col@status is used to set \mc@curr@col@status:
\def\mc@col@status#1{%
  \gdef\mc@curr@col@status(#1)}

And when \mc@set@col@status is executed we can simply set up the label by associating it with the \mc@curr@col@status and ignore the second argument:
\def\mc@set@col@status#1#2{%
  \global\expandafter\let\csname #1\endcsname \mc@curr@col@status}  
\AtEndDocument{\def\mc@set@col@status#1#2{%
  \ifnum #2=\mc@curr@col@status
  \@tempswatrue
  \fi}%}

Finally, as part of determining in which column we are, we used a switch inside \mc@col@status@write to determine if we are in the first column or not.
\newif\ifmc@firstcol
\mc@firstcoltrue

Index

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

Symbols
\@Mvi .......................... 241
\@footnotetext .......................... 963

B
\balance@columns 592
\balance@columns@out .......................... 555

C
\c@collectmore .......................... 304
\c@columnbadness 943
\c@finalcolumnbadness .......................... 943
\c@inrows .......................... 941
\c@unbalance .......................... 305
\col@number .......................... 304
\colbreak@box 1022
\columnbreak 899
\columns@percent 3
\columns@prule 3
\columns@prule@color 3, 372

D
\docolaction .......................... 1069
\doublecol@number 304

E
\emergencystretch 834
\endmulticols 243
\endmulticols* 976
\enough@room 135
\F
\flushcolumns 550
\G
\get@keptmarks 867
\I
\if@boxedmulticols 132
\ifsh@ntking 550
\init@mult@footins 226
\K
\kept@botm@argin 850
\kept@firstm@rk 850
\kept@topm@rk 850
\L
\leave@mult@footins 487
\LR@column@boxes 1024
\LR@multicol@columns 1061
\M
\maxbalancingoverflow 804
\mc@align@columns 1061
\mc@boxedresult 304
\mc@col@check@num 1068
\mc@col@status 1109
\mc@firstcol 1119
\mc@set@col@status 1111
\mult@cols 98
\mult@col@space 95
\mult@firstbox 806
\mult@footnotetext 92, 963
\mult@firstbox 806
\mult@g@rightbox 806
\mult@info 956
\mult@rightbox 806
\mult@column@out 377
\multicol@leftm@argin 240
\multicol@base@lineskip 3, 304
\multicol@min@depth@string 371
\multicol@pretolerance 3, 304
\multicols 71
\multicols* 963
\multicol@sep 3, 304
\multicol@tolerance 3, 304
\N
\newcolumn 1007
\O
\page@free 304
\P
\V
\vfilmaxdepth 375

Change History

v1.0c
\enough@room: Penalty 0 added to empty the contribution list. .......................... 10

v1.0d
General: All lines shortened to 72 or less. .......................... 1

v1.0e
General: Redefinition of description env. to use \descriptionmargin=5pt in documentation. .......................... 1
\prepare@multicols: \textwidth changed to \linewidth. .......................... 13
Setting of \columnwidth removed. .......................... 13
So this file will work with the ‘twocolumn’ command. .......................... 13

v1.0f
General: Changed \z@ to 0pt in redefinition of description. .......................... 1

v1.1a
General: \multicol@sep changed to \multicolsep. .......................... 1
\flushcolumns: \flushedcolumns renamed to \flushcolumns. .......................... 21

