Vertical alignments in plain TEX

Udo Wermuth

Abstract

Alignment structures are created in plain TEX using one of three methods: tabbing, horizontal or vertical alignment. Tabbing might be the easiest method; horizontal alignment with `\halign` has more complicated instructions. Both methods are described in great detail in Chapter 22 of *The TEXbook*. Vertical alignment with `\valign` is only briefly mentioned in *The TEXbook*; this might be the reason why it's rarely used. This article looks at `\valign` and presents a few tables that might benefit from it.

1 Introduction

The TEXbook [4] not only contains an in-depth description of plain TEX, its author Donald E. Knuth also writes in a style that is often entertaining. Or in his own words from page iv: "Computer system manuals usually make dull reading, but take heart: This one contains JOKES every once in a while, so you might actually enjoy reading it. (However, most of the jokes can only be appreciated properly if you understand a technical point that is being made—so read *carefully*.)" So you directly get a taste of his humor. But it might not always be obvious if he is joking. Neither the index nor exercise 27.5 helps us; we must find the jokes through our own reading.

One famous sentence that some readers might classify as a joke appears on page 249 in the *The TEXbook*: "People usually work with TEX at least a year before they find their first application for `\valign`; and then it's usually a one-row '`\valign {\vfil#\vfil\cr...}`'." In [3] T. Hoekwater comments on this statement: "Well, it took me a lot longer than that: nearly three decades, in fact." And M. Clark makes a similar judgment about Knuth's assessment in [2, p. 231]: "This might even be a conservative estimate. The only examples I have ever seen were rather forced (and could probably have been done another way)." (See also the appendix.)

This article looks at this rarely used command `\valign` [4, p. 249, p. 285]. It's implemented in the program TEX, i.e., it is one of TEX's *primitives*. Knuth programmed `\valign` and its sibling `\halign` to create vertical and horizontal alignments, respectively. No reader can identify the used primitive only from a typeset result: All alignments contain rows and columns and might have horizontal and/or vertical rules. With both commands one can create the image of joined cells across columns or rows. The only difference that I see between the two primitives is the amount of input needed to realize a certain output. Clark's statement seems to be correct and is valid for both. This text presents nine alignments with `\valign`. I find it not too difficult to realize them with `\halign` although the solutions are less pleasant to code and need sometimes more code.

Contents. The text expects that the reader is familiar with horizontal alignments created with the primitive `\halign`: Section 2 explains `\valign` for someone who knows `\halign`. Section 3 introduces a few macros that I find useful when I work with `\valign`; they are applied in sections 4–7. Next, in section 4 we look at three tables that seem to be more easily realized with `\valign` than `\halign`. Section 5 contains a table with spanned rows; a simple thing for `\valign`. The topic of section 6 is ruled tables. In section 7 `\halign` and `\valign` work together to build alignments. Finally, in section 8, I briefly mention unusual applications of `\valign`.

2 From horizontal to vertical alignment

TEX provides the primitive `\valign` to create vertical alignments. There is a close relationship between `\halign` and `\valign`. To describe what the latter primitive does, one can take the description of the former and exchange "horizontal" with "vertical" and "row" with "column" if the last two refer to the typeset output. For example, the sentence

> The primitive `\halign` is a vertical command that expects horizontal material in its entries: Each input row stands for the column entries in one row of the alignment.

becomes

> The primitive `\valign` is a horizontal command that expects vertical material in its entries: Each input row stands for the row entries in one column of the alignment.

These exchanges describe most aspects that one has to know to work with `\valign`. But the details are more complex and the consequences of this transposition should be explicitly stated at least once.

1. `\valign{...}` is a horizontal command: The command can be used inside a paragraph.

⇒ As a consequence the command starts a paragraph; thus TEX indents the alignment. One must start with '`\noindent\valign`' to get an alignment that's allowed to cover the full page width.

It also means that `\leftskip` & `\rightskip`, `\hangindent` & `\hangafter`, and `\parshape` influence the position of the whole alignment.

2. TEX rejects `\valign` in math modes. Thus, it can't be used for *alignment displays* ([4, p. 190]).

doi.org/10.47397/tb/44-3/tb138wermuth-valign

3. `\valign` has the same two-part structure as `\halign`: After a mandatory *preamble* with at least one *template*, any number of input rows with cell entries can follow. The number of templates must not be less than the number of entries in any input row.
⇒ The templates are now applied to the rows of the typeset columns.

4. '`#`', '`&`', `\cr`, and `\crcr` are used as in `\halign`. The last three don't structure the current alignment if they appear after a yet unmatched '`{`' inside `\valign`'s braces. (1) A `\cr`/`\crcr` ends the preamble and each input row, templates and cell entries are separated by the *alignment tab* '`&`', and the *parameter symbol* '`#`' defines the unique place in a template to insert a cell entry. (2) A *periodic preamble* has an additional '`&`' in front of one template: TeX repeatedly reuses the templates up to the preamble's end if it needs more for an input row. (3) Starting with the preamble's end, TeX expands the token after each (i) `\cr`/`\crcr` and (ii) '`&`' until it finds an unexpandable one. TeX checks if `\omit` or `\noalign` (only with case (i)) follows. (See nos. 8 and 10.)

5. The cell entries expect vertical material: Imagine that TeX builds a group through a `\vbox` with depth 0 pt. The vboxes are put together horizontally.
⇒ One consequence is that a letter (and other items; see [4, p. 283]) that starts a cell entry switches to horizontal mode, i.e., TeX starts a paragraph. As its width TeX uses the current width of the context, i.e., usually the full `\hsize`. Thus, with more than one column the `\hsize` must be changed locally.

A longer text should be placed in a `\vtop`, in which the `\hsize` might be reduced.

Single lines with horizontal material for the row entries should use an `\hbox`. This allows setting the column width using the keyword '`to`'; see no. 6.

In the case of entries consisting simply of hboxes that form the column we need `\strut` commands in each `\hbox`. Otherwise the distances between the hboxes might become unequal because TeX doesn't apply *interline glue* [4, p. 78] as in a paragraph.

6. In a `\valign` TeX computes and unifies the row heights, not the column widths.
⇒ The column widths must be given to TeX by other means if the text should be centered or set flush right. Using an `\hbox`, as suggested in no. 5, one could set the width of the column in the template. For example, use a construction like '`\hbox to \columnwidth`' in which the dimension register `\columnwidth` is set to an appropriate value.

7. `\tabskip` now specifies the distance between the rows. Its value outside of `\valign` creates white space above the table. Its last value adds white space after the alignment. As in an `\halign` the value must be changed inside the preamble.
⇒ However, it's a standard glue specification, not comparable to interline glue. (See also no. 5.)

8. `\omit` at the start of a cell entry works as before: TeX ignores the associated template from the preamble and uses '`#`' instead.

9. `\span` works as before: In the preamble it expands the following token; in input rows for cells it replaces '`&`' and combines the cells. Up to 255 `\span` commands can be used in sequence between cells.
⇒ Of course, in a `\valign` rows are spanned. The macro `\multispan` [4, p. 243] can be used if several row entries in a typeset column have to be combined.

10. `\noalign{...}` is used to insert horizontal material after a column of the alignment; it's allowed after a `\cr`/`\crcr` or another `\noalign`. A `\noalign` after the preamble's end puts its material to the left of the table. After the '`}`' TeX expands the next token to check for `\omit` or `\noalign`; cf. no. 4(3).
⇒ It can be used to specify the horizontal distance between the columns of the alignment comparable to `\tabskip` in an `\halign`. But there is no automation: TeX must get a value after each `\cr`.

11. `\everycr` works as before: Its tokens are inserted after every `\cr` and after a `\crcr` that's not immediately preceded by `\cr`, `\crcr`, or `\noalign`.
⇒ An '`\everycr={\noalign{\hskip ...}}`' sets the distance of columns without the need to repeat `\noalign` after every `\cr`. An `\unskip` after the closing brace of the `\valign` eliminates the white space to the right of the table. But see also no. 10.

