The tabulary package*

David Carlisle

2024/06/04

1 User Documentation

`\begin{tabulary}{`⟨*length*⟩`}{`⟨*pream*⟩`}` ... `\end{tabulary}`

The rather daft name may change in a later release but it is a pun on `tabularx`, which itself was a pun on `tabular*`...

These environments work pretty much like the standard tabular environment (or more correctly, the enhanced version from the array package) except that there are more possibilities for the column types.

LCRJ These new 'uppercase' column types are only activated in the `tabulary` environment. In order to make the total table width equal to ⟨*length*⟩ the `LCRJ` columns are converted to `p` columns (with `\raggedright`, `\centering`, or `\raggedleft` or normal justification respectively applied). The width of these converted columns is proportional to the natural width of the longest entry in each column.

To stop very narrow columns being too 'squeezed' by this process any columns that are narrower than `\tymin` are set to their natural width. This length may be set with `\setlength` and is arbitrarily initialised to 10 pt. (If you know that a column will be narrow, it may be preferable to use, say, `c` rather than `C` so that the tabulary mechanism is never invoked on that column.)

Similarly one very large entry can force its column to be too wide. So to prevent this, all columns with natural length greater than `\tymax` are set to the same width (with the proportion being taken as if the natuaral length was *equal* to `\tymax`). This is initially set to twice the text width..

Narrow `p` columns are sometimes quite hard to set, and so you may redefine the command `\tyformat` to be any declarations to make just after the `\centering` or `\ragged`... declaration. By default it redefines `\everypar` to insert a zero space at the start of every paragraph, so the first word may be hyphenated. (See DogBook).

As the environment makes a standard LaTeX box, it will be indented by the paragraph indent at the start of a paragraph, and so will not fit on a line if given argument `\textwidth` unless it is preceded by `\noindent` or is in a `center` environment or some other environment with zero paragraph indent.

*This file has version number v0.11, last revised 2024/06/04.

2 Features

You can use `\multicolumn` but if the multicolumn text turns out to be longer than the final calculated widths of the columns that it spans, then the final table will be too wide.

`\verb` doesnt work. (except in restricted version as in tabularx)

The whole table is evaluated twice, so take care with some TEX constructions that may have side effects like writing to files.

3 Options

The following package option is defined:

debugshow Causes a lot of stuff to appear on the terminal. I find this invaluable, you may find it less so.

4 Examples

With C columns

1	the rain in spain falls mainly on the plain	(an @ expr.)	the rain in spain falls mainly on the plain the rain in spain falls mainly on the plain
a	b	(an @ expr.)	c
a	b b	(an @ expr.)	c c
a			

With J columns

1	the rain in spain falls mainly on the plain	(an @ expr.)	the rain in spain falls mainly on the plain the rain in spain falls mainly on the plain
a	b	(an @ expr.)	c
a	b b	(an @ expr.)	c c
a			

With L, R and C columns, and a `\multicolumn`

1	the rain in spain falls mainly on the plain	the rain in spain falls mainly on the plain the rain in spain falls mainly on the plain	and now for something completely different
x	some multicolumn text across columns 2–4		
a	b	c	d
a	b b	c c	d d
a			

The following examples attempt to show the effect of the `\tymin` and `\tymax` parameters. One should also perhaps note that `\tymax` refers to the total column width (inluding any inter-column space, rules etc) but `\tymin` just refers to the width of the column entry (like the argument to the standard `p` column).

`\tymin=0pt`
`\tymax=\maxdimen`
Note how the first column is 'squeezed'. In fact it is in such a narrow column that even 'a' produces an overfull box warning!

a	b	c c c c c	d d d d d d d d d d d d d d d d d d d d
	b	c c c c c	d d d d d d d d d d d d d d d d d d d d
	b	c c c c c	d d d d d d d d d d d d d d d d d d d d
	b	c c c c c	d d d d d d d d d d d d d d d d d d d d
		c c c c c	d d d d d d d d d d d d d d d d d d d d
		c c c c c	d d d d d d d d d d d d d d d d d d d d
		c c	d d d d d d d d

`\tymin=20pt`
`\tymax=\maxdimen`
Here increase `\tymin` so that columns b and a are not so narrow. 'a' is set to its natural width, and 'b' is set to `\tymin`.

