## NAME

**tbl** - tbl language reference for mandoc

## DESCRIPTION

The **tbl** language formats tables. It is used within mdoc(7) and man(7) pages. This manual describes the subset of the **tbl** language accepted by the mandoc(1) utility.

Each table is started with a roff(7) **TS** macro, consist of at most one line of *Options*, one or more *Layout* lines, one or more *Data* lines, and ends with a **TE** macro. All input must be 7-bit ASCII.

### Options

If the first input line of a table ends with a semicolon, it contains case-insensitive options separated by spaces, tabs, or commas. Otherwise, it is interpreted as the first *Layout* line.

The following options are available. Some of them require arguments enclosed in parentheses:

**allbox** Draw a single-line box around each table cell.

**box** Draw a single-line box around the table. For GNU compatibility, this may also be invoked with **frame**.

**center** Center the table instead of left-adjusting it. For GNU compatibility, this may also be invoked with **centre**.

**decimalpoint**
Use the single-character argument as the decimal point with the **n** layout key. This is a GNU extension.

**delim** Use the two characters of the argument as eqn(7) delimiters. Currently unsupported.

**doublebox**
Draw a double-line box around the table. For GNU compatibility, this may also be invoked with **doubleframe**.

**expand** Increase the width of the table to the current line length. Currently ignored.

**linesize**
Draw lines with the point size given by the unsigned integer argument. Currently ignored.

**nokeep** Allow page breaks within the table. This is a GNU extension and currently ignored.

**nospaces**
Ignore leading and trailing spaces in data cells. This is a GNU extension.

**nowarn** Suppress warnings about tables exceeding the current line length. This is a GNU extension and currently ignored.

**tab** Use the single-character argument as a delimiter between data cells. By default, the horizontal tabulator character is used.

## Layout

The table layout follows an *Options* line or a roff(7) **TS** or **T&** macro. Each layout line specifies how one line of *Data* is formatted. The last layout line ends with a full stop. It also applies to all remaining data lines. Multiple layout lines can be joined by commas on a single physical input line.

Each layout line consists of one or more layout cell specifications, optionally separated by whitespace. The following case-insensitive key characters start a new cell specification:

**c** Center the string in this cell.

**r** Right-justify the string in this cell.

**l** Left-justify the string in this cell.

**n** Justify a number around its last decimal point. If no decimal point is found in the number, it is assumed to trail the number.

**s** Horizontally span columns from the last non-**s** layout cell. It is an error if a column span follows a _ or = cell, or comes first on a layout line. The combined cell as a whole consumes only one cell of the corresponding data line.

**a** Left-justify a string and pad with one space.

^ Vertically span rows from the last non-^ layout cell. It is an error to invoke a vertical span on the first layout line. Unlike a horizontal span, a vertical span consumes a data cell and discards the content.

_ Draw a single horizontal line in this cell. This consumes a data cell and discards the content. It may also be invoked with -.

= Draw a double horizontal line in this cell. This consumes a data cell and discards the content.

Each cell key may be followed by zero or more of the following case-insensitive modifiers:

**b** Use a bold font for the contents of this cell.

**d** Move content down to the last row of this vertical span. Currently ignored.

**e** Make this column wider to match the maximum width of any other column also having the **e** modifier.

**f** The next one or two characters select the font to use for this cell. One-character font names must be followed by a blank or period. See the roff(7) manual for supported font names.

**i** Use an italic font for the contents of this cell.

**m** Specify a cell start macro. This is a GNU extension and currently unsupported.

**p** Set the point size to the following unsigned argument, or change it by the following signed argument. Currently ignored.

**v** Set the vertical line spacing to the following unsigned argument, or change it by the following signed argument. Currently ignored.

**t** Do not vertically center content in this vertical span, leave it in the top row. Currently ignored.

**u** Move cell content up by half a table row. Currently ignored.

**w** Specify a minimum column width.

**x** After determining the width of all other columns, distribute the rest of the line length among all columns having the **x** modifier.

**z** Do not use this cell for determining the width of this column.

**|** Draw a single vertical line to the right of this cell.

**||** Draw a double vertical line to the right of this cell.

If a modifier consists of decimal digits, it specifies a minimum spacing in units of **n** between this column and the next column to the right. The default is 3. If there is a vertical line, it is drawn inside the spacing.

### Data

The data section follows the last *Layout* line. Each data line consists of one or more data cells, delimited by **tab** characters.

If a data cell contains only the two bytes '\^', the cell above spans to this row, as if the layout specification of this cell were ^.

