An environment for multicolumn output

Frank Mittelbach
Email: see top of the source file

Printed June 14, 2024

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

Abstract

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

Preface to version 1.8

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

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

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

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

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

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

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

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

\footnote{This file has version number v1.9h, last revised 2024/05/23.}

\footnote{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.

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.

For example:
\begin{verbatim}
\renewcommand \footnoterule{\% \kern-3pt\hbox to\textwidth{\hskip .6\textwidth \hrulefill }\% \kern2.6pt} \end{verbatim}

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 L\TeX\ 2\epsilon. 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 L\TeX, 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\footnote{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.)} to get together the few lines of code below and there is still a good chance that I missed something after all.

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

2 The User Interface

To use the environment one simply says
\begin{verbatim}
\begin{multicols}{⟨number⟩}
  ⟨multicolumn text⟩
\end{multicols}
\end{verbatim}

where ⟨number⟩ is the required number of columns and ⟨multicolumn text⟩ may contain arbitrary \TeX commands, except that floats and marginpars are not allowed in the current implementation\footnote{This is dictated by lack of time. To implement floats one has to reimplement the whole \TeX output routine.}.

As its first action, the multicols environment measures the current page to determine whether there is enough room for some portion of multicolumn output.
The space between columns is controlled by the \( \texttt{columnsep} \) parameter. 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 \( \texttt{columnseprule} \) 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 \( \texttt{columnseprulecolor} \). 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{multicol} environment. Please use this parameter with care or leave it alone; it is intended only for packing 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.\(^4\)

We therefore use a \texttt{multicol tolerance} parameter for the \texttt{tolerance} value inside the \texttt{multicol} 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{multicol} environment. Since \texttt{tolerance} is set to \texttt{multicol tolerance} at the beginning of every \texttt{multicol} environment one can locally overwrite this default by assigning \texttt{tolerance=\( \langle \text{desired value} \rangle \)}.

There also exists a \texttt{multicol pretolerance} parameter holding the value for \texttt{pretolerance} within a \texttt{multicol} 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{multicol} 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

\(^3\)Actually the added space may be less because we use \texttt{addvspace} (see the \LaTeX{} manual for further information about this command).

\(^4\)Look at the next paragraph, it was set with the \texttt{sloppy} declaration.

\(^5\)The reason behind this behavior is the asynchronous character of the \TeX{} \texttt{page builder}. 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.

\(^6\)This message will be generated even if there are no footnotes in this part of the text.
corresponding to this footnote might be set on the next one.\footnote{5} Therefore the \texttt{multicols} environment gives a warning message\footnote{6} 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{\textbackslash 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 \langle number \rangle, \TeX{} will consider \langle number \rangle more (or less) lines before making its final decision. So a value of \langle -1 \rangle 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{\textbackslash raggedcolumns} if you don’t want the bottom lines to be aligned. The default is \texttt{\textbackslash 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{\textbackslash 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 outs of material).

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

Unlike \texttt{\textbackslash collectmore}, the \texttt{\textbackslash unbalance} counter is reset to zero at the end of the environment so it only applies to one \texttt{\textbackslash 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{\textbackslash finalcolumnbadness}. 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{\textbackslash 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{\textbackslash multicolovershoot} the other \texttt{\textbackslash multicolundershoot}. They control the amount of space a column within the \texttt{\textbackslash multicols} environment is allowed to be “too full” or “too short” without affecting the column badness. They are set to 0pt and 2pt, respectively.

Finally, when doing the balancing at the end, columns may become higher than the remaining available space. In that case the algorithm aborts and instead generates a normal page. However, if the amount is not too large, e.g., a line or so, then it might be better to keep everything on the same page instead of starting a new page with just one line after balancing. So the parameter \texttt{\textbackslash maxbalancingoverflow} governs this process: only when the excess gets larger than its value balancing is aborted.
2.2 Not balancing the columns

Although this package was written to solve the problem of balancing columns, I got repeated requests to provide a version where all white space is automatically placed in the last column or columns. Since version v1.5u 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).

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

2.4 Floats inside a 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. Starred floats, e.g., \texttt{figure*}, denote page wide floats that are handled in a similar fashion as normal floats outside the multicols environment. However, they will never show up on the page where they are encountered. In other words, one can influence their placement by specifying a combination of t, b, and/or p in their optional argument, but h doesn’t work because the first possible place is the top of the next page. One should also note, that this means that their placement behavior is determined by the values of \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{\LMulticolcolumns} (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.
- \texttt{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]}.
- \texttt{Floats and marginpars not allowed inside ‘multicols’ environment!}
  - This message appears if you try to use the \texttt{\marginpar} command or an unsterrated version of the \texttt{figure} or \texttt{table} environment. Such floats will disappear!
- \texttt{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
a document where all text lines are aligned at an invisible grid, something that requires careful adjustment of many parameters and macros, e.g., heading definitions.

2.7 Tracing the output

To understand the reasoning behind the decisions \TeX\ makes when processing a \multicol\ 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 \multicol\ 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 \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 \hrule into your output, separating the part of text which had already been considered on the previous page from the rest. Clearly this setting should not be used for the final output. It will also activate even more debugging code for mark handling.

3 Prefaces to older versions

3.1 Preface to version 1.4

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

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

3.2 Preface to version 1.2

After the article about the \multicol\ environment was published in \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 \multicol\adjdemerits to be used for \adjdemerits, etc.) which would influence the line breaking algorithm.

But actually resetting such parameters to zero or even worse to a negative value won’t give better line breaks inside the \multicol\ environment. \TeX\’s line breaking algorithm will only look at those possible line breaks which can be reached without a badness higher than the current value of \tolerance (or \pretolerance in the first pass). If this isn’t pos-
The macro \balance@columns that contains the code for balancing gathered material is a macro without parameters. It assumes that the material for balancing is stored in the box \mult@box which is a \vbox. It also “knows” about all parameters set up by the \multicols environment, like \@colnumber, etc. It can also assume that \@colroom is the still available space on the current page.

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

Table 1: Interface description for \balance@columns

Table 4.1 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 \LaTeX, 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 \LaTeX on the \file{.dtx} file.

\begin{verbatim}
1 \(\ast\)driver
2 \documentclass[ltxdoc]
\end{verbatim}

We use the \balance@columns option when loading \multicols so that full tracing is produced. This has to be done before the \doc package is loaded, since \doc
\setemergencystretch: This is a hook for people who like to play around. It is supposed to set the \emergencystretch (dimen) register provided in the new \TeX{} 3.0. The first argument is the number of columns and the second one is the current \hsize. At the moment the default definition is 4pt \times #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 \texttt{multicol.sty} version 1.2. Both commands might be removed if good solutions to these open problems are found. I hope that these commands will prevent that nearly identical style files derived from this one are floating around.

otherwise requires \texttt{multicols} without any options.

First we set up the page layout suitable for this article.

We want a rule between columns.

We also want to ensure that a new \texttt{multicols} environment finds enough space at the bottom of the page.

When balancing columns we disregard solutions that are too bad. Also, if the last column is too bad we typeset it without stretch.

The index is supposed to come out in four columns. And we don’t show macro names in the margin.

4.2 Identification and option processing

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

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

The following redefinitions have to be moved until after the preamble because version 3 of \texttt{doc} resets them after the preamble (this is tmp, because hypdoc is not yet integrated, but as we all know, tmp solutions have a tendency to survive for a long time...).

Line numbers are very small for this article.

The following redefinitions have to be moved until after the preamble because version 3 of \texttt{doc} resets them after the preamble (this is tmp, because hypdoc is not yet integrated, but as we all know, tmp solutions have a tendency to survive for a long time...).

\begin{document}
\typeout{****************************************
\^^J* Expect some Under- and overfull boxes.
\'^{}}}****************************************}

\end{document}
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.

\def\multicols#1{\col@number#1\relax
If the user forgot the argument, \TeX{} will complain about a missing number at this point. The error recovery mechanism will then use zero, which isn’t a good choice in this case. So we should now test whether everything is okay. The minimum is two columns at the moment.

\ifnum\col@number<\tw@
\PackageWarning{multicol}{Using \number\col@number columns doesn’t seem a good idea.^^JTherefore I use two columns instead!}
\else  \fi

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

\ifnum\col@number>20
\PackageError{multicol}{Too many columns!}
\else  \fi

Within the environment we need a special version of the kernel \footnotetext command since the original sets the \hsize to \columnwidth which is not correct in the 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.

