ConTEXt basics for users: Table macros

Aditya Mahajan

Abstract

ConTEXt has four different table building macros. In the author's view the `table` macros, which are the oldest of the four, are the easiest to use for simple tables. This article gives a gentle introduction to these macros.

1 Introduction

Tables provide visual representation of information; understanding how to build them in a markup language can be difficult and frustrating. To make matters especially confusing for a new user, ConTEXt provides four different mechanisms to build tables — tables, tabulations, line tables, and natural tables— and the user must decide which one of the four to use. This can be a difficult decision, because there is no "one size fits all" solution. Each mechanism has its own strengths and weaknesses, and the best match depends on the application.

Tables can get fairly complicated. They can have fancy horizontal and vertical lines, colored rows, sophisticated MetaPost backgrounds for cells, and/or may need to be split across multiple pages. Some mechanisms are better than others at handling such advanced features.

However, most users rarely need these features; for the most part, they just want a simple, publication quality table. I think that out of the four table building macros, the `table` macros, which are based on Michael Wichura's TABLE package [1], are the easiest to use for simple tables.

I should note that this mechanism is no longer actively developed, and the more modern natural table macros are the unofficially recommended choice. However, I find natural tables to be a bit too verbose. The `table` macros take care of the *simple* tables. If you need to typeset *complicated* tables, you need to look into all the mechanisms and decide which one suits your needs best.

This article explains the basics of the `table` macros. It is cumbersome to separate out the features of Michael Wichura's TABLE package, and those added or adapted by Hans Hagen in the course of integrating with ConTEXt. For simplicity of exposition and with sincere apologies to Michael, I am going to refer to all these features as features of ConTEXt's `table` macros.

2 How to build tables

The basic structure of a table built with the `table` macros looks like this:

```
\starttable[preamble]
 \NC ... \NC \AR % first row
     ...         % middle rows
 \NC ... \NC \AR % last row
\stoptable
```

The *preamble* indicates the number and formatting of the columns. The column formatting is defined by "column specifiers", separated by vertical bars `|`; we also put bars at the beginning and end of the preamble. These `|` characters do *not* indicate vertical lines, as they do in LaTeX; they simply separate columns.

Each column specifier consists of keys, which are single characters, and the key combination indicates the formatting for that column. The number of columns is simply the number of column specifiers (that is, the number of `|`'s minus one).

The body of the table can contain an arbitrary number of rows. Each row has the form

```
\NC cell 1 \NC ... \NC cell n \NC \AR
```

The `\NC` (mnemonic: new column) control sequence is the usual column separator, and `\AR` (mnemonic: automatic row) is the usual row separator. Each column separator corresponds to one `|` in the preamble.

Let's consider an example. Suppose we want to produce the following table:

left	center	right
23	45	67

Each row consists of three columns, with the first one left aligned, the second center aligned, and the last right aligned. The preamble for this table is `[|l|c|r|]`. Here `l` is an abbreviation for left,[1] `c` for center, and `r` for right. Each column is as wide as necessary to accommodate the width of its contents. The input for the above table is:

```
\starttable[|l|c|r|]
  \NC left \NC center \NC right \NC \AR
  \NC 23   \NC 45     \NC 67    \NC \AR
\stoptable
```

The "extra" `\NC`'s at the end of each row are required. Omitting them can lead to hard-to-detect problems.

3 Specifying column widths

By default, each column is wide enough to accommodate the largest entry in that column. This may not

[1] `l` is actually *ell*. In the fonts used for this article, it is difficult to distinguish `l` (ell) from `1` (one). To make things easier for the reader, I will not use the digit `1` (one) in any example.

always be desired. If you want to ensure that each column has a certain *minimum* width, use `w(width)` in the column specifier. For example, if we want the middle column in the above table to be at least 4 cm wide, we can specify the preamble as

```
  \starttable[|l|cw(4cm)|r|]
```

which gives

left	center	right
23	45	67

On the other hand, if you want to specify a certain *maximum* width of a column, use the `p(width)` in the column specifier. Such columns are called paragraph columns. For example, suppose we want a column to be 5 cm wide:

TUGboat	The TUGboat journal is a unique benefit of joining TUG. It is currently published three times a year ...

We can use the following:

```
\starttable[|l|lp(5cm)|]
  \NC TUGboat
  \NC The TUGboat journal ...
  \NC \AR
\stoptable
```

This `lp(5cm)` key combination gives us left-justified text, i.e., ragged right. You can also use `rp(5cm)` to get right-justified (ragged left) text, and `cp(5cm)` to get centered text.

However, in most cases we want such paragraph columns to be justified. To achieve this we can use `xp(5cm)`, which gives:

TUGboat	The TUGboat journal is a unique benefit of joining TUG. It is currently published three times a year ...

4 Horizontal and vertical lines

To get horizontal lines that span the width of the entire table, you can use `\HL` between the rows where a line is desired.

By default, the width of the line (a.k.a. "rule") is 0.4 pt — TEX's default for lines. The linewidth can be controlled by the `rulethickness` option of `\setuptables`. For example, if you want 2 pt thick lines, use `\setuptables[rulethickness=2pt]`.

