NAME

groff_hdtbl – Heidelberger table macros for GNU roff

DESCRIPTION

The **hdtbl** macros consist of four base and three optional macros, controlled by about twenty arguments. The syntax is simple and similar to the **HTML** table model and nearly as flexible: You can write sequences of tokens (macro calls with their arguments and content data), separated by blanks and beginning with a macro call, into the same line to get compact and cleanly arrranged input. An advantage of **hdtbl** is that the tables are constructed without calling a preprocessor; this means that **groff**'s full macro capabilities are available. On the other hand, table processing with **hdtbl** is much slower than using the **tbl**(1) preprocessor. A further advantage is that the HTML-like syntax of **hdtbl** will be easily converted to HTML; this is not implemented yet.

USAGE

In this and the next section, we present examples to help users understand the basic workflow of **hdtbl**. First of all, you must load the *hdtbl.tmac* file. As with nearly all other groff macro packages, there are two possibilities to do so: Either add the line

```
.mso hdtbl.tmac
```

to your *roff* file before using any macros of the **hdtbl** package, or add the option

```
-m hdtbl
```

to the command line of groff (before the document file which contains **hdtbl** macros). Then you can include on or more tables in your document, where each one must be started and ended with the `.TBL` and `.ETB` macros, respectively.

In this man page, we approximate the result of each example in the *tty* format to be as generic as possible since **hdtbl** currently only supports the PS and PDF output devices.

The simplest well-formed table consists of just single calls to the four base table macros in the right order. Here we construct a table with only one cell.

```
.TBL
.TR
.TD
contents of the table cell
.ETB
```

A *tty* representation is

```
+-----------------------------------------------------+
| contents-of-the-table-cell                          |
+-----------------------------------------------------+
```

Equivalent to the above is the following notation.

```
.TBL .TR .TD "contents of the table cell" .ETB
```

By default, the formatted table is inserted into the surrounding text at the place of its definition. If the vertical space isn't sufficient, it is placed at the top of the next page. Tables can also be stored for later insertion.

Using '*row-number*`*`*column-number*' as the data for the table cells, a table with two rows and two columns can be written as

```
.TBL cols=2
.  TR .TD 1*1 .TD 1*2
.  TR .TD 2*1 .TD 2*2
.ETB
```

A *tty* representation is

```
+-------------------------+-------------------------+
```

```
| 1*1                       | 1*2                       |
+---------------------------+---------------------------+
| 2*1                       | 2*2                       |
+---------------------------+---------------------------+
```

Here we see a difference from HTML tables: The number of columns must be explicitly specified using the '`cols=`*m*' argument (or indirectly via the '`width`' argument, see below).

The contents of a table cell is arbitrary; for example, it can be another table, without restriction to the nesting depth. A given table layout can be either constructed with suitably nested tables or with proper arguments to `.TD` and `.TH`, controlling column and row spanning. Note, however, that this table

```
.TBL
.  TR
.    TD
.      nop 1*1 1*2
.  TR
.    TD
.      TBL cols=2 border=
.        TR
.          TD
.            nop 2*1
.          TD
.            nop 2*2
.      ETB
.ETB
```

and this table

```
.TBL cols=2
.  TR
.    TD colspan=2
.      nop 1*1 1*2
.  TR
.    TD
.      nop 2*1
.    TD
.      nop 2*2
.ETB
```

are similar but not identical (the use of `.nop` is purely cosmetic to get proper indentation).

The first table looks like

```
+-----------------------------------------------------+
| 1*1 1*2                                             |
+-----------------------------------------------------+
|                                                     |
| 2*1                         2*2                     |
|                                                     |
+-----------------------------------------------------+
```

and the second one like

```
+-----------------------------------------------------+
| 1*1 1*2                                             |
+---------------------------+-------------------------+
| 2*1                       | 2*2                     |
+---------------------------+-------------------------+
```

Here the latter table in a more compact form.

```
.TBL cols=2 .TR ".TD colspan=2" 1*1 1*2
.            TR .TD 2*1 .TD 2*2 .ETB
```

If a macro has one or more arguments (see below), and it is not starting a line, everything belonging to this macro including the macro itself must be enclosed in double quotes.

