Name

tbl − prepare tables for *groff* documents

Synopsis

tbl [**−C**] [*file* . . .]

tbl −−help

tbl −v
tbl −−version

Description

The GNU implementation of *tbl* is part of the *groff*(1) document formatting system. *tbl* is a *troff*(1) preprocessor that translates descriptions of tables embedded in *roff*(7) input files into the language understood by *troff*. It copies the contents of each *file* to the standard output stream, except that lines between **.TS** and **.TE** are interpreted as table descriptions. While GNU *tbl*'s input syntax is highly compatible with AT&T *tbl*, the output GNU *tbl* produces cannot be processed by AT&T *troff*; GNU *troff* (or a *troff* implementing any GNU extensions employed) must be used. Normally, *tbl* is not executed directly by the user, but invoked by specifying the **−t** option to *groff*(1). If no *file* operands are given on the command line, or if *file* is "**−**", *tbl* reads the standard input stream.

Overview

tbl expects to find table descriptions between input lines that begin with **.TS** (table start) and **.TE** (table end). Each such *table region* encloses one or more table descriptions. Within a table region, table descriptions beyond the first must each be preceded by an input line beginning with **.T&**. This mechanism does not start a new table region; all table descriptions are treated as part of their **.TS/.TE** enclosure, even if they are boxed or have column headings that repeat on subsequent pages (see below).

(Experienced *roff* users should observe that *tbl* is not a *roff* language interpreter: the default control character must be used, and no spaces or tabs are permitted between the control character and the macro name. These *tbl* input tokens remain as-is in the output, where they become ordinary macro calls. Macro packages often define **TS**, **T&**, and **TE** macros to handle issues of table placement on the page. *tbl* produces *groff* code to define these macros as empty if their definitions do not exist when the formatter encounters a table region.)

Each table region may begin with *region options,* and must contain one or more *table definitions;* each table definition contains a *format specification* followed by one or more input lines (rows) of *entries.* These entries comprise the *table data.*

Region options

The line immediately following the **.TS** token may specify region options, keywords that influence the interpretation or rendering of the region as a whole or all table entries within it indiscriminately. They must be separated by commas, spaces, or tabs. Those that require a parenthesized argument permit spaces and tabs between the option's name and the opening parenthesis. Options accumulate and cannot be unset within a region once declared; if an option that takes a parameter is repeated, the last occurrence controls. If present, the set of region options must be terminated with a semicolon (**;**).

Any of the **allbox**, **box**, **doublebox**, **frame**, and **doubleframe** region options makes a table "boxed" for the purpose of later discussion.

allbox Enclose each table entry in a box; implies **box**.

box Enclose the entire table region in a box. As a GNU extension, the alternative option name **frame** is also recognized.

center Center the table region with respect to the current indentation and line length; the default is to left-align it. As a GNU extension, the alternative option name **centre** is also recognized.

decimalpoint(*c***)**
Recognize character *c* as the decimal separator in columns using the **N** (numeric) classifier (see subsection "Column classifiers" below). This is a GNU extension.

delim(*xy*)
Recognize characters *x* and *y* as start and end delimiters, respectively, for *eqn*(1) input, and ignore input between them. *x* and *y* need not be distinct.

doublebox
Enclose the entire table region in a double box; implies **box**. As a GNU extension, the alternative option name **doubleframe** is also recognized.

expand
Spread the table horizontally to fill the available space (line length minus indentation) by increasing column separation. Ordinarily, a table is made only as wide as necessary to accommodate the widths of its entries and its column separations (whether specified or default). When **expand** applies to a table that exceeds the available horizontal space, column separation is reduced as far as necessary (even to zero). *tbl* produces *groff* input that issues a diagnostic if such compression occurs. The column modifier **x** (see below) overrides this option.

linesize(*n*)
Draw lines or rules (e.g., from **box**) with a thickness of *n* points. The default is the current type size when the region begins. This option is ignored on terminal devices.

nokeep Don't use *roff* diversions to manage page breaks. Normally, *tbl* employs them to avoid breaking a page within a table row. This usage can sometimes interact badly with macro packages' own use of diversions—when footnotes, for example, are employed. This is a GNU extension.

nospaces
Ignore leading and trailing spaces in table entries. This is a GNU extension.

nowarn
Suppress diagnostic messages produced at document formatting time when the line or page lengths are inadequate to contain a table row. This is a GNU extension.

tab(*c*) Use the character *c* instead of a tab to separate entries in a row of table data.

