An environment for multicolumn output

Frank Mittelbach
Email: see top of the source file
Printed August 31, 2020

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 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 \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 \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 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 \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

\docolaction{first}{middle}{last}

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 colaction when loading the package.

∗This file has version number v1.8y, last revised 2019/12/09.
†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.
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:

```latex
\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\textsuperscript{2e}.

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\textsuperscript{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

```latex
\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\textsuperscript{2}.

As its first action, the `multicols` environment measures the current page to determine whether there is enough room for some

\textsuperscript{1}I started with the algorithm given in the \TeXbook 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.)

\textsuperscript{2}This is dictated by lack of time. To implement floats one has to reimplement the whole \LaTeX output routine.
portion of multicolumn output. This is controlled by the \(\text{dimen}\) variable \texttt{\premulticols} which can be changed by the user with ordinary \LaTeX{} commands. If the space is less than \texttt{\premulticols}, a new page is started. Otherwise, a \texttt{\vskip} of \texttt{\multicolsep} is added.\footnote{Actually the added space may be less because we use \texttt{\addvspace} (see the \LaTeX{} manual for further information about this command).}

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

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

\begin{verbatim}
\begin{multicols}{3}
[\section{The User Interface}] ...
\end{verbatim}

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

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

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

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

The color of the rules separating the columns can be specified through \texttt{\columnsprulecolor}. 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{\multicolbaselineskip} which is added to the \texttt{\baselineskip} parameter inside the \texttt{\multicols} environment. Please use this parameter with care or leave it alone; it is intended only for package file designers since even small changes might produce totally unexpected changes to your document.

2.1 Balancing columns

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

Paragraphing in \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 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 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{\multicolpretolerance} parameter for the \texttt{\tolerance} value inside the \texttt{\multicols} environment. Its default value is 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{\multicolpretolerance} at the beginning of every \texttt{\multicols} environment one can locally overwrite this default by assigning \texttt{\tolerance=\(d\text{esired value}\)}). There also exists a \texttt{\multicolposttolerance} 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{The reason behind this behavior is the asynchronous character of the \TeX \ pagebuilder. However, this could be avoided by defining very complicated output routines which don't use \TeX primitives like \texttt{\_insert} but do everything by hand. This is clearly beyond the scope of a weekend problem.} Therefore the \texttt{\multicols} environment gives a warning message\footnote{This message will be generated even if there are no footnotes in this part of the text.} whenever it is unable to use all the material considered so far.

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

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

If there is only a small amount of material available for balancing 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 distributed to the earlier columns even if that means that later columns are empty or partially empty. This is controlled through the counter `\texttt{\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 `\texttt{\unbalance}' counter, to some positive \texttt{(number)}. This will make all but the right-most column \texttt{(number)} of lines longer than they would normally have been. `Lines' in this context refer to normal text lines (i.e. one \texttt{\baselineskip} apart): thus, if your columns contain displays, for example, you may need a higher \texttt{(number)} to shift something from one column into another. A negative value can make sense if you have \texttt{\minrows} and want to locally adjust that.

Unlike `\texttt{\collectmore}', the `\texttt{\unbalance}' counter is reset to zero at the end of the environment so it only applies to one \texttt{\multicols} environment.

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

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

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

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

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

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

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

Finally, when doing the balancing at the end, columns may become higher than the remaining available space. In that case the algorithm aborts and instead generates a normal
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 \texttt{multicols*} instead of 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 \LaTeX that it should break the first column after this particular line?”. The \texttt{pagebreak} command (which works with the two-column option of \LaTeX) is of no use here since it would end the collection phase of \texttt{multicols} and thus all columns on that page. So with version 1.5u the \texttt{columnbreak} command was added. If used within a paragraph it marks the end of the current line as the desired breakpoint. You can observe its effect on the previous page where three lines of text have been artificially forced into the second column (resulting in some white space between paragraphs in the first column).

2.4 Floats inside a \texttt{multicols} environment

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

2.5 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 \texttt{\RLmulticolcolumns} (switching to right-to-left typesetting) and \texttt{\LRmulticolcolumns} (switching to left-to-right typesetting) the latter being the default.

2.6 Warnings

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

\texttt{Underfull \hbox (badness \ldots)}

As the columns are often very narrow \LaTeX 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.

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

Floats and marginpars not allowed inside ‘multicols’ environment!

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

Very deep columns! Grid alignment might be broken

This message can only appear if the option \texttt{grid} was chosen. In that case it will show up if a column has a very large depth so that \texttt{multicols} is unable to back up to its baseline. This is only relevant if one tries to produce
2.7 Tracing the output

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

\texttt{⟨number⟩} = 1. \TeX will now tell you, whenever it enters or leaves a \texttt{multicol}s environment, the number of columns it is working on and its decision about starting a new page before or after the environment.

\texttt{⟨number⟩} = 2. In this case you also get information from the balancing routine: the heights tried for the left and right-most columns, information about shrinking if the \texttt{raggedcolumns} declaration is in force and the value of the ‘unbalance’ counter if positive.

\texttt{⟨number⟩} = 3. Setting \texttt{⟨number⟩} 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.

\texttt{⟨number⟩} ≥ 4. Setting \texttt{⟨number⟩} to such a high value will additionally place an \texttt{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 \texttt{multicol.bug} file this new release enhances the \texttt{multicol}s environment by allowing for balancing in arbitrary contexts. It is now, for example, possible to balance text within a \texttt{multicol}s or a \texttt{minipage} as shown in 2 where a \texttt{multicol}s environment within a \texttt{quote} environment was used. It is now even possible to nest \texttt{multicol}s environments.

The only restriction to such inner \texttt{multicol}s 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 \texttt{multicol}s: The balancing routine is now placed into a single macro which is called \texttt{balance@columns}. This means that one can easily try different balancing routines by rewriting this macro. The interface for it is explained in table 1. There are several improvements possible, one can think of integrating the \texttt{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 \texttt{multicol.dtx} directly!

3.2 Preface to version 1.2

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

But actually resetting such parameters to zero or even worse to a negative value won’t give better line breaks inside the \texttt{multicol}s environment. \TeX’s line breaking algorithm will only look at those possible line breaks which can be reached without a badness higher than the current value of \texttt{tolerance} (or \texttt{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@gfirstbox, the next in register \mult@firstbox + 2, ..., only the last one as an exception in register \mult@grightbox. Furthermore it has to set up 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 1: Interface description for \balance@columns

| Possible, then, as a last resort, T\hbox{E}X will produce overfull boxes. All those (and only those) possible break points will be considered and finally the sequence which results in the fewest demerits will be chosen. This means that a value of \(-1000\) for \adjdemerits instructs T\hbox{E}X to prefer visibly incompatible lines instead of producing better line breaks.

However, with T\hbox{E}X 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 T\hbox{E}X (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 L\hbox{A}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 T\hbox{E}X 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 T\hbox{E}Xs capabilities, really perfect solutions would need a different approach than it was done in T\hbox{E}Xs 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 T\hbox{E}X, 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 L\hbox{A}TEX on the .dtx file.

1 (*driver)
2 \documentclass{ltxdoc}

We use the balancingshow option when loading multicol\ls 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.

