formatting

Tabulating in ConTEXt
text flow tables

Hans Hagen
PRAGMA ADE
Ridderstraat 27
8061GH Hasselt NL
pragma@wxs.nl

abstract
This article describes the ConTEXt tabulate environment, which can be used to typeset tables that are part of the text flow. This mechanism differs from the TABLE based table mechanism, but recognizes the same preamble commands. It offers automatic width calculations when typesetting (multiple) paragraphs in tables and splits the tables over pages.

keywords
tables, alignment, ConTEXt

In a text, tabulated information can be included either in the text flow, fixed at the place it should appear, or it can be included at a preferred place, but given some freedom to float when there is no room. The tabulate commands discussed here take care of the fixed alternative, while the table commands are primary meant for floats. However, one is free to use whatever suits best, because none of them are limited to the cases mentioned.

While the table commands are in fact a layer around the TABLE package, the tabulate commands are written from scratch. Both have a definition preamble, and to keep things simple, the tabulate preamble keys are similar to those used in tables. We use `\NC` to as column separators, and `\NR` to go to the next row.

```
\starttabulate[|l|c|r|]
\NC this and that \NC left and right \NC here and there \NC \NR
\NC such and so   \NC up and down    \NC on and on      \NC \NR
\stoptabulate
```

this and that	left and right	here and there
such and so	up and down	on and on

The three keys mean:

`l` left aligned
`c` centered
`r` right aligned

There are also some spacing commands. These apply to both line and paragraph columns.

`i`*n* set space left
`j`*n* set space right
`k`*n* set space around

In these three keys *n* multiplies the spacing unit as set up with `\setuptabulate` (default: .5em).

```
\starttabulate[|l|k2c|r|]
\NC this and that \NC left and right \NC here and there \NC \NR
```

```
\NC such and so   \NC up and down    \NC on and on      \NC \NR
\stoptabulate
```

this and that left and right here and there
such and so up and down on and on

It is possible to specify the width of a column:

```
\starttabulate[|lw(4cm)|w(4cm)l|r|]
\NC this and that \NC left and right \NC here and there \NC \NR
\NC such and so   \NC up and down    \NC on and on      \NC \NR
\stoptabulate
```

this and that left and right here and there
such and so up and down on and on

The main reason for writing these tabulate commands, was that we wanted to typeset tables with paragraphs that would span the full available width. The related preamble keys are:

`w(`*d*`)` fixed width one liner
`p(`*d*`)` fixed width paragraph
`p` maximum width paragraph

In the next example the first column has an unknown width, the next one contains a left aligned paragraph and has a width of 4cm. The third column has 2cm width one–liner, while last paragraph column occupies the rest of the available width.

```
\starttabulate[|l|p(4cm)l|w(2cm)|p|]
...
\stoptabulate
```

A four column table with paragraph entries can be specified by saying:

```
\starttabulate[|p|p|p|p|]
...
\stoptabulate
```

Instead of retyping font switches, we can specify a font for a column. In the next table, the preamble specifies `[|lT|p|]`.

`B` **boldface**
`I` *italic*
`R` roman
`S` slanted
`T` teletype

For math there is:

`m` in line math mode
`M` display math mode

Using the `f` key, it is possible to define arbitrary font switches, for instance `f\bs`. There are some more hooks:

`f\command` font specification

`b{..}`	put before entry
`a{..}`	put after entry
`h\command`	do with entry (hook)

The next example shows how to apply a hook (`h`) and also puts something around en entry.

```
\starttabulate[|w(2cm)h\inframed|b{(}a{)}|p|]
\HC {Ugly} \NC indeed      \NC he said.     \NC \NR
\HC {Nice} \NC but useless \NC I would say. \NC \NR
\stoptabulate
```

Here `\inframed` keeps the frame within the normal line height and depth (this is a special case of `\framed`). Watch careful: hooked entries are marked with `\HC` and don't forget the braces, or whatever the hook–command expects! This example turns up as:

Ugly	(indeed)	he said.
Nice	(but useless)	I would say.

