An environment for multicolumn output*†

Frank Mittelbach

Abstract

This article describes the use and the implementation of the multicols environment. This environment allows switching between one and multicolumn format on the same page. Footnotes are handled correctly (for the most part), but will be placed at the bottom of the page and not under each column. LaTeX's float mechanism, however, is partly disabled in 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 Introduction

Switching between two column and one column layout is possible in LaTeX, but every use of `\twocolumn` or `\onecolumn` starts a new page. Moreover, the last page of two column output isn't balanced and this often results in an empty, or nearly empty, right column. When I started to write macros for `doc.sty` (see "The doc–Option", *TUGboat* volume 10 #2, pp. 245–273) I thought that it would be nice to place the index on the same page as the bibliography. And balancing the last page would not only look better, it also would save space; provided of course that it is also possible to start the next article on the same page. Rewriting the index environment was comparatively easy, but the next goal, designing an environment which takes care of footnotes, floats etc., was a harder task. It took me a whole weekend[1] to get together the few lines of code below and there is still a good chance that I missed something after all. Try it and, hopefully, enjoy it; and *please* direct bug reports and suggestions back to Mainz.

2 The User Interface

To use the environment one simply says

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

where ⟨*number*⟩ is the required number of columns and ⟨*multicolumn text*⟩ may contain arbitrary LaTeX commands, except that floats and marginpars are not allowed in the current implementation[2].

As its first action, the multicols environment measures the current page to determine whether there is enough room for some portion of multicolumn output. This is controlled by the ⟨*dimen*⟩ variable `\premulticols` which can be changed by the user with ordinary LaTeX commands. If the space is less than `\premulticols`, a new page is started. Otherwise, a `\vskip` of `\multicolsep` is added.[3]

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 `\postmulticols` available. Again we add `\multicolsep` or start a new page.

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

```
\begin{multicols}{3}
```

* 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 v1.1a, last revised 89/09/20, documentation dated 89/09/20.

[1] 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 `\addvspace` (see the LaTeX manual for further information about this command).

```
[\section{The User
          Interface}] ...
```

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

Separation of columns with vertical rules is achieved by setting the parameter `\columnseprule` 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 `\multicolbaselineskip` which is added to the `\baselineskip` parameter inside the multicols environment. Please use this parameter with care or leave it alone; it is intended only for style 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 `\tolerance`: this controlls the allowed 'looseness' (i.e. the amount of blank space between words). Its default value is 200 (the LATEX `\fussy`) which is too small for narrow columns. On the other hand the `\sloppy` declaration (which sets `\tolerance` to $10000 = \infty$) is too large, allowing really bad spacing.[4]

We therefore use a `\multicolstolerance` parameter for the `\tolerance` value inside the multicols environment. Its default value is 9999 which is less than infinity but 'bad' enough for most paragraphs in a multicolumn environment. Changing its value should be done outside the multicols environment. Since `\tolerance` is set to `\multicolstolerance` at the beginning of every multicol environment one can locally overwrite this default by assigning `\tolerance␣=␣`⟨*desired value*⟩.

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.[5] Therefore the multicols environment gives a warning message[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 `\pagebreak` command in the final document. Another way to influence the behavior of TEX in this respect is given by the counter variable 'collectmore'. If you use the `\setcounter` declaration to set this counter to ⟨*number*⟩, TEX will consider ⟨*number*⟩ more (or less) lines before making its final decision. So a value of -1 may solve all your problems at the cost of slightly less optimal columns.

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

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

[4] Look at the next paragraph, it was set with the `\sloppy` declaration.

[5] The reason behind this behavior is the asynchronous character of the TEX *page_builder*. However, this could be avoided by defining very complicated output routines which don't use TEX primitives like `\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.

2.2 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 ⟨*number*⟩ you then will get some tracing information on the terminal and in the transcript file:

⟨*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 `\raggedcolumns` declaration is in force and the value of the 'unbalance' counter if positive.

⟨*number*⟩ ≥ 3. Setting ⟨*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.

3 The Implementation

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

3.1 Starting and Ending the multicols 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.

```
\@ifundefined{mult@cols}{}{\endinput}
\typeout{Style option: `multicol'
   \fileversion\space <\filedate> (FMi)}
\typeout{English documentation
  \@spaces\@spaces\space<\docdate> (FMi)}
```

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.

```
  \ifnum\col@number<\@ne
     \@warning{Using `\number\col@number'
       columns doesn't seem a good idea.^^J
       I therefore use two columns instead}%
     \col@number\tw@ \fi
```

Now we can safely look for the optional arguments.

```
  \@ifnextchar[\mult@cols{\mult@cols[]}}
```

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

```
\def\mult@cols[#1]{\@ifnextchar[%
```

This argument should be a ⟨*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.

```
   {\mult@@cols{#1}}%
   {\mult@@cols{#1}[\premulticols]}}
```

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

```
\def\mult@@cols#1[#2]{%
  \ifnum\c@tracingmulticols>\z@
      \typeout{^^J^^JStarting multicolumn
               output with \the\col@number
               \space columns:^^J}\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.)

