The TABLE Manual

Michael J. Wichura

The University of Chicago

	´0	´1	´2	´3	´4	´5	´6	´7	
´00x	Γ	Δ	Θ	Λ	Ξ	Π	Σ	Υ	˝0x
´01x	Φ	Ψ	Ω	ff	fi	fl	ffi	ffl	
´02x	ı	ȷ	`	´	ˇ	˘	¯	˚	˝1x
´03x	¸	ß	æ	œ	ø	Æ	Œ	Ø	
´04x	˛	!	”	#	$	%	&	’	˝2x
´05x	(	)	*	+	,	-	.	/	
´06x	0	1	2	3	4	5	6	7	˝3x
´07x	8	9	:	;	¡	=	¿	?	
´10x	@	A	B	C	D	E	F	G	˝4x
´11x	H	I	J	K	L	M	N	O	
´12x	P	Q	R	S	T	U	V	W	˝5x
´13x	X	Y	Z	[	“	]	^	·	
´14x	‘	a	b	c	d	e	f	g	˝6x
´15x	h	i	j	k	l	m	n	o	
´16x	p	q	r	s	t	u	v	w	˝7x
´17x	x	y	z	–	—	˝	~	¨	
	˝8	˝9	˝A	˝B	˝C	˝D	˝E	˝F	

Printed: September 1988

This manual was written using TEX supplemented by the TABLE macros. The features described herein exist in Version 1.0 of TABLE.

The TABLE project was carried out using computer facilities supported in part by National Science Foundation Grants number DMS 86-01732 and DMS 87-03942 to the Department of Statistics at the University of Chicago.

PREFACE

TABLE is a collection of TEX macros which facilitate the construction of tables, such as the BUDGET TRANSFERS table

1970 Federal Budget Transfers (in billions of dollars)			
State	Taxes Collected	Money Spent	Net
New York	22.91	21.35	−1.56
New Jersey	8.33	6.96	−1.37
Connecticut	4.12	3.10	−1.02
Maine	0.74	0.67	−0.07
California	22.29	22.42	+0.13
New Mexico	0.70	1.49	+0.79
Georgia	3.30	4.28	+0.98
Mississippi	1.15	2.32	+1.17
Texas	9.33	11.13	+1.80

from the tbl manual. In general, a table consists of columns which may be independently left-adjusted, centered, right-adjusted, or aligned on decimal points. Headings may be placed over single columns or groups of columns. Table entries may contain equations or several rows of text. Horizontal and vertical lines may be drawn wholly or partially across the table. Of course, all these things could be done using TEX's primitive `\halign`, `\omit`, and `\span` commands; typically they're considerably easier to do with TABLE.

In writing TABLE the author drew upon some good ideas from existing table programs. TABLE's key system for specifying column formats is adapted from M. E. Lesk's tbl program. (LATEX has a format key system too, but TABLE's is both more extensive and more flexible.) The idea of letting the choice of entry separator determine whether or not a vertical line is drawn across a row is adapted from Michael Ferguson's INRSTEX program. (TABLE's separators are implemented differently, though, so that TABLE is recursive whereas the INRSTEX table-making macros are not.)

Examples and exercises are the life blood of any instruction manual, so this manual has lots of them. The more exercises you try to solve, the more you'll learn and the faster you'll learn it. Answers to all the exercises are given in Appendix A. Many of the examples are up front, in Section 1. That section was written to be a kind of mini-manual; you can learn enough from it to start using the TABLE macros to make tables of your own. Subsequent sections go over everything in detail and add embellishments.

For ease of reference, all of TABLE's format keys are tabulated in Appendix B. Similarly, TABLE's commands and parameters are tabulated in Appendices C and D; these two appendices exhibit all of TABLE's external control sequences.

TABLE can be used within LATEX as an enhanced version of LATEX's `tabular` environment.

Thanks go to Mike Beach, Rick Chappell, and especially Mike and April Frigge for their many helpful suggestions during the preparation of the TABLE manual.

Michael J. Wichura

Chicago, Illinois
September, 1988

CONTENTS

1. INTRODUCTION

1.1. EXAMPLES

Various features of the table-making macros are introduced here through a series of examples. This section doesn't try to explain everything; there are aspects of the examples you won't understand until you've read the rest of the manual. Moreover, some of the rules stated here aren't strictly true; definitive versions are left for later on. Nonetheless, there's enough information in this section to enable you to start using the macros to construct some tables of your own.

Example. To begin with, the WORLD POPULATION table

Year	World Population
8000 B.C.	5,000,000
50 A.D.	200,000,000
1650 A.D.	500,000,000
1850 A.D.	1,000,000,000
1945 A.D.	2,300,000,000
1980 A.D.	4,400,000,000

on page 246 of *The TEXbook* is produced by placing the following code between a pair of $$'s:

```
\BeginTable
  \def\AD{\csc\ a.d.} % (use Caps and Small Caps font)
  \def\BC{\csc\ b.c.}
  \def\C{\JustCenter}
  \BeginFormat
  |     r     |          r          |
  \EndFormat
  \_
  |  \C Year  |  World Population  |  \\+22
  \_
  |  8000\BC  |        5,000,000   |  \\+20
  |    50\AD  |      200,000,000   |  \\
  |  1650\AD  |      500,000,000   |  \\
  |  1850\AD  |    1,000,000,000   |  \\
  |  1945\AD  |    2,300,000,000   |  \\
  |  1980\AD  |    4,400,000,000   |  \\+02
  \_
\EndTable
```

The commands `\BeginTable` and `\EndTable` delineate the table environment. The `\def`initions made between `\BeginTable` and `\BeginFormat` are local to

the table. The format specification between `\BeginFormat` and `\EndFormat` stipulates two r`i`ght-adjusted columns. The `\_` commands draw horizontal lines across the whole table, while the '`|`'s separating the data items in the body of the table draw vertical lines in the rows in which they appear. The `\\` commands designate the ends of rows; the suffix `+22` to `\\` in the first row adds (`+`) some extra space (`2` points above and `2` points below) to the text for that row to separate it more from the adjacent horizontal lines. The `\C` before `Year` causes that word to be centered in its column.

Exercise 1. Set the WORLD POPULATION table without the suffixes '`+22`', '`+20`', and '`+02`' to `\\`. How does the result compare to the table in the text? (If you don't have a CAPS AND SMALL CAPS font, use, e.g., '`\sevenrm \ A.D.`'.)