\ifx\@footnotetext\mult@footnotetext
\else
\let\orig@footnotetext\@footnotetext
\let\@footnotetext\mult@footnotetext
\fi

Now we can safely look for the optional arguments.

\long\def\mult@footnotetext#1{\begingroup\columnwidth\textwidth\orig@footnotetext{#1}\endgroup}

The \mult@cols macro grabs the first optional argument (if any) and looks for the second one.

\def\mult@cols[#1]{\@ifnextchar[\mult@@cols{#1}}

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.

\ifx\@footnotetext\mult@footnotetext\else
\fi

\long\def\mult@footnotetext#1{\begingroup\columnwidth\textwidth\orig@footnotetext{#1}\endgroup}

The \mult@cols macro sets the first optional argument (if any) and looks for the second one.

\ifx\@footnotetext\mult@footnotetext
{\mult@@cols{#1}}\fi

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 multicols should start in vertical mode. If we are not already there we now force it with \par since otherwise the test for “inner” mode wouldn’t show if we are in a box.

\par\ifinner \@boxedmulticolstrue

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

\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 multicols environment.) For many releases this argument was typeset in a group to get a similar effect as \twocolumn[...] where the argument is also implicitly surrounded by braces. However, this conflicts with local changes done by things like sectioning commands (which account for the majority of commands used in that argument) messing up vertical spacing etc. later in the document so that from version v1.5q this argument is again typeset at the outer level.

#1\par\addvspace\multicolsep

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

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

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

\divide\tempcnta\prevdepth

\advance\tempcnta\one

\dimen@\prevdepth

\advance\dimen@ -\tempcnta\baselineskip

\advance\dimen@ \topskip

\kern-\dimen@

\fi

We start a new grouping level to hide all subsequent changes (done in \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

\begingroup

\setbox\mult@box

\vbox\bgroup

\color@setgroup

\endgroup

\endgroup

\if@boxedmulticols\MessageBreak

(boxed mode)\fi%

We may have to reset some parameters at this point, perhaps \@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” multicols code.

\newif\if@boxedmulticols
\@boxedmulticolsfalse
\newbox\mult@box
The \enough@room macro used above isn’t perfect but works reasonably well in this context. We measure the free space on the current page by subtracting \pagetotal from \pagegoal. This isn’t entirely correct since it doesn’t take the ‘shrinking’ (i.e. \pageshrink) into account. The ‘recent contribution list’ might be nonempty so we start with \par and an explicit \penalty.\footnote{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.

\[\text{137} \text{def\enough@room#1\%}\]

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

\[\text{138} \text{iffboxedmulticols\else}\]
\[\text{139} \par\]

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.

\[\text{140} \text{\bgroup}@nobreakfalse\addpenalty\z@\egroup}\]
\[\text{141} \page@free \pagegoal\]
\[\text{142} \text{advance } \page@free \ \text{–} \ \text{pagetotal}\]

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.

\[\text{143} \text{\setminus\spaceskip#1\relax}\]

Now we test whether tracing information is required:

\[\text{144} \text{\multi@info\z@}\]
\[\text{145} \{\text{Current page:\MessageBreak}\}
\text{height=}\%\]
\[\text{146} \text{\the\pagegoal: used \the\pagetotal}\]
\[\text{147} \text{\space → free=\the\page@free}\]
\[\text{148} \MessageBreak\]
\[\text{149} \text{needed \the@tempskipa}\]

Our last action is to force a page break if there isn’t enough room left.

\[\text{150} \text{iffdim \page@free <#1\newpage fi}\]
\[\text{151} \text{fi}\]

When preparing for multicolumn output several things must be done.

\[\text{154} \text{def\prepare@multicols\%}\]

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

\[\text{155} \text{\multicol@leftmargin@talleftmargin}\]
\[\text{156} \text{\talleftmargin@z@}\]
\[\text{157} \parshape@z@\]

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

\[\text{158} \text{\doublecol@number@col@number}\]
\[\text{159} \text{\multiply\doublecol@number\tw@}\]
\[\text{160} \text{\advance\doublecol@number\mult@rightbox}\]
\[\text{161} \text{\ifboxedmulticols}\]
\[\text{162} \text{\let@kept@firstmark\kept@firstmark}\]
\[\text{163} \text{\let@kept@botmark\kept@botmark}\]
\[\text{164} \text{\global@let@kept@firstmark\kept@firstmark\@empty}\]
\[\text{165} \text{\global@let@kept@botmark\kept@botmark\@empty}\]
\[\text{166} \text{\else}\]

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.

\[\text{167} \text{\nointerlineskip \{\topskip@z@\null\%}\]
\[\text{168} \text{\output\%}\]
\[\text{169} \text{\global@setbox\partial@page\vbox}\]
\[\text{170} \text{\{}%\]

\footnote{See the documentation of \endmulticols for further details.}
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 \texttt{\textbackslash premulticols} and we are near the bottom of the page then it can happen that not the \texttt{\textbackslash eject} is triggering this special output routine but rather the overfull main vertical list. In that case we get another breakpoint through the \texttt{\textbackslash eject} penalty. As a result this special output routine would be called twice and the contents of \texttt{\textbackslash partial@page}, i.e. the material before the \texttt{\textbackslash 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 \texttt{\textbackslash premulticols} by using a suitable value for the second argument.

171 (\texttt{\check})
172 \texttt{\ifvoid\partial@page\textbackslash else}
173 \texttt{\PackageError{multicol}\%}
174 \texttt{\Erreur{saving partial page}\%}
175 \texttt{\The part of the page before}
176 \texttt{the multicols environment was}
177 \texttt{nearly full with\textbackslash "the result that starting the environment}
178 \texttt{will produce an overfull}
179 \texttt{page. Some\textbackslash "text may be lost!}
180 \texttt{Please increase \textbackslash premulticols}
181 \texttt{either generally or for this\%}
182 \texttt{\"environment by specifying a}
183 \texttt{suitable value in the second}
184 \texttt{optional argument to\textbackslash "the}
185 \texttt{multicols environment.}
186 \texttt{\unvbox\partial@page}
187 \texttt{\box\last@line}
188 \texttt{\box\last@line}
189 \texttt{\relax}
190 (\texttt{\check})
191 \texttt{\unvbox\@cclv}
192 \texttt{\global\setbox\last@line\lastbox}
193 \%}

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

\texttt{\prep@keptmarks}

Finally we have to initialize \texttt{\kept@topmark} which should ideally be initialized with the mark that is current on “top” of this page. Unfortunately we can’t use \texttt{\topmark} because this register will not always contain what its name promises because \LaTeX{} sometimes calls the output routine for float management. Therefore we use the second best solution by initializing it with \texttt{\firstmark}. In fact, for our purpose this doesn’t matter as we use \texttt{\kept@topmark} only to initialize \texttt{\firstmark} and \texttt{\botmark} of a following page if we don’t find any marks on the current one.

195 \texttt{\global\let\set@mult@footins\relax}
196 \texttt{\textbackslash eject}

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

197 \texttt{\advance\@colroom-\ht\partial@page}

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

198 \texttt{\set@mult@vsize\relax}

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

199 \texttt{\output{\multicolsout}}

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

200 \texttt{\init@mult@footins}

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

201 \texttt{\reinsert@footnotes}

Inside \texttt{\multicols} a \texttt{\clearpage} is fairly useless as we aren’t supporting floats. In fact, it can cause harm as it doesn’t know about the \texttt{\partial@page} and may therefore result in making columns too long.

\footnote{During such a call the \texttt{\botmark} gets globally copied to \texttt{\topmark} by the \TeX{} program.}
So we change that to behave like \newpage but also check if there are any deferred floats. If so, perhaps the user tried to place them through that \clearpage (but that needs to be done before starting the multicol environment).

\begin{verbatim}
\def\clearpage{%
  \ifx\deferlist\empty\else
    \PackageError{multicol}{Deferred floats not cleared}\%
    \spaceinside\space and doesn’t
    \newpage\space and doesn’t
    \MessageBreak
  \fi

  \def\clearpage{\MessageBreak
    \newpage}%

  \move it before the multicol

  \environment if you need it.}%
\end{verbatim}

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

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

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

\begin{verbatim}
\vbrightness\@Mi \hbrightness5000
\tolerance\multicoltolerance
\end{verbatim}

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

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

For use with the new \TeX we set \emergencyystretch to \colnumber × 4pt. However this is only a guess so at the moment this is done in a macro \setemergencyystretch which gets the current \hspace and the number of columns as arguments. Therefore users are able to figure out their own formula.

\begin{verbatim}
\setemergencyystretch\colnumber\hspace
\end{verbatim}

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

9I'm not sure that I really want page-wide footnotes. But balancing of the last page can only be achieved with this approach or with a multi-path algorithm which is complicated and slow. But it’s a challenge to everybody to prove me wrong! Another possibility is to reimplement a small part of the fire_up procedure in \TeX (the program). I think that this is the best solution if you are interested in complex page makeup, but it has the disadvantage that the resulting program cannot be called \TeX thereafter.