MACROS AND ARGUMENTS

The order of macro calls and other tokens follows the HTML model. In the following list, valid predecessors and successors of all **hdtbl** macros are given, together with the possible arguments.

Macro arguments are separated by blanks. The order of arguments is arbitrary; they are of the form

`key=`*`value`*

or

`key='`*`value1`* [*`value2`* [...]]`'`

with the only exception of the optional argument of the macro `.ETB`, which is the string '`hold`'. Another possible form is

`"key=`*`value1`* [*`value2`* [...]]`"`

However, this is limited to the case where the macro is the first one in the line and not already enclosed in double quotes.

Argument values specified below as *c* are colors predefined by **groff** or colors defined by the user with the `.defcolor` request. Argument values *d* are decimal numbers with or without decimal point. Argument values *m* are natural numbers. Argument values *n* are numerical values with the usual **groff** scaling indicators. Some of the arguments are specific to one or two macros, but most of them can be specified with `.TBL`, `.TR`, `.TD`, and `.TH`. These common arguments are explained in the next subsection.

Most of the argument default values can be changed by the user by setting corresponding default registers or strings, as listed below.

`.TBL` [*`args`*]

Begin a new table.

predecessor: `.TD`, `.TH`, `.ETB`, `cell contents`
successor: `.CPTN`, `.TR`
arguments:

`border=`[*`n`*]
: Thickness of the surrounding box border. '`border=`' (no value) means neither a surrounding box border nor any horizontal or vertical separator lines between the table rows and cells. '`border=0`' suppresses the surrounding box border, but still allows separator lines between cells and rows.
Default: '`border=.1n`' (register '`t*b`').

`bc=`*`c`*
: Border color.
Default: '`bc=red4`' (string '`t*bc`').

`cols=`*`m`*
: Number of table columns. This argument is necessary if more than one column is in the table and no '`width`' arguments are present.
Default: '`cols=1`' (register '`t*cols`').

`cpd=`*`n`*
: Cell padding, i.e., the extra space between the cell space border and the cell contents.
Default: '`cpd=.5n`' (register '`t*cpd`').

`csp=`*`n`*
: Cell spacing, i.e., the extra space between the table border or vertical or horizontal lines between cells and the cellspace.
Default: '`csp=.5n`' (register '`t*csp`').

`tal=l|c|r`
: Horizontal alignment of the table, if it is smaller than the line width. '`tal=l`': left alignment. '`tal=c`': centered alignment. '`tal=r`': right alignment.

Default: 'tal=l' (register 't*tal').

width='*w1* [*w2* [...]]'
: Widths of table cells. *w1*, *w2*, ... are either numbers of type *n* or natural numbers with the pseudo-scaling indicator '%', with the meaning "percent of the actual line length (or column length for inner tables, respectively)". If there are less width values than table columns, the last width value is used for the remaining cells. The argument

```
width='1.5i 10%'
```

for example indicates that the first column is 1.5 inches wide; the remaining columns take 1/10 of the column length each.
Default: The table width equals the outer line length or column length; the columns have equal widths.

height=*n*
: Height of the table. If the table with its contents is lower than *n*, the last row is stretched to this value.

.CPTN [*args*]
: Text of caption.

The (optionally numbered) table caption. .CPTN is optional.

predecessor: .TBL
successor: .TR
arguments:

val=t|b
: Vertical alignment of the table caption. 'val=t': The caption is placed above the table. 'val=b': The caption is placed below the table.
Default: 'val=t' (string 't*cptn').