5 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.
6 \setlength{\textwidth}{39pc}
7 \setlength{\textheight}{54pc}
8 \setlength{\parindent}{1em}
9 \setlength{\parskip}{0pt plus 1pt}
10 \setlength{\headsep}{20pt}
11 \setlength{\columnsep}{1.5pc}
We want a rule between columns.
15 \setlength{\columnseprule}{.4pt}
We also want to ensure that a new \multicols environment finds enough space at the bottom of the page.
16 \setlength{\premulticols}{6\baselineskip}
When balancing columns we disregard solutions that are too bad. Also, if the last column is too bad we typeset it without stretch.

4.2 Identification and option processing

We start by identifying the package. Since it makes use of features only available in \LaTeX 2ε we ensure that this format is available. (Now this is done earlier in the file.)

8 (*package)
38 \% \NeedsTeXFormat{LaTeX2e}
39 \% \ProvidesPackage{multicol} [...].../...
41 \% v... 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 \onecolumn in which case \multicols does work, we don’t make this an error.

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

19 \setcounter{IndexColumns}{4}
20 \let\DescribeMacro\SpecialUsageIndex
21 \let\DescribeEnv\SpecialEnvIndex
22 \renewcommand{\PrintMacroName}[1]{\null}
23 \CodelineIndex
24 \%\DisableCrossrefs \% Partial index
25 \%\RecordChanges \% Change log
Line numbers are very small for this article.

26 \renewcommand{\theCodelineNo}{\scriptsize\rm\arabic{CodelineNo}}
27 \settowidth{\MacroIndent}{\scriptsize\rm 00\ }
30 \begin{document}
31 \typeout
32 {****************************************
33 ^^^J* Expect some Under- and overfull boxes.
34 ^^^J****************************************}
35 \DocInput{multicol.dtx}
36 \end{document}
37 (/driver)
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

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, \TeX{} will complain about a missing number at this point. The error recovery mechanism will then use zero, which isn’t a good choice in this case. So we should now test whether everything is okay. The minimum is two columns at the moment.

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 \TeX{} stack overflow as this would generate a self-referencing definition.

Now we can safely look for the optional arguments.

This argument should be a ⟨dimen⟩ denoting the minimum free space needed on the current page to start the environment. If the user didn’t supply one, we use \premulticols as a default.

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

```
\par
\ifinner \@boxedmulticolstrue
```

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

```
\else
  \ifnum \doublecol@number>\z@
  \@boxedmulticolstrue
  \fi
  \fi
```

Then we look to see if statistics are requested:

```
\mult@info\z@
{Starting environment with \the\col@number space columns}
```

In boxed mode we add some more info.

```
\if@boxedmulticols\MessageBreak (boxed mode)\fi
```

Then we measure the current page to see whether a useful portion of the multicolumn environment can be typeset. This routine might start a new page.

```
\enough@room{#2}%
```

Now we output the first argument and produce vertical space above the columns. (Note that this argument corresponds to the first optional argument of the \texttt{multicols} environment.) For many releases this argument was typeset in a group to get a similar effect as \texttt{\textbackslash two\textbackslash column[...]} 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.

```
\par\addvspace\multicolsep
```

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

```
\ifdim\prevdepth = -\@m\p@
  \else
```

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 \texttt{\baselineskip} grid. We know that the box produced by \texttt{multicols} has \texttt{\topskip} at its top so that also needs to be taken into account.

```
\divide\tempcnta@\tempcntb\baselineskip
\advance\tempcnta@\tempcntb\@ne
\advance\tempcntb@\prevdepth
\advance\tempcntd@\tempcntb\baselineskip
\advance\tempcntd@\topskip
\kern-\tempcntd@
```

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

```
\begingroup
\prepare@multicols
```

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

```
\if@boxedmulticols
\setbox\mult@box\vbox\bgroup
\color@setgroup
```

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

```
\fi
```

We finish by suppressing initial spaces.

```
\ignorespaces}
```

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

```
\newif\if@boxedmulticols
\@boxedmulticolsfalse
\newbox\mult@box
```

The \texttt{\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 \texttt{\pagetotal} from \texttt{\pagegoal}. This isn’t entirely correct since it doesn’t take the ‘shrinking’ (i.e. \texttt{\pageshrink}) into account. The ‘recent contribution list’ might be nonempty so we start with
\par and an explicit \penalty.\footnote{See the documentation of \endmulticols for further details.} Actually, we use \addpenalty to ensure that a following \addvspace will ‘see’ the vertical space that might be present. The use of \addpenalty will have the effect that all items from the recent contributions will be moved to the main vertical list and the \pagetotal value will be updated correctly. However, the penalty will be placed in front of any dangling glue item with the result that the main vertical list may already be overfull even if \TeX is not invoking the output routine.

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 \addvspace couldn’t detect preceding glue. So this was changed to \addpenalty. But this turned out to be not enough as \addpenalty will not add a penalty when \nobreak is true. Therefore we force this switch locally to false. As a result there may be a break between preceding text and the start of a 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
Now we test whether tracing information is required:
142 \mult@info\z@\relax
143 {Current page:\MessageBreak
144 height=\%
145 \the\pagetotal: used \the\pagetotal
146 \space -> free=\the\page@free
147 \MessageBreak
148 needed \the\@tempskipa
149 \space(for #1)}\%
Our last action is to force a page break if there isn’t enough room left.
150 \ifdim \page@free <\#1\newpage \fi
151 \fi}

When preparing for multicolumn output several things must be done.
152 \def\prepare@multicols{%
We start saving the current \@totalleftmargin and then resetting the \parshape in case we are inside some list environment. The correct indentation for the multicols environment in such a case will be produced by moving the result to the right by \multicols@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.
153 \multicols@leftmargin\@totalleftmargin
154 \@totalleftmargin\z@
155 \parshape\z@
We also set the register \doublecol@number for later use. This register should contain $2 \times \col@number$. This is also an indicator that we are within a multicols environment as mentioned above.
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
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.
165 \nointerlineskip \{\topskip\z@\null\%
166 \output{\global\setbox\partial@page\vbox
167 \global\setbox\current@page\vbox
168 %}
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.

\(\text{\PartialPage}\)

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

\(\text{\PartialPage}\)

Finally we have to initialize \Keep@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 al-
ways contain what its name promises because \TeX
sometimes calls the output routine for float manage-
ment.\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 pur-
pose this doesn’t matter as we use \Keep@topmark
only to initialize \Firstmark and \Botmark of a fol-
lowing page if we don’t find any marks on the current
one.

\(\text{\PartialPage}\)

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 \PartialPage to put the
actual free room into this variable.

\(\text{\PartialPage}\)

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 be-
cause we want to get the original value for \vsize
restored in case this multicols environment ends on
the same page where it has started.

\(\text{\PartialPage}\)

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

\(\text{\PartialPage}\)

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 docu-
ment, 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.

\(\text{\PartialPage}\)

For the same reason (page-wide footnotes), the
\texttt{\multicols} 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 \PartialPage was first processed). This
will reduce the free space (i.e. \pagetotal) by the
appropriate amount since we have changed the mag-
nification factor, etc. above.

\(\text{\PartialPage}\)

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 \PartialPage 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 start-
ing the multicols environment).

\(\text{\PartialPage}\)
Additionally, we advance \texttt{\set@floatcmds} which handles any redefinitions of \texttt{\set@floatcmds}. This macro is used to set up the parameters associated with footnote floats. It can be redefined by applications that require different amount of spaces when typesetting footnotes.

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 \texttt{\baselineskip - \topskip} if you think about it.
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.

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

In versions prior to 1.8r the balancing at the end of the environment was done by changing the output routine from \mult@col@number@out to \balance@columns@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.