One can use the hook for specific formatting purposes, like in:

item	**quantity**
figures	
tables	
formulas	

There are three specially typeset cells. Watch the braces!

```
\def\SomeBar#1{\blackrule[width=#1em]}
\starttabulate[|l|lh\SomeBar|]
\HL
\NC \bf item \NC \bf quantity \NC \NR
\HL
\NC figures  \HC  {5}         \NC \NR
\NC tables   \HC  {8}         \NC \NR
\NC formulas \HC {12}         \NC \NR
\HL
\stoptabulate
```

Until now, we used `\NC` to go to the next column. For special purposes one can use `\EQ` to force a column separator.

```
\starttabulate
\NC equal \EQ one can change the separator by changing the \type {EQ}
              variable with the tabulate setup command \NC \NR
\NC colon \EQ by default, a colon is used, but an equal sign suits
              well too \NC \NR
\stoptabulate
```

Typeset this turns up as:

equal : one can change the separator by changing the `EQ` variable with the tabulate setup command
colon : by default, a colon is used, but an equal sign suits well too

We've seen \NC for normal entries, \EQ for entries seperated by an equal sign, and \HC for hooked ones. There is also \HQ for a hooked entry with separator. When one does not want any formatting at all, \RC and \RQ can be used.

	normal	**raw**	**hook**
equal	\EQ	\RQ	\HQ
none	\NC	\RC	\HC

This small table shows all three categories. We've got 4 centered columns, either **bold** or verbatim and two cells are aligned different. This table is coded as:

```
\starttabulate[|*{4}{cBh\type|}]
\NC           \NC normal \NC raw   \NC hook  \NC \NR
\RC \bf equal \HC {\EQ}  \HC {\RQ} \HC {\HQ} \NC \NR
\RC \bf none  \HC {\NC}  \HC {\RC} \HC {\HC} \NC \NR
\stoptabulate
```

The equal sign, or whatever else symbol is set up, can also be forced by the e key in the preamble.

e insert an equal symbol in the next column

When we have multiple columns with similar templates, we can save some typing by repeating them. We have of course to make sure that the number of |'s is ok.

```
\starttabulate[|*{6}{k1pc|}]
\NC this and that \NC left and right \NC here and there \NC
    such and so   \NC up and down    \NC on and on      \NC \NR
\stoptabulate
```

Counting the |'s, we have $1 + 6 \times 1 = 7$ of them.

this and that	left and right	here and there	such and so	up and down	on and on

A better example of automatic width calculation is given below:

tables We use the \starttable command when we are typesetting tables that are separate entities, that may float and are sort of independent of the normal text flow.

tabulate The \starttabulate command is meant for typesetting tabular text in the normal text flow. Therefore automatic width calculation, as demonstrated here, comes in handy.

This was entered as:

```
\starttabulate[|l|p|]
\NC tables   \NC We use the \type {\starttable} command when we are
                typesetting tables that are separate entities, that
                may float and are sort of independent of the normal
                text flow. \NC \NR
\NC tabulate \NC The \type {\starttabulate} command is meant for
                typesetting tabular text in the normal text flow.
                Therefore automatic width calculation, as demonstrated
```

```
                here, comes in handy. \NC \NR
\stoptabulate
```

When no template is given, `[|l|p|]` is assumed, which sometimes saves some typing. There is however a better way to save time, because one can define specific tabulate environments, like:

```
\definetabulate[Three][|lB|lS|p|]
\startThree
\NC one \NC two \NC three four five six seven eight nine ten
                    eleven twelve and so on and on and on \NC \NR
\stopThree
```

one two three four five six seven eight nine ten eleven twelve and so on and on and on

The three tabulate commands can be summarized with:

```
\definetabulate[.1.][.2.][.3.]

.1.      name
.2.      name
.3.      text
```

The first argument identifies the tabulation. The second argument is optional and identifies a related tabulation. More on that later. The last argument always defines the preamble.

```
\starttabulate[...][..,..=..,..] ... \stoptabulate

...      text
..=..    see \useexternalfigure
```

The (optional) first argument holds the preamble, and the optional second one can be used to change settings.

```
\setuptabulate[...][..,..=..,..]

...          name
unit         dimension
indenting    yes no
before       command
after        command
inner        command
EQ           text
```

The optional argument specifies a related tabulation. By setting `indenting` to `yes`, the table is indented according to the current indentation scheme. Left and right skips are always taken into account! The unit concerns the spacing as set by the spacing keys `i`, `j` and `k`. Commands assigned to `inner` are executed before the first columns is typeset.

