Glisterings: Assemblies; Table talk

Peter Wilson

> It did a ghastly contrast bear
> To those bright ringlets glistering fair.
>
> *Marmion*, Sir Walter Scott

The aim of this column is to provide odd hints or small pieces of code that might help in solving a problem or two while hopefully not making things worse through any errors of mine.

> Eric, or, Little by Little
>
> *Book title*, Frederick W. Farrar

1 Assemblies

1.1 Adding to a macro

On occasions it is useful to be able to extend a pre-existing macro. For instance, to assemble a list of the names of the members of some organization, or the reviewers of some article, and then print them. In simple cases the LaTeX kernel
`\g@addto@macro{`⟨*macro*⟩`}{`⟨*addition*⟩`}`
can be used for this.

```
\makeatletter
\newcommand*{\member}[1]{%
  \@ifundefined{@members}{%
% a new list of members
% define it as the argument (member name)
    \newcommand{\@members}{#1}}{%
% a list exists, add the argument to it
    \g@addto@macro{\@members}{, #1}}}
\newcommand*{\showmembers}{%
  \ignorespaces\@members}
\newcommand*{\thememberss}{\showmembers
  \let\@members\relax}
\makeatother
```

The macro `\member{`⟨*name*⟩`}` can be used several times to add ⟨*name*⟩ to the `\@members` macro. The macro `\thememberss` can then be called to print the contents of `\@members` and clear `\@members` so a new list may be started. If you want to print the list more than once then use `\showmembers` which prints, but does not clear, the list.

```
\member{Fred} \member{Joe}
\member{Susan} \member{Faye}
```

`\thememberss` ⇒ Fred, Joe, Susan, Faye

For more complex additions, for instance when the macro to be extended takes arguments, then the patchcmd package [4] could be the answer.

Once having created a list of members it might have to be changed because one or more members have left. This is more complicated and I present it only as an example of what could be done.

The `\deletemember{`⟨*name*⟩`}` will go over the list of members, creating a new temporary working list with the exception of the ⟨*name*⟩ member, then replace the original list with the working one.

```
\makeatletter
\let\xpf\expandafter%  just a shorthand
\newcommand*{\deletemember}[1]{%
  \let\@tempmembers\relax
  \def\@dm@num{1}%
  \@for\member@:=\@members\do{%
    \ifnum\@dm@num<2\relax
      \def\t@mp@b{#1}%          initial entry
      \ifx\member@\t@mp@b%
        \def\@dm@num{0}%
      \else
        \def\t@mp@b{\space #1}% later entries
        \ifx\member@\t@mp@b%
          \def\@dm@num{0}%
        \fi
      \fi
    \fi
    \ifnum\@dm@num=0\relax
      \def\@dm@num{2}%
    \else
      \xpf\xpf\xpf\transfer\xpf{\member@}%
    \fi}%
  \let\@members\@tempmembers}
\makeatother
```

The coding of `\deletemember` is not straightforward. The LaTeX kernel's `\@for` construct is used to loop over the comma-separated entries in `\@members`, putting, in turn, each entry into the `\member@` macro. Due to the way that `\@members` is constructed, the name of the initial entry is recovered as `name`, while a later entry is recovered as ␣`name`; hence the two tests for the argument ⟨*name*⟩ against the recovered `\member@` name.

The `\@dm@num` macro is used to track the state of the process. At the start it is set to 1. If it is less than 2, attempts are made to match the argument with the current list name and if a match is found then `\@dm@num` is set to 0. After the argument check, if the argument is matched (`\@dm@num` = 0) then `\@dm@num` is reset to 2, otherwise the current member name is added to the working list. This all means that once the list name matches the argument then no further attempts at matching are needed or done, and the remaining original members are simply added to the working list. At the end the original list is set to the temporary working list.

The tricky part is that the current contents of `\member@`, not the macro itself, should be added

to the working list.[1] The bunch of \expandafters around the call to \transfer expands \member@ to its definition before it gets handed over as the argument to \transfer.

The macro \transfer{⟨*name*⟩} adds ⟨*name*⟩ to the macro \@tempmembers containing a list of comma separated names. It has the same general form as the earlier \member macro.

```
\makeatletter
\newcommand*{\transfer}[1]{%
  \@ifundefined{@tempmembers}{%
    \newcommand*{\@tempmembers}{#1}%
  }{%
    \g@addto@macro{\@tempmembers}{,#1}%
  }}
\makeatother
```

Here are some examples of adding and deleting members to and from the original member list above.

```
\member{Alice} \member{Bob} \member{Claire}
\member{David} \member{Erica}
```

\showmembers ⇒ Fred, Joe, Susan, Faye, Alice, Bob, Claire, David, Erica
\deletemember{David}\showmembers ⇒ Fred, Joe, Susan, Faye, Alice, Bob, Claire, Erica
\deletemember{Fred}\showmembers ⇒ Joe, Susan, Faye, Alice, Bob, Claire, Erica

```
\member{Xerxes} \member{Zeno}
```

\showmembers ⇒ Joe, Susan, Faye, Alice, Bob, Claire, Erica, Xerxes, Zeno
\deletemember{Miriam}\showmembers ⇒ Joe, Susan, Faye, Alice, Bob, Claire, Erica, Xerxes, Zeno

1.2 Piecing a paragraph

Ron Aaron wanted a different kind of assembly. He wrote [1]:
What I wish to do is accumulate text into a paragraph 'as I go'. My simple approach is to allocate a box, and then unbox and add the text. But this doesn't work as I intend:

```
\newbox\textbox
\def\addbox#1{%
  \setbox\textbox\vbox{
    \unvbox\textbox#1}}
\addbox{Hello}
\addbox{there!}
\box\textbox
```

What I get is each appended bit of text in a separate line. I've tried to '\unskip*' and '*\unkern*' etc. after the* \unvbox *but whatever I do I get a list of lines . . .*

Trying out Ron's example the result is:

Hello
there!