12. Vertical rules are now easily drawn: '`\noalign {\vrule}`' after a `\cr` puts a rule between two columns. After the `\cr` of the preamble this construction draws a rule to the left of the alignment.
⇒ As vertical rules are nowadays mostly applied to avoid confusion [1, 13.53], the advantage to draw them easily might not be very important.

After the definition of `\vrulefill`—the plain format doesn't define this command—$n \leq 256$ rows receive a vertical rule with the command sequence '`\multispan{`n`}\vrulefill`', i.e., this is the vertical analog of one of the methods to get horizontal rules in `\halign` [4, p. 246]. (See also no. 9.)

Horizontal rules are created by an '`\hrule#`' in a template like vertical rules in `\halign` [4, p. 246]. But there is often a second method if hboxes are used (see no. 5): Use '`\omit\hrule`' in a cell entry.

13. Like `\halign`, `\valign` accepts the keywords '`to`' and '`spread`' followed by a dimension.
⇒ Now these keywords set or change the height of the overall alignment, not its width.

Table 1: TEX's error messages for bad input with commands for vertical alignment

#	**The error is** ...	**Example of user input**	**TEX's (first) error message**
1	a \valign in math mode	$\valign{... or $$\valign{...	Missing $ inserted
... made in the construction of \valign:			
2	a forgotten keyword	\valign 100pt{...	Missing { inserted
3	a forgotten dimension	\valign to {...	Missing number, treated as zero
4	a forgotten {	\valign #...	Missing { inserted
5	a forgotten }	\valign{#\cr X\cr\bye	You can't use '\end' in internal vertical mode
... made in the construction of the preamble:			
6	a missing preamble	\valign{}\bye	Runaway preamble? with } Forbidden control sequence found while scanning preamble of \valign inserted: \cr } read again: \bye
7	a forgotten \cr	\valign{#}\bye	Runaway preamble? with } (see #6)
8	an additional }	\valign{#}\cr X\cr}\bye	Runaway preamble? with }\cr X\cr } \par (#6)
9	an additional {	\valign{#{\cr X\cr}\bye	Runaway preamble? with {\cr X\cr } \par (#6)
10	a forgotten #	\valign{\cr ...	Missing # inserted in alignment preamble
11	a forgotten &	\valign{##\cr ...	Only one # is allowed per tab
12	an incomplete \tabskip	\valign{#\tabskip\cr ...	Missing number, treated as zero
13	a missing unit	\valign{#\tabskip=0\cr ...	Illegal unit of measure (pt inserted)
14	a second && in preamble	\valign{#&&#&&#\cr ...	Missing # inserted in alignment preamble
15	an \omit in preamble	\valign{\omit#\cr x\cr}	Misplaced \omit
16	a \noalign in preamble	\valign{#\noalign...	Misplaced \noalign
... made in the contents of a cell entry:			
17	an unmatched {	\valign{#\cr {X\cr}	Missing } inserted
18	an unmatched }	\valign{#\cr X}\cr}	Missing \cr inserted
19	a # in the entry	\valign{#\cr #...	You can't use 'macro parameter #' in internal vertical mode
20	an additional & or \span	\valign{#\cr X&X\cr ...	Extra alignment tab has been changed to \cr
21	an \omit that's not first	\valign{#\cr X\omit...	Misplaced \omit
22	the use of \noalign	\valign{#\cr X\noalign{}...	Misplaced \noalign
23	the use of \raise or \lower	\valign{#\cr \lower...	You can't use '\lower' in internal vertical mode
24	the use of \spacefactor	\valign{#\cr \showthe\spacefactor...	Improper \spacefactor
25	a badly used \string\cr	\valign{#\cr \tt\string\cr\cr}	Misplaced \cr & Missing \cr inserted (100 times, i.e., an endless loop)
26	the use of the template's end although it's like \outer [4, p. 206]	\valign{#\cr \def\\#1{}\\\cr}	Runaway argument? Forbidden control sequence found while scanning use of \\ inserted: \par read again: \endtemplate
... made in the construction of a \noalign:			
27	a missing {	\valign{#\cr\noalign X}\cr}	Missing { inserted
28	a missing }	\valign{#\cr\noalign{X\cr}	Misplaced \cr
29	a vertical command	\valign{#\cr\noalign{\vss}}	Missing } inserted
30	the use of \moveleft or \moveright	\valign{#\cr \noalign{\moveleft...	You can't use '\moveleft' in restricted horizontal mode
31	the use of \prevdepth	\valign{#\cr \noalign{\showthe\prevdepth...	Improper \prevdepth
... made by an unsuitable \everycr for the \valign; here with the simple \valign{#\cr}:			
32	a missing \noalign	\everycr={\hbox{x}}	Missing \cr inserted (100 times)
33	a recursive \everycr	\everycr={\noalign{x}\cr}	TeX capacity exceeded, sorry
... made by a devious user who deserves no sympathy [4, p. 299]:			
34	the improper nesting of alignments	\valign{\halign#\cr\cr}	Emergency stop (interwoven alignment preambles are not allowed)
35	too many \span in series	\valign{&#\cr\multispan{257}...	This can't happen (256 spans) (a joke?!)

As with \halign [4, p. 238] \tabskip's glue specification (see no. 7) must be able to stretch or shrink to fulfill the request; otherwise TeX responds with an underfull vbox warning and writes a *prototype column* [4, p. 302f.] in the log file.

For example, with '\tabskip=0pt' a '\valign spread 1sp{#\cr \vbox to 1.1\vsize{\vss}\cr}' creates an underfull vbox warning as \tabskip cannot stretch. The overfull vbox is reported at the next page break.

14. With \valign, alignments with overfull typeset lines cause an overfull \hbox warning.
⇒ TeX builds a paragraph and the line-breaking procedure triggers the "Overfull \hbox" warning.

For example, set '\hsize=10pt\tabskip=0pt' and TeX accepts '\halign{&#\cr u&w\cr}'. But it complains about the identical looking output created with '\noindent\valign{#\cr \hbox{u}\cr \hbox{w}\cr}' and prints an overfull rule.

15. Alignments don't require programming in TeX but the rules for their construction must be strictly obeyed. Table 1 presents examples of what can be done wrong with a \valign and how TeX responds.

16. \valign is a horizontal command, so it interacts with the special integer \spacefactor [4, p. 76]; \halign interacts with \prevdepth [4, p. 79f.]. The \spacefactor's value in front of the \valign is used for a \noalign directly after the preamble. After a column the \spacefactor is 1000. The \noalign that follows another \noalign gets the former's final value. At \valign's end the current \spacefactor is used, i.e., either 1000 after a column, a \noalign's final value, or the value from the start if the \valign ends after the preamble. (See [4, p. 286].)

17. To look into a vertical alignment, use the command \showlists [4, p. 88f.], either with the \ddt technique [4, p. 248] or insert, for example, the following macro, say as \showalign1, before the closing brace of the alignment.

```
\def\showalign#1{% #1: value of \showboxdepth
% call it after a \cr; shows unset boxes in log
 \noalign{\scrollmode \showboxbreadth=100
  \showboxdepth=#1 \showlists \errorstopmode}}
```

There is one surprise: Spanned rows (see no. 9) are reported as "columns". This word is hardcoded in §185 of *TeX: The Program* [5].

3 Macros for vertical alignments

From the list in the previous section we must conclude that vertical alignments require some parameters and macros to ease the coding. At least this is necessary if the typeset cell entries are text or numbers. Most of the following code snippets are of general use. Maybe only the last one is more specialized and might need a change if you want to apply its code in other texts.