Table format specification

The table format specification is mandatory: it determines the number of columns in the table and directs how the entries within it are to be typeset. The format specification is a series of column *descriptors.* Each descriptor encodes a *classifier* followed by zero or more *modifiers.* Classifiers are letters (recognized case-insensitively) or punctuation symbols; modifiers consist of or begin with letters or numerals. Spaces, tabs, newlines, and commas separate descriptors. Newlines and commas are special; they apply the descriptors following them to a subsequent row of the table. (This enables column headings to be centered or emboldened while the table entries for the data are not, for instance.) We term the resulting group of column descriptors a *row definition.* Within a row definition, separation between column descriptors (by spaces or tabs) is often optional; only some modifiers, described below, make separation necessary.

Each column descriptor begins with a mandatory *classifier,* a character that selects from one of several arrangements. Some determine the positioning of table entries within a rectangular cell: centered, left-aligned, numeric (aligned to a configurable decimal separator), and so on. Others perform special operations like drawing lines or spanning entries from adjacent cells in the table. Except for "|", any classifier can be followed by one or more *modifiers;* some of these accept an argument, which in GNU *tbl* can be parenthesized. Modifiers select fonts, set the type size, and perform other tasks described below.

The format specification can occupy multiple input lines, but must conclude with a dot "**.**" followed by a newline. Each row definition is applied in turn to one row of the table. The last row definition is applied to rows of table data in excess of the row definitions.

For clarity in this document's examples, we shall write classifiers in uppercase and modifiers in lowercase. Thus, "**CbCb,LR.**" defines two rows of two columns. The first row's entries are centered and boldfaced; the second and any further rows' first and second columns are left- and right-aligned, respectively.

The row definition with the most column descriptors determines the number of columns in the table; any row definition with fewer is implicitly extended on the right-hand side with **L** classifiers as many times as necessary to make the table rectangular.

Column classifiers

The **L**, **R**, and **C** classifiers are the easiest to understand and use.

A, **a** Center longest entry in this column, left-align remaining entries in the column with respect to the centered entry, then indent all entries by one en. Such "alphabetic" entries (hence the name of the classifier) can be used in the same column as **L**-classified entries, as in "**LL,AR.**". The **A** entries are often termed "sub-columns" due to their indentation.

C, **c** Center entry within the column.

L, **l** Left-align entry within the column.

N, **n** Numerically align entry in the column. *tbl* aligns columns of numbers vertically at the units place. If multiple decimal separators are adjacent to a digit, it uses the rightmost one for vertical alignment. If there is no decimal separator, the rightmost digit is used for vertical alignment; otherwise, *tbl* centers the entry within the column. The *roff* dummy character **\&** in an entry marks the glyph preceding it (if any) as the units place; if multiple instances occur in the data, the leftmost is used for alignment.

If **N**-classified entries share a column with **L** or **R** entries, *tbl* centers the widest **N** entry with respect to the widest **L** or **R** entry, preserving the alignment of **N** entries with respect to each other.

The appearance of *eqn* equations within **N**-classified columns can be troublesome due to the foregoing textual scan for a decimal separator. Use the **delim** region option to make *tbl* ignore the data within *eqn* delimiters for that purpose.

R, **r** Right-align entry within the column.

S, **s** Span previous entry on the left into this column.

^ Span entry in the same column from the previous row into this row.

_, **−** Replace table entry with a horizontal rule. An empty table entry is expected to correspond to this classifier; if data are found there, *tbl* issues a diagnostic message.

= Replace table entry with a double horizontal rule. An empty table entry is expected to correspond to this classifier; if data are found there, *tbl* issues a diagnostic message.

| Place a vertical rule (line) on the corresponding row of the table (if two of these are adjacent, a double vertical rule). This classifier does not contribute to the column count and no table entries correspond to it. A **|** to the left of the first column descriptor or to the right of the last one produces a vertical rule at the edge of the table; these are redundant (and ignored) in boxed tables.