219 \set@floatcmds

Additionally, we advance \baselinestretch by \multicolbaselinestretch to allow corrections for narrow columns.

\begin{verbatim}
\advance\baselinestretch\multicolbaselinestretch
\end{verbatim}

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

\begin{verbatim}
\linewidth = (\colnumber - 1) × \columnsep \colnumber
\end{verbatim}

The formula above has changed from release to release. We now start with the current value of \linewidth so that the column width is properly calculated when we are inside a minipage or a list or some other environment. This will be achieved with:

\begin{verbatim}
\hspace\linewidth \advance\hspace\columnsep
\advance\hspace-\colnumber\columnsep
\divide\hspace\columnnumber
\end{verbatim}

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

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

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

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.

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

13
Since we have to set \texttt{\col@umber} columns on one page, each with a height of \texttt{\@colroom}, we have to assign \texttt{\vsize} = \texttt{\col@umber \times \@colroom} in order to collect enough material before entering the \texttt{\output} routine again. In fact we have to add another \texttt{(\col@umber - 1) \times (\baselineskip - \topskip)} if you think about it.

But this might not be enough since we use \texttt{\vs 19\split} 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 \texttt{#1} is \texttt{\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 \texttt{\@totalleftmargin}.

In versions prior to 1.8r the balancing at the end of the environment was done by changing the output routine from \texttt{\multi@column@out} to \texttt{\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 \texttt{\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 \texttt{multicols} environment is sensed we have to balance the gathered material. Depending on whether or not we are inside a boxed multicols different things must happen. But first we end the current paragraph with a \texttt{\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 \texttt{\color@endgroup} this can’t be done later in \texttt{\balance@Columns} as the color command will hide it.
If we are in an unrestricted `multicols` environment we end the current paragraph above with `\par` but this isn’t sufficient since \TeX’s `page_break` will not totally empty the contribution list. Therefore we must also add an explicit `\penalty`. Now the contribution list will be emptied and, if its material doesn’t all fit onto the current page then the output routine will be called before we change it. At this point we need to use `\penalty` not `\addpenalty` to ensure that a) the recent contributions are emptied and b) that the very last item on the main vertical list is a valid break point so that \TeX breaks the page in case it is overfull.

```
\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 `multicols` environment body was completely empty or if a multi-page `multicols` just ends at a page boundary we have the unusual case that the `\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 `multicols` (stored in `\partial@page`) 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
\ifmarktrace
  \mult@info\tw@  
\fi
```

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

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

```
\if@boxedmulticols
  \unvbox\mc@boxedresult
  \mc@colstatus@write
  \else
    \reinsert@footnotes
```

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

```
\global\c@unbalance\z@
```

Now it’s time to return any footnotes if we are in unrestricted mode. In boxed mode footnotes are kept inside, but in that case we have to first return the saved box to the page and then write another column status into the `.aux` file to support `\docolaction` in case we have nested environments.

```
\if@boxedmulticols
  \unvbox\mc@boxedresult
  \mc@colstatus@write
  \else
    \reinsert@footnotes
    \reinsert@footnotes
```

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

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

```
\ifdim \pagegoal=\maxdimen
  \global\vsize@colroom
  \else
    \enough@room\postmulticols
  \fi
```

There is one more thing to do: the balanced result of the environment is supposed to have a `\prevdepth` of zero as we backed up by its real `prevdepth` within `\page@sofar`. However if the balancing happened in the output routine then \TeX reverts to the

---

10This 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_break` 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.

11Actually, we are still in a group started by the `\begin` macro, so `\global` must be used anyway.
\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.

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

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

In the new \LaTEX{} release \texttt{\col@number} is already allocated by the kernel, so we don’t allocate it again.

\newcount\c@unbalance
\newcount\c@collectmore
\newcount\doublecol@number
\newcount\multicoltolerance
\newcount\multicolpretolerance
\newdimen\full@width
\newdimen\page@free
\newdimen\premulticols
\newdimen\postmulticols
\newskip\multicolsep
\newskip\multicolbaselineskip
\newbox\partial@page
\newbox\last@line
\newbox\mc@boxedresult

And here are their default values:
\c@unbalance = 0
\c@collectmore = 0
\col@number = 1
\multicoltolerance = 9999
\multicolpretolerance = -1
\premulticols = 50pt
\postmulticols = 20pt
\multicolsep = 12pt plus 4pt minus 3pt
\multicolbaselineskip = 0pt

4.4 The output routines

We first start with some simple macros. When typesetting the page we save the columns either in the box registers 0, 2, 4, ...) (locally) or 1, 3, 5, ...) (globally). This is \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.

\def\process@cols#1#2{\count@#1\relax
\loop\typeout{Looking at box \the\count@}
\relax\loop
\count@=\doublecol@number
\repeat}

12 You will see the reason for this numbering when we look at the output routines \texttt{\multi@column@out} and \texttt{\balance@columns@out}.

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

\def\page@sofar{\balance@columns
prepares its output in the even numbered scratch box registers. Now we output the columns gathered assuming that they are saved in the box registers 2 (left column), 4 (second column), ... However, the last column (i.e. the rightmost) should be saved in box register 0.12 First we ensure that the columns have equal width. We use \texttt{\process@cols} for this purpose, starting with \texttt{\count@ = \mult@rightbox}. Therefore \texttt{\count@} loops through \texttt{\mult@rightbox}, \texttt{\mult@rightbox + 2,... (to \doublecol@number)}.

\def\page@sofar{%
\def\page@sofar{\balance@columns
prepares its output in the even numbered scratch box registers. Now we output the columns gathered assuming that they are saved in the box registers 2 (left column), 4 (second column), ... However, the last column (i.e. the rightmost) should be saved in box register 0.12 First we ensure that the columns have equal width. We use \texttt{\process@cols} for this purpose, starting with \texttt{\count@ = \mult@rightbox}. Therefore \texttt{\count@} loops through \texttt{\mult@rightbox}, \texttt{\mult@rightbox + 2,... (to \doublecol@number)}.

\def\process@cols\mult@rightbox
\process@cols\mult@rightbox

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

\ifvoid\count@\relax\else\fi
\ifnum\count@<\doublecol@number
\repeat

16
Now we give some tracing information.
\count@\col@number \advance\count@\m@ne
\mult@info\z@

\MessageBreak
\the\full@width \space = indent + columns + sep =
\the\multicol@leftmargin \space
+ \the\col@number \space
x \the\hsize \space
+ \the\count@ \space
x \the\columnsep

At this point we should always be in vertical mode.
\ifvmode \else \errmessage{Multicol Error} \fi

Now we put all columns together in an \hbox of width \full@width (shifting it by \multicol@leftmargin to the right so that it will be placed correctly if we are within a list environment) and separating the columns with a rule if desired.

The box containing the columns has a large height and thus will always result in using \lineskip if the normal \baselineskip calculations are used. We therefore better cancel that process.
\nointerlineskip

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

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

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

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

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

By default the vertical rule between columns will be in \normalcolor.
\def\columnseprulecolor{\normalcolor}
Before we tackle the bigger output routines we define just one more macro which will help us to find our way through the mysteries later. \reinsert@footnotes will do what its name indicates: it reinserts the footnotes present in \footinbox so that they will be reprocessed by \TeX's page builder.