First we need to control the widths of certain columns; see section 2, no. 6. I declare a dimension register for this task.

```
\newdimen\Vcolumnwidth % set the column width
```

The entries themselves must often use an \hbox of the current width and they should have a \strut; see section 2, no. 5. Therefore, I use the following macros to work with the width. Note, I use short names for the control words as they are often used. They all start with a capital 'V'.

```
\def\VR#1{% #1: text, right aligned
 \hbox to \Vcolumnwidth{\strut\hss#1}}
\def\VC#1{% #1: text, centered
 \hbox to \Vcolumnwidth{\strut\hss#1\hss}}
\def\VL#1{% #1: text, left aligned
 \hbox to \Vcolumnwidth{\strut#1\hss}}
```

Sometimes the width doesn't play a rôle. As the command \omit works as before (see section 2, no. 8) I define another macro that excludes the template. But an hbox and a \strut are still required.

```
\def\Vo#1{% #1: text, output without template
 \omit\hbox{\strut#1}}
```

Besides \vrulefill a macro without an hbox and a strut is needed for horizontal rules; see section 2, no. 12.

```
\def\vrulefill{\leaders\vrule\vfill}% no 'V'
\def\Vh{\omit\hrule}% a horizontal rule
```

To span several columns, for example, in the headers, I use the following macro that uses an hbox whose width is a multiple of \Vcolumnwidth.

```
\def\Vc(#1)#2{% #1: # columns to span; #2: text
 \omit\hbox to 0pt{\hbox to #1\Vcolumnwidth{%
   \hss\strut#2\hss}\hss}}% no effect on 1st col
```

To initialize the dimension \Vcolumnwidth I provide two methods. The first is to specify the value, in "pt" units, directly in a cell entry.

```
\def\VI#1pt{% #1pt: a valid dimension for the
 \global\Vcolumnwidth=#1pt }%    column width
```

The second method is useful for column widths of numeric data. As the digits have all the same width only the maximum number of digits in the column must be given. With a second argument of '1' it can be used for text strings too.

```
\newdimen\Vcolwdmodule % one unit times a factor
\def\Vi(#1){% #1: text; its width is the unit
 {\setbox0=\hbox{#1}%
  \global\Vcolwdmodule=\the\wd0}}
\def\VII(#1){% #1: number; the factor
 \global\Vcolumnwidth=#1\Vcolwdmodule}
```

Glue between the columns is set by `\noalign`; see section 2, no. 10. Note, `\VD` uses ‘`PT`’ as an argument delimiter in the glue’s last component.

```
\def\VD#1PT{% #1: glue between columns
 \noalign{\hskip #1pt}}% valid after a \cr/\crcr
\def\Vd#1pt{\VD#1PT}% fixed width case
```

Longer text should be placed inside a `\vtop` (see section 2, no. 5) and the next macro allows for typesetting justified or ragged right. This macro must be called in the preamble with `\span`.

```
\def\Vv(#1)(#2){%#1: hsize frac; #2:rskip w/o em
 \vtop{\hsize=0.#1\hsize \parindent=0pt
  \leftskip=0pt \rightskip=#2em minus #2em
  \strut##\strut}}%usage: in preamble with \span
```

As said above, `\multispan` works with `\valign` and I created a variant tailored for this text that puts lines across rows. I assume that `\tabskip` and `\baselineskip` have only natural width.

```
\newtoks\Vlft \newtoks\Vrt % surround the rows
\def\Vmultispan#1/#2{% #1: # rows; #2: # lines
% center text lines of \Vmultirows with \vcenter
% (in an \hbox) of height (stored in \dimen255):
% #1\baselineskip + (#1-1)\tabskip
 \multispan{#1}\bgroup\dimen255=#1\baselineskip
 \advance\dimen255 by #1\tabskip
 \advance\dimen255 by -\tabskip
 \hbox\bgroup $\the\Vlft \vcenter to \dimen255
  \bgroup\vss \count255=#2\vbox\bgroup
   \Vmultirows}%  the macro leaves 4 groups open
\newdimen\Vrowskip % skip between text lines
\def\Vmultirows#1{%#1: a text line; calls itself
 \hbox{\strut#1}\advance\count255 by -1
 \ifnum\count255=0 %all lines done; close groups
  \egroup \vss\egroup \the\Vrt $\egroup \egroup
 \else\vskip\Vrowskip\expandafter\Vmultirows\fi}
```

4 Examples for `\valign`

Let’s state one point directly at the beginning. Although every alignment can be implemented with `\valign`, one should only use this primitive if it provides an advantage over `\halign`.

When we look at the list that was presented in section 2 on how `\valign` works, it becomes clear how an alignment should look to be a good use case. The typeset data should be connected column-wise and each row should have a different format. Table 2 fulfills these criteria.

Table 2: Some numbers in four positional systems

Decimal	10	100	991
Binary	`1010`	`1100100`	`1111011111`
Octal	*12*	*144*	*1737*
Hexadecimal	`0xA`	`0x64`	`0x3DF`

Here is one implementation with `\halign`.

Example 1: Implementation with `\halign`

```
\centerline{\bf Table 2: \rm Some numbers in
   four positional systems}\medskip
\everycr={}\tabskip=0pt % initialize
\centerline{\vbox{\halign{#\hfil\hskip4pt&&
      \hskip4pt\hfil#\cr % periodic preamble
   Decimal&         10&        100&
                 991\cr \noalign{\vskip3pt}
   Binary&    \tt 1010&\tt 1100100&
      \tt 1111011111\cr \noalign{\vskip3pt}
   Octal&     \it 12\/&  \it 144\/&
          \it 1737\/\cr \noalign{\vskip3pt}
   Hexadecimal&\tt 0xA&   \tt 0x64&
           \tt 0x3DF\cr}}}
```

Note, I keep ‘`\tabskip=0pt`’ to be able to use a periodic preamble and perfect centering, i.e., the values of the first and last application of `\tabskip` must be identical. I cannot write all binary numbers in one physical row because of the journal’s column width. But even with another width the line break is required if a few more numbers are added.

The table can be easily implemented with the primitive `\valign` if we use the macros from the above described toolset. Here numeric data is used so the column width gets determined by the number of digits in the binary representation. This number is specified as a table entry with the template `\VII(#)`. The other entries use the macro `\VR` to keep this column width. The exception is, of course, the first typeset column, in which we just output hboxes without the template.

Example 2: Implementation with `\valign`

```
\centerline{\bf Table 2: \rm Some numbers in
   four positional systems}\medskip
\everycr={}\tabskip=0pt \Vi(\tt0)%  initialize
\centerline{\valign{\VII(#)&\tabskip=3pt
   \VR{#}&\VR{\tt#}&\VR{\it#\/}&
   \VR{\tt0x#}\tabskip=0pt\cr% end of preamble
  0&\Vo{Decimal}&\Vo{Binary}&\Vo{Octal}&
    \Vo{Hexadecimal}\cr           \Vd8pt
  4& 10&        1010&   12&   A\cr\Vd4pt
  7& 100&    1100100&  144&  64\cr\Vd4pt
 10& 991& 1111011111& 1737& 3DF\cr}}
```

We moved the format of the entries into the templates so that the entries are easier to read. The entries in an input row belong together and are more easily understood than in the version with `\halign`.

Imagine adding another typeset column for the decimal number 128. In example 2 this is easy: Add the line “`8& 128& 10000000& 200& 80\cr\Vd4pt`” after the line for 100. In example 1 we have to work harder: All input rows must be changed, as each row receives one new entry.

In a new table we could turn the first column into headers; this supports `\halign`. But such a table cannot fill the double-column width if we add more numbers; `\valign` can do that better.

Of course, the general idea is to design an alignment in such a way that its contents communicates best with the readers. But the world isn't ideal and every alignment must also obey the constraints given by the available space, the document structure, or a journal's style guide. And there might be aesthetic reasons to format an alignment horizontally or vertically.

Let's look at an example that illustrates what I mean with the last sentence. The following table fits into the width of this journal's columns.

Table 3: No. of *TUGboat* articles by rating in 2022

	Issues of Volume 43		
Category	**43:1**	**43:2**	**43:3**
Reports and Notices	9	13	8
Introductory	4	4	5
Intermediate	3	12	8
Intermediate Plus	2	4	9
Advanced	3	3	6

Of course, this table can be implemented with `\halign` but less comfortably. With `\valign` I take one issue, count the articles on the page where the articles are rated by difficulty, and type one input row that becomes a typeset column. With `\halign` I have to add to each input row one number.