Example. Next, the code

```
\BeginTable
  \def\L{\JustLeft}
  \BeginFormat
  |   c    |    c     |      c          |
  \EndFormat
  \_
  |  \use3     AT\&T Common Stock       |  \\  \_
  |  Year  |  Price   |  Dividend        |  \\  \_
  |  1971  |  41--54  |   \$2.60         |  \\  \_
  |  ~~~2  |  41--54  |    ~2.70         |  \\  \_
  |  ~~~3  |  46--55  |    ~2.87         |  \\  \_
  |  ~~~4  |  40--53  |    ~3.24         |  \\  \_
  |  ~~~5  |  45--52  |    ~3.40         |  \\  \_
  |  ~~~6  |  51--59  |    ~~.95\rlap*   |  \\  \_
  "  \use3 \L  * (first quarter only)     "  \\
\EndTable
```

produces the classic AT&T COMMON STOCK table

AT&T Common Stock		
Year	Price	Dividend
1971	41–54	$2.60
2	41–54	2.70
3	46–55	2.87
4	40–53	3.24
5	45–52	3.40
6	51–59	.95*

* (first quarter only)

from the tbl manual (see also page 247 of *The TEXbook*). Here the format specification stipulates three centered columns. The entries for the first and last rows span all three columns because of the '`\use3`'s. (In general, '`\use`*c*', *c* being a single digit, uses *c* columns and the format of the last one.) TABLE

defines '~' to be a non-printing character having the width of a single digit; the numeric entries therefore line up properly with the corresponding column labels. No vertical lines are drawn at the ends of the bottom row because of the following simple rule: if instead of '|' you type '"' as a column separator, no vertical line will be drawn at the corresponding point in the table row.

Exercise 2. Set the first column of the WORLD POPULATION table using 'c' and '~'s.

Example. Here is the AT&T COMMON STOCK table again, in a more "open" style:

AT&T Common Stock

Year	*Price*	*Dividend*
1971	41–54	$2.60
2	41–54	2.70
3	46–55	2.87
4	40–53	3.24
5	45–52	3.40
6	51–59	.95*

* (first quarter only)

This version of the table is produced by a slight modification of the previous code:

```
\BeginTable
  \def\L{\JustLeft}
  \BeginFormat
  |    ck      |    ck       |      ck         |
  \EndFormat
  \_
  "  \use3  \it AT\&T Common Stock             "  \\+22
  "  \use3  \-                                 "  \\0
  "  \it Year  "  \it Price  "  \it Dividend   "  \\+22
  "  \-        "  \-         "   \-            "  \\0
  "  1971      "  41--54     "   \$2.60        "  \\+20
  "  ~~~2      "  41--54     "    ~2.70        "  \\
  "  ~~~3      "  46--55     "    ~2.87        "  \\
  "  ~~~4      "  40--53     "    ~3.24        "  \\
  "  ~~~5      "  45--52     "    ~3.40        "  \\
  "  ~~~6      "  51--59     "    ~~.95\rlap*  "  \\+02
  \_
  "  \use3 \L * (first quarter only)           "  \\+20
\EndTable
```

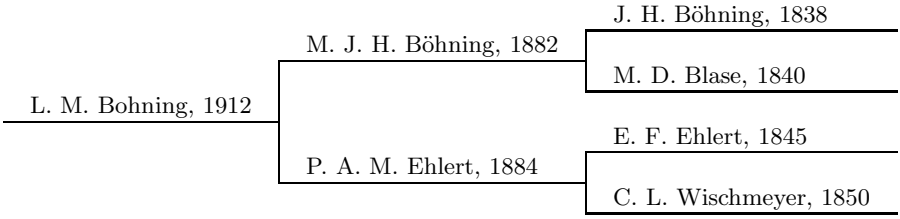
Three new features of the TABLE macros are employed here: (1) A 'k' in a column format makes that column a bit wider (by a kern having the width of a digit) on both the left and right. (2) A \- command within a column draws a horizontal line across that column; the line is exactly as wide as the

column. (3) The rows of a table are evenly spaced vertically because the `\\` commands insert struts into them. No strut, however, is inserted when `\\` is followed by '`0`'. The `\\0` rows above thus have precisely the height and depth (about $1/4$ point) of the horizontal lines they contain. ('`\\+`*hd*', *h* and *d* being single digits, adds *h* points to the height of the usual strut and *d* points to its depth.)

Exercise 3. Use '`k`' and `\-` to set the WORLD POPULATION table in open style.

Exercise 4. What would be the result of setting the AT&T COMMON STOCK table using the open-style code above, but with '`\\0`' replaced by '`\\`'?

Example. The FAMILY TREE chart



from page 248 of *The TEXbook* is produced by

```
\BeginTable
  \ninepoint   % (From Appendix E of The TeXbook)
  \Expand
  \def\B{Bohning}      \def\Eh{Ehlert}
  \def\BB{B\"ohning}   \def\W{Wischmeyer}
  \BeginFormat
  | l               | l                   | l                 |
  \EndFormat
  "                 "                     " J. H. \BB, 1838   " \\
  "                 "                     & \=                & \\0
  "                 " M. J. H. \BB, 1882 |                     \\
  "                 & \=                  &                     \\0
  "                 |                     | M. D. Blase, 1840 " \\
  "                 |                     & \=                & \\0
  " L. M. \B, 1912 |                                           \\
  & \=              &                                          \\0
  "                 |                     " E. F. \Eh, 1845   " \\
  "                 |                     & \=                & \\0
  "                 | P. A. M. \Eh, 1884 |                     \\
  "                 & \=                  &                     \\0
  "                 "                     | C. L. \W, 1850    " \\
  "                 "                     & \=                & \\0
\EndTable
```

Note that the columns are left-adjusted. The principle new feature here is the `\=` command which, like `\-`, draws a horizontal line across its column.

`\=`, however, differs from `\-` in two important respects: (1) A `\=` line extends into the white space between columns so as to join up with any neighboring vertical lines. (2) When a table is `\Expand`ed, as this one is, so as to be exactly as wide as the page, `\=` lines lengthen, but `\-` lines don't. There is also a difference in syntax; in place of the usual '`|`' and/or '`"`', you have to type a '`&`' for the column separator on each side of a `\=`.

