An environment for multicolumn output†
Frank Mittelbach

Abstract
This article describes the use and the implementation of the multicol 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. \TeX's float mechanism, however, is partly disabled in the current implementation and will be added in a later version. At the moment only floats contributed outside the scope of the environment will find their way into the actual output.

1 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 \multicoladjdemerits to be used for \adjdemerits, etc.) which would influence the line breaking algorithm.

But actually resetting such parameters to zero or even worse to a negative value won't give better line breaks inside the 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 possible, then, as a last resort, \TeX will produce overfull boxes. All those (and only those) possible break points will be considered and finally the sequence which results in the fewest demerits will be chosen. This means that a value of \-1000 for \adjdemerits instructs \TeX to prefer visibly incompatible lines instead of producing better line breaks.

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

Version 1.1a had a nice "feature": the penalty for using the forbidden floats was their ultimate removal from \TeX's \@freelist so that after a few \marginpars inside the multicol environment floats where disabled forever. (Thanks to Chris Rowley for pointing this out.) I removed this misbehaviour and at the same time decided to allow at least floats spanning all columns, e.g., generated by the \figure* environment. You can see the new functionality in table ?? which was inserted at this very point. However single column floats are still forbidden and I don't think I will have time to tackle this problem in the near future. As an advice for all who want to try: wait for \TeX 3.0. It has a few features which will make life much easier in multicolumn surroundings. Nevertheless we are working here at the edge of \TeX's capabilities, really perfect solutions would need a different approach than it was done in \TeX's page builder.

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

2 Introduction

Switching between two column and one column layout is possible in \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 col-

†Editor's note: This paper, with slight modification, is the basis for Mr. Mittelbach's citation as the Donald E. Knuth Scholar at the 1989 TUG Meeting.

†This file has version number v1.3c, last revised 91/04/08, documentation dated 91/03/14.
\setemergencystretch: This is a hook for people who like to play around. It is supposed to set the \emergencystretch register provided in the new TeX 3.0. The first argument is the number of columns and the second one is the current \hspace. At the moment the default definition is 4pt \times \#1, i.e. the \hspace isn't used at all. But maybe there are better formulae.

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

Table 1: The new commands of multicol.sty version 1.2. Both commands might be removed if good solutions to these open problems are found. I hope that these commands will prevent that nearly identical style files derived from this one are floating around.

umn. When I started to write macros for doc.sty (see “The doc-Option”, TeXboat 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.} 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.

3 The User Interface

To use the environment one simply says

\begin{multicols}{\#1} \multicolumn{\text{text}}{\end{multicols}

where \#1 is the required number of columns and \multicolumn may contain arbitrary \LaTeX{} commands, except that floats and marginpars are not allowed in the current implementation\footnote{This is dictated by lack of time. To implement floats one has to reimplement the whole \LaTeX{} output routine.}.

As its first action, the multicol environment measures the current page to determine whether there is enough room for some portion of multicol text output. This is controlled by the \dimen variable \premulticol which can be changed by the user with ordinary \LaTeX{} commands. If the space is less than \premulticol, a new page is started. Otherwise, a \vskip of \multicolsep is added.\footnote{Actually the added space may be less because we use \advspace (see the \LaTeX{} manual for further information about this command).}

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

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

\begin{multicols}{3}
\section{The User Interface} ...

If such text is unusually long (or short) the value of \premulticol might need adjusting to prevent a bad page break. We therefore provide a third argument which can be used to overwrite the default value of \premulticol just for this occasion.

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

Since narrow columns tend to need adjustments in interline spacing we also provide a (\skip) parameter called \multicolskip which is added to the \baselineskip parameter inside the multicols environment. Please use this parameter with care or leave it unset.

\section{The User Interface} ...

3 I started with the algorithm given in the TeXbook on page 417. Without this help a weekend would not have been enough.

2 This is dictated by lack of time. To implement floats one has to reimplement the whole \LaTeX{} output routine.

3 Actually the added space may be less because we use \advspace (see the \LaTeX{} manual for further information about this command).
alone; it is intended only for style file designers since even small changes might produce totally unexpected changes to your document.