```
   #1\par\addvspace\multicolsep
```

We start a new grouping level to hide all subsequent changes (done in `\prepare@multicols` for example) and finish by suppressing initial spaces.

```
   \begingroup
   \prepare@multicols\ignorespaces}
```

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.[7]

```
\def\enough@room#1{\par \penalty\z@
   \page@free \pagegoal
   \advance \page@free -\pagetotal
```

Now we test whether tracing information is required:

```
\ifnum \c@tracingmulticols>\z@
  \typeout{Current page:}%
  \message{\@spaces goal height=%
    \the\pagegoal: used \the\pagetotal
    \space -> free=\the\page@free}%
  \typeout{\@spaces needed \the#1
           (for \string#1)}\fi
```

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

```
\ifdim \page@free <#1\newpage \fi}
```

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

```
\def\prepare@multicols{%
  \output{\global\setbox\partial@page
               \vbox{\unvbox\@cclv}}\eject
```

Then we assign new values to \vbadness, \hbadness and \tolerance since it's rather hard for TeX to produce 'good' paragraphs within narrow columns.

```
\vbadness9999 \hbadness5000
\tolerance\multicoltolerance
```

We also set the register \doublecol@number for later use. This register should contain 2 × \col@number.

```
\doublecol@number\col@number
\multiply\doublecol@number\tw@
```

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

```
\advance\baselineskip\multicolbaselineskip
```

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

```
\advance\@colroom-\ht\partial@page
```

Since we have to set \col@umber columns on one page, each with a height of \@colroom, we have to assign \vsize = \col@number × \@colroom in order to collect enough material before entering the \output routine again.

```
\vsize\col@number\@colroom
```

But this might not be enough since we use \vsplit later to extract the columns from the gathered material. Therefore we add some 'extra lines,' the number depending on the value of the 'collectmore' counter.

```
\advance\vsize\c@collectmore\baselineskip
```

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

$$\frac{\texttt{\textbackslash columnwidth} - (\texttt{\textbackslash col@number} - 1) \times \texttt{\textbackslash columnsep}}{\texttt{\textbackslash col@number}}$$

This will be achieved with:

```
\hsize\columnwidth \advance\hsize\columnsep
\advance\hsize-\col@number\columnsep
\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 pagewide footnotes[8] this simple trick saves us from rewriting the \footnote macros.

```
\linewidth\hsize
```

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

```
\output{\multi@columnout}%
```

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

```
\multiply\count\footins\col@number
\multiply\skip \footins\col@number
```

[7] See the documentation of \endmulticols for further details.

[8] I'm not sure that I really want pagewide 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.

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

```
   \reinsert@footnotes}
```

When the end of the multicols environment is sensed we have to balance the gathered material. We end the current paragraph with `\par` but this isn't sufficient since TeXs *page_builder* will not totally empty the contribution list.[9] 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.

```
\def\endmulticols{\par\penalty\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`.

```
   \endgroup \reinsert@footnotes
```

We also set the 'unbalance' counter to its default. This is done globally since LaTeX counters are always changed this way.[10]

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

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.

```
   \enough@room\postmulticols
   \addvspace\multicolsep
```

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

```
   \ifnum\c@tracingmulticols>\z@
      \typeout{^^JEnding multicolumn
               output.^^J^^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\col@number
\newcount\doublecol@number
\newcount\multicoltolerance
                    \multicoltolerance = 9999
\newdimen\page@free
\newdimen\premulticols  \premulticols = 50pt
\newdimen\postmulticols \postmulticols= 20pt
\newskip\multicolsep
      \multicolsep = 12pt plus 4pt minus 3pt
\newskip\multicolbaselineskip
                    \multicolbaselineskip=0pt
```

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

```
\newbox\partial@page
```

3.2 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 PLAIN TeX policy to avoid an overflow of the save stack.

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

```
\def\process@cols#1#2{\count@#1\relax
     \loop #2%
     \advance\count@\tw@
     \ifnum\count@<\doublecol@number
   \repeat}
```

[9] This once caused a puzzling bug where some of the material was balanced twice, resulting in some overprints. The reason was the `\eject` which was placed at the end of the contribution list. Then the *page_builder* was called (an explicit `\penalty` will empty the contribution list), but the line with the `\eject` didn't fit onto the current page. It was then reconsidered after the output routine had ended, causing a second break after one line.

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

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

```
\def\page@sofar{\unvbox\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.[11] First we ensure that the columns have equal width. We use `\process@cols` for this purpose, starting with `\count@` = 0. Therefore `\count@` loops through 0, 2,... (to `\doublecol@number`).

```
    \process@cols\z@{\wd\count@\hsize}%
```

Now we put all columns together in an `\hbox` of width `\textwidth`

```
    \hbox to\textwidth{%
```

separating them with a rule if desired.

```
        \process@cols\tw@{\box\count@
          \hss\vrule\@width\columnseprule\hss}%
```

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

```
        \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. `\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*.

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

Now we can't postpone the difficulties any longer. The `\multicolumnout` 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 `\outputpenalty` is less than −10001, otherwise the penalty which triggered the output routine is higher. Therefore it's easy to distinguish both cases: we simply test this register.