A solution with `\halign` should have the first column as headers and the issues as rows. This table is wide and short: The headers dominate the column widths. The numbers in the cells aren't equidistant except one works with `\hidewidth` for the headers and an extremely stretching `\tabskip`. Would such a table look aesthetically pleasing? Does it help to turn two of the categories into two-line headers?

Here is the implementation with `\valign`. It shows how `\Vc` builds the centered header. Note, this works because the columns have no additional skips between them. Also note, the table has an additional 3 pt space between caption and first header as well as an additional 1 pt fixed-width distance between all other rows and after the table because of the two `\tabskip`s.

Example 3: Implementation of Table 3

```
\medskip\centerline{\spaceskip=3.2pt\bf Table 3:
 \rm No.\ of \TUB\ articles by rating in 2022}
\everycr={}\tabskip=3pt \Vcolumnwidth=35pt
\valign{\tabskip=1pt&\VC{#}\cr % periodic
  &\Vo{\bf Category}&\Vo{Reports and Notices}&
   \Vo{Introductory}&\Vo{Intermediate}&
   \Vo{Intermediate Plus}&\Vo{Advanced}\cr
  \Vc(3){\bf Issues of Volume 43}& % centered
   \bf 43:1&        9& 4&        3& 2& 3\cr
  &\bf 43:2&\llap{1}3& 4&\llap{1}2& 4& 3\cr
  &\bf 43:3&        8& 5&        8& 9& 6\cr}
```

The next example uses a `\vtop` through the macro `\Vv` from the toolset as longer text is output; see Table 4. The table fills the full `\hsize`.

Table 1, #25, shows that we cannot use '`\tt \string\cr`' in a cell entry. Therefore, I define the macro `\(...)` to typeset reserved tokens like `\cr`.

Digression: Note, however, because of section 2, no. 4(3), '`\noindent\valign{\tt#\cr \string\cr \cr}`' is valid input. In this case `\string` is expanded by TeX early as it appears after a `\cr` and it digests the next token, the `\cr`.

Example 4: Table 4; some text is omitted

```
\def\(#1){% #1: text; output \#1 in \tt
  {\tt\string\ \unskip#1}}
\tabskip=8pt %    space between title and table
```

Table 4: Alignment methods in TeX

Method	*Tabbing*	*Horizontal/vertical alignment*
Description	Sets fixed width columns via tabulators to align text that fits into the column width line by line. A user can create arbitrarily long alignments with this method. Three methods are available to set the tabulators: (1) a fixed number of columns, (2) column widths via a sample line, and (3) remove and set tabulators on the fly. It is implemented in plain TeX, using macros.	TeX calculates the necessary width (for horizontal)/height (for vertical) of each column/row after it reads all input lines of the alignment; thus TeX needs a lot of memory to store a very long alignment. Each column/row gets an individual format for its material through templates collected in the preamble of the alignment. This is a built-in function of the program TeX.
Used tokens	`\settabs`, `\columns`, `\+`, `&`, `\cr`, `\cleartabs`	`\halign`/`\valign`, `&`, `#`, `\tabskip`, `\cr`, `\crcr`, `\span`, `\omit`, `\noalign`

Table 5: Elementary properties of the digits

Digit	**Parity**	**Factors**	**One unique property**
0	even	neither prime nor composite	additive identity: $x + 0 = x$
1	odd		multiplicative identity: $x \times 1 = x$
2	even	prime	only 2 obeys $x + x = x \times x = x^x$
3	odd		unique sequence: prime followed by a square
4	even	square: 2×2	
5	odd	prime	unique: member of two twin prime pairs: 3 and 5, 5 and 7
6	even	composite: 2×3	smallest perfect number: $1 + 2 + 3 = 6$
7	odd	prime	smallest: not sum of < 4 squares as $1^2 + 1^2 + 1^2 + 2^2 = 7$
8	even	cube: $2 \times 2 \times 2$	unique: two integer powers that differ by 1 (nonelementary)
9	odd	square: 3×3	

```
\centerline{{\bf Table 4:}
  Alignment methods in \TeX}
\noindent\valign{\tabskip=10pt %   row distance
  \hbox{\strut\it #}&\span\Vv(415)(0)\vfil&
    \span\Vv(415)(3)\vfil\cr        % preamble
 \Vo{Method}&\Vo{Description}&
   \Vo{Used tokens}\cr              % 1st column
\VD 10pt plus 5pt minus 5PT % glue between cols
 Tabbing&
 Sets fixed width columns via tabulators to ...
  (3)~remove and set tabulators on the fly.\par
  It is implemented ...&
 \(settabs), \(columns), \(+), {\tt\&}, \(cr),
  \(cleartabs)\cr
\VD 20pt plus 10pt minus 10PT %   glue doubled
 Horizontal/vertical alignment&
 \TeX\ calculates the necessary width (for ...
  in the preamble of the alignment.\par
  This is a built-in function of the program
  \TeX.&
 \(halign)/\(valign), {\tt\&}, {\tt\#},
  \(tabskip), \(cr), \(crcr), \(span), \(omit),
  \(noalign)\cr}
```

The nice thing about the code in example 4 is that all elements of the alignment methods are kept together. That is, after the description of *tabbing* the relevant tokens are listed.

Of course, Table 4 can be implemented with `\halign`. Although I repeat myself, the input is less pleasant as it requires data from tabbing and alignment in each input row.

In this case, when we turn the first column into headers to improve the input, the typeset table looks unbalanced: a short text, a wide and long text, and a wide and short text. The third column should be made smaller but I'm convinced that such a table doesn't look better than Table 4.

5 Spanning rows with `\valign`

Another case for `\valign` are alignments in which rows have to be joined. Table 5 does not have a vertical structure but spanned rows. Three columns have their text left aligned and we need not calculate the column widths. Only the first column has to use `\Vcolumnwidth` and we can set it outside of the alignment.

I should mention that the use of the large brace forces us to assign `\tabskip` a certain value. The large delimiters aren't available in all sizes and so we have to make sure that two `\baselineskip` plus the `\tabskip` equal the height of an available brace.

Example 5: Implementation with `\valign`

```
\Vi(\bf Digit)\VII(1)% init for 1st column
\centerline{\bf Table 5: \rm Elementary
                properties of the digits}
\tabskip=6pt % get right distance for the \}
\Vlft={\left\}\mkern3mu}\Vrt={\right.}
\Vrowskip=2pt % initialization for \Vmultispan
\noindent\valign{&\hbox{\strut#}\cr % periodic
 \bf Digit& \VC{0}&\VC{1}&\VC{2}&\VC{3}&
   \VC{4}&\VC{5}&\VC{6}&\VC{7}&\VC{8}&\VC{9}\cr
\Vd6pt \bf Parity& even& odd&
  even& odd& even& odd& even& odd& even& odd\cr
\Vd6pt \bf Factors&
  \Vmultispan2/2{neither prime}{nor composite}&
  \Vmultispan2/1{prime}&
  square: $2\times2$&      prime&
  composite: $2\times3$&   prime&
  cube: $2\times2\times2$& square: $3\times3$\cr
\Vd10pt \bf One unique property&
  additive identity: $x+0=x$&
  multiplicative identity: $x\times1=x$&
  only 2 obeys $x+x=x\times x=x^x$&
  \Vmultispan2/1{unique sequence: prime
```

Table 6: Holidays in at least one federal state and well-known (local) celebration days in Germany, 2023

Day	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1	New Year				Labor Day						All Saint's	
2												
3										Unity Day		Advent Sun
4												
5												
6	Epiphany			Holy Thu								
7				Good Fri								
8			Women's Day			Corpus Christi		(Augsburg)				
9				Easter Sun	Europe Day							
10				Easter Mon								
11											11.11.	
12												
13												
14			Pi Day		Mother's Day							
15					Kalte Sofie			Assumption Day				
16												
17			St. Patrick									
18					Ascension Day							
19												
20		Rose Mon							Children's Day			
21												
22		Ash Wed									Rep&Pray	
23					Constitution Day							
24												Heiligabend
25												Christmas
26												Boxing Day
27						Siebenschläfer						
28					Whit Sun							
29					Whit Mon							
30					(Wäldchestag)							
31										Reformation		Silvester

```
   followed by a square}&
unique: member of two twin prime pairs:
   3 and 5, 5 and 7&
smallest perfect number: $1+2+3=6$&
smallest: not sum of $<4$ squares as
   $1^2+1^2+1^2+2^2=7$&
\Vmultispan2/1{unique: two integer powers
   that differ by 1 (nonelementary)}\cr}
```

Note: The unique property for the digits 8 and 9 is called *Catalan's Conjecture*. Eugène Catalan published it in 1844; Preda Mihăilescu proved it in 2002.