To change the table format within a *tbl* region, use the **.T&** token at the start of a line. It is followed by a format specification and table data, but *not* region options. The quantity of columns in a new table format thus introduced cannot increase relative to the previous table format; in that case, you must end the table region and start another. If that will not serve because the region uses box options or the columns align in an undesirable manner, you must design the initial table format specification to include the maximum quantity of columns required, and use the **S** horizontal spanning classifier where necessary to achieve the desired columnar alignment.

Attempting to horizontally span in the first column or vertically span on the first row is an error. Non-rectangular span areas are also not supported.

Column modifiers

Any number of modifiers can follow a column classifier. Arguments to modifiers, where accepted, are case-sensitive. If the same modifier is applied to a column specifier more than once, or if conflicting modifiers are applied, only the last occurrence has effect. The modifier **x** is mutually exclusive with **e** and **w**, but **e** is not mutually exclusive with **w**; if these are used in combination, **x** unsets both **e** and **w**, while either **e** or **w** overrides **x**.

b, **B** Typeset entry in boldface, abbreviating **f(B)**.

d, **D** Align a vertically spanned table entry to the bottom ("down"), instead of the center, of its range. This is a GNU extension.

e, **E** Equalize the widths of columns with this modifier. The column with the largest width controls. This modifier sets the default line length used in a text block.

f, **F** Select the typeface for the table entry. This modifier must be followed by a font or style name (one or two characters not starting with a digit), font mounting position (a single digit), or a name or mounting position of any length in parentheses. The last form is a GNU extension. (The parameter corresponds to that accepted by the *troff* **ft** request.) A one-character argument not in parentheses must be separated by one or more spaces or tabs from what follows.

i, **I** Typeset entry in an oblique or italic face, abbreviating **f(I)**.

m, **M** Call a *groff* macro before typesetting a text block (see subsection "Text blocks" below). This is a GNU extension. This modifier must be followed by a macro name of one or two characters or a name of any length in parentheses. A one-character macro name not in parentheses must be separated by one or more spaces or tabs from what follows. The named macro must be defined before the table region containing this column modifier is encountered. The macro should contain only simple *groff* requests to change text formatting, like adjustment or hyphenation. The macro is called *after* the column modifiers **b**, **f**, **i**, **p**, and **v** take effect; it can thus override other column modifiers.

p, **P** Set the type size for the table entry. This modifier must be followed by an integer *n* with an optional leading sign. If unsigned, the type size is set to *n* scaled points. Otherwise, the type size is incremented or decremented per the sign by *n* scaled points. The use of a signed multi-digit number is a GNU extension. (The parameter corresponds to that accepted by the *troff* **ps** request.) If a type size modifier is followed by a column separation modifier (see below), they must be separated by at least one space or tab.

t, **T** Align a vertically spanned table entry to the top, instead of the center, of its range.

u, **U** Move the column up one half-line, "staggering" the rows. This is a GNU extension.

v, **V** Set the vertical spacing to be used in a text block. This modifier must be followed by an integer *n* with an optional leading sign. If unsigned, the vertical spacing is set to *n* points. Otherwise, the vertical spacing is incremented or decremented per the sign by *n* points. The use of a signed multi-digit number is a GNU extension. (This parameter corresponds to that accepted by the *troff* **vs** request.) If a vertical spacing modifier is followed by a column separation modifier (see below), they must be separated by at least one space or tab.

w, **W** Set the column's minimum width. This modifier must be followed by a number, which is either a unitless integer, or a *roff* horizontal measurement in parentheses. Parentheses are required if the width is to be followed immediately by an explicit column separation (alternatively, follow the width with one or more spaces or tabs). If no unit is specified, ens are assumed. This modifier sets the default line length used in a text block.

x, **X** Expand the column. After computing the column widths, distribute any remaining line length evenly over all columns bearing this modifier. Applying the **x** modifier to more than one column is a GNU extension. This modifier sets the default line length used in a text block.

z, **Z** Ignore the table entries corresponding to this column for width calculation purposes; that is, compute the column's width using only the information in its descriptor.

n A numeric suffix on a column descriptor sets the separation distance (in ens) from the succeeding column; the default separation is **3n**. This separation is proportionally multiplied if the **expand** region option is in effect; in the case of tables wider than the output line length, this separation might be zero. A negative separation cannot be specified. A separation amount after the last column in a row is nonsensical and provokes a diagnostic from *tbl*.