a	b b b	c c c c	d d d d d d d d d d d d d d d d d d d d
	b	c c c c	d d d d d d d d d d d d d d d d d d d d
		c c c c	d d d d d d d d d d d d d d d d d d d d
		c c c c	d d d d d d d d d d d d d d d d d d d d
		c c c c	d d d d d d d d d d d d d d d d d d d d
		c c c c	d d d d d d d d d d d d d d d d d d d d
		c c c c	d d d d d d d d d d d d d d d d d d d d
		c c c c	d d d d d d d d d d d d d d d d d d d d
		c c c c	d d d d d d d d d

`\tymin=20pt`
`\tymax=200pt`
In the previous example, the large d column dominated the table, being a lot wider than the c column. By reducing `\tymax` can limit the width of column d producing more even column widths, but now producing an entry for d that is longer than that for c.

a	b b b	c c c c c c c c c c c	d d d d d d d d d d d
	b	c c c c c c c c c c c	d d d d d d d d d d d
		c c c c c c c c c c	d d d d d d d d d d d
			d d d d d d d d d d
			d d d d d d d d d d
			d d d d d d d d d d
			d d d d d d d d d d
			d d d d d d d d d d
			d d d d d d d d d d
			d d d d d d d d d d
			d d d d d d d

5 The Code

```
1 ⟨*package⟩
```

Rollback.

```
2 \DeclareRelease{v0.10}{1995/10/08}{tabulary-v010.sty}
3 \DeclareCurrentRelease{}{2024-06-01}
```

This version needs a current array package

```
4 \RequirePackage{array}[2024/05/23]
5 \catcode`\Z=14
6 \DeclareOption{debugshow}{\catcode`\Z=9\relax}
7 \ProcessOptions
```

`\arraybackslash` Borrowed from tabularx.

```
8 \def\arraybackslash{\let\\=\@arraycr}
```

`\@finalstrut` Bug fixed version from December 1995 LaTeX release. Old bug going back to LaTeX2.09...

```
9 \def\@finalstrut#1{%
10   \unskip\ifhmode\nobreak\fi\vrule\@width\z@\@height\z@\@depth\dp#1}
```

`\TY@count` Counter so that we know what column we are hacking around in.

```
11 \newcount\TY@count
```

`\tabulary` Top level macro for standard form.

```
12 \def\tabulary{%
13   \let\TY@final\tabular
14   \let\endTY@final\endtabular
15   \TY@tabular}
```

`\TY@tabular` Looks a lot like tabularx at this stage. Grab everything into a token register.

```
16 \def\TY@tabular#1{%
17   \edef\TY@{\@currenvir}%
18   {\ifnum0=`}\fi
```

At this point need to save locally things that tabulary will globally mess up. These are restored at the end of the environment.

```
19   \@ovxx\TY@linewidth
20   \@ovyy\TY@tablewidth
21   \count@\z@
22   \@tempswatrue
23   \@whilesw\if@tempswa\fi{%
24   \advance\count@\@ne
25   \expandafter\ifx\csname TY@F\the\count@\endcsname\relax
26     \@tempswafalse
27   \else
28     \expandafter\let\csname TY@SF\the\count@\expandafter\endcsname
29                     \csname TY@F\the\count@\endcsname
30     \global\expandafter\let\csname TY@F\the\count@\endcsname\relax
31     \expandafter\let\csname TY@S\the\count@\expandafter\endcsname
32                     \csname TY@\the\count@\endcsname
33   \fi}%
```

```
    \global\TY@count\@ne
    \TY@width\xdef{0pt}%
    \global\TY@tablewidth\z@
    \global\TY@linewidth#1\relax
Z\message{^^J^^JTable^^J%
Z        Target Width: \the\TY@linewidth^^J%
Z        \string\tabcolsep: \the\tabcolsep\space
Z        \string\arrayrulewidth: \the\arrayrulewidth\space
Z        \string\doublerulesep: \the\doublerulesep^^J%
Z        \string\tymin: \the\tymin\space
Z        \string\tymax: \the\tymax^^J}%
```

Placing this here means that nested tabulars will get this definition but that's probably OK, the extra code for LCR etc shouldn't do any harm

```
    \let\@classz\TY@classz
    \let\verb\TX@verb
    \toks@{}\TY@get@body}
```