Exercise 5. Set the little MORTALITY table to the right making use of `\=`.

	control		
	alive	dead	
low risk	5	3	8
high risk	4	26	30
	9	29	39

Example. This portion

American Chicken	*French Connection*	*Cooking Methods*
Squab	*Poussin*	Broil, Grill, Roast
Broiler	*Poulet Nouveau*	Broil, Grill, Roast
Fryer	*Poulet Reine*	Fry, Sauté, Roast
Rooster	*Poulard*	Roast, Poach, Fricassee
Fowl	*Poule de l'Année*	Stew, Fricassee
Rooster	*Coq*	Soup stock, Forcemeat

of the RECIPE table on page 236 of *The TEXbook* is set with

```
\BeginTable
  \OpenUp11
  \BeginFormat
  | rB           | cI             | l                      |
  \EndFormat
  " \sl American " \sl French     " \sl Cooking            " \\
  " \sl Chicken  " \sl Connection " \sl Methods            " \\+03
  " Squab        " Poussin        " Broil, Grill, Roast    " \\
  " Rooster      " Coq            " Soup stock, Forcemeat  " \\
\EndTable
```

The input lines for 'Broiler', 'Fryer', 'Rooster', and 'Fowl' are similar and have been omitted. This example illustrates the use of `\OpenUp` to increase the spacing between all the rows of a table; '`\OpenUp`*hd*', *h* and *d* being single digits, adds *h* points to the height and *d* points to the depth of the struts inserted by `\\` and `\\+`. The example also introduces two simple format keys, '`B`' and '`I`'. Entries in a '`B`' column are set in Bold face, just as though you had typed `\bf` in front of each entry. '`I`' is similar, but specifies Italic type (`\it`). The top two lines of the table are set in slanted type regardless of whether the format has '`B`' or '`I`' because, for example, '`\bf \sl`' gives the same result as '`\sl`' alone.

Exercise 6. Suppose you wanted all the names in the FAMILY TREE chart to be in italic type. What change would you make to the format for that table? (This is too easy.)

Example. The following WORD USAGE table excerpted from Efron and Thisted (1976) exhibits the number of words Shakespeare used exactly n times, as a function of n.

Number of words used exactly n times	n
14,376	1
4,343	2
2,292	3
7	99
5	100

The code for this is:

```
\BeginTable
  \def\C{\JustCenter}
  \BeginFormat
    | cn[14,376]                    | r           |
  \EndFormat
    " \C \it Number of words        " \Lower{$n$} " \\
    " \C \it used exactly $n$ times "             " \\
    "            14,376             "       1     " \\+30
    "             4,343             "       2     " \\
    "             2,292             "       3     " \\
    "                 7             "      99     " \\
    "                 5             "     100     " \\
\EndTable
```

One new format key, '`n`', is introduced here. Entries in a "numeric" column are aligned on their decimal points; in the case at hand the decimal points are implicit (e.g., '2,292' is taken to be '2,292.'). '`n`' needs to be told how wide the widest entry in the column is; that's why the "sample entry" '`14,376`' is specified in the format. ('`00,000`' would have worked just as well, since all 10 digits have the same width.) The `\C`'s in the heading are necessary to escape from the numeric format. Note that '`~`'s wouldn't suffice to align the first column of this table since some of the entries contain a comma, which doesn't have the width of a digit. `\Lower` lowers its argument a half-line; this feature is particularly useful in table headings.

Exercise 7. Use '`n`' to set the second column of the WORLD POPULATION table so that the column heading is centered over the population figures.

Exercise 8. Use '`n`' to set the STACK of numbers to the right. Specify the sample entry as '`00.0`'.

46
23
15
6.5
2.1

Example. Student's (1908) SOPORIFICS table

Additional Hours of Sleep gained
by the use of two tested drugs

Patient	*A*	*B*	*Difference* $B - A$
1	+0.7	+1.9	+1.2
2	−1.6	+0.8	+2.4
3	−0.2	+1.1	+1.3
4	−1.2	+0.1	+1.3
5	−0.1	−0.1	0
6	+3.4	+4.4	+1.0
7	+3.7	+5.5	+1.8
8	+0.8	+1.6	+0.8
9	0	+4.6	+4.6
10	+2.0	+3.4	+1.4
Mean	+0.75	+2.33	+1.58

produced in part by

```
\BeginTable
  \def\C{\JustCenter}
  \def\H#1{\C \Lower{\it #1}} % For Headings
  \def\Diff{\C \it Difference}
  \BeginFormat
  |   cn[00]    | cN[+00.00] | cN[+00.00] | cN[+00.00] |
  \EndFormat
  " \use4 \C \it Additional Hours of Sleep gained      " \\
  " \use4 \C \it by the use of two tested drugs        " \\+03
  \_
  | \H{Patient} |   \H{$A$}  |   \H{$B$}  |  \Diff     | \\+30
  |     {}      |     {}     |     {}     |  \C $B-A$  | \\+03
  \_
  |      1      |    +0.7    |    +1.9    |    +1.2    | \\+30
  |      2      |    -1.6    |    +0.8    |    +2.4    | \\
  |      5      |    -0.1    |    -0.1    |     0      | \\
  \_
  |  \C Mean    |    +0.75   |    +2.33   |    +1.58   | \\+33
  \_
\EndTable
```

illustrates the use of the format key 'N', which is like 'n', except that entries are set in math mode. This distinction is relevant whenever minus signs are involved, since '$-$' gives '−', whereas '-' gives only a short dash '-'. You can't have any empty entries in an 'n' or 'N' column; that's why '{}'s are used to fill out the row containing '$B-A$'. Moreover, at least one blank must follow each entry.

Exercise 9. Why do you suppose the author specified the sample entries for the '`N`' columns in the SOPORIFICS table as '`+00.00`' instead of '`+0.00`'?

Example. Consider next the STATISTICS table

Estimate	*Standard Error*	*Correlations*		
$\hat{\beta}_1 = -1.61$	0.38	1.0		
$\hat{\beta}_2 = -4.97$	0.47	0.53	1.0	
$\hat{\beta}_3 = -3.32$	0.43	0.57	0.66	1.0

which is coded as

```
\BeginTable
  \LongLines
  \def\B#1{\hat\beta_{#1}}
  \def\H#1{\JustCenter \Lower{\it #1}}
  \BeginFormat
  |  c m s6      | c            | l s    | l    | l      |
  \EndFormat
  \_4
  " \H{Estimate} " \it Standard " \use3 \H{Correlations} " \\+20
  "              " \it Error    "        "      "        " \\+02
  \_
  " \B1 = -1.61  " 0.38         " 1.0    "                 \\+30
  " \B2 = -4.97  " 0.47         " 0.53   " 1.0  "          \\
  " \B3 = -3.32  " 0.43         " 0.57   " 0.66 " 1.0    " \\+02
  \_4
\EndTable
```

`\LongLines` is like `\Expand` in that it makes `\_` lines extend all the way across the page. It differs from `\Expand` in that `\Expand` stretches the white space between columns, whereas `\LongLines` does not. As this example shows, `\_` may be followed by a single digit; the larger the digit, the darker is the line `\_` draws. `\_` alone is equivalent to `\_2`. The same conventions apply to `\-` and `\=`. The format key '`s`', followed by an optional single digit w, sets the inter-column space to the width of w digits. '`s`' alone is equivalent to '`s3`'. (For comparison, TEX's `\quad` and `\qquad` have the width of 2 and 4 digits respectively.) An '`s`' specification applies to the white space to the right of the current column *and all subsequent columns*, until it is overriden by another '`s`' key. The format key '`m`' specifies math mode; every entry in an '`m`' column is set as though you had typed it between a pair of '`$`'s. (You can escape from math mode by using a `\JustLeft`, `\JustCenter`, or `\JustRight` command.)

