The delarray package*

David Carlisle
carlisle@cs.man.ac.uk

1994/03/14

1 Examples

The addition to `array.sty` added in `delarray.sty` is a system of implicit `\left` `\right` pairs. If you want an array surrounded by parentheses, you can enter:
`\begin{array}({cc})` ...

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

Similarly if an environment equivalent to PLAIN TeX's `\cases` could be defined by:
`\begin{array}\{{lL}.` ...

$$f(x) = \left\{ \begin{array}{ll} 0 & \text{if } x = 0 \\ \sin(x)/x & \text{otherwise} \end{array} \right.$$

Here `L` is supposed to denote a column of left aligned L-R text. It may be defined via: `\newcolumntype{L}{>{$}l<{$}}`, as discussed in `array.sty`. Note that as the delimiters must always be used in pairs, the '`.`' must be used to denote a 'null delimiter'.

This feature is especially useful if the `[t]` or `[b]` arguments are also used. In these cases the result is not equivalent to surrounding the environment by `\left`...`\right`, as can be seen from the following example:

$$\begin{array}[t]({c}) 1\\2\\3 \end{array} \begin{array}[c]({c}) 1\\2\\3 \end{array} \begin{array}[b]({c}) 1\\2\\3 \end{array} \quad\mbox{not}\quad \left(\begin{array}[t]{c} 1\\2\\3 \end{array}\right) \left(\begin{array}[c]{c} 1\\2\\3 \end{array}\right) \left(\begin{array}[b]{c} 1\\2\\3 \end{array}\right)$$

```
\begin{array}[t]({c}) 1\\2\\3 \end{array}
\begin{array}[c]({c}) 1\\2\\3 \end{array}
\begin{array}[b]({c}) 1\\2\\3 \end{array}
\quad\mbox{not}\quad
\left(\begin{array}[t]{c} 1\\2\\3 \end{array}\right)
\left(\begin{array}[c]{c} 1\\2\\3 \end{array}\right)
\left(\begin{array}[b]{c} 1\\2\\3 \end{array}\right)
```

*This file has version number v1.01, last revised 1994/03/14.

2 The Macros

```
⟨*package⟩
\RequirePackage{array}[1994/02/03]
```

\@tabarray This macro tests for an optional bracket and then calls up `\@@array` or `\@@array[c]` (as default).

```
\def\@tabarray{\@ifnextchar[{\@@array}{\@@array[c]}}
```

\@@array This macro tests for an optional delimiter before the left brace of the main preamble argument. If there is no delimiter, `\@arrayleft` and `\@arrayright` are made a no-ops, and `\@array` is called with the positional argument. Otherwise call `\@del@array`.

```
\def\@@array[#1]{\@ifnextchar\bgroup
  {\let\@arrayleft\relax\let\@arrayright\relax\@array[#1]}%
  {\@del@array[#1]}}
```

\@del@array We now know that we have an `array` (or `tabular`) with delimiters.

```
\def\@del@array[#1]#2#3#4{%
```

The following line is completely redundant but it does catch errors involving delimiters before the processing of the alignment begins. A common error is likely to be omiting the '.' in a `\cases`-type construction. This causes the first token of the alignment to be gobbled, possibly causing lots of spurious errors before the cause of the error, the missing delimiter, is discovered as `\@arrayright` puts the alignment and the delimiters together.

```
  \setbox\z@\hbox{$\left#2\right#4$}%
```

In the case of a 'c' argument we do not need to rebox the alignment, so we can define `\@arrayleft` and `\@arrayright` just to insert the delimiters.

```
  \if#1c\def\@arrayleft{\left#2}\def\@arrayright{\right#4}%
```

Otherwise we (should) have a `[t] or [b]` argument, so first we store the alignment, without delimiters in box0.

```
  \else\def\@arrayleft{\setbox\z@}%
```

Then after the alignment is finished:

```
  \def\@arrayright{%
```

Calculate the amount the box needs to be lowered (this will be negative in the case of `[b]`). A little bit of arithmetic cf. the TEXBook, Appendix G, rule 8. We calculate the amount this way, rather than just taking the difference between the depth of box0 and the depth of the box defined below, as the depth of that box may be affected by the delimiters if `\delimitershortfall` or `\delimiterfactor` have non-standard values.

```
      \dimen@=\dp\z@
      \advance\dimen@-\ht\z@
      \divide \dimen@ by \tw@
      \advance\dimen@ by\fontdimen22 \textfont\tw@
```

Now lower the alignment and the delimiters into place.

```
      \lower\dimen@\hbox{$\left#2\vcenter{\unvbox\z@}\right#4$}}%
```

End the `\if#1c`

```
  \fi
```

Now that we have defined `\@arrayleft` and `\@arrayright`, call `\@array`.

```
  \@array[#1]{#3}}
```

⟨/package⟩

2.1 newarray.sty

All the features of the old `newarray` style option have been merged into the `array` or `delarray` options.