One can add horizontal lines to a table by `\HL`. This command automatically takes care of spacing:

```
\starttabulate[|l|p|]
\HL
\NC small  \NC They say, small is beautiful.            \NC \NR
\HL
\NC medium \NC It seems that the medium is the message. \NC \NR
\HL
\NC large  \NC Large T||shirts are always sold out.     \NC \NR
\HL
\stoptabulate
```

When a page break occurs at such a horizontal rule, the rule is automatically duplicated. One can force top, mid or bottom rules with \FL, \ML and \LL.

small	They say, small is beautiful.
medium	It seems that the medium is the message.
large	Large T–shirts are always sold out.

Although the tabulate environment is primary meant for use in the text flow, it is quite legal to use it in floating tables. When used as float, the spacing around a tabulation is automatically surpressed. In the text flow however, the tabulate commands adjust themselves to the current text width and indentation.

- This means that a table created with this environment can be used within for instance an itemize.

  see this As to be expected, paragraph entries are automatically adjusted to the smaller text width.

- This small table was entered as:

```
\starttabulate
\NC see this \NC As to be expected, paragraph entries are
             automatically adjusted to the smaller text
             width. \NC \NR
\stoptabulate
```

Although tables in themselves can be used to format text in columns, occasionally using for instance an itemize in a table makes sense. The next, a bit weird, example shows this.

1. first	□□□□□ this or that	α. alpha
2. second	□□□□□ such or so	β. beta
3. third	□□□□□ here or there	γ. gamma

In such situations, packing items just looks better.

```
\starttabulate[|p(2cm)|p(5cm)|p|]
\NC \startitemize[n,packed]
    \item first \item second \item third
    \stopitemize
\NC \startitemize[packed][items=5,width=2.5em,distance=.5em]
    \its this or that \its such or so \its here or there
    \stopitemize
\NC \startitemize[g,packed,broad]
```

```
    \item alpha \item beta \item gamma
    \stopitemize
\NC\NR
\stoptabulate
```

Because the table content is read in before it is processed, there are some limitations when on–the–fly `\catcode` changes are involved. In day–to–day use however, one will not run into trouble that fast. The tabulate environment is mildly E-TEX aware, so one can expect less `\catcode` related problems when using this TEX alternative.

Tabulations can for instance be used to compose tables with numbered material, like the one below:

not to much	1.220
pretty much	5.186
totalling up to	6.406

This table is entered as:

```
\starttabulate[|l|r|]
\NC not to much     \NC          1.220  \NC \NR
\NC pretty much     \NC          5.186  \NC \NR
\NC totalling up to \NC \overbar{6.406} \NC \NR
\stoptabulate
```

Sometimes one wants to align entries in such tables by hand, and like in the TABLE based table environment, we can use ~ for this purpose. This character inserts an one digit wide space. Normally such a tie produces an unbreakable space, so changing its meaning is to be specified in the preamble.

```
\starttabulate[|l|~c|]
\NC this much \NC ~12 \NC \NR
\NC that far  \NC 185 \NC \NR
\stoptabulate
```

The next few examples show how we can define related tabulations. Actually, the main reason for programming this mechanism, originates in the wish to re–implement the legend macros.

```
\definetabulate [legend]       [|emj1|i1|mR|]
\definetabulate [legend] [two] [|emj1|emk1|i1|mR|]
\setuptabulate  [legend]       [unit=.75em,EQ={=}]
```

Now we can say things like:

```
\startlegend
\NC w \NC the width of the box  \NC pt \NR
\NC h \NC the height of the box \NC pt \NR
\NC d \NC the depth of the box  \NC pt \NR
\stoplegend
```

This simple legend turns up as:

w	=	the width of the box	pt
h	=	the height of the box	pt
d	=	the depth of the box	pt

An additional entry is possible with the `two` alternative.

```
\startlegend[two]
\NC w \NC width  \NC the width of the box  \NC pt \NR
\NC h \NC height \NC the height of the box \NC pt \NR
\NC d \NC depth  \NC the depth of the box  \NC pt \NR
\stoplegend
```