When the end of the multicol 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.

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.

\begin{verbatim}
235 \def\set@mult@vsize#1{% 
236 \vsize@col@number@baselineskip 
237 \@tempdima=baselineskip 
238 \advance@tempdima-\topskip 
239 \vsize\col@number\vsize 
240 \@collectmore@baselineskip }

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

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

244 \balance@columns

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

245 \return@nonemptymark(first)\% 
246 \kept@firstmark 
247 \return@nonemptymark(bot)\% 
248 \kept@botmark 
249 \page@sofar 
250 \global\let\kept@firstmark 
251 \@kept@firstmark 
252 \@kept@botmark 
253 \@kept@botmark 
254 \*marktrace

255 \mult@info{tu@} 
256 \{Restore kept marks togetMessageBreak first: \meaning\kept@firstmark 
257 \MessageBreak bot\space\space: 
258 \MessageBreak \meaning\kept@botmark }\% 
259 \*marktrace

This finishes the code for the “boxed” case.

261 \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.

262 \ifdim\pagegoal=\maxdimen 
263 \ifvoid\colbreak@box\else 
264 \mult@info{Re-adding forced 
265 break(s) for splitting}\% 
266 \unvbox\colbreak@box\fi 
267 \fi

If we are in an unrestricted multicol environment we end the current paragraph above with \par but this isn’t sufficient since \TeX’s \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 \eject which was placed at the end of the contribution list. Then the \page_builder was called (an explicit \penalty will empty the contribution list), but the line with the \eject 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 \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.

\end{verbatim}
point we need to use \penalty not \addpenalty to ensure that a) the recent contributions are emptied and b) that the very last item on the main vertical list is a valid break point so that \TeX breaks the page in case it is overfull.

\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.

\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{\eject} 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{\eject} will get lost if we don’t put this back by hand.

\ifvbox\partial@page

\unvbox\partial@page\fi

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

\global\let\kept@firstmark\@empty
\global\let\kept@botmark\@empty

⟨*marktrace⟩

\mult@info\tw@

⟨/marktrace⟩

\prevdepth\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 write another column status into the \texttt{.aux} file to support \texttt{docolaction} in case we have nested environments.

\if@boxedmulticols
\mc@col@status@write
\else
\reinsert@footnotes
\fi

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 \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.

\prevdepth\z@

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

\mult@info\z@

⟨Ending environment⟩

\if@boxedmulticols
\space(boxed mode)\fi

\}

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

\newcount\c@unbalance
\newcount\c@collectmore

\global\c@unbalance\z@
\global\c@collectmore\z@

\footnote{Actually, we are still in a group started by the \texttt{\begin} macro, so \texttt{\global} must be used anyway.}
In the new \LaTeX{} release \texttt{\col@number} is already allocated by the kernel, so we don’t allocate it again.

\begin{verbatim}
302 \newcount\col@number
303 \newcount\doublecol@number
304 \newcount\multicoltolerance
305 \newcount\multicolpretolerance
306 \newdimen\full@width
307 \newdimen\page@free
308 \newdimen\premulticols
309 \newdimen\postmulticols
310 \newskip\multicolsep
311 \newskip\multicolbaselineskip
312 \newbox\partial@page
313 \newbox\last@line
\end{verbatim}

And here are their default values:

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

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

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

\section{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 \texttt{Plain TeX} policy to avoid an overflow of the save stack.

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

\begin{verbatim}
324 \def\process@cols#1#2{\count@#1\relax
325 \loop
326 \langle\texttt{debug}\rangle
327 \typeout{Looking at box \the\count@}
328 \langle\texttt{/debug}\rangle
329 \ifnum\count@<\doublecol@number
330 \repeat}
\end{verbatim}

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

\begin{verbatim}
335 \ifvoid\count@
336 \setbox\count@\hbox to\hsize{\hfill}
337 \else
338 \wd\count@\hsize
339 \fi}
\end{verbatim}

Now we give some tracing information.

\begin{verbatim}
343 \count@\col@number \advance\count@\m@ne
344 \mult@info\tw@\relax
345 \ifnum\count@<\doublecol@number
346 \repeat}
\end{verbatim}

\begin{verbatim}
333 \def\page@sofar{\%}
334 \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.\footnote{You will see the reason for this numbering when we look at the output routines \texttt{\multi@column@out} and \texttt{\balance@columns@out}.}

First we ensure that the columns have equal width. We use \texttt{\process@cols} for this purpose, starting with \texttt{\count@=\mult@rightbox}. Therefore \texttt{\count@} loops through \texttt{\mult@rightbox, \mult@rightbox + 2,...} (to \texttt{\doublecol@number}).

\begin{verbatim}
334 \process@cols\mult@rightbox
\end{verbatim}

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

\begin{verbatim}
335 {\ifvoid\count@
336 \setbox\count@\hbox to\hsize{\hfill}
337 \else
338 \wd\count@\hsize
339 \fi}
\end{verbatim}

Now we give some tracing information.

\begin{verbatim}
340 \count@\col@number \advance\count@\m@ne
341 \mult@info\tw@\relax
342 \ifnum\count@<\doublecol@number
343 \repeat}
\end{verbatim}

At this point we should always be in vertical mode.

\begin{verbatim}
350 \ifvmode\else\errmessage{Multicol Error}\fi
\end{verbatim}

Now we put all columns together in an \texttt{\hbox} of width \texttt{\full@width} (shifting it by \texttt{\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.

\mointerlineskip

As mentioned earlier we want to have the reference point of the box we put on the page being at the baseline of the last line of the columns but we also want to ensure that the box has no depth so that any following skip is automatically starting from that baseline. We achieve this by recording the depths of all columns and then finally backing up by the maximum. (perhaps a simpler method would be to assemble the box in a register and set the depth of that box to zero (not checked).

We need a global scratch register for this; using standard \TeX conventions we choose \dimen2 and initialize it with the depth of the character “p” since that is one of the depths that compete for the maximum.

\setbox\z@\hbox{p}\global\dimen\tw@\dp\z@

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 a similar look as far as the rules are concerned we force the depth to be at least the depth of a letter ‘p’ (which is what we set \dimen2 to above).

\rlap{\phantom p}\

The processed material might consist of a last line with a descender in which case the \prevedepth 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 \page@sofar is being called inside the OR then setting \prevedepth 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.

\prevedepth\z@
Now we can’t postpone the difficulties any longer. The `\multi@column@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 \texttt{\textasciitilde -10000}, otherwise the penalty which triggered the output routine is higher. Therefore it’s easy to distinguish both cases: we simply test this register.

\def\multi@column@out{\ifnum\outputpenalty <-\@M 
If this was a `\clearpage`, a float or a marginpar we call `\special` otherwise we construct the final page. For the next block of code see comments in section 7.2.

\ifvoid\colbreak@box\else
\mult@info{Re-adding forced break(s) for splitting}%
\setbox\@cclv\vbox{\unvbox\colbreak@box
\penalty-\@Mv
\unvbox\@cclv}%
\fi

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

\splittopskip\topskip
\splitmaxdepth\@maxdepth

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 `\partial@page` is already subtracted from `\@colroom` so we can use its value as a starter.

\dimen0\@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.

\divide\skip\footins\col@number
\ifvoid\footins \else
\leave@mult@footins
\fi

And there is one more adjustment that we have to make: if the user has issue 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.

