LaTeX table columns with fixed widths

Frank Mittelbach

While attending a workshop on ConTEXt at the TUG 2017 conference in Bachotek, I was introduced to some of the table generating macros of ConTEXt.

The functionality they offer[1] is not very different from what is offered through LaTeX environments, sometimes with similar, sometimes with somewhat different syntax.

But there was one noticeable exception: In LaTeX, simple columns that are either centered or aligned left or right always automatically calculate their width based on the content of the cells. If a fixed column width is needed, then one needs a rather complicated set of constructions (using `p` columns and setting up the alignment manually) resulting in rather unreadable source documents. In contrast, ConTEXt offers a simple preamble specification, such as `lw(3cm)` for something like "make this column 3cm with the material left aligned".

This seems like a natural task, so I was a little surprised that I have never seen LaTeX users complain about the missing functionality or more precisely about the roundabout way that is necessary to achieve this[2] in a LaTeX `tabular` environment, e.g., `>{\raggedright\arraybackslash}p{3cm}`).

On the other hand, given `\newcolumntype` provided by the array package, it should be fairly easy to provide a reasonable interface for such task. So my thought was to provide a column type `w` that takes two arguments: the alignment (such as `c`, `l` or `r`) and a dimension specifying the columns width. With that we can then specify a left-aligned column of 3 centimeters as `w{l}{3cm}` or even shorter as `wl{3cm}`. And since new column types can be defined from existing ones, we could shorten this further through a declaration such as

```
\newcolumntype{C}[1]{wc{#1}}
\newcolumntype{L}[1]{wl{#1}}
\newcolumntype{R}[1]{wr{#1}}
```

so that the necessary preamble specification then simply becomes `L{3cm}`.

Simple LaTeX code

So how can we generate columns with a fixed width? A simple approach is to put each cell into a `\makebox` as this command allows making boxes of a specific width (first optional argument) and allows specifying the alignment within the box (second optional argument).

The array package supports the preamble specifiers `>{...}` and `<{...}` through which we can put material before and after the cell content. However, they do not allow us to use the cell content as an argument to a command we place into `>{...}`, so instead we need to use a slightly more elaborate way using the `lrbox` environment.

So first we load the array package (if that hasn't happened already) and then we allocate a box register to temporarily hold the cell content.

```
\usepackage{array} \newsavebox\cellbox
```

Then we define the `w` column type as follows:

```
\newcolumntype{w}[2]{%
```

Before the cell content we start an `lrbox` environment to collect the cell material into the previously allocated box `\cellbox`.

```
  >{\begin{lrbox}\cellbox}%
```

Then comes a specifier for the cell content. We use `l`, but that doesn't matter as in the end we will always put a box of a specific width (`#2`) into the cells of that column, so `c` or `r` would give the same result.

```
  l%
```

At the end of the cell we end the `lrbox` environment so that all of the cell content is now in box `\cellbox`. As a final step we put that box into a `\makebox` using the optional arguments of that command to achieve the correct width and the desired alignment within that width.

```
  <{\end{lrbox}%
    \makebox[#2][#1]{\usebox\cellbox}}}
```

The code above only uses high-level LaTeX constructs. It could be made slightly more efficient by using internal functions.

Going more low-level

One of the issues with the code from the previous section is that it will always set its text at natural width even when that overflows the available space (as the box register is copied). Furthermore, using `\makebox` with optional arguments means that an overflow will be silently accepted, as that command uses `\hss` internally for alignment. Thus an overprint needs to be detected visually.

Both can be appropriate depending on the circumstances, but often enough some squeezing and a warning if something overflows may be the more appropriate action. Therefore a second version (tabular column type `W`) to address these limitations can be helpful as well.

[1] Perhaps more accurately the functionality presented during the workshop — this was most certainly only a fraction of that is possible.

[2] And this is not exactly equivalent either, if the material overflows the available space.

A possible implementation for this could be:

```
\newcolumntype{W}[2]
  {>{\begin{lrbox}\cellbox}%
   l%
   <{\end{lrbox}%
     \let\hss\hfil
     \makebox[#2][#1]{\unhbox\cellbox}}}
```

This is a bit sneaky, as it temporarily disables `\hss`, but given that the cell content is supposed to be fairly plain LR material this should be sufficient in essentially all cases.

Example usage

After providing this declaration we can now easily code tables with all or some column widths fixed, e.g.,

```
\begin{tabular}{|l|wr{12mm}|Wr{12mm}|r|}
  flexible & fixed (w)
              & fixed (W) & flexible  \\
  123      & 123   & 123  & 123       \\
  123456789 & 123456789
              & 123456789 & 123456789 \\
a b c d e & a b c d e
              & a b c d e & a b c d e
\end{tabular}
```

This gives the following result:

flexible	fixed (w)	fixed (W)	flexible
123	123	123	123
123456789	123456789	123456789	123456789
a b c d e	a b c d e	a b c d e	a b c d e

Observe the overfull box marks in the third row as the material is wider than `12mm`) while in the second column it overflows silently in rows 1, 3 and 4 (the latter being squeezed enough to fit in column 3). Also notice that `W` always overflows to the right while `w` overflows away from the alignment (i.e., to the left if the alignment is `r`).

Outlook

In my opinion it makes sense to predefine the `w` and `W` column types in the `array` package, as the short specifications such as `wc{1cm}` should be very useful to many people.

On the other hand `C`, `L` or `R` are likely to be used already for other purposes (e.g., indicating math columns), so it seems better to only mention those as one way to make use of the `w` or `W` column type if people desire to do so.

◇ Frank Mittelbach
Mainz, Germany
frank.mittelbach (at)
latex-project (dot) org
https://www.latex-project.org