`\TY@@mkpream` Saved version.

```
\let\TY@@mkpream\@mkpream
```

`\TY@mkpream` TY version.

```
\ExplSyntaxOn
\def\TY@mkpream{%
    \def\@addamp{%
      \if@firstamp \@firstampfalse \else
      \global\advance\TY@count\@ne
      \edef\@preamble{\@preamble & \noexpand\tbl_update_cell_data:}\fi
      \TY@width\xdef{0pt}}%
    \def\@acol{%
      \TY@subwidth\col@sep
      \@addtopreamble{\hskip\col@sep}}%
    \let\@arrayrule\TY@arrayrule
    \let\@classvi\TY@classvi
    \def\@classv{\save@decl
      \expandafter\NC@ecs\@nextchar\extracolsep{}\extracolsep\@@__tbl
      \sbox\z@{\d@llarbegin\@nextchar\d@llarend}%
      \TY@subwidth{\wd\z@}%
      \@addtopreamble{\d@llarbegin\the@toks\the\count@\relax\d@llarend}%
      \prepnext@tok}%
  \global\let\@mkpream\TY@@mkpream
  \TY@@mkpream}
\ExplSyntaxOff
```

`\TY@arrayrule` Pull this out so the colortbl support below can redefine

```
\def\TY@arrayrule{%
  \TY@subwidth\arrayrulewidth
  \@addtopreamble \vline}
```

\TY@classvi Pull this out so the colortbl support below can redefine

```
\def\TY@classvi{\ifcase \@lastchclass
  \@acol \or
  \TY@subwidth\doublerulesep
  \@addtopreamble{\hskip \doublerulesep}\or
  \@acol \or
  \@classvii
  \fi}
```

\TY@tab First run a tabular with all the column types fudged so that the widths of any rules or @-expresions are noted.

```
\def\TY@tab{%
  \setbox\z@\hbox\bgroup
```

Support displaymath by making it non-display in the first pass. (Other display environments defined in terms of $$ would need to be added here by packages that define them.)

```
  \let\[$\let\]$%
  \let\equation$\let\endequation$%
    \col@sep\tabcolsep
    \let\d@llarbegin\begingroup\let\d@llarend\endgroup
    \let\@mkpream\TY@mkpream
      \def\multicolumn##1##2##3{\multispan##1\relax}%
    \CT@start\TY@tabarray}
```

\TY@tabarray

```
\def\TY@tabarray{\@ifnextchar[{\TY@array}{\@array[t]}}
\def\TY@array[#1]{\@array[t]}
```

\TY@width Just a shorthand to access a column width macro.

```
\def\TY@width#1{%
  \expandafter#1\csname TY@\the\TY@count\endcsname}
```

\TY@subwidth Subtract a width from the current column width and also The total line table width and the target line width.

```
\def\TY@subwidth#1{%
  \TY@width\dimen@
  \advance\dimen@-#1\relax
  \TY@width\xdef{\the\dimen@}%
  \global\advance\TY@linewidth-#1\relax}
```

\endtabulary First run one modified tabular, making sure to add a blank row (cf longtable) to the end in case the user supplied last row is hidden by an hline or something.

```
\def\endtabulary{%
  \SuspendTagging {tabulary}%
  \gdef\@halignto{}%
```

Save values of counters, to reset after the trial

```
  \def\@elt##1{\global\value{##1}\the\value{##1}\relax}%
  \edef\TY@ckpt{\cl@@ckpt}%
```

```
  \expandafter\TY@tab\the\toks@
  \crcr\omit
  {\xdef\TY@save@row{}%
     \loop
    \advance\TY@count\m@ne
    \ifnum\TY@count>\z@
    \xdef\TY@save@row{\TY@save@row&\omit}%
    \repeat}\TY@save@row
  \endarray\global\setbox1=\lastbox\setbox0=\vbox{\unvbox1
    \unskip\global\setbox1=\lastbox}\egroup
  \ResumeTagging {tabulary}%
```

Check that `\tymin` is not too large.

```
  \dimen@\TY@linewidth
  \divide\dimen@\TY@count
  \ifdim\dimen@<\tymin
    \TY@warn{tymin too large (\the\tymin), resetting to \the\dimen@}%
    \tymin\dimen@
  \fi
```