Instead of actually reinserting the footnotes we insert an empty footnote. This will trigger insertion mechanism as well and since the old footnotes are still in their box and we are on a fresh page \skipfootins should be correctly taken into account.

\begin{verbatim}
375 \def\reinsert@footnotes{% 
376 \ifvoid\footins\else \insert\footins{}\fi}
\end{verbatim}

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

\begin{verbatim}
377 \def\vfilmaxdepth{\vskip \z@ \@plus .0001fil \@minus \@maxdepth}
\end{verbatim}

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{-10000}, otherwise the penalty which triggered the output routine is higher. Therefore it’s easy to distinguish both cases: we simply test this register.

\begin{verbatim}
379 \def\multi@column@out{% 
380 \ifnum\outputpenalty <-\@M 
381 \ifvoid\colbreak@box\else
382 \leave@mult@footins
383 \fi
384 \setbox\count@\vsplit\@cclv to\dimen@ 
385 \penalty\vfilmaxdepth
386 \unvbox\colbreak@box
387 }% 
388 \leave\mult@footins
389 \fi
\end{verbatim}

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

\begin{verbatim}
390 \splittopskip\topskip
391 \splitmaxdepth\@maxdepth
\end{verbatim}

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

\begin{verbatim}
392 \boxmaxdepth\@maxdepth
\end{verbatim}

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

\begin{verbatim}
393 \dimen@\@colroom
\end{verbatim}

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

\begin{verbatim}
394 \divide\skip\footins\col@number
395 \ifvoid\footins\else
396 \leave\mult@footins
397 \fi
398 \fi
\end{verbatim}

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

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

\begin{verbatim}
399 \ifvbox@kludgeins 
400 \ifdim \wd\@kludgeins \textgreater \z@ 
401 \shrinkingtrue
402 \fi
403 \fi
\end{verbatim}

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

\begin{verbatim}
404 \process@cols\mult@firstbox{% 
405 \setbox\count@ \vsplit\@cclv to\dimen@
\end{verbatim}
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\count@  \vbox to\dimen@  \unvbox\count@  \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  \ifnum\outputpenalty=\@M  \else  \penalty\outputpenalty  \fi

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

\ifvoid\footins  \else  \PackageWarning{multicol}{\meaning\kept@topmark}  \use@marktrace

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

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

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

\iffalse\empty\kept@firstmark  \let\firstmark\kept@topmark  \let\botmark\kept@topmark  \else  \let\firstmark\kept@firstmark  \let\botmark\kept@botmark  \fi

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

\let\topmark\kept@topmark

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

Now we use \TeX'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 \TeX.} Admittedly this is a funny way to do it.

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

\boxmaxdepth\maxdimen  \setbox\@cclv\vbox{\unvbox\partial@page \page@sofar}
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{multicols} environment.

After the page is shipped out we have to prepare the kept marks for the following page. \texttt{\@kept@firstmark} and \texttt{\@kept@botmark} reinitialized by setting them to \texttt{\@empty}. The value of \texttt{\@botmark} is then assigned to \texttt{\@kept@topmark}.

\begin{verbatim}
465 \@makecol \@outputpage
\end{verbatim}

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

\begin{verbatim}
475 \global \@colroom \@colht
476 \global \@mparbottom \z@ 
477 \global \@textfloatsheight \z@ 
\end{verbatim}

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

\begin{verbatim}
478 \process@deferreds
479 \@whiledo{\if@fcolmade \fi}{\@outputpage
480 \global \@colroom \@colht
481 \global \process@deferreds}%
\end{verbatim}

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

\begin{verbatim}
482 \mult@info \one
483 \{(Colroom:\MessageBreak
484 \the\@colht\space 
485 after float space removed 
486 = \the\@colroom \&gobble)\%
\end{verbatim}

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

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

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

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

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

\begin{verbatim}
489 \def\leave@mult@footins{}
490 \advance\dimen@-\skip\footins
491 \advance\dimen@-\ht\footins
492}
\end{verbatim}

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

\begin{verbatim}
493 \def\speci@ls{%
494 \ifnum \outputpenalty <- -\@Mi
495 \PackageError{multicol}{Document end inside multicols environment}\@ehd
496 \else
497 \fi}
\end{verbatim}

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

\begin{verbatim}
499 \ifnum \outputpenalty < -\@MM
500 \PackageError{multicol}{Document end inside multicols environment}\@ehd
501 \else
502
\fi
\end{verbatim}

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

\begin{verbatim}
500 \ifnum \outputpenalty = -\@Mv
501 \mult@info \one{Forced column break seen}% 
502 \global \vsize \pagetotal
503 \global \advance \vsize -\@Mv
504 \global \setbox \colbreak@box 
505 \vbox{% 
506 \ifvoid \colbreak@box 
507 \else
508 \unvbox \colbreak@box
509 \vbox-%\penalty-\@Mv
510 \fi
\end{verbatim}

As this is the place of a forced break we now remove vertical white space just in front of it (or some of it at least) as it is quite likely that the break is not exactly in the right place, e.g., after a display environment (if \LaTeX{} would break here by its own it would break before the space following the display). Thus we rebox box 255 once (using \texttt{\@maxdepth} and calling \texttt{\remove@discardable@items} inside). The depth of 255 will then give us the depth the box would have had if it would have been a natural break. We then unbox 255 to get it into the \texttt{\colbreak@box} and then back up by this depth.
This will position the bottom of the box at its natural baseline which is useful for balancing later on.

\ifnum\outputpenalty = -\@Mvi
\if\@fcolmade
\begingroup
\let\@tempb\@deferlist

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.

\gdef\@deferlist{}\%
\let\@elt\@scolelt

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.
\@tempb \endgroup
\fi}

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

\newif\ifshr@nking

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 \@bsphack...\@esphack.

\def\raggedcolumns{\@bsphack\shr@nkingtrue\@esphack}
\def\flushcolumns{\@bsphack\shr@nkingfalse\@esphack}

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 (eg 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.

\setbox\mult@box\vbox{%
  \ifvoid\colbreak@box\else
    \unvbox\colbreak@box
    \penalty-\@Mv
    \mult@info\@ne{Re-adding forced break(s) in balancing}\%
  \fi
  \unvbox\@cclv
}\vbox{%
  \remove@discardable@items
}\balance@columns

The last column again is a forced break, so here we discard white space as well as that is normally unwanted.

\iftoo@bad
  \mult@info\@ne
  {Balancing failed ... cut a normal page}%
\fi
\vbox{%
  \setbox\@cclv
  \vskip\topskip
  \vskip-\splittopskip
  \unvbox\mult@box
}\vbox{%
  \ifvbox\@kludgeins
    \insert\@kludgeins\{
    \unvbox\@kludgeins\}
  \fi
}\unvbox\partial@page

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

\ifvoid\@kludgeins
  \insert\@kludgeins\{
  \unvbox\@kludgeins\}
\fi

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

\return@nonemptymark{first}\%
\kept@firstmark
\return@nonemptymark{bot}\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 \multicolsep etc. this could produce funny results.

\penalty\z@
\fi}

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

This macro now does the actual balancing.

\def\balance@columns{%
  \mult@column@out
  \ifvoid\colbreak@box
    \@cclv\mult@box
  \else
    \\penalty\z@\fi
\}

We start by setting the kept marks by updating them with any marks from this box. This also means that if there was an \enlargethispage present it will apply to this page as \mult@column@out will look at the status of \@kludgeins.

\penalty-\@Mvi
\fi
\}
box may contain `\columnbreak` in which case doing some artificial splitting to get the marks out still fails to see all marks unless we take some special precaution in `\get@keptmarks` (which is now done).

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

Then follow values assignments to get the `\vsplitting` right. We use the natural part of `\topskip` as the natural part for `\splittopskip` and allow for a bit of undershoot and overshoot by adding some stretch and shrink.

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

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

The code above sets `\@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@ ≤ to \@tempdimena`.

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`.

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.
But we don’t allow nonsense values for a start.}

\ifnum\dimen@<\topskip
\mult@info\@ne
\the\dimen@ \space -> \the\topskip \space (corrected)\%
\dimen@\topskip
\fi

Now we try to find the final column height. We start by setting \vbadness to infinity (i.e. 10000) to 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@gfirstbox{\global\setbox\count@\vsplit\mult@grightbox to\dimen@}

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

\global\setbox\count@\vbox to\dimen@{\unvbox\count@}%

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

\ifnum\c@tracingmulticols>\@ne\@tempcnta\count@ \advance\@tempcnta-\mult@grightbox \divide\@tempcnta \tw@ \message{^^JColumn \number\@tempcnta space badness: \the\badness space}%
\fi

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

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

\setbox\mult@nat@firstbox\vbox{\unvcopy\mult@gfirstbox}%

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

\global\setbox\mult@grightbox\vbox{\unvbox\mult@grightbox}%

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

\setbox\mult@nat@gfirstbox
\vbox{\unvcopy\mult@gfirstbox}%
After \texttt{process@cols} has done its job we have the following situation:

\begin{verbatim}
box mult@rightbox ← all material
box mult@firstbox ← first column
box \mult@firstbox + 2 ← second column

... ...