The width of a specific line can be increased using an optional argument to `\HL`. This argument must be an integer, and it increases the width of the current line by that factor. For example:

```
\setuptables[rulethickness=0.03em]
\starttable[|l|l|r|]
  \HL[3]
  \NC Animal \NC Desc \NC Cost (\$) \NC \AR
  \HL
  \NC Gnat \NC per gram \NC 13.65 \NC \AR
  \NC Emu  \NC stuffed  \NC 33.33 \NC \AR
  \NC Gnu  \NC stuffed  \NC 92.50 \NC \AR
  \HL[3]
\stoptable
```

gives

Animal	Desc	Cost ($)
Gnat	per gram	13.65
Emu	stuffed	33.33
Gnu	stuffed	92.50

Here the first and last lines are $3 * 0.03\,\text{em} = 0.09\,\text{em}$ thick, while the middle line is the default 0.03 em. Notice that the height of the rows around the horizontal lines is automatically adjusted — that's the `A` in `\AR`.

Other row specifiers allow manual adjustment of space: `\NR` (new row), `\SR` (single row), `\FR` (first row), `\MR` (middle row), and `\LR` (last row). Depending on the surrounding rows, `\AR` is converted into one of these row specifiers.

One can use `\tracetablestrue` to see what ConTEXt is doing behind the scenes. If we add `\tracetablestrue` before calling the above table, we get:

Animal	Desc	Cost ($)	`\SR`
Gnat	per gram	13.65	`\FR`
Emu	stuffed	33.33	`\MR`
Gnu	stuffed	92.50	`\LR`

The first row was surrounded by two horizontal lines, so the `\AR` in the first row was changed into `\SR` (single row). This introduces some space above and below the row, so that the row is not too close to the horizontal lines.

The second row has a horizontal line above it, so the `\AR` was changed into `\FR` (first row). This adjusts the space above the row to provide some distance from the horizontal line.

The third row does not have a horizontal line above or below it, so the `\AR` is changed into `\MR` (middle row). This adds normal inter-row space above and below the row.

The last row had a horizontal line below it, so the `\AR` was changed into `\LR` (last row). This adjusts the space below the row to provide some distance from the horizontal line.

The \NR command sequence does not adjust space at all. It should be used only when such tight spacing is required.

If you feel that ConTEXt has made a wrong choice, you can use the desired row separator instead of \AR to end that particular row in the table.

Vertical lines are rarely considered good typography. However, if you insist, use \VL as the column separator instead of \NC. For example, this input:

```
\starttable[|l|cw(4cm)|r|]
  \HL
  \VL left \VL center \VL right \VL \AR
  \HL
  \VL 23   \VL 45     \VL 67    \VL \AR
  \HL
\stoptable
```

gives

left	center	right
23	45	67

This also shows that with the w key, the width of the middle column was increased, in this case with centered text.

5 Documentation

So far I have shown some very basic features of the table macros. A few more details of these macros are documented in the beginner's manual [2]. However, the manual does not list all features of these macros. Documentation of the features inherited from the TABLE package is available in its own manual, which is sold by PCTEX [3].

Unfortunately, there is no real documentation of the enhancements, especially with regards to color, done by ConTEXt. There are a few examples in the source file core-tab.tex [4], and a few more on the ConTEXt wiki [5].

So, in the next issue of this series, I will explain some additional features of table macros, such as specifying the font style and color of each column, spanning multiple rows and columns, controlling the space between the columns, and splitting tables across pages.

6 Other mechanisms

To conclude this installment, here is a brief overview of the other table-building mechanisms mentioned earlier.

The tabulate macros were added for building running text aligned blocks like formula legends and facts. They are capable of automatic width calculation of paragraph columns (similar to the tabularx package in LATEX) and splitting across pages. They are built on the same underlying principle as the table macros, and in due time will be backward compatible with them. However, at present they do not support vertical rules and colors. Documentation and examples of tabulations are given in a *MAPS* article [6] by Hans Hagen.

Line tables are experimental macros for building large tables that can be split horizontally and vertically. It is possible to repeat table header lines and entry columns. Unfortunately, these macros are largely undocumented.

Natural tables are the most configurable table macros. Their syntax is inspired from HTML tables. It is possible to change the color, background, and style of a single cell, row, or column, and of odd or even rows and columns. It is also easy to use MetaPost backgrounds. Again, there is no detailed documentation, but many examples are present in Hans's example document [7]; another interesting example is Willi Egger's MyWay [8].

Finally, the pros and cons of each of these mechanisms is summarized in an article in the ConTEXt Garden wiki [9].

Bibliography

[1] Michael Wichura: The TABLE Macro Package. http://www.pctex.com/kb/47.html.

[2] Hans Hagen: ConTEXt an excursion. http://www.pragma-ade.com/show-man-1.htm.

[3] Michael Wichura: PICTEX and TABLE manual. http://store.pctexstore.com/maclimprom.html.

[4] Hans Hagen: ConTEXt core macros — TABLE embedding. http://www.logosrl.it/context/modules/current/singles/core-tab_ebook.pdf.

[5] ConTEXt garden — Table. http://wiki.contextgarden.net/Table.

[6] Hans Hagen: Tabulating in ConTEXt — text flow tables. *NTG MAPS*, Spring 1999. http://www.ntg.nl/maps/pdf/22_28.pdf.

[7] Hans Hagen: Natural tables in ConTEXt. http://www.pragma-ade.com/general/manuals/enattab.pdf.

[8] Willi Egger: My Way — Use of natural table environment. http://dl.contextgarden.net/myway/NaturalTables.pdf.

[9] ConTEXt garden — Tables Overview. http://wiki.contextgarden.net/Tables_Overview.

◇ Aditya Mahajan
University of Michigan
adityam (at) umich dot edu