3.1 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 default \texttt{\sloppy} which is too small for narrow columns. On the other hand the \texttt{\sloppy} declaration (which sets \texttt{\tolerance} to 10000 = \infty) is too large, allowing really bad spacing.\footnote{Look at the next paragraph, it was set with the \texttt{\sloppy} declaration.}

We therefore use a \texttt{\multicol\tolerance} parameter for the \texttt{\tolerance} value inside the 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 multicol environment. Since \texttt{\tolerance} is set to \texttt{\multicol\tolerance} at the beginning of every multicol environment one can locally overwrite this default by assigning \texttt{\tolerance} = \texttt{\desired\value}. \texttt{\multicol\tolerance} generation of multicolumn output can be divided into two parts. In the first part we are collecting material for a page, shipping it out, collecting material for the next page, and so on. As a second step, balancing will be done when the end of the multicols environment is reached. In the first step \TeX{} might consider more material whilst finding the final columns than it actually use when shipping out the page. This might cause a problem if a footnote is encountered in the part of the input considered, but not used, on the current page. In this case the footnote might show up on the current page, while the footnotemark corresponding to this footnote might be set on the next one.\footnote{The reason behind this behavior is the asynchronous character of the \TeX{} \texttt{page\_builder}. However, this could be avoided by defining very complicated routines which don't use \TeX{} primitives like \texttt{\insert} but do everything by hand. This is clearly beyond the scope of a weekend problem.}

Therefore the multicols environment gives a warning message\footnote{This message will be generated even if there are no footnotes in this part of the text.} whenever it is unable to use all the material considered so far.

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

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

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

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

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

3.2 Tracing the output

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

\begin{verbatim}
<number> = 1. \TeX{} will now tell you, whenever it enters or leaves a multicols environment, the number of columns it is working on and its decision about starting a new page before or after the environment.
<number> = 2. In this case you also get information from the balancing routine: the heights tried for the left and right-most columns, information about shrinking if the \texttt{\raggedcolumns} declaration is in force and the value of the ‘unbalance’ counter if positive.
\end{verbatim}
\langle \text{number} \rangle \geq 3$. Setting \langle \text{number} \rangle to such a high value will additionally place an \texttt{\hrule} into your output, separating the part of text which had already been considered on the previous page from the rest. Clearly this setting should not be used for the final output.

## 4 The Implementation

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

### 4.1 Starting and Ending the \texttt{multicol} Environment

As always we begin by identifying the latest version of this file on the VDU and in the transcript file but we abort if this file was already read in.

\begin{verbatim}
1 \ifnum\mult@cols>0\relax
2 \typeout{Style option: \texttt{\multicol}}
3 \fileversion{space <\filedate> (PMI)}
4 \typeout{English documentation}
5 \spaces\spaces\spaces<\doctype> (PMI)}
\end{verbatim}

As mentioned before, the \texttt{multicol} environment has one mandatory argument (the number of columns) and up to two optional ones. We start by reading the number of columns into the \texttt{\col@number} register.

\texttt{\def\mult@cols#1{\col@number#1}}

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.

\texttt{\def\mult@cols#1{\col@number#1\relax}
1 \ifnum\col@number<2\relax
2 \warning{Using \texttt{"\textbackslash col@number"} doesn’t seem a good idea.}--}>)
3 I therefore use two columns instead\%}
4 \col@number=2 \fi}

Now we can safely look for the optional arguments.

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

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

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

This argument should be a \langle \text{dimen} \rangle denoting the minimum free space needed on the current page to start the environment. If the user didn’t supply one, we use \texttt{\premult@cols} as a default.

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

After removing all arguments from the input we are able to start with \texttt{\mult@cols}. First we look to see if statistics are requested:

\texttt{\def\mult@cols#1{\ifnum\mult@cols>0\relax
1 \warning{Style option: \texttt{\multicol}}
2 \fileversion{space <\filedate> (PMI)}
3 \typeout{English documentation}
4 \spaces\spaces\spaces<\doctype> (PMI)}

Then we measure the current page to see whether a useful portion of the \texttt{multicol} environment can be typeset. This routine might start a new page.

\texttt{\ifnum\mult@cols>0\relax}

Now we output the first argument and produce vertical space above the columns. (Note that this argument corresponds to the first optional argument of the \texttt{multicol} environment.)