box \mult@grightbox ← last column
\end{verbatim}

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

\begin{verbatim}
682    \ifnum\c@tracingmulticols>\@ne
683        \message{^^JFirst column
684            = \the\dimen@ space
685                (\the\ht\mult@nat@firstbox)}\fi
\end{verbatim}

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.

\begin{verbatim}
690    \ifnum\c@tracingmulticols>\@ne
691        \message{<> last column = \the\ht\mult@grightbox}\fi
\end{verbatim}

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

\begin{verbatim}
905    \ifdim\ht\mult@grightbox >\dimen@
907        \too@badtrue
908        \ifnum\c@tracingmulticols>\@ne
909            \typeout{Rejected: last column too large!}\fi
910    \else
911        \global\setbox\mult@grightbox
912            \vbox to\dimen@{
913                \unvbox\mult@grightbox\vfil}\
917        \ifnum\badness>\c@finalcolumnbadness
918            \global\setbox\mult@grightbox
919                \vbox to\dimen@{
920                    \unvbox\mult@grightbox\vfil}\
921            \fi
922        \fi
\end{verbatim}

Thus if \texttt{@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.

\begin{verbatim}
14
\begin{verbatim}
143    \ifnum\badness>\c@finalcolumnbadness
144        \global\setbox\mult@grightbox
145            \vbox to\dimen@{
146                \unvbox\mult@grightbox\vfil}\
147        \fi
148    \fi
\end{verbatim}
\end{verbatim}

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.

\begin{verbatim}
160    \ifnum\badness<\c@finalcolumnbadness
161        \global\setbox\mult@grightbox
162            \vbox to\dimen@{
163                \unvbox\mult@grightbox\vfil}\
164        \fi
165    \fi
\end{verbatim}

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

\begin{verbatim}
177    \ifnum\badness>\c@finalcolumnbadness
178        \global\setbox\mult@grightbox
179            \vbox to\dimen@{
180                \unvbox\mult@grightbox\vfil}\
181        \fi
182    \fi
\end{verbatim}

\begin{verbatim}
192    \ifnum\badness<\c@finalcolumnbadness
193        \global\setbox\mult@grightbox
194            \vbox to\dimen@{
195                \unvbox\mult@grightbox\vfil}\
196        \fi
197    \fi
\end{verbatim}

\begin{verbatim}
205    \ifnum\badness<\c@finalcolumnbadness
206        \global\setbox\mult@grightbox
207            \vbox to\dimen@{
208                \unvbox\mult@grightbox\vfil}\
209        \fi
210    \fi
\end{verbatim}

\begin{verbatim}
218    \ifnum\badness<\c@finalcolumnbadness
219        \global\setbox\mult@grightbox
220            \vbox to\dimen@{
221                \unvbox\mult@grightbox\vfil}\
222        \fi
223    \fi
\end{verbatim}
If we have unprocessed forced breaks we normally reiterate with a larger column size to fit them in eventually. However, if there are simply too many of them (e.g., 3 forced breaks but only 2 columns to balance) then this will never succeed and we would continue growing the columns until we hit the largest possible column size. So in addition we check how big the column size is compared to available room and if we exceed this by \texttt{maxbalancingoverflow} we give up and instead of balancing cut another normal page. To be indicate this case we set \texttt{forcedbreak@leftover} to true.

\begin{verbatim}
\ifdim\dimen@ < \@tempdima \maxbalancingoverflow
  \typeout{Rejected: unprocessed forced break(s) in last column!}
\else
  \forcedbreak@leftovertrue
  \typeout{Failed: columns too large with unprocessed forced break(s)!}
\fi
\fi
\fi
\fi
\end{verbatim}

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

\begin{verbatim}
\ifdim\ht\mult@nat@firstbox<\dimen@ \last@try
  \too@badtrue
  \typeout{Retry: using natural height of first column!}
  \dimen@\ht\mult@nat@firstbox
  \last@try\dimen@
  \advance\dimen@-\p@
\fi
\fi
\fi
\fi
\end{verbatim}

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@}.

\begin{verbatim}
\iftoo@bad
\repeat
\ifforcedbreak@leftover \too@badtrue \else
\fi
\fi
\end{verbatim}

At that point \texttt{dimen@} 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 \texttt{not} if we are building a boxed multicolumn (in the latter case the current \texttt{@colroom} is irrelevant since the produced box might be moved anywhere at a later stage).

\begin{verbatim}
\if@boxedmulticols\else
\ifdim\dimen@>\@colroom
  \dimen@\@colroom
\fi
\fi
\end{verbatim}

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{TP\TeX}book page 81). This would mean that extra \texttt{\topskip} is added when the boxes are returned to the page-builder via \texttt{\page@sofar}.

\begin{verbatim}
\vfillmaxdepth
\end{verbatim}

when putting the final column together we want overfull information:

\begin{verbatim}
\vfuzz\z@
\setbox\count0\vbox to\dimen@ \page@sofar
\vfuzz\z@
\end{verbatim}

\texttt{\plus\multicolundershoot} \texttt{\minus\multicolovershoot} \texttt{\un vbox\@tempcnta} \texttt{\ifshrink\vfilmaxdepth\fi}
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).

\begin{verbatim}
\ifnum\badness>\@M
    \vfuzz\maxdimen % no overfull warning
    \setbox\@tempboxa \vbox to\dimen@{
        \vskip-\maxbalancingoverflow
        \unvcopy\count@}%
    \ifnum\badness>\@M
        \mult@info\@ne
        \{Balanced column more than
        \the\maxbalancingoverflow\space
        too large\}%
    \else
        \mult@info\@ne
        \{Balanced column too large, but less than
        \the\maxbalancingoverflow\}%
    \fi
\fi
\end{verbatim}

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

\begin{verbatim}
\ifnum\numexpr\count20-\count14-1<40
    \count14=\@cclv
\fi
\newbox\mult@rightbox
\newbox\mult@grightbox
\newbox\mult@firstbox
\newbox\mult@gfirstbox
\newbox\@tempa
\newbox\@tempa
\newbox\@tempa
\newbox\@tempa
\newbox\@tempa
\newbox\@tempa
\newbox\@tempa
\newbox\@tempa
\let\@tempa\relax
\end{verbatim}

\section{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., \(2 \times \maxcols + 1\)) 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.

\begin{verbatim}
\ifnum\numexpr\count20-\count14-1<40
    \% this is = 2 * 20
    \count14=\@cclv
\fi
\end{verbatim}

\section{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.

\begin{verbatim}
\ifundefined{emergencystretch}
    \{newdimen\emergencystretch\}
\else
\fi
\end{verbatim}

My tests showed that the following formula worked pretty well. Nevertheless the \texttt{\setemergencystretch} macro also gets \texttt{\hsize} as second argument to enable the user to try different formulae.
Even if this should be used as a hook we use a @ in the name since it is more for experts.

This is cheap (deferring the floats until after the current page) but any other solution would go deep into \TeX's output routine and I don’t like to work on it until I know which parts of the output routine have to be reimplemented anyway for \TeXX.

We have to add the float to the \texttt{deferlist} because we assume that outside the \texttt{multicols} environment we are in one column mode. This is not entirely correct, I already used the \texttt{multicols} environment inside of \TeX's \texttt{twocolumn} declaration but it will do for most applications.

5.1 Maintaining the mark registers

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

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

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.

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.

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.

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.

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}.

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.

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
Now we split the box to the maximal possible dimension. This should split off the full contents of the box so that effectively everything is split off. As a result \texttt{\splitfirstmark} and \texttt{\splitbotmark} will contain the first and last mark in the box respectively.

Unfortunately, a simple \texttt{\vsplit} is not enough if the material contains forced breaks in that case we first have to get rid of those which is why we do this work in a separate macro that can call itself recursively.

\begin{verbatim}
\def\mc@get@all@box@marks #1{%
  \setbox\@tempboxa\vsplit#1to\maxdimen
  \ifvoid#1\else
    \setbox#1\vbox{\unvbox\@tempboxa \unvbox#1}
    \mc@get@all@box@marks #1%
  \fi
}
\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}
\def\set@keptmarks{%
  \ifx\kept@firstmark\@empty
    \mult@info\tw@
    \{Set kept first mark:\MessageBreak
    \meaning\kept@firstmark
    \@gobbletwo
  \fi
}
\end{verbatim}

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

\begin{verbatim}
\def\prep@keptmarks{%
  \if@boxedmulticols \else
    \get@keptmarks\partial@page
  \fi}
\end{verbatim}

The \texttt{\prep@keptmarks} function is used to initialize the kept marks from the contents of \texttt{\partial@page}, i.e. the box that holds everything from the top of the current page prior to starting the \texttt{multicols} environment. However, such a box is only available if we are not producing a boxed \texttt{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 \LaTeX{} sequence is
a series of self-canceling glues so if we remove them recursively we are usually fine.