\ifvbox\@kludgeins
\advance\dimen0 -\ht\@kludgeins
\shr@nkingtrue
\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).

\process@cols\mult@gfirstbox{\setbox\count@ \vsplit\@cclv to\dimen@}

After splitting we update the kept marks.

\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.

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

Then the last column follows.

\setbox\mult@rightbox \vsplit\@cclv to\dimen@
\set@keptmarks
\setbox\mult@rightbox\vbox to\dimen@
\{\unvbox\mult@rightbox
\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
\fi
\unnum\outputpenalty=\@M
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 \texttt{multicol} produced this warning regardless of whether or not footnotes were present, resulting in many unnecessary warnings.

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

If the ‘\texttt{tracingmulticols}’ counter is 4 or higher we also add a rule.

\texttt{\ifnum \c@tracingmulticols>\thr@@ \hrule\allowbreak \fi}

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

\texttt{\ifx\@empty\kept@firstmark \let\firstmark\kept@topmark \let\botmark\kept@topmark \else \let\firstmark\kept@firstmark \let\botmark\kept@botmark \fi}

We also initialize \texttt{\topmark} with \texttt{\kept@topmark}. This will make this mark okay for all middle pages of the \texttt{multicol} environment.

\texttt{\let\topmark\kept@topmark}

\texttt{\message{Use kept top mark:} \meaning{\kept@topmark}}

\texttt{\message{Produce first mark:} \meaning{\firstmark}}

\texttt{\message{Produce bot mark:} \meaning{\botmark}}

\texttt{\global\let\colroom\colht}

With a little more effort we could have done better. If we had, for example, recorded the shrinkage of the material in \texttt{\partial@page} it would be now possible to try higher values for \texttt{\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 \texttt{\boxmaxdepth} needs to be unrestricted so we set it back now as it was changed above.

\texttt{\boxmaxdepth\maxdimen}

The macro \texttt{\@makecol} adds all floats assigned for the current page to this page. \texttt{\@outputpage} ships out the resulting box. Note that it is just possible that such floats are present even if we do not allow any inside a \texttt{multicol} environment.

\texttt{\@makecol\@outputpage}

After the page is shipped out we have to prepare the keep marks for the following page. \texttt{\kept@firstmark} and \texttt{\kept@botmark} is then assigned to \texttt{\kept@topmark}.

\texttt{\let\topmark\kept@topmark}

\texttt{\global\let\colroom\colht}

\texttt{\global\@mparbottom \z@}

\texttt{\global\@textfloatsheight \z@}

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

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

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

This macro is used to subtract the amount of space occupied by footnotes for the current space from the space available for the current column. The space current column is stored in \dimen@. See above for the description of the default action.

We left out two macros: \process@deferreds and \speci@ls.

If the document ends in the middle of a multicols environment, e.g., if the user forgot the \end{multicols}, \TeX adds a very negative penalty to the end of the galley which is intended to signal the output routine that it is time to prepare for shipping out everything remaining. Since inside multicols the output routine of \TeX 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.

For the next block of code see comments in section 7.2.
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.

Additionally we empty the \@currlist to avoid later error messages when the \LaTeX output routine is again in force. But first we have to place the boxes back onto the \@freelist. (\@elt's default is \relax so this is possible with \xdef.)

Our next action is to (globally) empty \@deferlist and assign a new meaning to \@elt. Here \@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 \@toplist or \@botlist) or should wait for further processing.

Now we call \@tempb which has the form

\@elt(box register)\@elt(box register)...

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

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

The actual definitions are simple: we just switch to true or false depending on the desired action. To avoid extra spaces in the output we enclose these changes in \oslashpack...\eslashpack.

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

For this we need to put the contents of box 255 into \mult@box. For the next block of code see also comments in section 7.2. All forced breaks except the last are inside \colbreak@box so all we have to do is to concatenate this box with box \@cclv and put a penalty in between. Here we test if \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 \iftoo@bad is set to true.

If during balancing the columns got too long the flag \iftoo@bad is set to true.
We also have to re-add the end of environment penalty since after this page we may want balance the remaining material.
\penalty-\@Mvi
%

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

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 \multicolumn@out. This will also put the explicit \texttt{\textbackslash eject} penalty back so the current \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).
\else
If the balancing went ok, we are in the position to apply \texttt{\textbackslash page@sofar}. But first we have to set \texttt{\textbackslash vsize} to a value suitable for one column output.
\global\vsize@colrom
\global\advance\vsize@colrom\ht@partial@page

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

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

Then we return the first and bottom mark and the gathered material to the main vertical list.
\return@nonemptymark(first)\texttt{\textbackslash kept@firstmark}
\return@nonemptymark(bot)\texttt{\textbackslash kept@botmark}
\page@sofar

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{\textbackslash multicolsep} etc. this could produce funny results.
\penalty\z@
\fi
%

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

This macro now does the actual balancing.
\def\balance@columns{%
We start by setting the kept marks by updating them with any marks from this box. This has to be done \texttt{\textbackslash page@sofar}. But first we have to set \texttt{\textbackslash vsize} to a value suitable for one column output.
\global\vsize@colrom
\global\advance\vsize@colrom\ht@partial@page

We then continue by resetting trying to remove any discardable stuff at the end of \texttt{\textbackslash multicolor@out}. 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{\textbackslash splittopskip} glue at the beginning.
\setbox\multicolor@out\vbox{%
\penalty-\@M
\unvbox\multicolor@out
%
%}

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

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

The next step is a bit tricky: when \TeX{} assembles material in a box, the first line isn’t preceded by interline glue, i.e. there is no parameter like \texttt{\boxtopskip} in \TeX{}. This means that the baseline of the first line in our box is at some unpredictable point depending on the height of the largest character in this line. But of course we want all columns to align properly at the baselines of their first lines. For this reason we have opened \texttt{\textbackslash multicolor@out} 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 \split@topskip is inserted at the top of \mult@box which is exactly what we like to achieve.

Next we try to find a suitable starting point for the calculation of the column height. It should be less than the height finally chosen, but large enough to reach this final value in only a few iterations.

The formula which is now implemented will try to start with the nearest value which is a multiple of \baselineskip. The coding is slightly tricky in \TeX{} and there are perhaps better ways ...

\@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 number (the number of scaled points) then divide this by \baselineskip (also in scaled points) and then multiply this result with \baselineskip assigning the result to \dimen@. This makes \dimen@ \leq to \@tempdimena.

\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 subtract one \baselineskip.

\divide\dimen@-\baselineskip
\ifnum\dimen@>\@tempdima
\mult@info\@ne
\@gobbletwo
\dimen@\topskip
\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+\@tempdima\baselineskip\the\baselineskip\div@tempdima
\ifnum\dimen@>\@tempdima
\mult@info\@ne
\{Start value
\the\dimen@ \space ->
\the\@tempdima \space (corrected for minrows)\the\@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@-\unbalance\baselineskip

We type out statistics if we were asked to do so.

\mult@info\@ne
\{Balance columns\oneline:
\ifnum\unbalance=\z@\else
\{off balance=\number\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)\%\the\@tempdima
\fi

Now we try to find the final column height. We start by setting \vbadness to infinity (i.e. 10000) to suppress underfull box reports while we are trying to find an acceptable solution. We do not need to do it in a group since at the end of the output routine everything will be restored. The setting of the final columns will nearly always produce underfull boxes with badness 10000 so there is no point in warning the user about it.