This related tabulation inherits the settings from the parent tabulation. Of course we could have defined `\startlegendtwo`, but we wanted downward compatibility with existing macros.

w = $width$ = the width of the box pt
h = $height$ = the height of the box pt
d = $depth$ = the depth of the box pt

The related fact macros are defined as:

```
\definetabulate [fact] [|R|ecmj1|i1mR|]
\setuptabulate  [fact] [unit=.75em,EQ={=}]
```

The first column is typeset in roman, the next one is separated from the first one by an equal sign, is centered, is typeset in math mode, and gets a bit more space afterwards. The last column is typeset in math mode, but inside there we switch to roman; some extra space is added in front. So:

```
\startfact
\NC width  \NC w \NC 48pt \NR
\NC height \NC h \NC  9pt \NR
\NC depth  \NC d \NC  3pt \NR
\stopfact
```

Indeed gives:

width w = 48pt
height h = 9pt
depth d = 3pt

In real life the definitions shown here also have something assigned to `inner`, which enables more compact specifications:

```
\startfact
\\ width  \\ w \\ 48pt \\
\\ height \\ h \\  9pt \\
\\ depth  \\ d \\  3pt \\
\stopfact
```

We show one last example, demonstrating the automatic paragraph width calculation. This example also shows that the last `\NC` is redundant.

```
\starttabulate[|Bl|p|Bl|]
\NC Example \NC \input tufte \NC Edward Tufte \NR
\stoptabulate
```

Example We thrive in information–thick worlds because of our marvelous and everyday capacity to select, edit, single out, structure, highlight, group, pair, merge, harmonize, synthesize, focus, organize, condense, reduce, boil down, choose, categorize, catalog, classify, list, abstract, scan, look in- **Edward Tufte**

to, idealize, isolate, discriminate, distinguish, screen, pigeonhole, pick over, sort, integrate, blend, inspect, filter, lump, skip, smooth, chunk, average, approximate, cluster, aggregate, outline, summarize, itemize, review, dip into, flip through, browse, glance into, leaf through, skim, refine, enumerate, glean, synopsize, winnow the wheat from the chaff and separate the sheep from the goats.

As can be expected, these commands have their dutch and german counterparts:

english	dutch	german
\starttabulate	\starttabulatie	\starttabulator
\definetabulate	\definieertabulatie	\definieretabulator
\setuptabulate	\steltabulatiein	\stelletabulatorein

The preamble commands and column and row separators are independant of the interface.

Bug or Feature?
misleading font messages

Hans Hagen
Taco Hoekwater

abstract

TEX error messages and warnings are not always that clear. Sometimes confusion is due to optimizations in TEX the program. We will discuss the not always honest \the\font.

keywords
fonts, error messages

TEX is written in a time when computer resources were not as generous as in our days. This means that in TEX the program there are optimizations wherever possible. Sometimes these optimizations have unexpected side effects.

Make yourself a file saying:

```
\font\one  =cmr10
\font\two  =cmr10
\font\three=cmr10

\one   One   \message{\the\font}
\two   Two   \message{\the\font}
\three Three \message{\the\font}
```

When this file is run through TEX, you will see that in all occasions, TEX reports that font \three is used. This is due to the fact that TEX only loads a font once, which in itself is one of the reasons why preloading fonts in plain TEX can speed up font definitions in production runs (the TEXbook explains this).

The undesirable side effect is that, although the font is still accessible by the old name(s), the last name used in defining one is reported back in messages concerning for instance overfull boxes. The degree of confusion arising from this situation depends on the way the macro package names fonts. Especially when a fall back mechanism is implemented, users can start searching for the wrong causes of the problem. Imagine a message mentioning problems with typesetting text (an overfull box for example) where a sans serif font is referred to, while in fact a serif is used as fall back.

This side effect can revelate itself rather disguised in high level font mechanisms. It took both authors quite some time nailing down the origin of these confusing messages, especially because one would expect the opposite: report the old name. Finding the source of the problem was complicated by the fact that (depending on the application) it occurs independant of grouping, and therefore is pretty hard to trace. We hope that in a next release of E-TEX there will be an appropriate warning issued in the log file when this low level optimization takes place.