Special care is needed with handling \vspace* as that corresponds to \penalty10000, \vskip <skip>, followed by \vskip 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 multicol code sometimes add an explicit penalty at the end of a column we first attempt to remove it in case it is there.

\skip0=0pt
\edef\the@zero@skip{\the\skip0}
\def\remove@discardable@items{%
  \unpenalty
  Save a previous skip (if there) and then remove it, we can’t really tell the difference between no skip an a skip of zero but that’s life.
  \edef\@tempa{\the\lastskip}%
  %\typeout{s1=\@tempa}%
  \unskip
  If it was a zero skip (or none) we save the next previous skip (if any).
  \ifx\@tempa\the@zero@skip
   \edef\@tempb{\the\lastskip}%
   %\typeout{s2=\@tempb}%
   If this one again was zero (or more likely not there in the first place) we stop.
   \else
   Otherwise we remove this “real” skip. Then we look if it was preceded by a penalty of 10000 (i.e., a \nobreak)
   \unskip
   \ifnum\lastpenalty=\@M
     \typeout{p=\lastpenalty}%
   \else
   Otherwise we recurse.
   \fi
  \else
  \remove@discardable@items
  \fi
  \else
  \if
  \remove@discardable@items
  \fi
%

\newif\iftoo@bad
\def\too@badtrue{\global\let\iftoo@bad\iftrue}
\def\too@badfalse{\global\let\iftoo@bad\iffalse}
\newif\ifforcedbreak@leftover
\newif\iftoo@bad
\newcount\c@minrows
\c@minrows=1
\newcount\c@columnbadness
\c@columnbadness=10000
\newcount\c@finalcolumnbadness
\c@finalcolumnbadness=9999
\newdimen\last@try
\newdimen\multicolovershoot
\newdimen\multicolundershoot
\multicolovershoot=0pt
\multicolundershoot=2pt
\newbox\mult@nat@firstbox

\mult@info#1#2{%
\ifnum\c@tracingmulticols>#1%
\GenericWarning{(multicol)\@spaces\@spaces}%
\fi
\fi}

A helper for producing info messages
\def\mult@info#1#2{%
\ifnum\c@tracingmulticols>#1%
\GenericWarning{(multicol)\@spaces\@spaces}%
\fi
\fi}

6 Fixing the \columnwidth

If we store the current column width in \columnwidth we have to redefine the internal \@footnotetext macro to use \textwidth for the width of the footnotes rather than using the original definition.

Starting with version v1.5r this is now done in a way that the original definition is still used, except that locally \columnwidth is set to \textwidth.

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

7 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.

\newcommand*{\nobalance}{% 
\ifinner \PackageWarning{multicol}{\multicols* inside a box does not make sense.\MessageBreak Going to balance anyway}\fi 
\else \ifinner \PackageWarning{multicol}{\multicols* inside a box does not make sense.\MessageBreak Going to balance anyway}\fi 
\else \PackageWarning{multicol}{\multicols* inside a box does not make sense.\MessageBreak Going to balance anyway}\fi 
\fi 

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.

\def\balance@columns@out{\multicols@out \penalty-\@Mvi \fi \begin{multicols} \end{multicols}}

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.

\newcommand*\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 \end{multicols}}

7.2 Manual column breaking

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

For this reason the current implementation uses a completely different approach. In a nutshell it extends the L\TeX output routine handling by introducing an additional penalty flag (i.e., a penalty which is forcing but higher than -10000 so that the output routine can look at this value and thus knows why it has been called).
Inside the output routine we test for this value and if it appears we do two things: save the galley up to this point in a special box for later use and reduce the \vsize by the height of the material seen. This way the forcing penalty is now hidden in that box and we can restart the collection process for the remaining columns. (This is done in \specs above.)

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

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

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

\def\LR@column@boxes{%
    We loop through the columns with \process@cols
    \mc@col@status@write
    The typeset box followed by the column rule material
    \box\mult@rightbox
}

This is modeled after \newpage but for column breaks.

As you will have noticed, we started with box register \mult@firstbox (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 \if@breakpen.

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

The box followed by the column rule material

As you will have noticed, we started with box register \mult@firstbox (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 \if@breakpen.

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

7.3 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.

We loop through the columns with \process@cols
\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 if, for 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.

1046 \def\RL@column@boxes{%
First step is to put all rules in the right place (without adding the comes which are instead represented by a space of $\hsize$.

1047 \process@cols\mult@firstbox{%
1048 \hskip\hsize
1049 \hss\{\columnseprulecolor\vrule
1050 \@width\columnseprule\}hss
1051 }%
1052 \hskip\hsize

At this point in the code our typesetting reference point is at the right end of the rightmost column (or rather where that column should appear).

We are now typesetting all columns by first backing up by their width (which is $\hsize$) then typesetting the box and then backing up again, but this time further, i.e., also across the column separation. That will then enable us to typeset the next column using the same approach until we are done with all but the final column.

1053 \process@cols\mult@firstbox{%
1054 \ifdim\dp\count@>\dimen\tw@
1055 \global\dimen\tw@\dp\count@ \fi
1056 \hskip-\hsize
1057 \mc@col@status@write
1058 \box\count@
1059 \hskip-\hsize
1060 \hskip-\columnsep
1061 }%

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

1062 \ifdim\dp\mult@rightbox>\dimen\tw@
1063 \global\dimen\tw@\dp\mult@rightbox \fi
1064 \hskip-\hsize
1065 \mc@lastcol@status@write
1066 \box\mult@rightbox
1067 \hskip-\hsize

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:

1068 \hskip\full@width
1069 }

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.

1070 \newcommand\LRmulticolcolumns
1071 \{\let\mc@align@columns
1072 \RL@column@boxes\}
1073 \newcommand\LRmulticolcolumns
1074 \{\let\mc@align@columns
1075 \LR@column@boxes\}

The default is left-to-right:

1076 \LRmulticolcolumns

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.

1077 \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).

1078 \newcommand\docolaction{%

First check is the support is enabled.

1079 \ifx\mc@col@status@write\relax
1080 \PackageError{multicol}{Option ‘colaction’ not selected}%
1081 {\string\docolaction\space

The \docolaction scans for a star and optional argument and 3 mandatory ones, but we do this in chunks (not having xparse available).
requires the use of the 'colaction' option on the package)%
\fi

Then prepare \mc@col@type.
\global\advance\mc@col@check@num\@ne
\edef\mc@col@type{\expandafter\ifx\csname mc@col-\the\mc@col@check@num\endcsname\relax0\else\csname mc@col-\the\mc@col@check@num\endcsname\fi}%

Finally check for a star, record this information and then call \@docolaction to do the rest.
\@ifstar{\@docolactionstartrue \@docolaction}{}
\@ifstarfalse\@docolaction{}

\newcommand\@docolaction[4][1]{%
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{\string\mc@set@col@status\mc@col@status{lcol-1}{0}\mc@col@status{lcol-2}{0}\mc@col@status{lcol-3}{0}\mc@col@status{lcol-4}{0}}}

Where 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
If column is unknown we use the default action or the action denoted by the optional argument (so that arg can take the value 1, 2, 3).
\ifcase #1\or #2\or #3\or #4\fi
\else
Otherwise we know (or think we know) that this is a first, middle, or last column:
\#2 % 1 First col
\#3 % 2 any middle col
\#4 % 3 last col
\else
\ERRORwrongdefaultgiven
\fi
\if@docolactionstar \else \@docolactioncheck \fi
}

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

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

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

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 @tempsva to false which will trigger the “Label(s) may have changed” warning.
\AtEndDocument{\def\mc@set@col@status{1#2}{% \ifnum #2=\mc@curr@col@status\else
Finally, as part of determining in which column we are, we used a switch inside `\mc@col@status@write` to determine if we are in the first column or not.