Exercise 10. Use '`m`' to set the 2-BY-2 table to the right.

y_{11}	y_{12}	n_1
y_{21}	y_{22}	n_2
m	$n-m$	n

Example. The SPECIAL FUNCTIONS table

Name	*Definition*
Gamma	$\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}\,dt$
Sine	$\sin(x) = \frac{1}{2i}(e^{ix} - e^{-ix})$
Error	$\operatorname{erf}(z) = \frac{2}{\sqrt{\pi}}\int_0^z e^{-z^2}\,dz$
Bessel	$J_0(z) = \frac{1}{\pi}\int_0^\pi \cos(z\sin\theta)\,d\theta$
Zeta	$\zeta(s) = \sum_{k=1}^\infty k^{-s} \quad (\Re s > 1)$

from the tbl manual results from

```
\BeginTable
  \OpenUp99
  \def\erf{\mathop{\rm erf}}
  \BeginFormat
  |4 l     |  r M o0   | \M                                  |4
  \EndFormat
  \_4
  |\it Name| \use2 \JustCenter \it Definition        | \\+{-5}{-5}
  \_
  | Gamma  | \Gamma(z) " = \int_0^\infty t^{z-1}e^{-t}\,dt | \\ \_
  | Sine   | \sin(x)   " = {1\over 2i}(e^{ix} - e^{-ix})   | \\ \_
  | Error  | \erf(z)   " = {2\over \sqrt\pi} \int_0^z
                           e^{-z^2}\,dz                   | \\ \_
  | Bessel | J_0(z)    " = {1\over \pi} \int_0^\pi
                           \cos(z\sin \theta)\,d\theta    | \\ \_
  | Zeta   | \zeta(s)  " = \sum_{k=1}^\infty k^{-s}
                                        \quad (\Re s>1) | \\+02
  \_4
\EndTable
```

As this example shows, a '|' *in the format line* can be followed by a single digit; the larger the digit, the darker are all the corresponding vertical lines in the table. Notice the three new format keys, 'M', '\M', and 'o': (1) 'M' is like 'm', except that column entries are set in display style. (2) '\M' is a variant of 'M' which is used to set left-adjusted column entries that start with a relation ('=', '<', '>', '≤', '≥', etc.). (3) 'o' is like 's', but affects only the white space just to the right of the current column. 'o0' is used above to eliminate the usual inter-column space between the second and third columns. As you

know from the RECIPE example, '`\OpenUp99`' adds 9 points to the height and depth of the strut which TABLE uses to maintain even spacing of the rows of a table. The suffix '`+{-5}{-5}`' to `\\` in the first row makes a further change to the strut for that row, adding -5 points to both its height and depth; the strut on the first row is thus 4 $(= 9 - 5)$ points higher and deeper than usual.

Exercise 11. Use '`s`', '`o`', '`m`' and '`\m`' (which is like '`\M`' without display style) to set the PAIRED EQUATIONS from page 242 of *The TEXbook*:

$$\begin{aligned} V_i &= v_i - q_i v_j, & X_i &= x_i - q_i x_j, & U_i &= u_i, & \text{for } i \neq j; \\ V_j &= v_j, & X_j &= x_j, & U_j &= u_j + \textstyle\sum_{i \neq j} q_i u_i. \end{aligned}$$

Use '`\OpenUp11`' to increase the spacing between the rows.

Example. The MATH SPACING table

		Right Atom							
		Ord	Op	Bin	Rel	Open	Close	Punct	Inner
	Ord	0	1	(2)	(3)	0	0	0	(1)
	Op	1	1	*	(3)	0	0	0	(1)
	Bin	(2)	(2)	*	*	(2)	*	*	(2)
Left	Rel	(3)	(3)	*	0	(3)	0	0	(3)
Atom	Open	0	0	*	0	0	0	0	0
	Close	0	1	(2)	(3)	0	0	0	(1)
	Punct	(1)	(1)	*	(1)	(1)	(1)	(1)	(1)
	Inner	(1)	1	(2)	(3)	(1)	0	(1)	(1)

on page 170 of *The TEXbook* is made by

```
\BeginTable
  \Expand \ninepoint
  \BeginFormat
s0|lIs1| l   | cw5 | cw5 | cw5 | cw5 | cw5 | cw5 | cw5 | cw5 | .
  "    "     " \use8 \it Right Atom                            "\\
  "    "     " Ord " Op  " Bin " Rel "Open "Close"Punct"Inner"\\+22
  "    "     & \use8 \=                                        &\\0
  "    "Ord  |  0  "  1  " (2) " (3) "  0  "  0  "  0  " (1) |\\+20
  "    "Op   |  1  "  1  "  *  " (3) "  0  "  0  "  0  " (1) |\\
  "    "Bin  | (2) " (2) "  *  "  *  " (2) "  *  "  *  " (2) |\\
  "Left"Rel  | (3) " (3) "  *  "  0  " (3) "  0  "  0  " (3) |\\
  "Atom"Open |  0  "  0  "  *  "  0  "  0  "  0  "  0  "  0  |\\
  "    "Close|  0  "  1  " (2) " (3) "  0  "  0  "  0  " (1) |\\
  "    "Punct| (1) " (1) "  *  " (1) " (1) " (1) " (1) " (1) |\\
  "    "Inner| (1) "  1  " (2) " (3) " (1) "  0  " (1) " (1) |\\+02
  "    "     & \use8 \=                                        &\\0
\EndTable
```

The only new things here are the two format keys '`w`' and '`.`'. The columns with the '`w5`' format have a minimum width of 5 digits; this feature keeps the spacing of those columns uniform. The key '`.`' is a convenient abbreviation for `\EndFormat`.

Exercise 12. Set the following ANSWER SHEET from a statistics exam. Use '`w(1.25in)`' and '`w(2in)`' to make the second and third columns $1\frac{1}{4}$ and 2 inches wide, respectively.