Now take the last row apart, cf longtable or appendix D.

```
  \setbox\tw@=\hbox{\unhbox\@ne
    \loop
\@tempdima=\lastskip
\ifdim\@tempdima>\z@
Z   \message{ecs=\the\@tempdima^^J}%
   \global\advance\TY@linewidth-\@tempdima
\fi
    \unskip
    \setbox\tw@=\lastbox
    \ifhbox\tw@
Z     \message{Col \the\TY@count: Initial=\the\wd\tw@\space}%
      \ifdim\wd\tw@>\tymax
        \wd\tw@\tymax
Z       \message{> max\space}%
Z     \else
Z       \message{ \@spaces\space}%
      \fi
  \TY@width\dimen@
Z \message{\the\dimen@\space}%
  \advance\dimen@\wd\tw@
Z \message{Final=\the\dimen@\space}%
   \TY@width\xdef{\the\dimen@}%
      \ifdim\dimen@<\tymin
Z       \message{< tymin}%
        \global\advance\TY@linewidth-\dimen@
        \expandafter\xdef\csname TY@F\the\TY@count\endcsname
                                                    {\the\dimen@}%
       \else
      \expandafter\ifx\csname TY@F\the\TY@count\endcsname\z@
Z       \message{***}%
        \global\advance\TY@linewidth-\dimen@
        \expandafter\xdef\csname TY@F\the\TY@count\endcsname
                                                    {\the\dimen@}%
        \else
```

```
Z         \message{> tymin}%
          \global\advance\TY@tablewidth\dimen@
          \global\expandafter\let\csname TY@F\the\TY@count\endcsname
                                                              \maxdimen
        \fi\fi
        \advance\TY@count\m@ne
    \repeat}%
```

A bit cheap just doing this four times, but prevents any possibilities of looping....

```
    \TY@checkmin
    \TY@checkmin
    \TY@checkmin
    \TY@checkmin
```

Reset the counter.

```
    \TY@count\z@
```

Reset the LCRJ column definition to set paragraphs to the calculated widths.

```
    \let\TY@box\TY@box@v
```

Restore counter values.

```
    \TY@ckpt
```

Run a second tabular, and for the star form, unbox it.

```
  {\expandafter\TY@final\the\toks@\endTY@final}%
```

Finish off by restoring global commands.

```
  \count@\z@
  \@tempswatrue
  \@whilesw\if@tempswa\fi{%
  \advance\count@\@ne
  \expandafter\ifx\csname TY@SF\the\count@\endcsname\relax
    \@tempswafalse
  \else
    \global\expandafter\let\csname TY@F\the\count@\expandafter\endcsname
                    \csname TY@SF\the\count@\endcsname
    \global\expandafter\let\csname TY@\the\count@\expandafter\endcsname
                    \csname TY@S\the\count@\endcsname
  \fi}%
  \TY@linewidth\@ovxx
  \TY@tablewidth\@ovyy
    \ifnum0=`{\fi}}
```

`\TY@checkmin` Check that no column is squeezed below `\tymin`. If it is, fix the width of that column to `\tymin` and try again re-computing the ratio. (The new ratio will be smaller, and may squeeze yet more rows, so need to iterate this, currently just do it four times.)

```
\def\TY@checkmin{%
  \let\TY@checkmin\relax
\ifdim\TY@tablewidth>\z@
  \Gscale@div\TY@ratio\TY@linewidth\TY@tablewidth
% \changes{v0.9}{2008/12/01}
%      {\cs{TY@linewidth}}
 \ifdim\TY@tablewidth <\TY@linewidth
   \def\TY@ratio{1}%
```

```
 \fi
\else
  \TY@warn{No suitable columns!}%
  \def\TY@ratio{1}%
\fi
\count@\z@
Z \message{^^JLine Width: \the\TY@linewidth,
Z             Natural Width: \the\TY@tablewidth,
Z             Ratio: \TY@ratio^^J}%
\@tempdima\z@
\loop
\ifnum\count@<\TY@count
\advance\count@\@ne
  \ifdim\csname TY@F\the\count@\endcsname>\tymin
    \dimen@\csname TY@\the\count@\endcsname
    \dimen@\TY@ratio\dimen@
    \ifdim\dimen@<\tymin
Z     \message{Column \the\count@\space ->}%
      \global\expandafter\let\csname TY@F\the\count@\endcsname\tymin
      \global\advance\TY@linewidth-\tymin
      \global\advance\TY@tablewidth-\csname TY@\the\count@\endcsname
      \let\TY@checkmin\TY@@checkmin
    \else
      \expandafter\xdef\csname TY@F\the\count@\endcsname{\the\dimen@}%
      \advance\@tempdima\csname TY@F\the\count@\endcsname
    \fi
  \fi
Z \dimen@\csname TY@F\the\count@\endcsname\message{\the\dimen@, }%
\repeat
Z \message{^^JTotal:\the\@tempdima^^J}%
}
```