```
\def\multi@columnout{%
   \ifnum\outputpenalty <-\@Mi
```

If this was a float or a marginpar we call `\speci@ls`

```
   \speci@ls \else
```

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

```
   \splittopskip\topskip
   \splitmaxdepth\maxdepth
```

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

```
   \dimen@\@colroom
```

But we must also substract 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\col@number
   \ifvoid\footins \else
      \advance\dimen@-\skip\footins
      \advance\dimen@-\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@cols\tw@{\setbox\count@
         \vsplit\@cclv to\dimen@}%
```

Then the last column follows.

```
   \setbox\z@\vsplit\@cclv to\dimen@
```

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

```
   \ifvoid\@cclv \else
       \unvbox\@cclv
       \penalty\outputpenalty
```

If the 'tracingmulticols' counter is 3 or higher we also add a rule. And in any case we inform the user about our bad luck.

```
       \ifnum \c@tracingmulticols>\tw@
                 \hrule\allowbreak \fi
       \@warning{I moved some lines to
                  the next page.^^J
           \@spaces Footnotes on page
         \thepage\space might be wrong}\fi
```

[11] You will see the reason for this numbering when we look at the output routines `\multi@columnout` and `\balance@column`.

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

Now we use LaTeX's standard output mechanism.[12] Admittedly this is a funny way to do it.

```
\setbox\@cclv\vbox{\page@sofar}%
```

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 multicols 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@tracingmulticols>\@ne
  \typeout{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 `\partial@page` is now empty and `\@colroom` might be reduced by the space reserved for floats.

```
\global\vsize\col@number\@colroom
\global\advance\vsize
   \c@collectmore\baselineskip
```

We also have to readjust the `\footins` skip register.

```
\multiply\skip\footins\col@number\fi}
```

We left out two macros: `\process@deferreds` and `\speci@ls`. 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\speci@ls{%
   \typeout{Floats and marginpars not
            allowed inside `multicols'
            environment!}%
   \unvbox\@cclv\reinsert@footnotes
```

Additionally we empty the `\@currlist` to avoid later error messages when the LaTeX output routine is again in force.

```
\gdef\@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.

```
    \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}
```

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. Therefore we start by assigning the values used by `\vsplit`.

```
\def\balance@columns{%
   \splittopskip\topskip
   \splitmaxdepth\maxdepth
```

[12] 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.

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

```
\setbox\z@\vbox{\unvbox\@cclv}\dimen@\ht\z@
```

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@\col@number\topskip
\advance\dimen@-\col@number\baselineskip
\divide\dimen@\col@number
```

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.

```
\ifnum\c@tracingmulticols>\@ne
   \typeout{Balance columns:
     \ifnum\c@unbalance=\z@\else
     (off balance=\number\c@unbalance)\fi}%
\fi
```

Now we try to find the final column height. Everything is done in a group so as to hide the changes to register contents. We start by setting `\vbadness` to infinity (i.e. 10000) to suppress underfull box reports while we are trying to find an acceptable solution.

```
{\vbadness\@M \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 unsucessful trial. We use `\process@cols` for this purpose, starting with 1. 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\@ne{\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\@ne\copy\z@
```

Using `\vsplit` 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.

```
{\process@cols\thr@@{\global\setbox\count@
                          \vsplit\@ne to\dimen@}}%
```

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

box 0	$\longleftarrow$	all material
box 3	$\longleftarrow$	first column
box 5	$\longleftarrow$	second column
$\vdots$		$\vdots$
box 1	$\longleftarrow$	last column

We report the height of the first column.

```
\ifnum\c@tracingmulticols>\@ne
   \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.

```
\ifshr@nking \global\setbox\thr@@
              \vbox{\unvbox\thr@@}%
   \ifnum\c@tracingmulticols>\@ne
      \message{ after shrinking
                \the\ht\thr@@}\fi\fi
```

Then we give information about the last column.

```
\ifnum\c@tracingmulticols>\@ne
   \message{<> last column = \the\ht\@ne}%
   \typeout{}\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\@ne >\ht\thr@@
\global\advance\dimen@\p@
\repeat}%
```

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.[13]

```
\dimen@\ht\thr@@
```

Then we move the contents of the odd-numbered box registers to the even-numbered ones, shrinking them if requested.

```
\process@cols\z@{\@tempcnta\count@
     \advance\@tempcnta\@ne
     \setbox\count@\vtop to\dimen@
        {\unvbox\@tempcnta
         \ifshr@nking\vfill\fi}}%
```

[13] The value of `\dimen@` may differ from the height of box register 3 when we use the `\raggedcolumns` declaration.

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

```
    \global\vsize\@colroom
    \global\advance\vsize\ht\partial@page
    \page@sofar}
```

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

Index

Italic numbers denote the pages where the corresponding entry is described, underlined numbers point to the definition, all others indicate the places where it is used.

◇ Frank Mittelbach
Electronic Data Systems
(Deutschland) GmbH
Eisenstraße 56
D-6090 Rüsselsheim
Federal Republic of Germany
Bitnet: pzf5hz@drueds2