6 Rules in vertical alignments

Some readers might be proud to have reached the level "TeX Master" [4, p. 245] and they might want to see how to earn this title with `\valign`.

Let's start with a calendar that shows all holidays for 2023 in Germany; see Table 6. (I made the output fit into this journal's page width. We want to typeset such a table; so it shouldn't be important that it isn't presented in one or two landscape pages.) Not all listed days are a holiday in every federal state. Some aren't a holiday at all and the days in parentheses are only locally celebrated.

With a `\valign` we create the table columnwise, i.e., month by month. Thus we can define macros that list the holidays in sequence to insert them into the table in the correct sequence just by calling the macro. The floating holidays are separately listed. Of course, they have to be reassigned every year.

Example 6: Holidays in Germany
With the macro `\(...)` I shorten some names in the output as the column width for each month in Table 6 is rather small and I want to avoid overprinting.

TeX input

```
\begingroup \eightpoint % see TeXbook, p.415
  \countdef\Hfcnt=1 \countdef\Hvcnt=3
  \countdef\Cfcnt=5 \countdef\Cvcnt=7
  \edef\RESETcounters{% save current values
```

```
    \global\count1=\the\count1
    \global\count3=\the\count3
    \global\count5=\the\count5
    \global\count7=\the\count7 }
  \global\Hfcnt=0 \global\Hvcnt=0
  \global\Cfcnt=0 \global\Cvcnt=0
\def\(#1){\ignorespaces}% to shorten names
\def\Hf{% public holidays with fix date
 \ifcase\Hfcnt New Year\('s Day)\or Epiphany\or
 Women's Day\or Labor Day\or (Augsburg\( Peace
 Festival))\or Assumption Day\or \(World)
 Children's Day\or \(German) Unity Day\or
 Reformation\( Day)\or All Saint's\( Day)\or
 Christmas\( Day)\or Boxing Day\fi
 \global\advance\Hfcnt by 1 }
\def\Hv{% public holidays with variable date
 \ifcase\Hvcnt Good Fri\(day)\or Easter
 Sun\(day)\or Easter Mon\(day)\or Ascension
 Day\or Whit Sun\(day)\or Whit Mon\(day)\or
 Corpus Christi\or Rep\(entance)\&Pray\(er Day)
 \fi \global\advance\Hvcnt by 1 }
\def\Cf{% special days with fix date
 \ifcase\Cfcnt Pi Day\or St.\ Patrick\('s Day)
 \or Europe Day\or Kalte Sofie\or Constitution
 Day\or Siebenschl\"afer\or 11.11.\or
 Heiligabend\or Silvester\fi
 \global\advance\Cfcnt by 1 }
\def\Cv{% special days with variable date
 \ifcase\Cvcnt Rose Mon\(day)\or Ash
 Wed\(nesday)\or Holy Thu\(rsday)\or Mother's
 Day\or (W\"aldchestag)\or Advent Sun\(day)\fi
 \global\advance\Cvcnt by 1 }
\def\tblstrut{% an own strut for this table
 \vrule height 8pt depth 4pt width 0pt }
\Vi(Day\ )% initialize first column; no \VII
\def\\#1#2{% #1, #2: digits
 \omit\hbox to\Vcolwdmodule{% use the unit
    \tblstrut\hfil\number#1#2\ }}
```

Now we just have to enter the macro names into a 31-day scheme for each month.

I prefer to draw only a few rules. So I use just one vertical rule and for each three-day group a horizontal rule. This is enough to identify the day of a holiday quickly.

The vertical rule is simply drawn by a `\vrule` in a `\noalign`; see section 2, no. 12. But note that the spacing to the left and the right is controlled in the macro `\\` and in the preamble, respectively.

The horizontal rules are included in the columns. This is possible as an `\hrule` gets the width of its context, i.e., here it's the width of the column. The `\Vh`s in the 5th, 9th, ..., 41st entry of the months also help to find the entry for each day.

Example 6 continued: The calendar with `\valign`

```
\everycr={}\tabskip=0pt
\Vcolumnwidth=37.4pt % ridiculously small
\centerline{\bf Table 6: \rm Holidays in at
 least one federal state and well-known (local)
 celebration days in Germany, 2023}\medskip
\noindent\valign{\VL{\hskip1pt#}&& % first row
  \VL{\hskip1pt\tblstrut\sixrm#}\cr % periodic
\Vo{Day }& \\01&\\02&\\03&\Vh&\\04&\\05&\\06&
\Vh&\\07&\\08&\\09&\Vh&\\10&\\11&\\12&\Vh&\\13&
\\14&\\15&\Vh&\\16&\\17&\\18&\Vh&\\19&\\20&\\21&
\Vh&\\22&\\23&\\24&\Vh&\\25&\\26&\\27&\Vh&\\28&
\\29&\\30&\Vh&\\31\cr        \noalign{\vrule}
 Jan&\Hf&   &   &\Vh&   &   &\Hf&\Vh&   &   &
        &\Vh&   &   &   &\Vh&   &   &   &\Vh&
        &   &   &\Vh&   &   &   &\Vh&   &   &
        &\Vh&   &   &   &\Vh&   &   &   &\Vh&
        \cr
 Feb&   &   &   &\Vh&   &   &   &\Vh&   &   &
        &\Vh&   &   &   &\Vh&   &   &   &\Vh&
        &   &   &\Vh&   &\Cv&   &\Vh&\Cv&   &
        &\Vh&   &   &   &\Vh&   &   &   &\Vh&
        \cr
...
 Dec&   &   &\Cv&\Vh&   &   &   &\Vh&   &   &
        &\Vh&   &   &   &\Vh&   &   &   &\Vh&
        &   &   &\Vh&   &   &   &\Vh&   &   &
     \Cf&\Vh&\Hf&\Hf&   &\Vh&   &   &   &\Vh&
     \Cf\cr}
\RESETcounters \endgroup
```

Other than example 6 the next example doesn't seem to benefit from `\valign`. Well, the table does not prefer `\halign` either. That is, there is neither a horizontal nor a vertical structure in Table 7.

Table 7: Lo-Shu

4	9	2
3	5	7
8	1	6

It's a magic square, i.e., sums of columns, rows, and diagonals always give the same result. It's probably been known in China for > 2000 years but received the above-stated name only centuries after its discovery.

I think Table 7 is a very good exercise for TeX Masters if they want to start to think in vertical structures. Here is an implementation with `\valign`; its output is shown in Table 7. Note, this time I set `\leftskip` in the code; see section 2, no. 1.