`\TY@@checkmin` Saved value

```
\let\TY@@checkmin\TY@checkmin
```

`TY@linewidth` Stores the target width.

```
\newdimen\TY@linewidth
```

`\tyformat` What to do with columns

```
\def\tyformat{\everypar{{\nobreak\hskip\z@skip}}}
```

`tymin` Columns narrower than this are not fudged.

```
\newdimen\tymin
\tymin=10pt
```

`tymin` Columns wider than this are all treated alike and set to the same width, to stop one particularly long entry hijacking the entire table.

```
\newdimen\tymax
\tymax=2\textwidth
```

`\@testpatch` Also add `LCRJ` although these don't do anything useful except in `tabulary`.

```
\def\@testpach{\@chclass
 \ifnum \@lastchclass=6 \@ne \@chnum \@ne \else
  \ifnum \@lastchclass=7 5 \else
   \ifnum \@lastchclass=8 \tw@ \else
    \ifnum \@lastchclass=9 \thr@@
   \else \z@
   \ifnum \@lastchclass = 10 \else
   \edef\@nextchar{\expandafter\string\@nextchar}%
   \@chnum
   \if \@nextchar c\z@ \else
    \if \@nextchar l\@ne \else
     \if \@nextchar r\tw@ \else
%     \if \@nextchar s6 \else
   \if \@nextchar C7 \else
    \if \@nextchar L8 \else
     \if \@nextchar R9 \else
     \if \@nextchar J10 \else
   \z@ \@chclass
   \if\@nextchar |\@ne \else
    \if \@nextchar !6 \else
     \if \@nextchar @7 \else
      \if \@nextchar <8 \else
       \if \@nextchar >9 \else
  10
  \@chnum
  \if \@nextchar m\thr@@\else
   \if \@nextchar p4 \else
    \if \@nextchar b5 \else
   \z@ \@chclass \z@ \@preamerr \z@ \fi \fi \fi \fi\fi \fi \fi\fi \fi
%   \fi
     \fi  \fi  \fi  \fi  \fi  \fi \fi \fi \fi \fi \fi}
```

`\TY@classz` Here hacked around without the respect Frank's code deserves...

```
\def\TY@classz{%
  \@classx
  \@tempcnta\count@
  \ifx\TY@box\TY@box@v
    \global\advance\TY@count\@ne
  \fi
  \let\centering c%
  \let\raggedright\noindent
  \let\raggedleft\indent
  \let\arraybackslash\relax
  \prepnext@tok
  \ifnum\@chnum<6
    \global\expandafter\let\csname TY@F\the\TY@count\endcsname\z@
  \fi
  \ifnum\@chnum=6
    \global\expandafter\let\csname TY@F\the\TY@count\endcsname\z@
  \fi
  \@addtopreamble{%
    \ifcase\@chnum
      \hfil\hskip1sp%
```

```
      \d@llarbegin\insert@column\d@llarend\do@row@strut\hfil \or
      \hskip1sp%
       \d@llarbegin \insert@column \d@llarend\do@row@strut\hfil \or
       \hfil\hskip1sp%
       \d@llarbegin \insert@column \d@llarend\do@row@strut \or
    \setbox\ar@mcellbox\vbox
    \@startpbox{\@nextchar}\insert@pcolumn \@endpbox
    \ar@align@mcell
    \do@row@strut\or
      \vtop \@startpbox{\@nextchar}\insert@pcolumn \@endpbox \or
      \vbox \@startpbox{\@nextchar}\insert@pcolumn \@endpbox \or
      \d@llarbegin \insert@column \d@llarend \do@row@strut \or% dubious "s" case
      \TY@box\centering\or
      \TY@box\raggedright\or
      \TY@box\raggedleft\or
      \TY@box\relax
    \fi}\prepnext@tok}
```