Table data

The table data come after the format specification. Each input line corresponds to a table row, except that a backslash at the end of a line of table data continues an entry on the next input line. (Text blocks, discussed below, also spread table entries across multiple input lines.) Table entries within a row are separated in the input by a tab character by default; see the **tab** region option above. Excess entries in a row of table data (those that have no corresponding column descriptor, not even an implicit one arising from rectangularization of the table) are discarded with a diagnostic message. *roff* control lines are accepted between rows of table data and within text blocks. If you wish to visibly mark an empty table entry in the document source, populate it with the **\&** *roff* dummy character. The table data are interrupted by a line consisting of the **.T&** input token, and conclude with the line **.TE**.

Ordinarily, a table entry is typeset rigidly. It is not filled, broken, hyphenated, adjusted, or populated with additional inter-sentence space. *tbl* instructs the formatter to measure each table entry as it occurs in the input, updating the width required by its corresponding column. If the **z** modifier applies to the column, this measurement is ignored; if **w** applies and its argument is larger than this width, that argument is used instead. In contrast to conventional *roff* input (within a paragraph, say), changes to text formatting, such as font selection or vertical spacing, do not persist between entries.

Several forms of table entry are interpreted specially.

- If a table row contains only an underscore or equals sign (**_** or **=**), a single or double horizontal rule (line), respectively, is drawn across the table at that point.
- A table entry containing only **_** or **=** on an otherwise populated row is replaced by a single or double horizontal rule, respectively, joining its neighbors.
- Prefixing a lone underscore or equals sign with a backslash also has meaning. If a table entry consists only of **_** or **\=** on an otherwise populated row, it is replaced by a single or double horizontal rule, respectively, that does *not* (quite) join its neighbors.
- A table entry consisting of **\R***x*, where *x* is any *roff* ordinary or special character, is replaced by enough repetitions of the glyph corresponding to *x* to fill the column, albeit without joining its neighbors.
- On any row but the first, a table entry of **\^** causes the entry above it to span down into the current one.

On occasion, these special tokens may be required as literal table data. To use either **_** or **=** literally and alone in an entry, prefix or suffix it with the *roff* dummy character **\&**. To express **_**, **\=**, or **\R**, use a *roff* escape sequence to interpolate the backslash (**\e** or **\[rs]**). A reliable way to emplace the **\^** glyph sequence within a table entry is to use a pair of *groff* special character escape sequences (**\[rs]\[ha]**).

Rows of table entries can be interleaved with *groff* control lines; these do not count as table data. On such lines the default control character (**.**) must be used (and not changed); the no-break control character is not recognized. To start the first table entry in a row with a dot, precede it with the *roff* dummy character **\&**.

Text blocks

An ordinary table entry's contents can make a column, and therefore the table, excessively wide; the table then exceeds the line length of the page, and becomes ugly or is exposed to truncation by the output device. When a table entry requires more conventional typesetting, breaking across more than one output line (and thereby increasing the height of its row), it can be placed within a *text block.*

tbl interprets a table entry beginning with "**T{**" at the end of an input line not as table data, but as a token starting a text block. Similarly, "**T}**" at the start of an input line ends a text block; it must also end the table entry. Text block tokens can share an input line with other table data (preceding **T{** and following **T}**). Input lines between these tokens are formatted in a diversion by *troff*. Text blocks cannot be nested. Multiple text blocks can occur in a table row.

Text blocks are formatted as was the text prior to the table, modified by applicable column descriptors. Specifically, the classifiers **A**, **C**, **L**, **N**, **R**, and **S** determine a text block's *alignment* within its cell, but not its *adjustment.* Add **na** or **ad** requests to the beginning of a text block to alter its adjustment distinctly from other text in the document. As with other table entries, when a text block ends, any alterations to formatting parameters are discarded. They do not affect subsequent table entries, not even other text blocks.

If **w** or **x** modifiers are not specified for *all* columns of a text block's span, the default length of the text block (more precisely, the line length used to process the text block diversion) is computed as $L \times C/(N+1)$, where L is the current line length, C the number of columns spanned by the text block, and N the number of columns in the table. If necessary, you can also control a text block's width by including an **ll** (line length) request in it prior to any text to be formatted. Because a diversion is used to format the text block, its height and width are subsequently available in the registers **dn** and **dl**, respectively.