Quantity	*Estimate*	*Standard Error*
μ_U		
μ_G		
$\mu_U - \mu_G$		

Example. The NEW YORK AREA ROCKS table

New York Area Rocks		
Era	Formation	Age (years)
Precambrian	Reading Prong	>1 billion
Paleozoic	Manhattan Prong	400 million
Mesozoic	Newark Basin, including Stockton, Lockatong, and Brunswick formations; also Watchungs and Palisades.	200 million
Cenozoic	Coastal Plain	On Long Island 30,000 years; Cretaceous sediments redeposited by recent glaciation.

from the tbl manual is set with

```
\BeginTable
  \def\C{\JustCenter}   \OpenUp11
  \BeginFormat
  | l 9          | l 9 p(1.5in)    | l 9 p(1.5in)   | .
  \_
  | \use3 \C \it New York Area Rocks                | \\ \_
  | \C Era       | \C Formation    | \C Age (years) | \\ \_
  | Precambrian | Reading Prong   | $>1$ billion   | \\ \_
  | Paleozoic   | Manhattan Prong | 400 million    | \\ \_
  | Mesozoic    | Newark Basin, including Stockton,
    Lockatong, and Brunswick formations; also
    Watchungs and Palisades.      | 200 million    | \\ \_
  | Cenozoic    | Coastal Plain   | On Long Island
    30,000 years; Cretaceous sediments redeposited
    by recent glaciation.                          | \\ \_
\EndTable
```

Two new format keys, '`p`' and '`9`', are introduced here. The key '`p`', followed immediately by '`(`⟨dimen⟩`)`', sets each entry in its column in "paragraph" mode (more TEXnically, in internal vertical mode) with a line size of ⟨dimen⟩. '`p`' column entries can contain displayed equations, `\obeylines` constructions, hanging indentation, and other such vertical mode material. `\OpenUp` works with '`p`' entries because TABLE puts the upper half of one of its struts on the top line of each entry, and the lower half on the bottom line. '`9`' specifies nine point type. This key, however, isn't one of TABLE's built-in format keys because there isn't much call for it. The author set up '`9`' for use in this manual by typing the instruction

```
\NewFormatKey 9{\ReadFormatKeys b{\ninepoint}}
```

near the start of his input file. This made the key '`9`' an abbreviation for '`b{\ninepoint}`', which instructs TEX to read the command `\ninepoint` before working on each entry in the column involved. '`b`' is one of TABLE's most useful format keys; its argument can be any string of tokens properly balanced with respect to braces. '`b`'s companion is '`a`', which specifies tokens for TEX to read after each entry. The ideas introduced here are developed at length in Sections 3.1.10 and 3.2; it is enough for now to realize that TABLE's key system can be extended at will.

Exercise 13. Why aren't the first two rows of the NEW YORK AREA ROCKS table in nine point type?

Exercise 14. Code the following RECURRENCE CRITERIA table from the theory of Markov chains X with countable state space I. Set the line sizes for the three columns to 70, 170, and 40 points, respectively.

Classification	*Criterion*	*Reference*
X is positive recurrent if	I is finite.	Theorem 3.15
X is positive recurrent if and only if	the equations $$u_k = \sum_{j\in I} u_j p_{jk}, \quad k \in I,$$ have a non-trivial nonnegative summable solution.	Theorem 4.2
X is recurrent (positive or null) if and only if	for some arbitrary $i_0 \in I$, the equations $$x_i = \sum_{j\neq i_0} p_{ij} x_j, \quad i \neq i_0,$$ have no bounded non-trivial solution.	Theorem 5.14

Exercise 15. Guess how TABLE defines its math mode key '`m`' in terms of '`b`' and '`a`'.

This section is very nearly finished. Before going on to the next one, why don't you consolidate what you've learned "by example" by working the following review exercises.

Exercise 16. Use '`c`' and '`n`' to set the first and third columns of the AT&T COMMON STOCK table (either version).

Exercise 17. Use '`~`'s to set the COMPOSITION OF FOODS table

Composition of Foods			
Food	Percent by Weight		
	Protein	Fat	Carbo-hydrate
Apples	.4	.5	13.0
Halibut	18.4	5.2	
Lima beans	7.5	.8	22.0
Milk	3.3	4.0	5.0
Mushrooms	3.5	.4	6.0
Rye bread	9.0	.6	52.7

from the tbl manual. If you look closely, you'll note that '`Food`' is a bit higher than '`Carbo-`'; '`\Raise2{Food}`' will lift '`Food`' up by 2 points.

Exercise 18. Set the following table giving the number of LIZARDS observed at various sites (the data is from Schoener (1970)).

			Time of Day								
			Early			*Mid-day*			*Late*		
S	*D*	*H*	*G*	*O*	*Total*	*G*	*O*	*Total*	*G*	*O*	*Total*
Sunny	≤ 2	< 5	20	2	22	8	1	9	4	4	8
		≥ 5	13	0	13	8	0	8	12	0	12
	> 2	< 5	8	3	11	4	1	5	5	3	8
		≥ 5	6	0	6	0	0	0	1	1	2
Shady	≤ 2	< 5	34	11	45	69	20	89	18	10	28
		≥ 5	31	5	36	55	4	59	13	3	16
	> 2	< 5	17	15	32	60	32	92	8	8	16
		≥ 5	12	1	13	21	5	26	4	4	8

S, sunny/shady; *D*, diameter (in); *H*, height (ft); *G, grahami*; *O, opalinus*.

Exercise 19. Code the FORMAT KEY USAGE table on the next page up to the entry for the 2-BY-2 sample table. Use the format key '`T`' to specify `Type writer` type, and '`b{\csc}`' to specify CAPS AND SMALL CAPS. Enter '`\m`' and '`\M`' as '`\tt\string\m`' and '`\tt\string\M`', respectively.

		Format Key																	
Sample table	*Page*	b	B	c	I	k	l	m	\m	M	\M	n	N	o	p	r	s	T	w	.
2-BY-2	8			X				X												
ANSWER SHEET	11			X	X			X											X	X
COMMAND SUMMARY	14						X								X					
COMMON STOCK$_1$	2			X																
COMMON STOCK$_2$	3			X		X														
COMP. OF FOODS	13			X			X													X
FAMILY TREE	4						X													
FORMAT KEY USAGE	14	X		X			X					X					X	X		
LIZARDS	13			X	X		X	X				X		X						X
MATH SPACING	10			X	X		X										X		X	X
MORTALITY	5			X												X				
NEW YORK ROCKS	11						X								X					X
PAIRED EQUATIONS	10							X	X					X		X	X			
RECIPE	5		X	X	X		X									X				
RECURRENCE CRIT.	12						X								X					
SOPORIFICS	7			X								X	X							
SPECIAL FUNCTIONS	9						X			X	X			X		X				
STACK	6			X								X								
STATISTICS	8			X			X	X									X			
WORD USAGE	6			X								X				X				
WORLD POPULATION	1															X				

Exercise 20. Code the start of the following COMMAND SUMMARY table. Enter the '`|`' and '`"`' in the first column as '`\tt \VBar`' and '`\tt \DQuote`'. (Why can't you just enter '`|`' and '`"`' directly?)