\texttt{\begin{verbatim}
1 \par\addvspace\multicolssep
2 \begin{group}
3 \prepare@multicols\ignorespaces}
\end{verbatim}

The \texttt{\enough@room} macro used above isn’t perfect but works reasonably well in this context. We measure the free space on the current page by subtracting \texttt{\pagetotal} from \texttt{\pagegoal}. This isn’t entirely correct since it doesn’t take the ‘shrinking’ (i.e., \texttt{\pageshrink}) into account. The ‘recent contribution list’ might be nonempty so we start with \texttt{\par} and an explicit \texttt{\penalty}. Actually, we use \texttt{\addpenalty} to ensure that a following \texttt{\addvspace} will ‘see’ the vertical space that might be present.

\texttt{\begin{verbatim}
1 \enough@room#1{\par \addpenalty\texttt{\par}}
2 \pagefree \pagegoal
3 \advance \pagefree - \pagetotal}
\end{verbatim}

Now we test whether tracing information is required:

\texttt{\begin{verbatim}
1 \ifnum \c@tracingmulticols>0\relax
2 \typeout{Current page: \texttt{\par}}
3 \message{\spaces\spaces goal height=\%}
4 \the\pagegoal: \texttt{used \the\pagetotal}
5 \space \rightarrow \free=\the\pagefree\%}
\end{verbatim}

\footnote{See the documentation of \texttt{\endmulticols} for further details.}
33 \typeout{\@spaces needed \the\@fil}
34 (for \string\@fil)\fi

Our last action is to force a page break if there isn't enough room left.
35 \ifdim\page\@free<\@fil

When preparing for multicolumn output several things must be done. First we remove everything from the 'current page' and save it in the box \partial@page.
36 \def\prepare@multicols{%

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.
37 \noindent\skip000\relax
38 \output{(\global\setbox\partial@page
39 \vbox{\movebox\@cmv
40 \setbox\@x@\lastbox
41 }\eject)

Then we assign new values to \badness, \badness and \tolerance since it's rather hard for \TeX{} to produce 'good' paragraphs within narrow columns.
42 \badness10001 \badness5000
43 \tolerance\multicol@tolerance

Since nearly always the first pass will fail we ignore it completely telling \TeX{} to hyphenate directly.
44 \pretolerance0

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

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

We also set the register \doublecol@number for later use. This register should contain \( \col@number \).
47 \doublecol@number\col@number
48 \multiply\doublecol@number\times\@fil

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

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

Since we have to set \col@number columns on one page, each with a height of \col@room, we have to assign \vs{size} = \col@number \times \col@room in order to collect enough material before entering the \output routine again.
51 \vs{size}\col@number\col@room

But this might not be enough since we use \vs{split} later to extract the columns from the gathered material. Therefore we add some 'extra lines,' the number depending on the value of the 'collectmore' counter.
52 \advance\vs{size}\collectmore\baselineskip

The \hsize of the columns is given by the formula:
\[
\frac{\text{columnwidth} - (\col@number - 1) \times \text{columnsep}}{\col@number}
\]

This will be achieved with:
53 \hsize\col@number\advance\hsize\vs{size}\columnsep
54 \advance\hsize\col@number\columnsep
55 \divide\hsize\col@number

We also set \linewidth to \hsize but leave \columnwidth unchanged. This is inconsistent, but \columnwidth is used only by floats (which aren't allowed in their current implementation) and by the \footnote macro. Since we want pagewise footnotes\footnote{I'm not sure that I really want pagewise footnotes. But balancing of the last page can only be achieved with this approach or with a multi-path algorithm which is complicated and slow. But it's a challenge to everybody to prove me wrong! Another possibility is to reimplement a small part of the \texttt{fine_up} procedure in \TeX{} (the program). I think that this is the best solution if you are interested in complex page makeup, but it has the disadvantage that the resulting program cannot be called \TeX{} thereafter.} this simple trick saves us from rewriting the \footnote macros.
56 \linewidth\hsize
Now we switch to a new \output routine which will be used to put the gathered column material together.

\output{\multicolumnout}%

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 pagewise footnotes). If, on the other hand, footnotes are typeset at the very end of the document, our scheme still works since \count\footins is zero then, so it will not change.