v1.2a
\balance@columns: Group around main loop removed. .......................... 23
\prepare@multicols: \pretolerance -1 because it nearly never succeeds. .......................... 13
\set@floatcmds added. .......................... 13
\set@emergency@stretch added. .......................... 13
\vbadness 10001 now. .......................... 13
\set@floatcmds: Macro added. .......................... 27
\set@emergency@stretch: Macro added. .......................... 27
\spec\s: Float boxes freed. ................. 21
v1.3a
\balance@columns: Changed \vtop to \vbox. 26
v1.3b
\endmulticols: Do \penalty with \addpenalty 15
\enough@room: Do \penalty with \addpenalty 11
\multicols: Minimum of two columns ........ 9
v1.3c
\balance@columns: \global\advance left over 25
from older code. .......................... 1
Limit column height to \@colroom. 26
\endmulticols: Check closing env. 15
\multicolumn@out: \unboxing avoided. 19
\multicols: Check if footnotes are actually 19
present before issuing a warning.
Unnecessary code removed 20
\prepare@multicols: null inserted and
removed in output 11
\reinsert@footnotes: \unboxing avoided. 18
v1.3d
\@unbalance: \col@number set to one. 16
v1.4a
General: Added support for multicol in inner
mode ................................. 1
\balance@columns: Changed to proper 25
\endlinechar \message 1
\multicolor@out: \botmark set to 10
\splitbotmark .......................... 19
\prepare@multicols: Checking for text losses.
Conditional code for boxed mode added. 11
kept marks initiated 12
v1.4b
\balance@columns: New algorithm for start
height ................................. 23
v1.4c
\endmulticols: But ignore \nobreak in
\addpenalty ......................... 15
\enough@room: But ignore \nobreak in
\addpenalty ......................... 11
\multicolor@out: Type set optional arg inside 10
\prepare@multicols: Using ................ 13
v1.4d
\balance@columns: \vsize added to tracing 23
info ................................. 10
\par added to allow for correct inner test 10
v1.4e
\multicolor@out: \global was probably wrong but
at least unnecessary 10
\multicolor@out: Only change \kept@topmark if \kept@topmark
non-empty 20
v1.4f
General: Added mark tracing with
tracingmulticols ≥ 2 1
\kept@topmark: Init to double brace pair ... 28
v1.4i
\multicolor@out: Set \kept@topmark to
\botmark ....................... 20
\prepare@multicols: \kept@topmark
initialized. 12
\setan@emergencystretch: Setting of
\emergencystretch on top removed. 27
v1.4j
\multicolors: Maximum of 5 columns (temp) ... 9
v1.4l
\multicolors: \@totalleftmargin now in
\prepare@multicols .................. 10
\page@sofar: use \multicolor@leftmargin
instead of \@totalleftmargin 16, 17
\prepare@multicolors: saved
\@totalleftmargin .................. 11
v1.4m
\endmulticols: Check \partial@page being
emptied 15
v1.4n
\return@nonemptymark: Make marks robust . 28
v1.4o
\prepare@multicols: \topskip locally zeroed. 11
v1.4p
\multicolor@out: Use different \vsize
setting .......................... 20
\prepare@multicolors: Code moved to
\setmult@vsize ...................... 12
Use different \vsize setting 12
\setmult@vsize: Macro added. 13
v1.5?
\balance@columns: Allow columns to come out
a bit long or short 23
Do splitting to zero here 23
Initialize \last@try 24
Show natural size 24
\endmulticols: Splitting off zero box moved to
\balance@columns 14
\leave@mult@footins: Macro added 20
\multicolors: Penalty moved to later point 10
\multicolor@out: Use \leave@mult@footins 18
\prepare@multicolors: Use \init@mult@footins 12
v1.5a
\balance@columns: New box mechanism ....... 24
\LR@columns: New box mechanism 32
\multicolor@out: New box mechanism 18, 19
\multicolors: Allow 10 columns again 9
\page@sofar: New box mechanism 16, 17
\prepare@multicolors: Add offset to
\doublecolnumber 11
v1.5b
\balance@columns: New badness mechanism ... 24
v1.5c
\balance@columns@out: added badness at
output routine exit 22
\endmulticols: Again use \penalty 15
\multicolor@out: Support \clearpage 18
\spec\s: Support \clearpage 20
v1.5d
\multicols@out: reinit \topmark .................. 19

v1.5c
\enough@room: Assign arg to skip register to be able to output value .................. 11

v1.5g
\set@floatcmds: Updated since floats have changed .................. 27

v1.5h
\balance@columns: Get kept marks first ........... 22
\page@sofar: Check for void boxes ................. 16

v1.5i
\page@sofar: But don’t remove original code. .. 16

v1.5j
\set@floatcmds: Updated since floats have changed again .................. 27

v1.5l
General: Try hard to explain unresolved reference that happens if \OnlyDescription is used .................. 7
\set@floatcmds: Added \@largefootcheck .. 27

v1.5m
General: Applied improvement of documentation, kindly done by Robin Fairbairns. .................. 1

v1.5n
\@footnotetext: Redefinition added pr/2664. .. 30
\prepare@multicols: Setting of \columnwidth added again pr/2664. .................. 13

v1.5o
\multicols: Redefinition of \@footnotetext only within env pr/2689. ................. 9