`\TY@box` The argument is `\centering` etc depending on whether LCRJ is used. However in this version the entries are set in horizontal mode with definitions mimicing the standard lcr columns. Later `\TY@box` will be redefined to `\TY@box@v` which really sets the entries in vertical mode.

```
\def\TY@box#1{%
  \ifx\centering#1%
    \hfil\hskip1sp%
    \d@llarbegin\insert@column\d@llarend\do@row@strut \hfil \else
  \ifx\raggedright#1%
      \hskip1sp%
      \d@llarbegin \insert@column \d@llarend\do@row@strut \hfil \else
  \ifx\raggedleft#1%
    \hfil\hskip1sp%
    \kern\z@ \d@llarbegin \insert@column \d@llarend\do@row@strut \else
  \ifx\relax#1%
       \d@llarbegin \insert@column \d@llarend\do@row@strut
  \fi  \fi  \fi  \fi}
```

`\TY@box@v` The version to use in a final run, set the `CLRJ` columns in a parbox of the appropriate width.

```
\def\TY@box@v#1{%
      \vtop \@startpbox{\csname TY@F\the\TY@count\endcsname}%
              #1\arraybackslash\tyformat
                                \insert@pcolumn\@endpbox}
```

`\TY@tablewidth` The natural width of the table on the first run.

```
\newdimen\TY@tablewidth
```

`\Gscale@div` Stolen from graphics package.

```
\def\Gscale@div#1#2#3{%
  \setlength\dimen@{#3}%
  \ifdim\dimen@=\z@
    \PackageError{graphics}{Division by 0}\@eha
    \dimen@#2%
  \fi
```

```
  \edef\@tempd{\the\dimen@}%
  \setlength\dimen@{#2}%
  \count@65536\relax
  \ifdim\dimen@<\z@
    \dimen@-\dimen@
    \count@-\count@
  \fi
  \loop
    \ifdim\dimen@<8192\p@
      \dimen@\tw@\dimen@
      \divide\count@\tw@
  \repeat
  \dimen@ii=\@tempd\relax
  \divide\dimen@ii\count@
  \divide\dimen@\dimen@ii
  \edef#1{\strip@pt\dimen@}}
```

`\TY@get@body` Place all tokens as far as the first `\end` into a token register. Then call `\TY@find@end` to see if we are at `\end{tabulary}`.

```
\long\def\TY@get@body#1\end
  {\toks@\expandafter{\the\toks@#1}\TY@find@end}
```

`\TY@find@end` If we are at `\end{tabulary}`, call `\end{tabulary}`, otherwise add `\end{...}` to the register, and call `\TY@get@body` again.

```
\def\TY@find@end#1{%
  \def\@tempa{#1}%
  \ifx\@tempa\TY@\def\@tempa{\end{#1}}\expandafter\@tempa
  \else\toks@\expandafter
    {\the\toks@\end{#1}}\expandafter\TY@get@body\fi}
```

`\TY@warn` Warning messages.

```
\def\TY@warn{%
  \PackageWarning{tabulary}}
```

```
\catcode`\Z=11
```