\multiply\count\footins\colnumber
\multiply\skip\footins\colnumber

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

\reinsert@footnotes

When the end of the \multicols environment is sensed we have to balance the material. We end the current paragraph with \par but this isn’t sufficient since \TeX’s \pagebuilder will not totally empty the contribution list.\footnote{This once caused a puzzling bug where some of the material was balanced twice, resulting in some overprints. The reason was the \eject which was placed at the end of the contribution list. Then the \pagebuilder was called (an explicit \penalty will empty the contribution list), but the line with the \eject didn’t fit onto the current page. It was then reconsidered after the output routine had ended, causing a second break after one line.} Therefore we must also add an explicit \penalty. Now the contribution list will be emptied and, if its material doesn’t all fit onto the current page then the output routine will be called before we change it.

\edef\endmulticols{\par\addpenalty\z@}

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

\output{\balance@columns}@eject

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.

\checkend{\multicols}%
\endgroup \reinsert@footnotes

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

\global\c@unbalance\z@%}
\typeout{""Ending multicolumn\n output.""":""J:\"fi}  

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

\newcount\c@unbalance \c@unbalance = 0
\newcount\c@collectmore \c@collectmore = 0
\newcount\c@tracingmulticols \c@tracingmulticols = 0
\newcount\colnumber \newcount\doublecolnumber
\newcount\multicoltolerance \multicoltolerance = 9999
\newdimen\page@free
\newdimen\premulticols \premulticols = 50pt
\newdimen\postmulticols \postmulticols = 20pt
\newskip\multicolsep
\multicolsep = 12pt plus 4pt minus 3pt
\newskip\multicolskip
\multicolskip = 0pt

We also need a box into which the “current page” can be put.

\newbox\partial@page
4.2 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 \textsc{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{\count0=1} (\texttt{\count0} is a scratch register defined in \textsc{Plain TeX}), processes argument \#2, adds two to \texttt{\count0}, processes argument \#2 again, etc. until \texttt{\count0} is higher than \texttt{\doublecolnumber}. It might be easier to understand it through an example, so we first define it and explain its usage afterwards.

```latex
\def\process@cols#2\relax{
  \loop #2
  \advance\count0\count@\relax
  \ifnum\count0<\doublecolnumber
    \repeat
}
```

We now define \texttt{page@sofar} to give an example of the \texttt{process@cols} macro. \texttt{page@sofar} should output everything on the 'current page'. So we start by unboxing \texttt{\partial@page} (i.e. the part above the multicol environment). If the \texttt{\partial@page} is void (i.e. if the multicol environment started on a new page or if we typeset several pages within the multicol environment) this will produce nothing.

```latex
\def\page@sofar\unbox\partial@page
```

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 right-most) should be saved in box register 0.\footnote{You will see the reason for this numbering when we look at the output routines \texttt{\multicolumn} and \texttt{\balance@column}.} First we ensure that the columns have equal width. We use \texttt{process@cols} for this purpose, starting with \texttt{\count0 = 0}. Therefore \texttt{\count0} loops through 0, 2, ... (to \texttt{\doublecolnumber}).

```latex
\def\process@cols\relax#2\times\wd\count@\hspace{#2}\relax
{\loop
  \process@cols\relax#2\relax
  \advance\count0\count@\relax
  \ifnum\count0<\doublecolnumber
    \repeat
}
```

Now we put all columns together in an \texttt{\hbox} of width \texttt{\textwidth}, separating them with a rule if desired.

```latex
\hbox to\textwidth
\hspace{#2}
```

As you will have noticed, we started with box register 2 (i.e. the left column). So this time \texttt{\count0} looped through 2, 4, ... Finally we add box 0 and close the \texttt{\hbox}.

```latex
97 \box\z\}
```

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. \texttt{\reinsert@footnotes} will do what its name indicates: it reinserts the footnotes present in \texttt{footinbox} so that they will be reprocessed by \textsc{TeX}'s \texttt{page builder}.

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

```latex
98 \def\reinsert@footnotes{\ifvoid\footins\else
  \insert\footins\fi}
```

Now we can't postpone the difficulties any longer. The \texttt{\multicolumn} 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 is sensed. In the latter case the \texttt{\outputpenalty} is less than \texttt{-1000}, otherwise the penalty which triggered the output routine is higher. Therefore it's easy to distinguish both cases: we simply test this register.

```latex
\def\multicolumn\relax{
  \ifnum\outputpenalty<0\omit
```