.TR [*args*]
: Begin a new table row.

predecessor: .TBL, .CPTN, .TD, .TH, .ETB, cell contents
successor: .TD, .TH
arguments:

height=*n*
: The height of the row. If a cell in the row is higher than *n*, this value is ignored; otherwise the row height is stretched to *n*.

.TD [*args* [*cell contents*]]
: Begin a table data cell.

.TH [*args* [*cell contents*]]
: Begin a table header cell.

Arguments and cell contents can be mixed. The macro .TH is not really necessary and differs from .TD only in three default settings, similar to the <TH> and <TD> HTML tags: The contents of .TH is horizontally and vertically centered and typeset in boldface.

predecessor: .TR, .TD, .TH, .ETB, cell contents
successor: .TD, .TH, .TR, .ETB, cell contents
arguments:

colspan=*m*
: The width of this cell is the sum of the widths of the *m* cells above and below this row.

rowspan=*m*
: The height of this cell is the sum of the heights of the *m* cells left and right of this column.

Remark: Overlapping of column and row spanning, as in the following table fragment (the overlapping happens in the second cell in the second row), is invalid and causes incorrect results.

```
.TR .TD 1*1 ".TD 1*2 rowspan=2" .TD 1*3
.TR ".TD 2*1 colspan=2"         .TD 2*3
```

A working example for headers and cells with **colspan** is

```
.TBL cols=3
.  TR ".TH colspan=2" header1+2 .TH header3
.  TR .TD 1*1 .TD 1*2 .TD 1*3
.  TR .TD 2*1 ".TD colspan=2" 2*2+3
.ETB
```

This looks like

```
+------------------------------+---------------+
|          header1+2           |    header3    |
+--------------+---------------+---------------+
| 1*1          | 1*2           | 1*3           |
+--------------+---------------+---------------+
| 2*1          | 2*2+3                         |
+--------------+-------------------------------+
```

A working example with **rowspan** is

```
.TBL cols=3
.  TR
.  TD 1*1
.  TD rowspan=2 1+2*2
.  TD 1*3
.
.  TR
.  TD 2*1
.  TD 2*3
.ETB
```

which looks like

```
+--------------+---------------+---------------+
| 1*1          | 1+2*2         | 1*3           |
+--------------+               +---------------+
| 2*1          |               | 2*3           |
+--------------+---------------+---------------+
```

.ETB [**hold**]

End of the table.

This macro finishes a table. It causes one of the following actions.

- If the argument 'hold' is given, the table is held until it is freed by calling the macro `.t*free`, which in turn prints the table immediately, either at the current position or at the top of the next page if its height is larger than the remaining space on the page.
- Otherwise, if the table is higher than the remaining space on the page, it is printed at the top of the next page.
- If neither of the two above constraints hold, the table is printed immediately at the place of its definition.

predecessor: `.TD`, `.TH`, `.ETB`, `cell contents`
successor: `.TBL`, `.TR`, `.TD`, `.TH`, `.ETB`, `cell contents`

arguments:

`hold` Prevent the table from being printed until it is freed by calling the macro `.t*free`. This argument is ignored for inner (nested) tables.

`.t*free` [*n*]

Free the next held table or *n* held tables. Call this utility macro to print tables which are held by using the 'hold' argument of the `.ETB` macro.

Arguments common to `.TBL`, `.TR`, `.TD`, and `.TH`

The arguments described in this section can be specified with the `.TBL` and `.TR` macros, but they are eventually passed on to the table cells. If omitted, the defaults take place, which the user can change by setting the corresponding default registers or strings, as documented below. Setting an argument with the `.TBL` macro has the same effect as setting it for all rows in the table. Setting an argument with a `.TR` macro has the same effect as setting it for all the `.TH` or `.TD` macro in this row.

`bgc=`[*c*]

The background color of the table cells. This includes the area specified with the 'csp' argument. The argument 'bgc=' (no value) suppresses a background color; this makes the background transparent.

Default: 'bgc=bisque' (string 't*bgc').