\vbadness\@M

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

\vfuzz \maxdimen

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

\last@try-\p@ \loop

In order not to clutter up \TeX{}'s valuable main memory with things that are no longer needed, we empty all globally used box registers. This is necessary if we return to this point after an unsuccessful trial. We use \process@cols for this purpose, starting with \mult@grightbox. Note the extra braces around this macro call. They are needed since \Plain{} \TeX{}'s \loop...\repeat mechanism cannot be nested on the same level of grouping.

\process@cols\mult@grightbox
\{\global\setbox\count@\box\voidb@x\}

The contents of box \mult@box are now copied globally to box \mult@grightbox. (This will be the right-most column, as we shall see later.)
We start with the assumption that the trial will be successful. If we end up with a solution that is too bad we set \texttt{too@bad} to \texttt{true}. We also assume that all forced breaks (if any) will be used during balancing. If this is not the case we record this in \texttt{forcedbreak@leftover}.

Using \texttt{\vsplit} we extract the other columns from box register \texttt{\mult@grightbox}. This leaves box register \texttt{\mult@box} untouched so that we can start over again if this trial was unsuccessful.

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

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

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

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.

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

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

- box \texttt{\mult@rightbox} ←− all material
- box \texttt{\mult@gfirstbox} ←− first column
- box \texttt{\mult@gfirstbox} + 2 ←− second column
  ... ...
- box \texttt{\mult@grightbox} ←− last column

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

If \texttt{\raggedcolumns} is in force older releases of this file also shrank 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 \texttt{\raggedcolumns} means allows for all columns to run slightly short.

Then we give information about the last column.

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.

\footnote{With \TeX{} version 3.141 it is now possible to use \LaTeX{}'s \texttt{\newlinechar} in the \texttt{\message} command, but people with older \TeX{} versions will now get \texttt{\newline} instead of a new line on the screen.}
We check whether our trial was successful. The test used is very simple: we merely compare the first and the last column. Thus the intermediate columns may be longer than the first if \raggedcolumns is used. If the right-most column is longer than the first then we start over with a larger value for \dimen@.

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

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

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

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

\verb+\iftoo@bad+
\verb+⟨/badness⟩+
\verb+\advance\dimen@-\p@+
\verb+\repeat+

If we come out of the loop with the switch \texttt{forcedbreak@leftover} set to true then balancing has failed and we should cut a normal page. We indicate this below with \texttt{\too@badtrue} when any of the columns get too high, so we set this flag here too in order to get the same processing logic.

\verb+\ifforcedbreak@leftover+
\verb+\too@badtrue+
\verb+\else+
\verb+\dimension@ holds the height that was determined by the balancing loop. If that height for the columns turns out to be larger than the available space (which is \texttt{@colroom}) we squeeze the columns into the space assuming that they will have enough shrinkability to allow this.\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.}

However, this squeezing should only be done if we are balancing columns on the main galley and \emph{not} if we are building a boxed multicol (in the latter case the current \texttt{@colroom} is irrelevant since the produced box might be moved anywhere at a later stage).

Then we move the contents of the odd-numbered box registers to the even-numbered ones, shrinking them if requested. We have to use \texttt{\vbox} not \texttt{\vtop} (as it was done in the first versions) since otherwise the resulting boxes will have no height (\texttt{\TeXbook} page 81). This would mean that extra \texttt{\topskip} is added when the boxes are returned to the pagebuilder via \texttt{\page@sofar}.

Amount that balancing is allowed to overflow the available column space. We default to 12pt which means about one line in most layouts.

\newdimen\maxbalancingoverflow
\maxbalancingoverflow=12pt

When putting the final column together we want overfull information:

\verb+\vfuzz\z@+
\verb+\setbox\count@\vbox to\dimen@+
\verb+\{%+
\verb+\vskip\z@+
\verb+\@plus-\multicolundershoot+
\verb+\@minus-\multicolovershoot+
\verb+\unvbox\@tempcnta+
\verb+\ifshr@nking\vfilmaxdepth\fi+
\verb+\}%+

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 \texttt{\maxbalancingoverflow} we accept it still to avoid getting very little material for the next page (which we would then have difficulties to balance).

\verb+\ifnum\badness>\@M+
\verb+\vfuzz\maxdimen% no overfull warning+
\verb+\setbox\@tempboxa\vbox to\dimen@+
\verb+\{\vskip-\maxbalancingoverflow+
\verb+\unvcopy\count@}+
\verb+\ifnum\badness>\@M+
\verb+\mult@info@one+
\verb+{Balanced column more than+
\verb+the\maxbalancingoverflow\space+
\verb+too large}}%+

Fail the balancing attempt:

\verb+\too@badtrue+
\verb+\else+

Otherwise report that there is a problem but within the accepted boundary.

\verb+\mult@info@one+
\verb+{Balanced column+
\verb+too large, but less than+
\verb+the\maxbalancingoverflow}\%+
\verb+\fi+
\verb+\fi+
\verb+\}%+

Finally end the \texttt{\ifforcedbreak@leftover} conditional.

\verb+\fi+
\verb+\fi+

\verb+\page@sofar+
\verb+\count@\vbox to\dimen@+
\verb+\{%+
\verb+\dimen@-\p@+
\verb+\iftoo@bad+
\verb+\repeat+
\verb+\too@badtrue+
\verb+\else+
\verb+\dimension@ holds the height that was determined by the balancing loop. If that height for the columns turns out to be larger than the available space (which is \texttt{@colroom}) we squeeze the columns into the space assuming that they will have enough shrinkability to allow this.\footnote{Should get cleaned up as we now have two different routes to reach this part of the processing.}

However, this squeezing should only be done if we are balancing columns on the main galley and \emph{not} if we are building a boxed multicol (in the latter case the current \texttt{@colroom} is irrelevant since the produced box might be moved anywhere at a later stage).

Then we move the contents of the odd-numbered box registers to the even-numbered ones, shrinking them if requested. We have to use \texttt{\vbox} not \texttt{\vtop} (as it was done in the first versions) since otherwise the resulting boxes will have no height (\texttt{\TeXbook} page 81). This would mean that extra \texttt{\topskip} is added when the boxes are returned to the pagebuilder via \texttt{\page@sofar}.

Amount that balancing is allowed to overflow the available column space. We default to 12pt which means about one line in most layouts.
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 \TeX 3.0 \texttt{\emergencystretch} is undefined 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 formulæ.