v1.5p
\balance@columns: Do not reset
\mult@gfirstbox (pr2739) .................. 25
\endmulticols*: Macro added .................. 31
\mult@cols: And removed the group again six years later .................. 10
\multicols: Macro added .................. 30

v1.5q
\@footnotetext: Use \@footnotetext but with local change to \columnwidth. ........... 30
\mult@footnotetext: Macro removed again ... 30
\multicols: Use \@footnotetext but with local change to \columnwidth. ........... 9

v1.5s
\speci@ls: check for \stop penalty pr/2873 .. 20

v1.5t
\return@nonemptymark: re-add \mark command which was commented out by mistake at some point in 1998 (pr/2978) .................. 28

v1.5u
\balance@columns@out: Support \columnbreak 21
\colbreak@box: Macro added .................. 32
\columnbreak: Macro added .................. 31
\mult@column@out: Support \columnbreak .. 18
\speci@ls: Support \columnbreak .................. 20

v1.5v
\balance@columns: Added tracing statements for trial unsuccessful .................. 25
Check last column if it contains forced break and reject trial if that is the case .... 25
\balance@columns@out: Added debug statements for column break support .... 21
\mult@column@out: Added debug statements for column break support .................. 18
\speci@ls: Added debug statements for column break support .................. 20

v1.5w
\multicols: Make \@footnotetext long to allow multi-paragraph footnotes. ........... 9

v1.5x
\endmulticols: Detect and fix problem if a multicols ends at the top of a page .... 15

v1.5y
\balance@columns: Limit column height only in unrestricted mode (pr/3212) ........... 26

v1.5z
\page@sofar: Ensure that column rule has always \normalcolor .................. 17

v1.5a1
\c@finalcolumnbadness: Change wrong default for \multicolovershoot to zero (pr/3465). 30
\mult@cols: Add a kern to cancel potential depth of previous line .................. 10
\page@sofar: Suppress interline glue at this point .......................... 17

v1.6a
General: New option grid .................. 9
\LR@column@boxes: Preparing for adjusting \prevdepth .................. 32
\mult@cols: Adjust spacing .................. 10
\page@sofar: Preparing for adjusting \prevdepth .................. 17

v1.6b
\page@sofar: Different info display .................. 16

v1.6c
\set@mult@vsize: Collect one line per column more .................. 14

v1.6d
\endmulticols: Catch problem with \columnbreak in last line .................. 14

v1.6e
\multicols: Avoid self-referencing definition of \@footnotetext (pr/3618) .................. 9

v1.6f
\balance@columns: \colbreak guard in the wrong position .................. 25
need to use \mult@grightbox in the loop .... 25
\columnseprulecolor: Make the color of the rule a hook .......................... 17
\page@sofar: Make the color of the rule a hook .......................... 17