If this was a float or a marginpar we call \texttt{\spec@ls}

```latex
\omit\spec@ls\else
```

otherwise we contract the final page. Actually a \texttt{\clearpage} will be silently accepted, producing the same effects as a \texttt{\newpage}, since we didn't distinguish between a penalty of \texttt{-10000} and \texttt{-10001} (produced by a \texttt{\clearpage}). Let us now consider the normal case. We have to \texttt{\colorbox} the columns from the accumulated material in box 255. Therefore we first assign appropriate values to \texttt{\splittopskip} and \texttt{\splitmaxdepth}.

```latex
103 \splittopskip\opkip
104 \splitmaxdepth\maxdepth
```

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

```latex
105 \dimen0\colroom
```

\footnote{You will see the reason for this numbering when we look at the output routines \texttt{\multicolumn} and \texttt{\balance@column}.}
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.

\divide\skip\footins\colnumber
\ifvoid\footins \else
\advance\dimen0\textwidth
\advance\dimen0=ht\footins \fi

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

\process\col\tw \setbox\count0
\vsplit\cclv to\dimen0

If \raggedcolumns is in force we add a \vfill at the bottom by unboxing the splitted box.

\ifshr\king
\setbox\count0\vbox to\dimen0
{\unvbox\count0\vfill} \fi
\%  

Then the last column follows.

\setbox\z@\vsplit\cclv to\dimen0
\ifshr\king
\setbox\z@\vbox to\dimen0
{\unvbox\z@\vfill} \fi

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

\ifvoid\cclv \else
\unvbox\cclv \fi

\penalty\outputpenalty

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

\ifvoid\footins\else
\warning{I moved some lines to the next page.} \fi
\spaces\footnotes on page \the\page\space might be wrong\fi
\%  

If the 'tracingmulticol' counter is 3 or higher we also add a rule.

\ifnum \c@tracingmulticol\tw0
\hrule \allowbreak \fi
\fi

With a little more effort we could have done better. If we had, for example, recorded the shrinkage of the material in \partialpage it would be now possible to try higher values for \dimen0 (i.e. the column height) to overcome the problem with the nonempty box 255. But this would make the code even more complex so I skipped it in the current implementation.

Now we use \LaTeX's standard output mechanism.\footnote{This will produce a lot of overhead since both output routines are held in memory. The correct solution would be to redesign the whole output routine used in \LaTeX.}

Admittedly this is a funny way to do it.

\makecol\outputpage

The macro \makecol adds all floats assigned for the current page to this page. \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 \multicol environment.

\makecol\outputpage

Now we reset \colroom to \colht which is \LaTeX's saved value of \textheight.

\global\colroom=\colht

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

\process\deferreds

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

\ifnum\c@tracingmulticol=0
\typout{Colroom: the\colht\space  
after float space removed  
= \the\colroom} \fi

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

\global\vsize=\colnumber\colroom
\global\advance\vsize
\c@collectmore=\baselineskip

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

\fi

We left out two macros: \process\deferreds and \specious. If we encounter a float or a marginpar in the current implementation we simply warn the user that this is not allowed. Then we reinsert the page and its footnotes.

\def\specious\%
\typout{Floats and marginpars not
allowed inside `multicol' environment)
\unbox\@cvl\reinsert\@footnotes

Additionally we empty the \@currlist to avoid later error messages when the \LaTeX output routine is again in force. But first we have to place the boxes back onto the \@freeplist. (\@elts default is \relax so this is possible with \def.)
\def\@freeplist{\@freeplist\@currlist}
\def\@currlist{}

\process@deferreds is a simplified version of \LaTeX's \@startpage. We first call the macro \@floatplacement to save the current user parameters in internal registers. Then we start a new group and save the \@deferlist temporarily in the macro \@tempb.
\def\process@deferreds{%
  \@floatplacement
  \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.

  \def\@deferlist{%
    \let\@elt\@scolelt

  Now we call \@tempb which has the form
  \@elt{\texttt{box register}}\@elt{ }box register...

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

  \begin{group}
  The \raggedcolumns and \flushcolumns declarations are defined with the help of a new \if...
  \newif\ifshrink

  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\@ids{\allowbreak\verb}@esphack.

  \def\raggedcolumns{%
    \@bsphack\shrinktrue@esphack

  \def\flushcolumns{%
    \@bsphack\shrinkfalse@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. Therefore we start by assigning the values used by \@split.