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

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.
ize 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}
837 \def\kept@topmark{{}{}}
838 \let\kept@firstmark\@empty
839 \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}
840 \def\return@nonemptymark#1#2{%
841 \ifx#2\@empty
842 \else
\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}
843 \meaning#2}
844 \if\fi
845 \fi}
846 \meaning#2}
847 \nobreak
848 \fi
849 \fi}
\end{verbatim}

Since the contents of the mark may be arbitrary \LaTeX{} code we better make sure that it doesn’t get expanded any further. (Some expansion have been done already during the execution of \texttt{\markright} or \texttt{\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}
850 \expandafter}%
851 \meaning#2}
852 \nobreak
853 \fi}
\end{verbatim}

We don’t want any breakpoint between such a returned mark and the following material (which is usually just the box where the mark came from).

\begin{verbatim}
854 \def\get@keptmarks#1{%
\end{verbatim}

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

\begin{verbatim}
855 \meaning#1 \empty
856 \meaning#1 \empty
\end{verbatim}

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

\begin{verbatim}
857 \meaning#1 \empty
858 \meaning#1 \empty
\end{verbatim}

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}
859 \begin{group}
860 \vbadness\@M
861 \setbox#1\copy#1%
862 \setbox#1\vsplit#1to\maxdimen
863 \set@keptmarks
864 \end{group}
\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.

\begin{verbatim}
865 \set@keptmarks
866 \endgroup
\end{verbatim}

For debugging we take again a look at the box dimension which shouldn’t have changed.

\begin{verbatim}
867 \meaning#1 \empty
868 \meaning#1 \empty
\end{verbatim}

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

\begin{verbatim}
870 \def\set@keptmarks{%
871 \if\fi}
\end{verbatim}

If \texttt{\kept@firstmark} is empty we assume that it isn’t set. This is strictly speaking not correct as we loose the ability to have marks that are explicitly empty, but for standard \LaTeX{} application it is sufficient. If it is non-empty we don’t change the value—within the output routines it will then be restored to \texttt{\@empty}.

\begin{verbatim}
874 \if\fi\kept@firstmark\empty
\end{verbatim}
We now put the contents of `\splitfirstmark` into `\kept@firstmark`. In the case that there wasn't any mark at all `\kept@firstmark` will not change by that operation.

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

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.

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.

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 \TeX 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 Opt`. 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.

Save a previous skip (if there) and then remove it, we can’t really tell the difference between no skip and a skip of zero but that’s life.

If it was a zero skip (or none) we save the next previous skip (if any).

Otherwise we remove this “real” skip. Then we look if it was preceded by a penalty of 10000 (i.e., a `\nobreak`)

If so this was a `\vspace*` or something equivalent to it. Therefore we reintroduce the skips and stop. Otherwise we recurse.

If the first skip was a non-zero skip we recurse as well.

If the first skip was a non-zero skip we recurse as well.
6 Fixing the \textwidth

If we store the current column width in \textwidth we have to redefine the internal \textwidth 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 \textwidth is set to \textwidth.

This solves two problems: first redefinitions of \textwidth done by a class will correctly survive and second if multicols is used inside a minipage environment the special definition of \textwidth 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 Further extensions

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

7.1 Not balancing the columns

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

The code for this environment was suggested by Matthias Clasen.

\footnotetext done by a class will correctly survive and second if multicols is used inside a minipage environment the special definition of \textwidth 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.

If we store the current column width in \textwidth we have to redefine the internal \textwidth 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 \textwidth is set to \textwidth.

This solves two problems: first redefinitions of \textwidth done by a class will correctly survive and second if multicols is used inside a minipage environment the special definition of \textwidth 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.

If we aren’t balancing we change the \balance@columns@out to work like the normal output routine that cuts normal pages. However, there is a catch: In case the last page we cut (after seeing the end of the environment) is actually larger than a page (for example, if it contains more \columnbreak commands than columns) we end up with some leftover material that is returned to the main galley, but now the environment end penalty is missing. So we add another one here too. Of course that shouldn’t be done if there is really only a single final page, but fortunately in that case we have just finished a page and any penalty on the recent contributions will be discarded, thus the extra one is harmless—puh.

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

We do this as follows: first we ensure that we are back in vertical mode and then we cancel out \lastskip if it was positive (in case of a negative glue we assume that it was deliberate, for a deliberate positive glue one needs to use \vspace*). We can't simply use \remove@discardable@items here as this only works inside boxes but we are here on the main vertical list.

Then we back up by \prevdepth but not more than \boxmaxdepth so that a baseline of the last box is now at the bottom. This way the material will align properly in case something like \vfill spreads it out after all. Finally we append \vfil to put white space at the bottom of the column, but we only do this if we aren't anyway doing \raggedcolumns.

\begin{verbatim}
\namedef{endmulticols*}{% \par \ifdim\lastskip>\z@ \vskip-\lastskip \fi \ifdim \prevdepth>\z@ \vskip-\ifdim \prevdepth>\boxmaxdepth \boxmaxdepth \else \prevdepth \fi \fi \ifshr\nking\else \vfil \fi \else \prevdepth \fi \end{multicols}}
\end{verbatim}

\subsection{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 \vspliiting 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 \LaTeX output routine handling by introducing an additional penalty flag (i.e., a penalty which is forcing but higher than -10000 so that the output routine can look at this value and thus knows why it has been called).

Inside the output routine we test for this value and if it appears we do two things: save the galley up to this point in a special box for later use and reduce the \vsiz 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 \specs above.)

In the output routines that do the \vspliiting 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.

\begin{verbatim}
\ifnum\col@number<\tw@ \PackageError{multicol}{{noexpand\columnbreak outside multicol}}{This command can only be used within a multicol or multicol* environment.} \else \ifvmode \penalty -\@Mv \relax \else \@bsphack \vadjust{\penalty -\@Mv \relax} \@esphack \fi \fi
\end{verbatim}

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

\begin{verbatim}
\ifnum\col@number<\tw@ \PackageError{multicol}{{noexpand\columnbreak outside multicol}}{This command can only be used within a multicol or multicol* environment.} \else \ifvmode \penalty -\@Mv \relax \else \@bsphack \vadjust{\penalty -\@Mv \relax} \@esphack \fi \fi
\end{verbatim}

Need a box to collect the galley up to the column break.

\begin{verbatim}
\newbox\colbreak@box
\end{verbatim}

\subsection{Supporting right-to-left languages}

\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.

\begin{verbatim}
\def\LR@column@boxes{% \ifnum\col@number<\tw@ \PackageError{multicol}{{noexpand\columnbreak outside multicol}}{This command can only be used within a multicol or multicol* environment.} \else \ifvmode \penalty -\@Mv \relax \else \@bsphack \vadjust{\penalty -\@Mv \relax} \@esphack \fi \fi
\end{verbatim}

We loop through the columns with \process@cols

\begin{verbatim}
\process@cols\mult@gfirstbox{%}
\end{verbatim}
If the depth of the current box is larger than the maximum found so far in `\dimen2` we update that register for later use.

```latex
\ifdim\dp\count@>\dimen\tw@
  \global\dimen\tw@\dp\count@ \fi
```

If the `colaction` option is given we write out status information about the current column, otherwise the next command does nothing.

```latex
\mc@col@status@write
```

The typeset box followed by the column rule material

```latex
\box\count@
```

As you will have noticed, we started with box register `\mult@gfirstbox` (i.e. the left column). So this time `\count@` looped through 2, 4,... (plus the appropriate offset). Finally we add box `\mult@rightbox` and we are done. Again we may have to update `\dimen\tw@`.

```latex
\ifdim\dp\count@>\dimen\tw@
  \global\dimen\tw@\dp\count@ \fi
```

If the `colaction` option is given we write out status information about the last column, otherwise the next command does nothing.

```latex
\mc@lastcol@status@write
\box\mult@rightbox
```

Assembling the boxes for right to left typesetting is far more complicated. When I first tried to build a solution for this my thinking was that all that is necessary to do is to reverse the order of the columns. But such an approach produces a subtle bug: If we work this way then the first column put on the page will be the last column of the text to read. and this means that the order in which \TeX\ executes write statements or assemblies mark material will not happen in the order of the textual flow. So for, if example each column contains a section command then these sections will appear in reverse order in the table of content.