Command	*Usage*
`|`	End entry, with trailing vertical line.
`"`	End entry, without trailing vertical line.
`\\`	End row, with standard strut.
`\\+`*hd*	End row, adding h points to the height and d points to the depth of the strut for that row.
`\\0`	End row, without standard strut.
`\OpenUp` *hd*	Add h points to the height and d points to the depth of struts.
`\_`	Draw horizontal line across whole table.
`\-`	Draw horizontal line exactly the width of the current column.
`\=`	Draw horizontal line across current column and half-way into the neighboring inter-column white spaces.
`\use` *c*	Span c columns, using format of the last one.
`\JustCenter`	Center entry, omitting stipulated format.
`\Expand`	Stretch table to full width of page.
`\LongLines`	Make `\_` lines extend across full width of page.
`\Lower`	Lower argument a half-line.

1.2. OVERVIEW

If you were to think of TABLE as a new car, the preceding examples would constitute the test drive — they show how TABLE handles in a variety of different situations. Now it's time for you to browse through the owner's manual in the next four sections. There you'll find complete instructions on how to work TABLE's various gadgets and on how to read her gauges and indicators. Section 2 on TABLE's quantum systems is a prerequisite to the other sections, so be sure to read it first. Sections 3 and 4 discuss the use of TABLE's format keys and commands, respectively; this material can be read as the need arises.

Here is some terminology: So far as TABLE is concerned, a *table* is an array of *entries* laid out in *rows* and *columns*. The entries in a row are aligned on their baselines; the entries in a column are typically either centered, left-adjusted, right-adjusted, or aligned by decimal points. The entries in a column usually have the same *format* (italic, numeric, mathematical, or whatever), but exceptions are permitted. The *width* of a column is the width of its widest entry. Similarly, the *height* of a row is the height of its tallest entry; the *depth* of a row is the depth of its deepest entry. A *ruled* table is one with horizontal and/or vertical lines, which may extend partially or wholly across the table. With TABLE even quite complicated formats may be specified with ease, and ruled tables are no harder to construct than unruled ones.

Tables are laid out in the input file according to the following outline:

```
\BeginTable
  prologue
  format section
  data section
\EndTable
```

The (possibly empty) *prologue* contains definitions (like `\def\L{\JustLeft}`) and declarations (like `\Expand`) that facilitate the construction of, or affect the design of, the current table only. The mandatory *format section*, which starts with the command `\BeginFormat` and ends with the command `\EndFormat`, describes the format of the columns. The *data section* consists of the table entries, laid out row by row. TABLE ignores blanks before and after the separators '`|`' and '`"`', so entries can be freely positioned in the input lines. It is not required that the '`|`'s and '`"`'s on the input lines line up with the corresponding '`|`'s in the format section. Indeed, several rows can be entered on one input line, or one row on several input lines. Nonetheless, it has been the author's experience that the closer he made the layout of a table in the input file look like the real thing, the less likely he was to make a mistake.

2. TABLE'S QUANTUM SYSTEMS

In the introductory examples you were told that "the format key '`s`', followed by an optional digit w, sets the inter-column space to the width of w digits" and "'`s`' alone is equivalent to '`s3`'". These statements are, however, only half-truths. The full story is as follows: TABLE has an `\InterColumnSpaceUnit` parameter which specifies the units in which inter-column space is measured and an `\InterColumnSpaceFactor` parameter which specifies how many units to use in the absence of an explicit declaration. The default for the inter-column space unit is 0.5 em[1] — the width of a digit. Thus, for example, '`s5`' ordinarily specifies a gap the width of 5 digits. The default for the inter-column space factor is 3, so '`s`' alone is ordinarily equivalent to '`s3`'. You can, however, set the unit and factor parameters to whatever you find convenient. For example, the assignment

'`\InterColumnSpaceUnit=.125in`'

sets the unit parameter to $1/8$ inch, so that '`s5`' specifies a gap of $5 \times 1/8'' = 5/8''$. And

'`\InterColumnSpaceFactor=2`'

sets the factor parameter to 2, so that '`s`' alone is equivalent to '`s2`'.

Exercise 21. What commands will set the inter-column space unit to two millimeters and the inter-column space factor to 5? What then would be the width of the gaps specified by: (1) '`s1`'; (2) '`s`'; (3) '`s8`'?

This unit-and-factor system lets you specify the width of inter-column space with a minimum of typing. TABLE has similar systems for specifying the thickness of horizontal and vertical lines, the width of columns, and the size of kerns.[2] The relevant parameters and their defaults are laid out in the QUANTUM SYSTEMS table on the next page.

All four systems work the same way. The quantity involved, be it inter-column space, line thickness, or whatever, comes in nonnegative integer multiples of the system unit. You can specify any size multiple you want: 0, 1, 2, 10, 52, 111, ... (enter, for example, '`s52`'). In the absence of an explicit declaration, the system factor is used as a multiplier.

---

[1] The default unit actually has stretch and shrink components which are ignored here for the sake of simplicity.

[2] There are also unit-and-factor systems for the height and depth of struts, the distance by which entries are raised or lowered, and the size of vertical gaps in a table. These systems, however, operate slightly differently from the others, so we defer discussion of them to Sections 4.9, 4.10, and 4.14.

QUANTUM SYSTEMS

Quantity	*Unit/Default*	*Factor/Default*
Inter-column space	`\InterColumnSpaceUnit` .5 em plus 1 fil minus .25 em	`\InterColumnSpaceFactor` 3
Line thickness	`\LineThicknessUnit` .00333 in	`\LineThicknessFactor` 2
Column width	`\ColumnWidthUnit` .5 em	`\ColumnWidthFactor` 10
Size of kerns	`\KernUnit` .5 em	`\KernFactor` 1

Exercise 22. The breadth of the horizontal lines drawn by TABLE's `\_`, `\-`, and `\=` commands is governed by the line-thickness system. Assuming the defaults in the QUANTUM SYSTEMS table are in force, how thick are the lines drawn by: (1) `\_1`; (2) `\-`; (3) `\=4`; (4) `\_20`? What would the answers be when the line-thickness unit and factor are 0.4 pt and 1, respectively?

Exercise 23. The format key '`w`' utilizes the column width system. What widths are specified by: (1) '`w1`'; (2) '`w`'; (3) '`w30`', when TABLE's defaults are in force? When `\ColumnWidthUnit=.5in` and `\ColumnWidthFactor=2`?