If a data cell contains only the single character '_' or '=', a single or double horizontal line is drawn across the cell, joining its neighbours. If a data cell contains only the two character sequence '_' or '\=', a single or double horizontal line is drawn inside the cell, not joining its neighbours. If a data line contains nothing but the single character '_' or '=', a horizontal line across the whole table is inserted without consuming a layout row.

In place of any data cell, a text block can be used. It starts with **T{** at the end of a physical input line. Input line breaks inside the text block neither end the text block nor its data cell. It only ends if **T}** occurs at the beginning of a physical input line and is followed by an end-of-cell indicator. If the **T}** is followed by the end of the physical input line, the text block, the data cell, and the data line ends at this point. If the **T}** is followed by the **tab** character, only the text block and the data cell end, but the data line continues with the data cell following the **tab** character. If **T}** is followed by any other character, it does not end the text block, which instead continues to the following physical input line.

## EXAMPLES

String justification and font selection:

```
.TS
rb c  lb
r  ci l.
r           center     l
ri          ce         le
right       c          left
.TE
```

**r**center**l**
ri *ce* le
right *c* left

Some ports in OpenBSD 6.1 to show number alignment and line drawing:

```
.TS
box tab(:);
```

```
r| l
r  n.
software:version
_
AFL:2.39b
Mutt:1.8.0
Ruby:1.8.7.374
TeX Live:2015
.TE

+------------+-------------+
|  software|version  |
+------------+-------------+
|       AFL      2.39b |
|       Mutt   1.8.0   |
|      Ruby1.8.7.374 |
|TeX       2015       |
|Live                 |
+-------------------------+
```

Spans and skipping width calculations:

```
.TS
box tab(:);
lz  s | rt
lt| cb| ^
^ | rz  s.
left:r
l:center:
:right
.TE

+-----------+---+
|left      |r |
|l|center| |
| |    right |
+-+-------------+
```

Text blocks, specifying spacings and specifying and equalizing column widths, putting lines into individual cells, and overriding **allbox**:

```
.TS
allbox tab(:);
le le||7 lw10.
The fourth line:_:line 1
of this column:=:line 2
determines:_:line 3
the column width.:T{
This text is too wide to fit into a column of width 17.
T}:line 4
T{
No break here.
T}::line 5
.TE
```

```
+------------------------------+------------------------------++----------------+
|The fourth                    +------------------------------+|line            |
|line                          +-                            -+|1               |
+------------------------------+------------------------------++----------------+
|of this                       +==============================+|line            |
|column                        +=                            =+|2               |
+------------------------------+------------------------------++----------------+
|determines                    |------------------------------||line            |
|                              |                               ||3               |
+------------------------------+------------------------------++----------------+
|the column                    |This text is too wide to       ||line            |
|width.                        |fit into a column of           ||4               |
|                              |width 17.                      ||                |
+------------------------------+------------------------------++----------------+
|No break                      |                               ||line            |
|here.                         |                               ||5               |
+------------------------------+------------------------------++----------------+
```

These examples were constructed to demonstrate many **tbl** features in a compact way. In real manual pages, keep tables as simple as possible. They usually look better, are less fragile, and are more portable.

## COMPATIBILITY

The mandoc(1) implementation of **tbl** doesn’t support mdoc(7) and man(7) macros and eqn(7) equations inside tables.

## SEE ALSO

mandoc(1), man(7), mandoc_char(7), mdoc(7), roff(7)

M. E. Lesk, *Tbl -- A Program to Format Tables*, June 11, 1976.

## HISTORY

The tbl utility, a preprocessor for troff, was originally written by M. E. Lesk at Bell Labs in 1975. The GNU reimplementation of tbl, part of the groff package, was released in 1990 by James Clark. A standalone tbl implementation was written by Kristaps Dzonsons in 2010. This formed the basis of the implementation that first appeared in OpenBSD 4.9 as a part of the mandoc(1) utility.

## AUTHORS

This **tbl** reference was written by Kristaps Dzonsons <*kristaps@bsd.lv*> and
Ingo Schwarze <*schwarze@openbsd.org*>.

## BUGS

In **-T utf8** output mode, heavy lines are drawn instead of double lines. This cannot be improved because the Unicode standard only provides an incomplete set of box drawing characters with double lines, whereas it provides a full set of box drawing characters with heavy lines. It is unlikely this can be improved in the future because the box drawing characters are already marked in Unicode as characters intended only for backward compatibility with legacy systems, and their use is not encouraged. So it seems unlikely that the missing ones might get added in the future.