colortbl support.

```
\AtBeginDocument{
\@ifpackageloaded{colortbl}{%
\expandafter\def\expandafter\@mkpream\expandafter#\expandafter1%
  \expandafter{%
    \expandafter\let\expandafter\CT@setup\expandafter\relax
    \expandafter\let\expandafter\CT@color\expandafter\relax
    \expandafter\let\expandafter\CT@do@color\expandafter\relax
    \expandafter\let\expandafter\color\expandafter\relax
    \expandafter\let\expandafter\CT@column@color\expandafter\relax
    \expandafter\let\expandafter\CT@row@color\expandafter\relax
    \expandafter\let\expandafter\CT@cell@color\expandafter\relax
    \@mkpream{#1}}
\let\TY@@mkpream\@mkpream
\def\TY@classz{%
  \@classx
  \@tempcnta\count@
  \ifx\TY@box\TY@box@v
```

```
    \global\advance\TY@count\@ne
  \fi
  \let\centering c%
  \let\raggedright\noindent
  \let\raggedleft\indent
  \let\arraybackslash\relax
  \prepnext@tok
\expandafter\CT@extract\the\toks\@tempcnta\columncolor!\@nil
  \ifnum\@chnum<6
    \global\expandafter\let\csname TY@F\the\TY@count\endcsname\z@
  \fi
  \ifnum\@chnum=6
    \global\expandafter\let\csname TY@F\the\TY@count\endcsname\z@
  \fi
  \@addtopreamble{%
    \setbox\z@\hbox\bgroup\bgroup
    \ifcase\@chnum
      \hskip\stretch{.5}\kern\z@
      \d@llarbegin\insert@column\d@llarend\hskip\stretch{.5}\or
      \kern\z@%<<<<<<<<<<<<<<<<<<<<<<<<<<<<
       \d@llarbegin \insert@column \d@llarend \hfill \or
      \hfill\kern\z@ \d@llarbegin \insert@column \d@llarend \or
    \setbox\ar@mcellbox\vbox
    \@startpbox{\@nextchar}\insert@pcolumn \@endpbox
    \ar@align@mcell
    \do@row@strut \or
      \vtop \@startpbox{\@nextchar}\insert@pcolumn \@endpbox \or
      \vbox \@startpbox{\@nextchar}\insert@pcolumn \@endpbox \or
      \d@llarbegin \insert@pcolumn \d@llarend \or% dubious s case
      \TY@box\centering\or
      \TY@box\raggedright\or
      \TY@box\raggedleft\or
      \TY@box\relax
    \fi
 \egroup\egroup
\begingroup
  \CT@setup
  \CT@column@color
  \CT@row@color
  \CT@cell@color
  \CT@do@color
\endgroup
        \@tempdima\ht\z@
        \advance\@tempdima\minrowclearance
        \vrule\@height\@tempdima\@width\z@
\unhbox\z@
}\prepnext@tok}%

    \def\TY@arrayrule{%
      \TY@subwidth\arrayrulewidth
      \@addtopreamble{{\CT@arc@\vline}}}%

    \def\TY@classvi{\ifcase \@lastchclass
      \@acol \or
      \TY@subwidth\doublerulesep
```

```
      \ifx\CT@drsc@\relax
        \@addtopreamble{\hskip\doublerulesep}%
      \else
        \@addtopreamble{{\CT@drsc@\vrule\@width\doublerulesep}}%
      \fi\or
      \@acol \or
      \@classvii
      \fi}%
}{%
\let\CT@start\relax
}
```

end of at begin document

```
}
```

`\TX@warn` `\verb` support, uses same csnames as in TX so they share code if both loaded (this version names tabulary in the warning though). See tabularx for documentation.

```
{\uccode`\*=`\ %
\uppercase{\gdef\TX@verb{%
  \leavevmode\null\TX@vwarn
  {\ifnum0=`}\fi\ttfamily\let\\\ignorespaces
  \@ifstar{\let~*\TX@vb}{\TX@vb}}}}
\def\TX@vb#1{\def\@tempa##1#1{\toks@{##1}\edef\@tempa{\the\toks@}%
    \expandafter\TX@v\meaning\@tempa\\ \\\ifnum0=`{\fi}}\@tempa!}
\def\TX@v#1!{\afterassignment\TX@vfirst\let\@tempa= }
\begingroup
\catcode`\*=\catcode`\#
\catcode`\#=12
\gdef\TX@vfirst{%
  \if\@tempa#%
    \def\@tempb{\TX@v@#}%
  \else
    \let\@tempb\TX@v@
    \if\@tempa\space~\else\@tempa\fi
  \fi
  \@tempb}
\gdef\TX@v@*1 *2{%
  \TX@v@hash*1##\relax\if*2\\\else~\expandafter\TX@v@\fi*2}
\gdef\TX@v@hash*1##*2{*1\ifx*2\relax\else#\expandafter\TX@v@hash\fi*2}
\endgroup
\def\TX@vwarn{%
  \@warning{\noexpand\verb may be unreliable inside tabularx/y}%
  \global\let\TX@vwarn\@empty}
```

`\tblcrcrn` Patch to ensure `\tbl_crcr:n` ends with `\crcr`. (Will be fixed in later LaTeX releases.

```
⟨package⟩
```

```
\ExplSyntaxOn
\cs_set:Npn \@tempa #1 {
     \int_compare:nNnT \g__tbl_col_int > 0
         {
           \tbl_count_missing_cells:n {#1}
           \cr
```

```
463           }
464         }
465 \ifx\@tempa\tbl_crcr:n
466 \cs_set:Npn \tbl_crcr:n #1 {
467     \int_compare:nNnT \g__tbl_col_int > 0
468         {
469           \tbl_count_missing_cells:n {#1}
470         }
471     \crcr
472   }
473 \fi
474 \let\@tempa\@undefined
475 \ExplSyntaxOff
```

476 ⟨/package⟩

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

B

C

D

E

F

T

U

V

W

X

Z

Change History