  \def\@balance{columns}%
  \@splittopskip\@opkip

  \def\@maxdepth{maxdepth}
  \def\@maxdepth{maxdepth}

  Next we measure the length of the current page and at the same time save it in box register 0.

  \setbox0\box{\unbox\@cvl\dimen@ht\@x}

  Then 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.

  \advance\dimen@colnumber\@opkip

  \advance\dimen@colnumber\baselineskip

  \divide\dimen@colnumber
  
  At the user's request we start with a higher value (or lower, but this usually only increases the number of tries).

  \advance\dimen@c@unbalance\baselineskip

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

  \ifmmode\set@tracingmulticols@end

  \typeout{Balance columns:

  \ifmmode\set@unbalance=\@z\else

  (off balance=\scol@unbalance)\fi%}

  \fi

  Now we try to find the final column height. We start by setting \@badness 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.

  \@badness@\@in

  In order not to clutter up \LaTeX'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 1. Note the extra braces around this macro call. They are needed since \LaTeX's \loop+\@ids{}\allowbreak\verb+\repeate mechanism cannot be nested on the same level of grouping.

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

  The contents of box 0 are now copied globally to box 1. (This will be the right-most column, as we shall see later.)

  \global\setbox\one@copy\@z
Using \vspli t we extract the other columns from box register 1. This leaves box register 0 untouched so that we can start over again if this trial was unsuccessful.

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

- box 0 ← all material
- box 3 ← first column
- box 5 ← second column
  ...
- box 1 ← last column

We report the height of the first column.

\ifnum\c@tracingmulticol=1
\message{\spaces First column = \the\ht\thr@@}\fi

If \raggedcolumns is in force we also shrink the first column to its natural height and optionally inform the user.

\ifhmode
\message{after shrinking \the\ht\thr@@}\fi

Then we give information about the last column.

\ifnum\c@tracingmulticol=1
\message{< last column = \the\ht\one}\fi

We check whether our trial was successful. The test used is very simple: we merely compare the first and the last column. Thus the intermediate columns may be longer than the first if \raggedcolumns is used. If the right-most column is longer than the first then we start over with a larger value for \dimen@.

\ifdim\ht\one >\ht\thr@@
\advance\dimen@ by\fi

Now we save the actual height of box register 3 (i.e. the left column) in the \dimen register \dimen@ since otherwise this information will be lost when processing the code below.\footnote{The value of \dimen@ may differ from the height of box register 3 when we use the \raggedcolumns declaration.}

\dimen@ =\ht\thr@@

If the determined height for the columns turns out to be larger than the available space (which is \@colroom) we squeeze the columns into the space assuming that they will have enough shrinkability to allow this.\footnote{This might be wrong, since the shrinkability that accounts for the amount of material might be present only in some columns. But it is better to try then to give up directly.}

\ifdim\dimen@ <\@colroom \dimen@ =\@colroom \fi

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

\ifnum\c@tracingmulticol=1
\message{\spaces First column = \the\ht\thr@@}\fi

\ifhmode
\message{after shrinking \the\ht\thr@@}\fi

This will bring us into the position to apply \page@sofar. But first we have to set \vsiz e to a value suitable for one column output.

\global\vsize =\@colroom
\global\advance\vsize by\ht\partial@page
\page@sofar

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

5 New macros and hacks for version 1.2

If we don't use \TeX 3.0 \emergencystretc h is undefined so in this case we simply add it as an unused \dimen register.

\ifdefined\emergencystretc h
{\newdimen\emergencystretc h}\else
\emergencystretc h 1pt

My tests showed that the following formula worked pretty well. Nevertheless the \setemergencystretc h macro also gets \vsize as second argument to enable the user to try different formulae.

\def\setemergencystretc h#1#2{\vsize\setemergencystretc h\dimen@\advance\dimen@ by#1\divide\dimen@ by#2}
\def\setemergencystretc h 4pt
\def\setemergencystretc h 1pt

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

\let\dblfloat\@dblfloat
This is cheap (deferring the floats until after the current page) but any other solution would go deep into \LaTeX's output routine and I don't like to work on it until I know which parts of the output routine have to be reimplemented anyway for 2.10 and 3.0.

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 \LaTeX's \texttt{twocolumn} declaration but it will do for most applications.

Frank Mittelbach
Eichenweg 29
D-6500 Mainz 1
Federal Republic of Germany
Bitnet: \texttt{pf55dez@druede}