For this reason some amount of gymnastics is needed to add the columns in their natural flow.

```latex
\def\RL@column@boxes{%
  \ifdim\dp\mult@rightbox>\dimen\tw@
    \global\dimen\tw@\dp\mult@rightbox \fi
```

If the `colaction` option is given we write out status information about the last column, otherwise the next command does nothing.

```latex
\mc@lastcol@status@write
\box\mult@rightbox
```

The approach for the final column is similar only that we do not have to back up over any column gap.

```latex
\ifdim\dp\mult@rightbox>\dimen\tw@
  \global\dimen\tw@\dp\mult@rightbox \fi
```

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:

```latex
\hskip\full@width
```

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

```latex
\newcommand\\LRmulticolcolumns{\let\mc@align@columns\RL@column@boxes}
\newcommand\\RLmulticolcolumns{\let\mc@align@columns\LR@column@boxes}
```

The default is left-to-right:

```latex
\RLmulticolcolumns
```

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.

```latex
\process@cols\mult@gfirstbox{%
  \hskip\hsize
  \mc@align@columns
  \box\mult@gfirstbox
  \hskip\hsize
  \h skip\-\hsize
  \mc@col@status@write
  \box\count@
  \hskip\hsize
```

The approach for the final column is similar only that we do not have to back up over any column gap.

```latex
\ifdim\dp\mult@rightbox>\dimen\tw@
  \global\dimen\tw@\dp\mult@rightbox \fi
```

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:

```latex
\hskip\full@width
```

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

```latex
\newcommand\\LRmulticolcolumns{\let\mc@align@columns\RL@column@boxes}
\newcommand\\RLmulticolcolumns{\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).

We use one counter \mc@col@check@num to generate us unique label names. Each time we execute \docolaction we increment this counter to get a new name.

```
\newcount\mc@col@check@num
```

The generated “labels” are named \mc@col\the\mc@col@check@num and they hold as values the numbers 1, 2, or 3 denoting the current column type.

The \docolaction scans for a star and optional argument and 3 mandatory ones, but we do this in chunks (not having xparse available).

```
\newcommand\docolaction{\@docolaction}
```

First check is the support is enabled.

```
\ifx\mc@col@status@write\relax
\PackageError{multicol}{Option 'colaction' not selected}{\string\docolaction}
\fi
```

Then prepare \mc@col@type.

```
\edef\@docolactioncheck{\write\@auxout\mc@set@col@status\mc@col\the\mc@col@check@num\mc@col@type}\if@docolactionstar\@docolactioncheck\@docolaction\else\@docolactioncheck\fi
```

Finally check for a star, record this information and then call \docolaction to do the rest.

```
\if@docolactionstar
\edef\@docolactioncheck{\write\@auxout\mc@set@col@status\mc@col\the\mc@col@check@num\mc@col@type}
\else
\@docolactioncheck\fi
\{\@docolactiontrue\docolaction\}@docolactionfalse\docolaction\}
```

How does the column number get associated with our label? We do this by writing another line into the aux file. Here are the preparations.

```
\edef\@docolactioncheck{\write\@auxout\mc@set@col@status\mc@col\the\mc@col@check@num\mc@col@type}%
```

Where do we do the actual \write depends on the whether or not we gave seen a * . If yes, we do it first and then execute the code argument, otherwise we execute that code first and check at the point after that.

```
\if@docolactionstar\docolactioncheck\fi
```

We prefix with 0 so that an unknown label (that returns \relax ) will result in case 0

```
\ifcase \mc@col@type\relax
\else
\ERRORwrongdefaultgiven
\fi
```

Otherwise we know (or think we know) that this is a first, middle, or last column:

```
\ifcase #1\or #2\or#3\or#4\fi
```

```
\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@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.

When the aux file is read in \mc@col@status is used to set \mc@curr@col@status:
\begin{verbatim}
1081 \def\mc@col@status#1{%
1082 \gdef\mc@curr@col@status[#1}%
\end{verbatim}

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:
\begin{verbatim}
1083 \def\mc@set@col@status#1#2{%
1084 \global\expandafter\let\csname #1\endcsname\mc@curr@col@status}
\end{verbatim}

The above definition is being used when the .aux file is read in at the beginning. At the end we need a different definition to test if another typesetting run is needed. There we compare the value used in the current run (stored in the second argument) with the value used on the next run. If those two values differ we set @tempswa to false which will trigger the “Label(s) may have changed” warning.
\begin{verbatim}
1086 \AtEndDocument{\def\mc@set@col@status#1#2{%
1087 \ifnum #2=\mc@curr@col@status\else
1088 \@tempswatrue
1089 \fi}%
1090}
\end{verbatim}

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.
\begin{verbatim}
1091 \newif\ifmc@firstcol
1092 \mc@firstcoltrue
\end{verbatim}
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 954
\balance@columns 582
\balance@columns@out 546
\c@collectmore 299
\c@columnbadness 933
\c@finalcolumnbadness 933
\c@minrows 931
\c@unbalance 300
\col@number 299
\colbreak@box 996
\columnnbreak 980
\columns@p 3
\columns@rulecolor 3, 366
\docolaction 1043
\doublecol@number 299
\emergency@stretch 823
\endmulticols 242
\endmulticols* 967
\enough@room 136
\flushcolumns 541
\get@keptmarks 856
\if@boxedmulticols 133
\ifshr@ning 541
\init@mult@footins 226
\Kept@botmark 839
\Kept@firstmark 839
\Kept@topmark 839
\leave@mult@footins 481
\LR@column@boxes 998
\LRmulticol@columns 1035
\maxbalancingoverflow 794
\mc@align@columns 1035
\mc@col@check@num 1042
\mc@col@status 1083
\mc@firstcol 1094
\mc@set@col@status 1085
\mult@cols 98
\mult@firstbox 796
\mult@footnotetext 954
\partial@page 299
\postmulticols 3, 299
\premulticols 2, 299
\preferred@keptmarks 598
\prepare@multicols 154
\process@cols 324
\process@deferreds 531
\raggedcolumns 541
\reinsert@footnotes 367
\remove@discardable@items 902
\return@nonemptymark 842
\RL@column@boxes 1011
\RLmulticol@columns 1035
\set@floatcmds 828
\set@keepmarks 872
\set@mult@vsize 230
\set@emergency@stretch 823
\special@ls 485
\page@free 299
\page@sofar 333
\vfilmaxdepth 369

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
\prepare@multicols: \textwidth changed to \linewidth. 13
Setting of \columnwidth removed. 13
So this file will work with the ‘wocolumn’ command. 13
General: Redefinition of description env. to use \descriptionmargin=5pt in documentation. 1
v1.0f
General: Changed \z@ to 0pt in redefinition of description. 1
v1.1a
\flushcolumns: \flushedcolumns renamed to \flushcolumns. 21
General: \multicol@sep changed to \multicol@sep. 1
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
\special@ls: Float boxes freed. 21
General: Added mark tracing with
\multicols: Minimum of two columns ..
\balance@columns: Changed \vtop to \vbox ..
\endmulticols: Do \penalty with \addpenalty
\enough@room: Do \penalty with \addpenalty
\multicols: Maximum of 5 columns (temp) ..
\prepare@multicols: \kept@topmark initialized.
\set@mult@vsize: Setting of \vsize locally zeroed.
\prepare@multicols: Use \vsize setting.
\prepare@multicols: Code moved to
\set@mult@vsize: Use different \vsize setting.
\set@mult@vsize: Macro added.
\balance@columns: Allow columns to come out
\balance@columns: New algorithm for start
\multicols: But ignore \nobreak in
\balance@columns: New box mechanism.
\multicols: But ignore \nobreak in
\multicols: Typeset optional arg inside group
\prepare@multicols: Using
\multicols: \vtop added to tracing info.
\multicols: \vtop added to tracing info.
\par added to allow for correct inner test.
\multicols: \vtop was probably wrong but
\multicols: \vtop added to tracing info.
\multicols: Retrace env. non-empty.
\kept@topmark: Init to double brace pair.
\kept@topmark: Added mark tracing with
\multicols: Added mark tracing with
\multicols: Set \kept@topmark to
General: Applied improvement of documentation, kindly done by Robin Fairbairns. 1