Example 7: Rules with `\valign`

```
\noindent\bf Table 7: \rm Lo-Shu\medskip
\def\balign{\leftskip=12pt \noindent}%       use
\def\ealign{\par\leftskip=0pt }\relax% \leftskip
\def\digstrut{% a strut to center the digits
  \vrule height 9.2pt depth 2.8pt width 0pt }
\everycr={}\tabskip=0pt \Vcolumnwidth=20pt
\balign\valign{\hrule#&\vskip1pt\hrule#&
  \VC{\digstrut#}&\hrule#&\VC{\digstrut#}&
  \hrule#&\VC{\digstrut#}&\hrule#&
  \vskip1pt\hrule#\cr% preamble with 9 templates
    \multispan9\vrulefill\cr         % left rule
    &\multispan7\hbox to1pt{\hss}&\cr      % gap
    &&\multispan6\vrulefill&\cr% inner left rule
```

Table 8: SI prefixes

Name	Symbol	Multiplying factor		Name	Symbol	Multiplying factor
quetta	Q	10^{30}		quecto	q	10^{-30}
ronna	R	10^{27}		ronto	r	10^{-27}
yotta	Y	10^{24}		yocto	y	10^{-24}
zetta	Z	10^{21}		zepto	z	10^{-21}
exa	E	10^{18}		atto	a	10^{-18}
peta	P	10^{15}		femto	f	10^{-15}
tera	T	10^{12}	(trillion)	pico	p	10^{-12}
giga	G	10^{9}	(billion)	nano	n	10^{-9}
mega	M	10^{6}	(million)	micro	µ	10^{-6}
kilo	k	10^{3}	(thousand)	milli	m	10^{-3}
hecto	h	10^{2}		centi	c	10^{-2}
deca	da	10^{1}		deci	d	10^{-1}

Source: `https://www.bipm.org/en/measurement-units/si-prefixes`

```
&&  4&&  3&&  8&&\cr     % first data column
&&\multispan6\vrulefill&\cr           % rule
&&  9&&  5&&  1&&\cr    % second data column
&&\multispan6\vrulefill&\cr           % rule
&&  2&&  7&&  6&&\cr     % third data column
&&\multispan6\vrulefill&\cr   % right border
&\multispan7\hbox to1pt{\hss}&\cr
\multispan9\vrulefill\cr}\ealign
```

In this table the above-mentioned `\vrulefill` is used. The horizontal rules are realized by the method that associates them with templates, i.e., it is the dual method used for the vertical rules in an implementation with `\halign`.

7 Combining both methods

Above, I wrote all alignments can be realized with `\halign`. I don't find it complicated if one follows the description in Chapter 22 of *The TEXbook* [4]. People who have difficulties with the required discipline of this approach can benefit from a combination of `\halign` and `\valign`. Let's study two cases.

As the first example let's look at a *double-up* table, like Table 8. It is one of the rare cases in which vertical rules are allowed according to [1, 13.53]. I implement the two halves of the table as horizontal alignments and a `\valign` glues them together with a vertical rule (see section 2, no. 12).

Example 8: Combine `\halign` and `\valign`

The font `cmmu10.mf` is built from the font `cmmi10.mf` by setting `slant:=0` and `math_fitting:=false`. Note, in a macro the '`#`' of an alignment must be doubled [4, p. 203].

TEX input

```
\font\tenmu=cmmu10 \let\hw=\hidewidth
\def\lhalf{\vbox{\offinterlineskip \tabskip=0pt
 \halign to 0.4\hsize{\tabskip=20pt
      \strut##\hfil&\hfil##\hfil& % Name, Symbol
      $10^{##}$\hfil\tabskip=5pt& % Factor
      ##\hfil\tabskip 0pt plus 100pt\cr % (text)
  Name\hw& \hw Symbol\hw&
   \omit\span Multiplying factor\hw\cr
 \noalign{\vskip 3pt}
  quetta& Q&  30\cr   ronna&  R&  27\cr
  yotta&  Y&  24\cr   zetta&  Z&  21\cr
  exa&    E&  18\cr   peta&   P&  15\cr
  tera&   T&  12& (trillion)\cr
  giga&   G&   9& (billion)\cr
  mega&   M&   6& (million)\cr
  kilo&   k&   3& (thousand)\cr
  hecto&  h&   2\cr   deca& \hw da\hw& 1\cr}}}
```

The main aspects of the preamble are repeated for the right half. The difference is that it has one less column because there is no "(text)".

Example 8 continued: The second half

```
\def\rhalf{\vbox{\offinterlineskip \tabskip=0pt
 \halign to 0.4\hsize{\tabskip=20pt
      \strut##\hfil&\hfil##\hfil& % Name, Symbol
      $10^{##}$\hfil              % Factor
        \tabskip=5pt plus 100pt\cr% no (text)
  Name\hw& \hw Symbol\hw&
   \omit Multiplying factor\hw\cr% no \span
 \noalign{\vskip 3pt}
  quecto& q& -30\cr   ronto&  r& -27\cr
  yocto&  y& -24\cr   zepto&  z& -21\cr
  atto&   a& -18\cr   femto&  f& -15\cr
  pico&   p& -12\cr   nano&   n&  -9\cr
  micro&  \tenmu\char22& -6\cr % upright $\mu$
  milli&  m&  -3\cr   centi&  c&  -2\cr
  deci&   d&  -1\cr}}}
```

(A side note: The organization wants us to typeset all physical prefixes upright, including the Greek μ for "micro". I created `cmmu10.mf` as a simple realization of an upright math italic font. It's sufficient for Table 8 and doesn't look too bad in text: ng, µg, mg. AMS Euler Roman looks too different: µg.)

The last step is to take the halves and create a one-row vertical alignment with two columns.

Example 8 continued: Application of `\valign`

```
\leftline{\bf Table 8: \rm SI prefixes}
\tabskip=4pt % space around alignment
\noindent\valign{#\cr % preamble
    \lhalf\cr
\noalign{\hskip 10pt\vrule\hskip 40pt}
    \rhalf\cr}
\par\leftline{{\it Source:}
  {\ninett https://www.bipm.org/en/%
   measurement-units/si-prefixes}}
```

Sure, this example is too simple to show all advantages of the method but I hope the reader understands the concept.

The next example is like Table 7: Neither alignment method is preferred. But it's clear that nested alignments help. I use `\valign` for the larger alignment as its output can be directly applied in a paragraph. There is no need to put the alignment into a `\vbox` with `\offinterlineskip`, which is necessary if `\halign` is used; see the code of the inner blocks. (Compare also to the discussion in the appendix.)

We want to typeset sudoku puzzles. As we have to typeset a lot of them in order to explain how to solve one, we implement a simplified input method that doesn't require us to use `\halign` or `\valign`. The idea is to enter the code on the left to get the result on the right:

```
\sudokublocks
~~3~~~5~~
974~~~613
8~~~~~~~4
8~14~97~2
3~9~~~5~1
4~25~19~6
2~~~~~~~4
156~~~237
~~8~~~1~~
~~~end~~~
```

		3	9	7	4	8		
5			6	1	3			4
8		1	3		9	4		2
4		9				5		1
7		2	5		1	9		6
2			1	5	6			8
		4	2	3	7	1		

Each input line represents one 3×3 block of the sudoku. A tilde is used for an empty cell. The sentinel `~~~end~~~` finishes the input. (The input encoding might not be the best choice, but, again, for this article it's sufficient and seems to require the smallest amount of extra code compared to other methods.)

First, we code the command that the users enter; it calls the macro which reads the input.

Example 9: Input for sudoku

```
\def\SPsudokuend{~~~end~~~}% the sentinel
\newif\ifSPbadinput % true: <> 9 input rows
\countdef\SPblockcnt=8 % counts the blocks
\def\sudokublocks{% user macro to start input
 \SPbadinputfalse  % assume all cells get input
 \begingroup           % closed in \SPblocksend
  \SPblockcnt=1 \def~{\phantom0}%~ = empty cell
  \let\next=\SPrdblock \next}% initialize \next
```

Next we read one input line representing one sudoku block. I assume the sentinel is always present.

Example 9 continued: Reading a sudoku block