***roff* interface**

The register **TW** stores the width of the table region in basic units; it can't be used within the region itself, but is defined before the **.TE** token is output so that a *groff* macro named **TE** can make use of it. **T.** is a Boolean-valued register indicating whether the bottom of the table is being processed. The **#T** register marks the top of the table. Avoid using these names for any other purpose.

tbl also defines a macro **T#** to produce the bottom and side lines of a boxed table. While *tbl* itself arranges for the output to include a call of this macro at the end of such a table, it can also be used by macro packages to create boxes for multi-page tables by calling it from a page footer macro that is itself called by a trap planted near the bottom of the page. See section "Limitations" below for more on multi-page tables.

GNU *tbl* internally employs register, string, macro, and diversion names beginning with the numeral **3**. A document to be preprocessed with GNU *tbl* should not use any such identifiers.

Interaction with *eqn*

tbl should always be called before *eqn*(1). (*groff*(1) automatically arranges preprocessors in the correct order.) Don't call the **EQ** and **EN** macros within tables; instead, set up delimiters in your *eqn* input and use the **delim** region option so that *tbl* will recognize them.

GNU *tbl* enhancements

In addition to extensions noted above, GNU *tbl* removes constraints endured by users of AT&T *tbl*.

- Region options can be specified in any lettercase.
- There is no limit on the number of columns in a table, regardless of their classification, nor any limit on the number of text blocks.
- All table rows are considered when deciding column widths, not just those occurring in the first 200 input lines of a region. Similarly, table continuation (**.T&**) tokens are recognized outside a region's first 200 input lines.
- Numeric and alphabetic entries may appear in the same column.
- Numeric and alphabetic entries may span horizontally.

Using GNU *tbl* within macros

You can embed a table region inside a macro definition. However, since *tbl* writes its own macro definitions at the beginning of each table region, it is necessary to call end macros instead of ending macro definitions with "**..**". Additionally, the escape character must be disabled.

Not all *tbl* features can be exercised from such macros because *tbl* is a *roff* preprocessor: it sees the input earlier than *troff* does. For example, vertically aligning decimal separators fails if the numbers containing them occur as macro or string parameters; the alignment is performed by *tbl* itself, which sees only **\$1**, **\$2**, and so on, and therefore can't recognize a decimal separator that only appears later when *troff* interpolates a macro or string definition.

Using *tbl* macros within conditional input (that is, contingent upon an **if**, **ie**, **el**, or **while** request) can result in misleading line numbers in subsequent diagnostics. *tbl* unconditionally injects its output into the source document, but the conditional branch containing it may not be taken, and if it is not, the **lf** requests that *tbl* injects to restore the source line number cannot take effect. Consider copying the input line counter register **c.** and restoring its value at a convenient location after applicable arithmetic.

Options

−−help displays a usage message, while **−v** and **−−version** show version information; all exit afterward.

−C Enable AT&T compatibility mode: recognize **.TS** and **.TE** even when followed by a character other than space or newline. Furthermore, interpret the uninterpreted leader escape sequence **\a**.

Limitations

Multi-page tables, if boxed and/or if you want their column headings repeated after page breaks, require support at the time the document is formatted. A convention for such support has arisen in macro packages such as *ms*, *mm*, and *me*. To use it, follow the **.TS** token with a space and then "**H**"; this will be interpreted by the formatter as a **TS** macro call with an **H** argument. Then, within the table data, call the **TH** macro; this informs the macro package where the headings end. If your table has no such heading rows, or you do not desire their repetition, call **TH** immediately after the table format specification. If a multi-page table is boxed or has repeating column headings, do not enclose it with keep/release macros, or divert it in any other way. Further, the **bp** request will not cause a page break in a "**TS H**" table. Define a macro to wrap **bp**: invoke it normally if there is no current diversion. Otherwise, pass the macro call to the enclosing diversion using the transparent line escape sequence **\!**; this will "bubble up" the page break to the output device. See section "Examples" below for a demonstration.