Exercise 24. The format key '`k`' utilizes the kern size system. How wide are the kerns specified by (1) '`k1`'; (2) '`k`'; (3) '`k4`', when TABLE's defaults apply? When `\KernUnit=.1in` and `\KernFactor=3`?

Exercise 25. Even though kerns of negative width make sense in TEX (they act as backspaces), '`k-2`' is not a legitimate use of TABLE's format key '`k`'. Why not?

From time to time you may want to specify a value that isn't an integer multiple of a system unit, without changing the system unit itself. TABLE lets you escape from the unit/factor systems by entering '`(`⟨value⟩`)`' in place of the usual unit multiplier, ⟨value⟩ designating the actual value you want to use. For example, '`s(.5in)`' specifies an inter-column gap of half an inch, and '`\_(1pt)`' draws a line that is one point thick. For line thicknesses, column widths, and sizes of kerns, ⟨value⟩ must be a dimension (see Chapter 10 of *The TEXbook*). For inter-column spaces, ⟨value⟩ must be the kind of rubber space TEX calls glue (see Chapter 12 of *The TEXbook*). For example, '`s(.25in plus 1fil minus .1in)`' specifies an inter-column space with a natural width of $1/4''$; this space can expand indefinitely but can contract by no more than $1/10''$.

Here's a summary of the foregoing rules, presented in terms of the system for inter-column space. The rules for the other systems are similar. If the key '`s`' is followed immediately by '`(`⟨value⟩`)`', TABLE sets the inter-column space

to ⟨value⟩. If '`s`' is immediately followed by an unsigned integer[3] i, TABLE sets the inter-column space to i multiples of the inter-column space unit. If '`s`' is followed by anything else — in particular, a blank — TABLE sets the inter-column space to (inter-column space factor) × (inter-column space unit).

Exercise 26. The format key '`p`' utilizes the column width system. Which of the following specifications are invalid, and why? (1) '`p4`'. (2) '`p 4`'. (3) '`p4.5`'. (4) '`p45`'. (5) '`p(4.5\ColumnWidthUnit)`'. (6) '`p(4.5in)`'. (7) '`p(4.5 inches)`'. (8) '`p(4.5″)`'. (9) '`p()`'.

In later sections of this manual, the notation ⟨spec$_{ICS}$⟩ will be used to designate any legitimate specification for inter-column space and ⟨value$_{ICS}$⟩ will be used to designate the corresponding value. Thus ⟨spec$_{ICS}$⟩ could be '`(3mm)`', in which case ⟨value$_{ICS}$⟩ would be 3 millimeters. Or ⟨spec$_{ICS}$⟩ could be '3', in which case ⟨value$_{ICS}$⟩ would be 3 multiples of the inter-column space unit. Or ⟨spec$_{ICS}$⟩ could be null, in which case ⟨value$_{ICS}$⟩ would be the default number of multiples of the inter-column space unit. The generic terms for specifications and values in the other systems are, in obvious notation, ⟨spec$_{LT}$⟩ and ⟨value$_{LT}$⟩, ⟨spec$_{CW}$⟩ and ⟨value$_{CW}$⟩, and ⟨spec$_{K}$⟩ and ⟨value$_{K}$⟩.

If you make a change to one of the quantum system parameters in the prologue to a table, the change applies just to that table. If, however, you make the change outside a table, the change applies to all subsequent tables, until the parameter is reset.[4]

The command `\NormalTableUnits` resets all the unit parameters to their default values.[5] This brings up a fine point that was glossed over earlier. TABLE stores the unit parameters as fixed dimensions, so '0.5 em' actually refers to the width of a digit in the font in effect when the unit parameters were last set; that font may be different from the current font. To properly match the unit parameters to the current font, you should specify `\NormalTableUnits` whenever you set a table in a different size type[6] than that in effect when the TABLE macros were loaded.

---

[3] The integer should be terminated by a blank. This rule isn't mandatory, but if you follow it, you'll never get into trouble.

[4] Or until the group in which the change was made ends. See Chapter 5 of *The TEXbook*.

[5] You can change the defaults `\NormalTableUnits` resets; see Section 4.11.

[6] `\NormalTableUnits` also sets the default for the strut system unit. This parameter is very sensitive to the type size; even so small a difference as one point is quite noticeable.

3. FORMAT KEYS

Each table must have a format section which describes the layout of the columns — their alignment, the type of entries they contain, the amount of white space after them, and so on. The format section for a table with n columns takes the form

```
\BeginFormat
  | ⟨keys1⟩ | ⟨keys2⟩ | ... | ⟨keysn⟩ |
\EndFormat
```

Here for $i = 1, 2, \ldots, n$, $\langle \text{keys}_i \rangle$ denotes a (possibly empty) string of *format keys*, separated by optional blanks, which succinctly describes the layout of the i^{th} column. TABLE has an extensive set of built-in format keys that cover all the usual formats, and then some; these keys are described in Section 3.1. Moreover, TABLE lets you define new format keys, much in the way that TEX lets you define new macros; this feature is described in Section 3.2. Key systems are convenient because of their ease of use, but there is some danger of a miscommunication arising between you and the computer. Section 3.3 discusses TABLE's format diagnostics which let you see exactly how the computer is interpreting your requests.

Exercise 27. How many '`|`'s are there in a format section?

3.1. BUILT-IN KEYS

This section describes TABLE's built-in keys, grouped according to function. Most of the keys are amply illustrated in Section 1— see the index on page 14. The examples here are limited to features that were only lightly touched on before. You will need to have read about TABLE's "quantum systems" (Section 2) before taking up Sections 3.1.5 through 3.1.9.

3.1.1. ALIGNMENT KEYS — '`c`', '`l`', '`r`'

These keys specify the alignment of items within a column: '`c`' centers, '`l`' justifies `l`eft, and '`r`' justifies `r`ight. If none of these keys is specified for a column, you get '`c`' by default. If more than one is specified for a column, the last one prevails.

3.1.2. FONT SELECTION KEYS — '`B`', '`I`', '`R`', '`S`', '`T`', '`f`'

Entries in a '`B`' column are set in **B**oldface type, just as though you had typed `\bf` before each entry. '`I`', '`R`', '`S`', and '`T`' are similar, but specify *I*talic (`\it`), Roman (`\rm`), *S*lanted (`\sl`), and Typewriter (`\tt`) type, respectively. You can specify any `f`ont you wish with the key '`f`⟨font identifier⟩'. For example,

'f\bf' is equivalent to 'B'. Also, given that \csc selects CAPS AND SMALL CAPS, 'f\csc' will set its column in THAT FONT.

If you don't make any selection, you get the font in effect when the table began (unless you specify a different font in the prologue to the table). If you make more than one font selection for a column, the first one prevails.