General: Try hard to explain unresolved reference that happens if \OnlyDescription is used. 7

\footnotetext: Redefinition added pr/2664. 30
\prepare@multicols: Setting of \columnwidth added again pr/2664. 13
\multicols: Redefinition of \footnotetext only within env pr/2689. 9

\footnotetext: Use \footnotetext but with local change to \columnwidth. 30
\multicols: Macro removed again. 30
\multicols: Use \footnotetext but with local change to \columnwidth. 9
\speci@s: check for \top penalty pr/2873 20
\return@nonemptymark: re-add \mark command which was commented out by mistake at some point in 1998 (pr/2978). 28
\balance@columns@out: Support \columnbreak 21
\columnbreakbox: Macro added. 31
\break@box: Macro added. 31
\multi@column@out: Support \columnbreak 18
\speci@s: Support \columnbreak 20
\balance@columns: Added tracing statements for trial unsuccessful. 25
\balance@columns: Added debug statements for column break support. 21
\multi@column@out: Added debug statements for column break support. 18
\speci@s: Added debug statements for column break support. 20
\multicols: Make \footnotetext long to allow multi-paragraph footnotes. 9
\endmulticols: Detect and fix problem if a multicols ends at the top of a page. 15
\balance@columns: Limit column height only in unrestricted mode (pr/3212). 26
\page@sofar: Ensure that column rule has always \normalcolor. 16
\column@boxes: Preparing for adjusting \pre@depth. 31, 32
\multi@cols: Adjust spacing. 10
\page@sofar: Preparing for adjusting \pre@depth. 17
\columnbreak: New option grid. 8
\page@sofar: Different info display. 16
\set@mult@vsize: Collect one line per column more. 14
\endmulticols: Catch problem with \columnbreak in last line. 14
\multicols: Avoid self-referencing definition of \footnotetext (pr/3618). 9
\balance@columns: /\columnbreak guard in the wrong position. 25
\columnbreakbox: Need to use \multigridbox in the loop. 24
\column@box: Make the color of the rule a hook. 17
\page@sofar: Make the color of the rule a hook. 16
\set@floatcmds: Added \minipagefalse. 27
\set@floatcmds: Use \@floatcheck to better support the modifications done by the float package. 27
v1.8v
  \endmulticols*: Redesign the whole approach. 31
  \multi@column@out: Set \boxmaxdepth 18
  \vfilmmaxdepth: Use only ‘0.0001fil’ for stretching 17
v1.8w
  \balance@columns: 26
  \remove@discardable@items removed 22
  Do not use \remove@discardable@items here 24
  Finish the new conditional 26
  Init \ifforcedbreak@leftover 23
  Watch out for columns growing too far in case of forced breaks 25
  \balance@columns@out: Add 31
  \remove@discardable@items at the end of the last column when balancing 21
  No additional penalty here 22
  Use \0Mv and not \break in case this forced break is not used on this page 21

v1.8x
  \balance@columns*: And a bit more redesign because of the change in \remove@discardable@items 31
  \multi@column@out:
    \remove@discardable@items removed 18
  \speci@ls: Remove discardable items just before a forced break 20
  General: The new switch 29

v1.8y
  \balance@columns: Added additional tracing if column overflows 25
v1.8z
  \remove@discardable@items: Another rewrite of \remove@discardable@items hopefully okay now 29
v1.8a
  \multi@column@out: Reset \@textfloatsheight after page finishes 19
  \c@unbalance: \col@number already initialized in the kernel, so not initializing it in the package in case the document is in two-column (pr/4435) 16
  \endmulticols*: Ensure we are back in vmode before using \prevdepth (pr/4448) 31

v1.8b
  \multi@column@out: Reset \@mparbottom after page finishes 19

v1.8c
  \endmulticols: Add \color@endgroup to prevent color leak 14
  \mult@cols: Add \color@setgroup to prevent color leak 10

v1.8d
  \multi@column@out: Reset \@mparbottom 19

v1.8e
  \LR@column@boxes: Support docolaction 31, 32
  \RL@column@boxes: Support docolaction 32
  General: Support docolaction 9, 32

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) 22
  Do not report overfull 23
  Use \vfilmmaxdepth 25, 26
  \endmulticols: Set \prevdepth for current vlist when returning from multicol environment 15
  \endmulticols*: Use \vfilmmaxdepth 31
  \multi@column@out: Use \vfilmmaxdepth 18
  \vfilmmaxdepth: Macro added (pr/4395) 17

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 \vfii in this case 25

\vfilmaxdepth: Remove discardable items just at the end of split boxes 24
\multi@column@out: Only re-add output penalty if it was explicitly set 18
Support for \enlargethispage 32

v1.8k
  \endmulticols*: Add \color@endgroup to prevent color leak 14
  \mult@cols: Add \color@setgroup to prevent color leak 10

v1.8l
  \multi@column@out: Reset \@mparbottom after page finishes 19

v1.8m
  \remove@discardable@items: Another rewrite of \remove@discardable@items hopefully okay now 29

v1.8n
  \c@unbalance: \col@number already initialized in the kernel, so not initializing it in the package in case the document is in two-column (pr/4435) 16
  \endmulticols*: Ensure we are back in vmode before using \prevdepth (pr/4448) 31

v1.8o
  \prepare@multicols: Make \clearpage behave like \newpage (pr/4511) 12

v1.8p
  \multi@column@out: Reset \boxmaxdepth 19

v1.8q
  \prepare@multicols: Make \clearpage behave like \newpage (pr/4511) 12

v1.8r
  \0Mvi: Macro added 14
v1.8s
  \balance@columns@out: Re-add the final penalty 21
  \endmulticols: Use special penalty to signal end of environment 15
  \speci@ls: Handling end of env through special penalty 20

v1.8t
  \endmulticols: Support for docolaction (issue/39) 15
v1.8t
\multicols*: Re-add end penalty for
\multicols* environment to guard against
leftovers (git/53) .......................... 30
v1.8u
\docolaction: Support star with \docolaction 33
v1.8v
\multicolumnout: Removed dead code, the
case where this can go wrong is too obscure
to worry about it (gh/101) ................... 18
v1.8w
\balancecolumns: Provide minrows counter
for balancing ............................. 23
\c@minrows: Provide minrows counter for
balancing ................................. 29
v1.8x
\balancecolumns: Use \@maxdepth not
\maxdepth (gh/190) ...................... 22
\speciols: Use \@maxdepth not \maxdepth
(gh/190) ................................. 20
\vfilmaxdepth: Use \@maxdepth not \maxdepth
(gh/190) ................................. 17
General: Use \@maxdepth not \maxdepth
(gh/190) ................................. 8
v1.8y
\multigfirstbox: Allow for 20 columns
(gh/237) ................................. 27
\multicols: Allow for 20 columns (gh/237) . 9