```
\def\SPrdblock#1
{% #1: 9 elements (digits and ~) EOL delimited
 \def\nxt{#1}\ifx\nxt\SPsudokuend % the sentinel
   \let\next=\SPblocksend   % true: end of input
  \else   % store '.#1' in \SPblock<\SPblockcnt>
   \expandafter     % build the block macro name
   \def\csname SPblock\romannumeral\SPblockcnt
        \endcsname{%  we need at least 9 elements
    .#1{}{}{}{}{}{}{}{}{}}%empty groups are okay
   \ifnum\SPblockcnt<10 \advance\SPblockcnt by 1
   \else \SPbadinputtrue % >9 blocks is an error
  \fi\fi \next}
\def\SPblocksend{% prints sudoku if input okay
 \ifnum\SPblockcnt=10 \else \SPbadinputtrue \fi
 \ifSPbadinput              % not nine input lines
  \errmessage{You need exactly nine data rows}%
 \else \SPprint     % no error: print the puzzle
 \fi\endgroup}
```

Next we turn our attention to the topic of this article. First, I implement a 3×3 alignment with thin rules and centered digits. The macro gets nine arguments for the nine cells and builds one cell of the larger alignment. Note the parameter text starts and ends with a period. Later I explain why the first period is important in this implementation.

Example 9 continued: A sudoku block

```
\def\Vs{\vrule height9.2pt depth2.8pt width0pt }
\def\SPmkcell.#1#2#3#4#5#6#7#8#9.{% #1--#9:block
 \vbox{\offinterlineskip \tabskip=3.5pt
  \everycr={}\halign{% one 3x3 block, thin rules
     \Vs ##&& \vrule ### \Vs ##\cr   % periodic
    #1&& #2&& #3\cr \noalign{\hrule}
    #4&& #5&& #6\cr \noalign{\hrule}
    #7&& #8&& #9\cr}}}
```

As in example 8, the '`#`' is doubled in the preamble as this is required in a macro to get a single '`#`'. (The strut `\Vs` centers the digits vertically.)

Finally, here is the macro that typesets the sudoku puzzle. Note the macro `\SPmkcell` expects expanded `\SPblock`⟨`\SPblockcnt`⟩ macros and as TeX expands a macro after '`&`' in the entries for an alignment (see section 2, no. 4(3)) we can use `\SPmkcell` in the preamble.

Example 9 continued: The complete sudoku

```
\def\SPprint{% typeset 9 \SPblock<\SPblockcnt>s
 \everycr={\noalign{\vrule width 1pt }}%
 \tabskip=0pt
  \valign{\hrule height 1pt ##&&\SPmkcell##.&
     \hrule height 1pt ##\cr % periodic preamble
   &\SPblocki  &&\SPblockiv&&\SPblockvii &\cr
   &\SPblockii &&\SPblockv &&\SPblockviii&\cr
   &\SPblockiii&&\SPblockvi&&\SPblockix  &\cr}}
```

Because of the `\SPblock`⟨*n*⟩s the necessary transposition of the input rows of `\sudokublocks` is easily accomplished.

Let's look at the important point of why we need the period in the `\SPblock...` macros. TeX expands after the '`&`' the argument of `\SPmkcell` to check if it starts with `\omit`. Without the period and with '`~`' as the first character in an input row, TeX expands this active character, finds no `\omit`, but leaves us with the seven tokens "`\let\if@=\iftrue \h@true\ph@nt0`", i.e., with the tokens from the expanded "`\phantom0`". These tokens throw errors as they aren't expected by `\SPmkcell`. The period avoids this problem.

Sometimes TeX's alignment methods are used for structures that don't look like a table. Next, let's study one such structure: a tournament. See Table 9 on the next page.

Example 10: Shortcuts for the teams

```
\def\AR{Argentina}   \def\AU{Australia}
\def\BR{Brazil}      \def\CH{Switzerland}
\def\EN{England}     \def\ES{Spain}
\def\FR{France}      \def\HR{Croatia}
\def\JP{Japan}       \def\KR{South Korea}
\def\MA{Morocco}     \def\NL{Netherlands}
\def\PL{Poland}      \def\PT{Portugal}
\def\SN{Senegal}     \def\US{USA}
```

Clearly, there are four columns in Table 9. Its typeset row elements are somewhat complex: The teams of the match appear above a rule and the result below it. Sometimes a vertical rule appears at the right end of such an element above and sometimes below the horizontal rule. And the elements for two matches, the finals, have no vertical rules.

To get a perfect alignment of the described elements two other elements are needed. One is completely empty but has the height and width of the other elements. The second is empty except for a vertical rule at the right end. The structure of these basic elements is realized by two `\halign`s.

Example 10 continued: The elements

```
\def\tblstrut{% use an own strut
 \vrule height 10pt depth 4pt width 0pt}
\def\Match#1#2-#3-#4#5{% #1, #2: teams;
 % #3: result; #4, #5: \relax or \omit for rule/
 % no rule at right of lines 1 (#4) and 3 (#5)
 \halign{\VC{\tblstrut##}&\vrule##\cr % preamble
  {#1}--{#2}&#4\cr         % the match (plus \vrule?)
  \multispan2\hrulefill\cr      % horizontal rule
  #3&#5\cr}}%          the result (plus \vrule?)
\def\Empty#1{% #1: \relax/\omit for rule/no rule
 \halign{\VC{\tblstrut##}&\vrule##\cr % preamble
  \relax&#1\cr
  \omit\phantom{\vrule height0.4pt}&#1\cr
  \relax&#1\cr}}
```

With the help of macros the five different cases can be easily implemented. Here I use short names as these macros are later used in the big alignment: 'f', 't', 'b' stand for "final", "top", and "bottom", respectively. A pair of "top" and "bottom" elements is connected via vertical rules as the winners of these matches build an element in the next column.

Example 10 continued: The five building blocks

```
\def\Mf#1#2-#3-{% #1, #2: teams; #3: result
   \vbox{\offinterlineskip\tabskip=0pt
      \Match{#1}{#2}-#3-\omit\omit}}%    no rules
\def\Mt#1#2-#3-{% #1, #2: teams; #3: result
   \vbox{\offinterlineskip\tabskip=0pt
      \Match{#1}{#2}-#3-\omit\relax}}
\def\Mb#1#2-#3-{% #1, #2: teams; #3: result
   \vbox{\offinterlineskip\tabskip=0pt
      \Match{#1}{#2}-#3-\relax\omit}}
\def\E{\vbox{% an empty block
   \offinterlineskip\tabskip=0pt\Empty\omit}}
\def\R{\vbox{% an empty block with a rule
   \offinterlineskip\tabskip=0pt\Empty\relax}}
```

The last step is to create the tournament with a `\valign`. I use a `\valign` as it represents the four columns better than an `\halign`. Each input row in the vertical alignment stands for a round of the tournament; in an `\halign` the input rows cross the tournament rounds.

Nevertheless, we must add the empty elements and thus, we have to do some counting in the columns. To make the input easier and shorter I decided to create blocks of three `\E` and `\R` as there is some pattern.

Note, the small column width stretches the input; otherwise top and bottom matches could be placed in one input row. Note also that the `\hsize` is 39 pc = 468 pt and this width minus 2 pt (1.6 pt are needed for the vertical rules) is distributed to the four columns; a tight fit.

Example 10 continued: Apply a `\valign`

```
\def\EEE{\E&\E&\E}\def\RRR{\R&\R&\R}% short-cuts
\centerline{\bf Table 9: \rm Knockout matches of
  the FIFA World Cup 2022}
\medskip \hrule height 1pt \medskip % use \hsize
\begingroup\offinterlineskip \tabskip=0pt
\noindent\valign{\VI#&&#\cr% periodic preamble
124pt&\Vo{\bf\qquad Round of 16}\vskip6pt&
         \Mt{\bf\NL}\US-3:1-&            \R&
         \Mb{\bf\AR}\AU-2:1-&
      \E&\Mt\JP{\bf\HR}-1:1, 1:1, (1:3)-&\R&
         \Mb{\bf\BR}\KR-4:1-&
      \E&\Mt{\bf\EN}\SN-3:0-&            \R&
         \Mb{\bf\FR}\PL-3:1-&
      \E&\Mt{\bf\MA}\ES-0:0, 0:0, (3:0)-&\R&
         \Mb{\bf\PT}\CH-6:0-\cr
124pt&\Vo{\bf\qquad Quarter-finals}&
```

Table 9: Knockout matches of the FIFA World Cup 2022

Round of 16	Quarter-finals	Semi-finals	Final
Netherlands–USA 3:1			
	Netherlands–**Argentina** 2:2, 2:2, (3:4)		
Argentina–Australia 2:1			
		Argentina–Croatia 3:0	
Japan–**Croatia** 1:1, 1:1, (1:3)			
	Croatia–Brazil 0:0, 1:1, (4:2)		
Brazil–South Korea 4:1			
			Argentina–France 2:2, 3:3, (4:2)
England–Senegal 3:0			
	England–**France** 1:2		
France–Poland 3:1			
		France–Morocco 2:1	
Morocco–Spain 0:0, 0:0, (3:0)			**Third place play-off**
	Morocco–Portugal 1:0		**Croatia**–Morocco 2:1
Portugal–Switzerland 6:0			

Results: First entry for normal time, second for extra time; goals in penalty shootout with parentheses.
Source: `https://en.wikipedia.org/wiki/2022_FIFA_World_Cup_knockout_stage`