`fgc=`*c* The foreground color of the cell contents.

Default: 'fgc=red4' (string 't*fgc').

`ff=`*name*

The font family for the table. *name* is one of the groff font families, for example `A` for the Avant-Garde fonts or `HN` for Helvetica-Narrow.

Default: The font family found before the table (string 't*ff').

`fst=`*style*

The font style for the table. One of `R`, `B`, `I`, or `BI` for roman, **bold**, *italic*, or ***bold italic***, respectively. As with **roff**'s `.ft` request the 'fst' argument can be used to specify the font family and font style together, for example 'fst=HNBI' instead of 'ff=HN' and 'fst=BI'.

Default: The font style in use right before the table (string 't*fst').

`fsz='`*d1* [*d2*]`'`

A decimal or fractional factor *d1*, by which the point size for the table is changed, and *d2*, by which the vertical line spacing is changed. If *d2* is omitted, value *d1* is taken for both.

Default: 'fsz='1.0 1.0'' (string 't*fsz').

`hal=l|c|b|r`

Horizontal alignment of the cell contents in the table. 'hal=l': left alignment. 'hal=c': centered alignment. 'hal=b': both (left and right) alignment. 'hal=r': right alignment.

Default: 'hal=b' (string 't*hal').

`val=t|m|b`

Vertical alignment of the cell contents in the table for cells lower than the current row. 'val=t': alignment below the top of the cell. 'val=m': alignment in the middle of the cell. 'val=b': alignment above the cell bottom.

Default: 'val=t' (string 't*val').

`hl=[s|d]`

Horizontal line between the rows. If specified with `.TD` or `.TH` this is a separator line to the cell below. 'hl=' (no value): no separator line. 'hl=s': a single separator line between the rows. 'hl=d': a double separator line.

The thickness of the separator lines is the half of the border thickness, but at least 0.1 inches. The distance between the double lines is equal to the line thickness.

Remark: Together with 'border=0' for proper formatting the value of 'csp' must be at least .05 inches for single separator lines and .15 inches for double separator lines.

Default: 'hl=s' (string 't*hl').

`vl=[s|d]`

Vertical separator line between the cells. If specified with `.TD` or `.TH` this is a separator line to the cell on the right. '`vl=s`': a single separator line between the cells. '`vl=d`': a double separator line. '`vl=`' (no value): no vertical cell separator lines. For more information see the documentation of the '`hl`' argument above.
Default: '`vl=s`' (string '`t*vl`').

HDTBL CUSTOMIZATION

Before creating the first table, you should configure default values to minimize the markup needed in each table. The following example sets up defaults suitable for typical papers:

```
.ds t*bgc white\" background color
.ds t*fgc black\" foreground color
.ds t*bc black\"  border color
.nr t*cpd 0.1n\"  cell padding
```

The file **examples/common.roff** provides another example setup in the "minimal Page setup" section.

A table which does not fit on a partially filled page is printed automatically on the top of the next page if you append the little utility macro `t*hm` to the page header macro of your document's main macro package. For example, say

```
.am pg@top
.  t*hm
..
```

if you use the **ms** macro package.

The macro `t*EM` checks for held or kept tables, and for missing `ETB` macros (table not closed). You can append this macro to the "end" macro of your document's main macro package. For example:

```
.am pg@end-text
.  t*EM
..
```

If you use the **ms** macro package.

BUGS AND SUGGESTIONS

Please send your commments to the groff mailing list ⟨groff@gnu.org⟩ or directly to the author.

AUTHORS

The **hdtbl** macro package was written by Joachim Walsdorff ⟨Joachim.Walsdorff@urz.uni–heidelberg.de⟩.

SEE ALSO

groff(1)
provides an overview of GNU *roff* and details how to invoke *groff* at the command line.

groff(7)
summarizes the *roff* language and GNU extensions to it.

tbl(1) describes the traditional *roff* preprocessor for tables.