\newif\ifmc@firstcol
\mc@firstcoltrue
Index

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

Symbols

\@Mvi .......................... 243
\@footnotetext .......................... 972

B
\balance@columns .......................... 594
\balance@columns@out .......................... 557

C
\c@collectmore .......................... 306
\c@columnbadness .......................... 952
\c@finalcolumnbadness .......................... 952
\c@minrows .......................... 950
\c@unbalance .......................... 306
\col@number .......................... 306
\colbreak@box ......................... 1031
\columnbreak .......................... 998
\columnnep .......................... 3
\columnneprule .......................... 3
\columnneprulecolor .......................... 3, 374

D
\docolaction .......................... 1078
\doublecol@number .......................... 306

E \emergency@stretch .......................... 836
\endmulticols .......................... 245
\endmulticols* .......................... 985
\enough@room .......................... 137
\F \flushcolumns .......................... 552
\G \get@keptmarks .......................... 869

I \if@boxedmulticols .......................... 134
\ifshr@nking .......................... 552
\init@mult@footins .......................... 228
\K \kept@botmark .......................... 852
\kept@firstmark .......................... 852
\kept@topmark .......................... 852
\L \leave@mult@footins .......................... 489
\\LR@column@boxes .......................... 1033
\\LRmulticol@boxes .......................... 1070
\\LRmulticol@columns .......................... 1070

M \maxbalancingoverflow .......................... 806
\mc@align@columns .......................... 1070
\mc@boxedresult .......................... 306
\mc@col@check@num .......................... 1077
\mc@col@status .......................... 1118
\mc@firstcol .......................... 1128
\mc@get@all@box@marks .......................... 885
\mc@set@col@status .......................... 1120
\mult@cols .......................... 109
\mult@colsep* .......................... 3
\mult@firstbox .......................... 808
\mult@footnotetext .......................... 94, 972
\mult@gfirstbox .......................... 808
\mult@grightbox .......................... 808
\mult@info .......................... 965
\mult@rightbox .......................... 808
\multi@column@out .......................... 379
\\multicol@leftmargin .......................... 242
\\multicol@baselineskip .......................... 3, 306
\\multicol@mind@string .......................... 373
\\multicol@pretolerance .......................... 3, 306
\multicols .......................... 73
\multicols* .......................... 972
\multicol@sep .......................... 3, 306
\multicol@tolerance .......................... 3, 306
\newcolumn .......................... 1016

P \page@free .......................... 306
\vfil@maxdepth .......................... 377

R \raggedcolumns .......................... 552
\reinsert@footnotes .......................... 375
\remove@discardable@items .......................... 921
\return@nome@mark .......................... 855
\RL@column@boxes .......................... 1046
\RLmulticol@columns .......................... 1070

S \set@floatcmds .......................... 841
\set@keptmarks .......................... 13
\set@mult@vsize .......................... 232
\setemergencystretch .......................... 836
\speci@ls .......................... 493

V \page@sofar .......................... 340
\partial@page .......................... 306
\post@multicols .......................... 3, 306
\pre@multicols .......................... 2, 306
\prepare@multicols .......................... 154
\process@cols .......................... 331
\process@deferreds .......................... 542

Change History

v1.0c
\enough@room: Penalty 0 added to empty the contribution list. .......................... 11
v1.0d
General: All lines shortened to 72 or less. .......................... 1
v1.0e
General: Redefinition of description env. to use \descriptionmargin=5pt in documentation. .......................... 1
\prepare@multicols: \textwidth changed to \linewidth. .......................... 13
Setting of \columnwidth removed. .......................... 13
So this file will work with the ‘twocolumn’ command. .......................... 13
v1.0f
General: Changed \z@ to 0pt in redefinition of description. .......................... 1
v1.1a
General: \multicol@sep changed to \multicol@sep. .......................... 1
\\flushcolumns: \flushedcolumns renamed to \flushcolumns. .......................... 21
v1.2a
\balance@columns: Group around main loop removed. .......................... 24
\prepare@multicols: \pretolerance -1 because it nearly never succeeds. .......................... 13
\set@floatcmds added. .......................... 13
\setemergencystretch added. .......................... 13
<table>
<thead>
<tr>
<th>Version</th>
<th>Change/Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1.3a</td>
<td>balance@columns: Changed \rtop to \vbox. \multicols: Minimum of two columns.</td>
</tr>
<tr>
<td>v1.3b</td>
<td>endmulticols: Do penalty with \addpenalty \enough@room: Do \penalty with \addpenalty \multicols: Minimum of two columns.</td>
</tr>
<tr>
<td>v1.3c</td>
<td>balance@columns: g@global advance left over from older code. Limit column height to @colroom \multicols: \multicols: Minimum of two columns.</td>
</tr>
<tr>
<td>v1.4a</td>
<td>General: Added support for multicol in inner mode.</td>
</tr>
<tr>
<td></td>
<td>balance@columns: Changed to proper \endlinechar in \message. \multicol@box: Forgotten braces added.</td>
</tr>
<tr>
<td></td>
<td>\multicols@out: \botmark set to \splitbotmark. \prepare@multicols: Checking for text losses. \prepare@multicols: Used.</td>
</tr>
<tr>
<td></td>
<td>\multicols@out: \botmark set to \splitbotmark. \prepare@multicols: Checking for text losses. \prepare@multicols: Used.</td>
</tr>
<tr>
<td>v1.4d</td>
<td>balance@columns: New algorithm for start height.</td>
</tr>
<tr>
<td>v1.4e</td>
<td>endmulticols: But ignore \nobreak in \addpenalty \enough@room: But ignore \nobreak in \addpenalty \multicols: Typeset optional arg inside group \prepare@multicols: Using.</td>
</tr>
<tr>
<td>v1.4f</td>
<td>balance@columns: \vsize added to tracing info \multicols@out: \vsize added to tracing info \par added to allow for correct inner test.</td>
</tr>
<tr>
<td>v1.4g</td>
<td>\multicols@out: \global was probably wrong but at least unnecessary \multicols@out: Only change \kept@topmark if \kept@topmark non-empty.</td>
</tr>
<tr>
<td>v1.4h</td>
<td>General: Added mark tracing with tracingmulticols\geq 2. \kept@topmark: Init to double brace pair. \multi@column@out: Set \kept@topmark to \botmark.</td>
</tr>
</tbody>
</table>

v1.4f | \multicols@out: Use different \vsize setting. \prepare@multicols: Code moved to \set@mult@vsize. Use different \vsize setting. \set@mult@vsize: Macro added. |

v1.4m | endmulticols: Check \partial@page being emptied. \return@nonemptymark: Make marks robust. \return@nonemptymark: Make marks robust. |

v1.4n | \prepare@multicols: \topskip locally zeroed. \prepare@multicols: \topskip locally zeroed. |

v1.5a | \prepare@multicols: Use different \vsize setting. \prepare@multicols: Code moved to \set@mult@vsize. Use different \vsize setting. \set@mult@vsize: Macro added. |

v1.5b | \prepare@multicols: Add offset to \doublecolnumber. \prepare@multicols: Add offset to \doublecolnumber. |

v1.5c | \prepare@multicols: New box mechanism. \prepare@multicols: New box mechanism. |

v1.5d | \prepare@multicols: New box mechanism. \prepare@multicols: New box mechanism. |

v1.5e | \prepare@multicols: New box mechanism. \prepare@multicols: New box mechanism. |

v1.5f | \prepare@multicols: New box mechanism. \prepare@multicols: New box mechanism. |

v1.5g | \prepare@multicols: New box mechanism. \prepare@multicols: New box mechanism. |

v1.5h | \prepare@multicols: New box mechanism. \prepare@multicols: New box mechanism. |

v1.5i | \prepare@multicols: New box mechanism. \prepare@multicols: New box mechanism. |

v1.5j | \prepare@multicols: New box mechanism. \prepare@multicols: New box mechanism. |

v1.5k | \prepare@multicols: New box mechanism. \prepare@multicols: New box mechanism. |

v1.5l | \prepare@multicols: New box mechanism. \prepare@multicols: New box mechanism. |

v1.5m | \prepare@multicols: New box mechanism. \prepare@multicols: New box mechanism. |

v1.5n | \prepare@multicols: New box mechanism. \prepare@multicols: New box mechanism. |

v1.5o | \prepare@multicols: New box mechanism. \prepare@multicols: New box mechanism. |

v1.5p | \prepare@multicols: New box mechanism. \prepare@multicols: New box mechanism. |

v1.5q | \prepare@multicols: New box mechanism. \prepare@multicols: New box mechanism. |

v1.5r | \prepare@multicols: New box mechanism. \prepare@multicols: New box mechanism. |

v1.5s | \prepare@multicols: New box mechanism. \prepare@multicols: New box mechanism. |

v1.5t | \prepare@multicols: New box mechanism. \prepare@multicols: New box mechanism. |

v1.5u | \prepare@multicols: New box mechanism. \prepare@multicols: New box mechanism. |

v1.5v | \prepare@multicols: New box mechanism. \prepare@multicols: New box mechanism. |

v1.5w | \prepare@multicols: New box mechanism. \prepare@multicols: New box mechanism. |

v1.5x | \prepare@multicols: New box mechanism. \prepare@multicols: New box mechanism. |

v1.5y | \prepare@multicols: New box mechanism. \prepare@multicols: New box mechanism. |
v1.5d \speci@ls: Support \clearpage 

v1.5e \multicolor\columnout: reinit \topmark 

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

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

v1.5h \balance@columns: Get kept marks first 

v1.5i \page@sofar: Check for void boxes 

v1.5j \speci@ls: Support \columnbreak 

v1.5k \balance@columns: Added debug statements for column break support 

v1.5l \speci@ls: Added debug statements for column break support 

v1.5m \multicolors: Make \footnotetext long to allow multi-paragraph footnotes. 