```
    \E&\Mt\NL{\bf\AR}-2:2, 2:2, (3:4)-&\RRR&
       \Mb{\bf\HR}\BR-0:0, 1:1, (4:2)-&
  \EEE&\Mt\EN{\bf\FR}-1:2-&           \RRR&
       \Mb{\bf\MA}\PT-1:0-\cr
109pt&\Vo{\bf\qquad Semi-finals}&
  \EEE&\Mt{\bf\AR}\HR-3:0-&\RRR&\RRR&\R&
       \Mb{\bf\FR}\MA-2:1-\cr
109pt&\Vo{\bf\qquad Final}&
  \EEE&\EEE&\E&
       \Mf{\bf\AR}\FR-2:2, 3:3, (4:2)-&
  \EEE&\E&\omit
     \vtop to 28.4pt{% 2 rows plus rule height
      \vskip 14.4pt % +12pt => 2pt white space
```

```
      \hbox to \Vcolumnwidth{\strut\bf\hss
        Third place play-off\hss}\vfil}&
       \Mf{\bf\HR}\MA-2:1-\cr}\endgroup
```

There is one more task to do: The source of the data should be stated in a footnote to the table [1, 13.44].

Example 10 continued: A footnote

```
\medskip \hrule height 1pt \medskip\noindent
{\it Results:} First entry for normal time,
second for extra time; goals in penalty shootout
with parentheses.\par\noindent {\it Source:}
{\tt\catcode`\_=12 https://en.wikipedia.org/%
 wiki/2022_FIFA_World_Cup_knockout_stage}
```

8 Other use cases for \valign

One macro of plain TeX based on `\halign` is very useful. It's the macro `\ooalign` [4, p. 356], for overprinting characters. A new symbol can be created by this macro if this symbol is just a combination of characters available in the Computer Modern fonts.

Example 11: Using `\ooalign`

```
\def\smiley{{% a group is required for \ooalign
 \mathsurround=0pt %  required with inline math
 \ooalign{\hidewidth \raise.3ex\hbox{% the eyes
    $\cdot\mkern 2mu\cdot$}\hidewidth\cr
  \hidewidth \lower.2ex\hbox{%        the mouth
    $\scriptscriptstyle \smile$}\hidewidth\cr
  \Orb\cr}}}%                          the head
```

Now we can write `\smiley` and get: ☺.

There is no need to implement a replacement of `\ooalign` with `\valign`. But we learned that one of `\halign`'s features can be utilized to overprint characters.

What makes `\valign` unique to turn it into an unusual application? The next example uses the fact that TeX adjusts the row heights (see section 2, no. 6) to move a character vertically; see also [6, p. 36]. In the second application we use the fact that it's easy to stack characters one above the other with the help of `\valign` (see section 2, nos. 5 and 7).

Case 1: Between digits the hyphen is too low as it is placed at a height that looks good for lowercase letters. But with telephone numbers a hyphen is used, thus we need a fix for these numbers.

Example 12: Moving a symbol with `\valign`

```
\def\D-{% center ‘-’ with respect to a digit
 {\everycr={}\tabskip=0pt \valign{##\cr% 1 row
   \vphantom0\cr   % invisible; defines height
   \vfil\hbox{-}\vfil\cr}}}%  centered wrt ‘0’
```

It is easy to raise the hyphen with the above defined `\D-`: The input "`555\D-6789 555-6789`" typesets "555-6789 555-6789". It also works with other font sizes.

Case 2: In former times lowercase German umlauts (ä, ö, ü) were written in texts that were typeset in "Fraktur" (blackletter) sometimes with an 'e' above the vowel instead of using two periods.

Example 13: Stacking symbols with `\valign`

```
\def\e"#1{% #1: a, o, u; put an ‘e’ above it
 {\everycr={}\tabskip=0pt \setbox0=\hbox{#1}%
  \valign{##\tabskip=0.18ex&% distance e--vowel
      ##\tabskip=0pt\cr      % two-row preamble
    \hbox to \wd0{\hss\fivebf e\hss}& % the ‘e’
    \box0\cr}}}%                output the vowel
```

Now we can *simulate* the old writing with Computer Modern Roman. For example, TeX typesets the input "`Albrecht D\e"urer, Hans Sch\e"aufelin`" as "Albrecht Duͤrer, Hans Schaͤufelin". Note, however, kerning information is lost; for example, vaͤ va.

References

[1] *The Chicago Manual of Style*, 15th edition, Chicago, Illinois: University of Chicago Press, 2003.

[2] Malcolm Clark, *A plain TeX primer*, Oxford University Press, 1992.

[3] Taco Hoekwater, "A use case for `\valign`", *ConTeXt Group Journal* 2018, 7–18. `articles.contextgarden.net/journal/2018/7-18.pdf`

[4] Donald E. Knuth, *The TeXbook*, Volume A of *Computers & Typesetting*, Boston, Massachusetts: Addison-Wesley, 1984.

[5] Donald E. Knuth, *TeX: The Program*, Volume B of *Computers & Typesetting*, Boston, Massachusetts: Addison-Wesley, 1986.

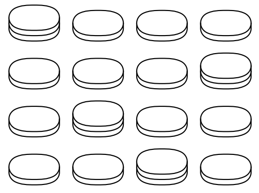
[6] Peter Wilson, "Glisterings: Glyphs, long labels", *TUGboat* **35**:1 (2014), 36–38. `tug.org/TUGboat/tb35-1/tb109glister.pdf`

Appendix

I don't know if Clark was aware of answer 22.∞ in `texbook.tex`, i.e., the source of [4], that applies `\valign`. Knuth's macros `\one` and `\two` are as simple as possible. Sure, `\halign` can be used but only with templates like '`\vbox{\offinterlineskip#}`'. Or we redefine the macros, for example, with a more complex `\ooalign` (see section 8). Such macros can also be used in Knuth's `\valign` answer [4, p. 335].

Example 14: An `\halign` solution of exercise 22.∞

```
\font\manual=manfnt % symbols for The TeXbook
\def\pennytop{% circular symbol like a coin
   \hbox to 24pt{\manual\char’130\hfil}}
\def\pennyedge{% like \pennytop but not closed
   \hbox{\manual\char’133}}
\def\OA#1{% #1: hboxed char on top of \pennyedge
   {\ooalign{\raise2pt#1\cr\pennyedge\cr}}}
\def\one{\OA\pennytop}\def\two{\OA{\hbox{\one}}}
\halign{&\tabskip=0pt#\cr% periodic
   \two&\one&\one&\one\cr \noalign{\vskip6pt}
   \one&\one&\one&\two\cr \noalign{\vskip6pt}
   \one&\two&\one&\one\cr \noalign{\vskip6pt}
   \one&\one&\two&\one\cr}
```



The alignment was created with more effort. As Clark indicated: It can be done the other way too. But as shown here, sometimes one needs more code.

◇ Udo Wermuth
Dietzenbach, Germany
`u dot wermuth (at) icloud dot com`