The squashed vertical spacing between the lines is real, not an artifact of this article.

In responding, Philip Taylor [5], having said that using a \vbox would be difficult, then gave two suggestions; either use an \hbox directly or a token-list register. His \hbox solution (and my example) is:

```
\newbox\textbox
\def\addbox #1{%
  \setbox \textbox = \hbox
  \bgroup
    \unhbox \textbox #1%
  \egroup}
\addbox{Hello}
\addbox{ World!}
\addbox{ Now isn't that a rather
         common saying?}
\unhbox\textbox
```

with the result:

Hello World! Now isn't that a rather common saying?

At various points after this I have used code like
`\addbox{ (n) text}`
as an example of assembling a paragraph piece by piece and at the end showing the result via:
`\unhbox\textbox`

```
\addbox{(1) Start of a paragraph.}
```

Philip's second solution uses a token register:

```
\newtoks\texttoks
\def\addtoks #1{%
  \texttoks =
  \expandafter {\the \texttoks #1}}
\addtoks {Goodbye}
\addtoks { \emph{vain} world. Ah, the
           weariness in that statement
           does one no good.}
\the \texttoks
```

and the example produces:

Goodbye *vain* world. Ah, the weariness in that statement does one no good.

```
\addbox{ (2) After an interruption
        add more.}
```

Note that with both of Philip's solutions you have to explicitly incorporate spaces where you want

[1] If the macro is added then the list will consist of nothing but a series of \member@, thus all expanding to the identical name (the current definition of \member@ when the list is printed).

them to occur in the assembled paragraph. It seemed, though, that Ron really wanted to use a `\vbox` but I have neither seen nor been able to come up with satisfactory code.

```
\addbox{ (3) This is the end
        of the piecewise
        paragraph.}
\unhbox\textbox
```

Now print the piecewise paragraph giving:

(1) Start of a paragraph. (2) After an interruption add more. (3) This is the end of the piecewise paragraph.

> Beneath those rugged elms, that yew-tree's shade,
> Where heaves the turf in many a mouldering heap,
> Each in his narrow cell for ever laid,
> The rude forefathers of the hamlet sleep.
>
> —*Elegy Written in a Country Churchyard*, Thomas Gray

2 Table talk

Arbo [2] wanted a tabular layout like the one shown in Figure 1 and tried using code like this to produce it.

```
\begin{tabular}{r|c|l}
\hline
First & Second & Third \\
Text &
  \multicolumn{1}{c|c|c|c|}%
                {C 1 & C 2 & C 3 & C 4}
  & More text \\
Words &
  \multicolumn{1}{c|c|c}%
                {C 5 & C 6 & C 7}
  & Text \\
Title &
  \multicolumn{1}{c|c}%
             {C 8 & C 9}
  & Some text \\
\hline
\end{tabular}
```

If you try it you will find, like Arbo, that it doesn't work, resulting in a string of error messages beginning with:

First	Second				Third
Text	C 1	C 2	C 3	C 4	More text
Words	C 5	C 6		C 7	text
Title	C 8			C 9	Some text

Figure 1: Desired tabular layout

```
! Missing } inserted.
<inserted text>
                }
l.6945  {C 1 & C 2 & C 3 & C 4}
```

The problem is that `\multicolumn` merges multiple columns into one whereas the requirement here was to split one column into several.

Donald Arseneau [3] responded that *'They don't look aligned at all, so don't call them columns'*, and provided code for an `\addcell` macro. Arbo modified it very slightly to center the `\vlines`, with the final version as follows:

```
\newcommand{\addcell}{\unskip\hfill
  \hspace\tabcolsep\vline\hspace\tabcolsep
  \hfill % added by Arbo
  \ignorespaces}
```

Using this, the tabular in Figure 1 is created by:

```
\begin{tabular}{|r|c|l|}\hline
First & Second & Third \\
Text & C 1
       \addcell C 2 \addcell C 3 \addcell C 4
     & More text \\
Words & C 5
        \addcell C 6 \addcell C 7
      & text \\
Title & C 8 \addcell  C 9 & Some text \\
\hline
\end{tabular}
```

References

[1] Ron Aaron. How to append text to a paragraph (in an existing vbox)? Post to `xetex` mailing list, 16 July 2010.

[2] Arbo. How to produce multiple columns within a multicolumn. Post to `comp.text.tex` newsgroup, 2 November 2010.

[3] Donald Arseneau. Re: How to produce multiple columns within a multicolumn. Post to `comp.text.tex` newsgroup, 2 November 2010.

[4] Michael J. Downes. The patchcmd package, 2000. `http://ctan.org/pkg/patchcmd`.

[5] Philip Taylor. Re: How to append text to a paragraph (in an existing vbox)? Post to `xetex` mailing list, 16 July 2010.

◇ Peter Wilson
12 Sovereign Close
Kenilworth, CV8 1SQ
UK
herries dot press (at)
earthlink dot net