v1.5n \page@sofar: Detect and fix problem if a multicolors ends at the top of a page 

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

v1.5p \multicolors: Redefinition of \footnotetext only within env \pr/2689. 

v1.5q \balance@columns: Do not reset \multicolor@firstbox (pr/2739) 

v1.5r \footnotetext: Use \footnotetext but with local change to \columnwidth 

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

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

v1.5v \balance@columns: Added tracing statements for trial unsuccessful 

v1.5w \multicolors: Make \footnotetext long to allow multi-paragraph footnotes. 

v1.5x \endmulticolors: Detect and fix problem if a multicolors ends at the top of a page 

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

v1.5z \page@sofar: Different info display 

v1.6a General: New option grid 

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

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

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

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

v1.6f \balance@columns /colbreak guard in the wrong position 

v1.6g \page@sofar: Make the color of the rule a hook 

v1.6h \set@floatcmds: Added \minipagefalse 

General: Applied improvement of documentation, kindly done by Robin Fairbairns. 

\\%footnotetext: Redefinition added pr/2689. 
\prepare@multicolors: Setting of \columnwidth added again pr/2689. 

\multicolors: Redefinition of \footnotetext 

\\%footnotetext: Use \footnotetext but with local change to \columnwidth 
\multicolor@footnotetext: Macro removed again. 
\multicolors: Use \footnotetext but with local change to \columnwidth. 

\\%footnotetext: Use \footnotetext but with local change to \columnwidth. 
\multicolor@footnotetext: Macro removed. 
\multicolors: Use \footnotetext but with local change to \columnwidth. 

\\%footnotetext: Use \footnotetext but with local change to \columnwidth. 
\multicolor@footnotetext: Macro removed. 
\multicolors: Use \footnotetext but with local change to \columnwidth. 

\\%footnotetext: Use \footnotetext but with local change to \columnwidth. 
\multicolor@footnotetext: Macro removed. 
\multicolors: Use \footnotetext but with local change to \columnwidth. 

\\%footnotetext: Use \footnotetext but with local change to \columnwidth. 
\multicolor@footnotetext: Macro removed. 
\multicolors: Use \footnotetext but with local change to \columnwidth.
v1.6h
\set@floatcmds: Use \@endfloatbox to better support the modifications done by the float package .......................... 28

v1.7a
General: RL language support added .......................... 32

v1.7b
General: RL language support fixed .......................... 32
\page@sofar: RL language support fixed ...................... 17

v1.8a
\balance@columns: Balancing concept improved .......................... 27
\balance@columns@out: Balancing concept improved .......................... 22
Support for \enlargethispage ........................................ 22
\maxbalancingoverflow: \maxbalancingoverflow parameter added .......................... 27
\multi@column@out: Only re-add output penalty if it was explicitly set ...................... 19
Support for \enlargethispage ........................................ 18

v1.8b
\balance@columns: Remove discardable items at the end of split boxes .......................... 24
\multi@column@out: And 20odd years later conclude that this was wrong and unboxing is always needed .......................... 19
Remove discardable items at the end of split boxes .......................... 19

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

v1.8d
\multi@column@out: Reset \@mparbottom after page finishes .......................... 20

v1.8e
General: Support \doc@action .................................. 9, 33
\LR@column@boxes: Support \doc@action ...................... 32
\RL@column@boxes: Support \doc@action ...................... 33

v1.8f
\endmulticols: Discard spaces before adding \color@endgroup .......................... 14

v1.8g
\page@sofar: Now adjusting \prevdepth .......................... 17
Resetting \prevdepth in the right place .......................... 17
Warn if value is exceeded not when equal .......................... 17

v1.8h
\balance@columns: All column boxes should obey \maxdepth (pr/4395) .......................... 23
Do not report overflow .......................... 24
Use \vfilmaxdepth ........................................ 25, 26
\endmulticols: Set \prevdepth for current vlist when returning from multicols environment .......................... 16
\endmulticols*: Use \vfilmaxdepth .......................... 31
\multi@column@out: Use \vfilmaxdepth .......................... 19
\vfilmaxdepth: Macro added (pr/4395) .......................... 18

v1.8i
\endmulticols*: Add \null to hide the final fill and only add vertical space if not doing \raggedcolumns .......................... 31

v1.8j
\balance@columns: Use \vfil in this case .......................... 25
\endmulticols*: Redesign the whole approach. .......................... 31
\multi@column@out: Set \boxmaxdepth .......................... 18
\vfilmaxdepth: Use only `0.0001fil' for stretching .......................... 18

v1.8k
General: The new switch .......................... 30
\balance@columns: .......................... 26
\remove@discardable@items removed .......................... 23
Do not use \remove@discardable@items here .......................... 24
Finish the new conditional .......................... 27
Init \ifforcedbreak@leftover .......................... 24
Watch out for columns growing too far in case of forced breaks .......................... 26
\balance@columns@out: Add \remove@discardable@items at the end of the last column when balancing .......................... 22
No additional penalty here .......................... 22
Use \@Mv and not \break in case this forced break is not used on this page .......................... 22
\endmulticols*: And a bit more redesign because of the change in \remove@discardable@items .......................... 31
\multi@column@out: \remove@discardable@items removed .......................... 19
\speci@ls: Remove discardable items just before a forced break .......................... 21

v1.8l
\balance@columns: Added additional tracing if column overflows .......................... 26

v1.8m
\remove@discardable@items: Another rewrite of \remove@discardable@items hopefully okay now .......................... 30

v1.8n
\multi@column@out: Reset \@textfloat@height after page finishes .......................... 20

v1.8o
\@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

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

v1.8q
\prepare@multicols: Make \clearpage behave like \newpage (pr/4511) .......................... 13

v1.8r
\@Mvi: Macro added .......................... 14
\balance@columns@out: Re-add the final penalty .......................... 22
<table>
<thead>
<tr>
<th>Version</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1.8s</td>
<td>Use special penalty to signal end of environment</td>
</tr>
<tr>
<td>v1.8t</td>
<td>Handling end of env through special penalty</td>
</tr>
<tr>
<td>v1.8u</td>
<td>Support for \docolaction (issue/39)</td>
</tr>
<tr>
<td>v1.8v</td>
<td>Re-add end penalty for \multicols* environment to guard against leftovers (git/53)</td>
</tr>
<tr>
<td>v1.8w</td>
<td>\docolaction: Support star with \docolaction</td>
</tr>
<tr>
<td>v1.8x</td>
<td>\multicolumnout: Removed dead code, the case where this can go wrong is too obscure to worry about it (gh/101)</td>
</tr>
<tr>
<td>v1.8y</td>
<td>Provide minrows counter for balancing (gh/190)</td>
</tr>
<tr>
<td>v1.8z</td>
<td>Use \maxdepth not \maxdepth (gh/190)</td>
</tr>
<tr>
<td>v1.9a</td>
<td>\multicols: Support for \docolaction</td>
</tr>
<tr>
<td>v1.9b</td>
<td>General: Swap names \mult@gfirstbox and \mult@firstbox</td>
</tr>
<tr>
<td>v1.9c</td>
<td>\mult@gfirstbox: Allow for 20 columns (gh/237)</td>
</tr>
<tr>
<td>v1.9d</td>
<td>\multicolmindepthstring: Make column min depth customizable (gh/1002)</td>
</tr>
<tr>
<td>v1.9e</td>
<td>\get@keptmarks: Get rid of forced breaks for mark extraction (gh/1130)</td>
</tr>
<tr>
<td>v1.9f</td>
<td>\mc@get@all@box@marks: Get rid of forced breaks for mark extraction (gh/1130)</td>
</tr>
</tbody>
</table>