Double horizontal rules are not supported by *grotty*(1); single rules are used instead. *grotty* also ignores half-line motions, so the **u** column modifier has no effect. On terminal devices ("*nroff* mode"), horizontal rules and box borders occupy a full vee of space; this amount is doubled for **doublebox** tables. Tables using these features thus require more vertical space in *nroff* mode than in *troff* mode: write **ne** requests accordingly. Vertical rules between columns are drawn in the space between columns in *nroff* mode; using double vertical rules and/or reducing the column separation below the default can make them ugly or overstrike them with table data.

A text block within a table must be able to fit on one page.

Using **\a** to put leaders in table entries does not work in GNU *tbl*, except in compatibility mode. This is correct behavior: **\a** is an *uninterpreted* leader. You can still use the *roff* leader character (Control+A) or define a string to use **\a** as it was designed: to be interpreted only in copy mode.

```
.ds a \a
.TS
box center tab(;);
Lw(2i)0 L.
Population\*a;6,327,119
.TE
```

Population.....6,327,119

A leading and/or trailing **|** in a format specification, such as "**|LCR|.**", produces an en space between the vertical rules and the content of the adjacent columns. If no such space is desired (so that the rule abuts the content), you can introduce "dummy" columns with zero separation and empty corresponding table entries before and/or after.

```
.TS
center tab(#);
R0|L C R0|L.
_
#levulose#glucose#dextrose#
_
.TE
```

These dummy columns have zero width and are therefore invisible; unfortunately they usually don't work as intended on terminal devices.

levulose	glucose	dextrose

Examples

It can be easier to acquire the language of *tbl* through examples than formal description, especially at first.

```
.TS
box center tab(#);
Cb Cb
L L.
Ability#Application
Strength#crushes a tomato
Dexterity#dodges a thrown tomato
Constitution#eats a month-old tomato without becoming ill
Intelligence#knows that a tomato is a fruit
Wisdom#chooses \f[I]not\f[] to put tomato in a fruit salad
Charisma#sells obligate carnivores tomato-based fruit salads
.TE
```

Ability	Application
Strength	crushes a tomato
Dexterity	dodges a thrown tomato
Constitution	eats a month-old tomato without becoming ill
Intelligence	knows that a tomato is a fruit
Wisdom	chooses *not* to put tomato in a fruit salad
Charisma	sells obligate carnivores tomato-based fruit salads

The **A** and **N** column classifiers can be easier to grasp in visual rendering than in description.

```
.TS
center tab(;);
CbS,LN,AN.
Daily energy intake (in MJ)
Macronutrients
.\" assume 3 significant figures of precision
Carbohydrates;4.5
Fats;2.25
Protein;3
.T&
LN,AN.
Mineral
Pu-239;14.6
_
.T&
LN.
Total;\[ti]24.4
.TE
```

Daily energy intake (in MJ)	
Macronutrients	
Carbohydrates	4.5
Fats	2.25
Protein	3
Mineral	
Pu-239	14.6
Total	~24.4

Next, we'll lightly adapt a compact presentation of spanning, vertical alignment, and zero-width column modifiers from the *mandoc* reference for its *tbl* interpreter. It rewards close study.

```
.TS
box center tab(:);
Lz  S | Rt
Ld| Cb| ^
^ | Rz  S.
left:r
l:center:
:right
.TE
```

left		r
	center	
l		right

Row staggering is not visually achievable on terminal devices, but a table using it can remain comprehensible nonetheless.

```
.TS
center tab(|);
Cf(BI) Cf(BI) Cf(B), C C Cu.
n|n\f[B]\[tmu]\f[]n|difference
1|1
2|4|3
3|9|5
4|16|7
5|25|9
6|36|11
.TE
```

n	***n*×*n***	**difference**
1	1	
2	4	3
3	9	5
4	16	7
5	25	9
6	36	11

Some *tbl* features cannot be illustrated in the limited environment of a portable man page.

We can define a macro outside of a *tbl* region that we can call from within it to cause a page break inside a multi-page boxed table. You can choose a different name; be sure to change both occurrences of "BP".

```
.de BP
.  ie '\\n(.z'' .bp \\$1
.  el \!.BP \\$1
..
```

See also

"Tbl—A Program to Format Tables", by M. E. Lesk, 1976 (revised 16 January 1979), AT&T Bell Laboratories Computing Science Technical Report No. 49.

The spanning example above was taken from *mandoc*'s man page for its *tbl* implementation ⟨https://man.openbsd.org/tbl.7⟩.

groff (1), *troff* (1)