v1.6g
\set@floatcmds: Added \@minipagefalse .................. 27
v1.6h
\setfloatcmds: Use \@endfloatbox to better support the modifications done by the float package .......................... 27
v1.7a
General: RL language support added .......................... 32
v1.7b
General: RL language support fixed .......................... 32
\page@sofar: RL language support fixed .......................... 17
v1.8a
\balance@columns: Balancing concept improved 26
\balance@columns@out: Balancing concept improved .......................... 22
Support for \enlargethispage .......................... 22
\maxbalancingoverflow:
\maxbalancingoverflow parameter added 27
\multi@column@out: Only re-add output penalty if it was explicitly set .......................... 19
Support for \enlargethispage .......................... 18
v1.8b
\balance@columns: Remove discardable items at the end of split boxes .......................... 24
\multi@column@out: And 20odd years later conclude that this was wrong and unboxing is always needed. .......................... 19
Remove discardable items at the end of split boxes .......................... 19
v1.8c
\endmulticols: Add \color@endgroup to prevent color leak .......................... 14
\multicol@col: Add \color@setgroup to prevent color leak .......................... 10
v1.8d
\multi@column@out: Reset \@mparbottom after page finishes .......................... 20
v1.8e
General: Support \docolaction 9, 33
\LR@column@boxes: Support \docolaction 32
\LR@column@boxes: Support \docolaction 33
v1.8f
\endmulticols: Discard spaces before adding \color@endgroup .......................... 14
v1.8g
\page@sofar: Now adjusting \prevdepth .......................... 17
Resetting \prevdepth in the right place .......................... 17
Warn if value is exceeded not when equal .......................... 17
v1.8h
\balance@columns: All column boxes should obey \maxdepth (pr/4395) .......................... 23
Do not report overfull .......................... 24
Use \vfilmadepth 25, 26
\endmulticols: Set \prevdepth for current vlist when returning from multicols environment .......................... 15
\endmulticols*: Use \vfilmadepth 31
\multi@column@out: Use \vfilmadepth 19
\vfilmadepth: Macro added (pr/4395) .......................... 18
v1.8i
\endmulticols*: Add \null to hide the final fill and only add vertical space if not doing \raggedcolumns .......................... 31
v1.8j
\balance@columns: Use \vfil in this case 25
\endmulticols*: Redesign the whole approach. 31
\multi@column@out: Set \boxmaxdepth 18
\vfilmadepth: Use only \textwidth for stretching .......................... 18
v1.8k
General: The new switch .......................... 30
\balance@columns: .......................... 26
\remove@discardable@items removed 22
Do not use \remove@discardable@items here 24
Finish the new conditional .......................... 27
Init \ifforcedbreak@leftover 24
Watch out for columns growing too far in case of forced breaks 25
\balance@columns@out: Add \remove@discardable@items at the end of the last column when balancing. 22
No additional penalty here 22
Use \@Mv and not \break in case this forced break is not used on this page 21
\endmulticols*: And a bit more redesign because of the change in \remove@discardable@items 31
\multi@column@out:
\remove@discardable@items removed 19
\special@ls: Remove discardable items just before a forced break 20
v1.8l
\balance@columns: Added additional tracing if column overflows .......................... 25
\remove@discardable@items: Another rewrite of \remove@discardable@items hopefully okay now 31
\special@ls: Remove discardable items just before a forced break 20
\special@ls: Remove discardable items just before a forced break 20
v1.8m
\vfilmadepth: Add \@textfloatsheight after page finishes .......................... 20
v1.8n
\@textfloatsheight: Add \null to hide the final fill and only add vertical space if not doing \raggedcolumns 31
v1.8o
\@textfloatsheight: Add \null to hide the final fill and only add vertical space if not doing \raggedcolumns 31
v1.8p
\prepare@multicols: Make \clearpage behave like \newpage (pr/4511) .......................... 12
v1.8r
\@Mvi: Macro added .......................... 14
\balance@columns@out: Re-add the final penalty 22
\endmulticols: Use special penalty to signal end of environment ................. 15
\special: Handling end of env through special penalty ......................... 21

v1.8s
\endmulticols: Support for \docolaction (issue/39) ......................... 15

v1.8t
\multicols*: Re-add end penalty for \multicols* environment to guard against leftovers (git/53) ......................... 31

v1.8u
\docolaction: Support star with \docolaction 33

v1.8v
\multi@column@out: Removed dead code, the case where this can go wrong is too obscure to worry about it (gh/101) ................. 18

v1.8w
\balance@columns: Provide minrows counter for balancing .................. 23
\c@minrows: Provide minrows counter for balancing ........................ 30

v1.8x
General: Use \@maxdepth not \maxdepth (gh/190) .......................... 9
\balance@columns: Use \@maxdepth not \maxdepth (gh/190) ............. 23
\special@ls: Use \@maxdepth not \maxdepth (gh/190) ....................... 20

v1.8y
\vfilmaxdepth: Use \@maxdepth not \maxdepth (gh/190) ..................... 18
\mult@gfirstbox: Allow for 20 columns (gh/237) .......................... 27
\multicolmindepthstring: Make column min depth customizable (gh/698) .......................... 17

v1.9a
\columnbreak: Added optional argument for conditional break ............ 31
\newcolumn: Macro added .................................................. 32

v1.9b
General: Swap names \mult@gfirstbox and \mult@firstbox (gh/701) .......... 1
\columnbreak: Corrected error message text (gh/703) ...................... 31
\mult@gfirstbox: Drop one unnecessary box allocation (gh/701) .......... 27

v1.9c
General: Added rollback to v1.8 .................................. 1

v1.9d
\newcolumn: Guard the \vfill (sx/624940) .. 32

v1.9e
\endmulticols: Delay returning boxed multicols (gh/1002) ............... 14, 15

v1.9f
\multicolmindepthstring: Make column min depth customizable (gh/698) ................. 17
\page@sofar: Make column min depth customizable (gh/698) ............... 17