Exercise 28. What font do you get from the key sequence 'BIRST'? From 'f\sevenrm'?

3.1.3. MATH KEYS — 'm', 'M', '\m', '\M'

Entries in an 'm' column are set in math mode, exactly as though you had typed each entry between a pair of '$'s. 'M' is like 'm', but additionally specifies display style: each entry in an 'M' column is set as though you had typed '$\displaystyle{' before it and '}$' after it.

'\m' and '\M' are variants of 'm' and 'M' that you should use to set the right halves of equations that are lined up on a relation ('=', '≤', '≥', etc.), as in the SPECIAL FUNCTIONS example. '\m' and '\M' ensure that TEX will produce the proper spacing around the relation. (For an example of bad spacing, see Exercise 30 below.) Moreover, these keys automatically left-justify the column entries.

No more than one math key may be specified for any given column.

Exercise 29. How would the entry '1\over 2' look in an: (1) 'm' column; (2) 'M' column?

Exercise 30. Reset the SPECIAL FUNCTIONS table, changing '\M' to 'lM' in the format for the third column. How does the result compare to the original?

3.1.4. NUMERIC KEYS — 'n', 'N'

TABLE's 'n' format will align numeric items on an implicit or explicit decimal point, as in the TABLETTE to the right. There the decimal points in '6.25' and '.125' are explicit, whereas the decimal point after '25' is implicit. 'N' is like 'n', but additionally sets the column entries in math mode.

25
6.25
.125

'n' and 'N' will often give the same results. There are, however, some distinctions: (1) Commas look better with 'n'. For example, 'N' typesets '1,234' as '1, 234', whereas 'n' gives '1,234'; to get the proper spacing after the comma with 'N', you'd have to enter '1{,}234'. (2) Minus signs look better with 'N'. For example, 'N' typesets '-12' as '−12', whereas 'n' gives '-12'. (3) 'N' is mandatory for items containing sub- or super-scripts, such as $.0_632$ and 3.2×10^{-7}.

The ensuing discussion is worded in terms of 'n', but applies equally well to 'N'. To carry out the alignment of an 'n' column, TABLE needs to know at the outset how far the column entries will extend to the left and right of the decimal point. To convey this field-width information, you specify

n[⟨sample integer part⟩.⟨sample decimal part⟩]

on the format line when at least one column entry has an explicit decimal point, or as

`n[`⟨sample integer part⟩`]`

when all the decimal points are implicit. Here, ⟨sample integer part⟩ should be the widest integer part of any entry, and ⟨sample decimal part⟩ should be the widest decimal part of any entry. For the TABLETTE above, the format would be '`n[25.125]`'. Since the digits 0, 1, 2, ..., 9 all have the same width, '`n[00.000]`' would work just as well. This particular specification can be entered more simply as '`n2.3` '. In general, you can type '`n`k`.`l ' in place of '`n[`$\underbrace{000\ldots0}_{k\text{ digits}}$`.`$\underbrace{000\ldots0}_{l\text{ digits}}$`]`' and '`n`$k$ ' in place of '`n[`$\underbrace{000\ldots0}_{k\text{ digits}}$`]`'. The trailing blank in '`n`k`.`l ' and '`n`k ' is very important; be sure you don't omit it. Don't use the abbreviated notation when either the ⟨sample integer part⟩ or ⟨sample decimal part⟩ contains a character other than a digit.

For example,

```
\BeginTable
\OpenUp11  \def\C{\JustCenter}
\BeginFormat
|  n4  | n2.0 | n0.3 |   n5.5  | N[0000.0\times10^{-0.0}] | .  \_
| \C A |  \C  | \C C |  \C D   |  \C E                    | \\ \_
| 1234 |  22. | .123 |  12345  | 1234.5\times10^{-8.2}    | \\ \_
|   23 |   2  | .1   |  12345. |   45.0\times10^{-4}      | \\ \_
|    0 |   0. | .23  |  12.345 |    2.                    | \\ \_
|  233 |  12. | .445 |  .12345 |   {}                     | \\ \_
\EndTable
```

produces this pedagogical N-AND-N table:

A		C	D	E
1234	22.	.123	12345	$1234.5 \times 10^{-8.2}$
23	2	.1	12345.	45.0×10^{-4}
0	0.	.23	12.345	2.
233	12.	.445	.12345	

Notice that only explicit decimal points are typeset, and decimal parts are not filled out with zeroes: what you type is what you get.

'`n`' and '`N`' entries must have at most one '`.`' that isn't "hidden" from view inside braces (as in '`{-8.2}`'). If no '`.`' is visible, TABLE assumes an implicit decimal point to the right of the entry. Otherwise, the visible '`.`' is used as an explicit decimal point. This algorithm is admittedly crude, but it is fast and handles simple cases as one would expect.

You need to *pay particular attention to blanks* in '`n`' and '`N`' entries. TABLE requires that each such entry be followed by at least one blank and that any imbedded blanks be hidden between braces (as in '`{123 456}`'). Moreover, no

empty entries are allowed. You can enter what would otherwise be an empty entry as '`{}`', or '`\JustCenter`', or '`\omit`'[7]; don't use '`~`'.

You mustn't specify more than one '`n`' or '`N`' for any given column. Moreover, '`n`' and '`N`' can't be used with any of the font selection[8] or math keys. If '`n`' or '`N`' is used with any of the kern keys '`i`', '`j`', and '`k`', the numeric key must come first. The '`[...]`' form of the column width key '`w`' (see below) can't be used with '`n`' or '`N`'.

Exercise 31. Code the TABLETTE.

Exercise 32. Attempting to set the MICROSCOPIC table $\begin{matrix}\text{My Data}\\ 1{,}234.2\\ 23.815\end{matrix}$, B. L. User entered

```
\BeginTable
  \BeginFormat
  | cN4.3    | .
  " My Data  " \\
  " 1,234.2  " \\
  "    23.815" \\
\EndTable
```

What four mistakes did he make?

Exercise 33. Where is the decimal point in: (1) '`82`'; (2) '`8.2`'; (3) '`{8.2}`'; (4) '`ABC`'; (5) '`AB.C`'?

3.1.5. INTER-COLUMN SPACE KEYS — '`s`', '`o`'

'`s`⟨spec$_{ICS}$⟩' sets the width of the white space to the right of the current column and all subsequent columns to ⟨value$_{ICS}$⟩.[9] An '`s`' specification remains in effect until it is overriden by a new one. It may, however, be interrupted temporarily by an '`o`' specification. '`o`⟨spec$_{ICS}$⟩' is like '`s`⟨spec$_{ICS}$⟩', but applies only to the white space just to the right of the current column; the preceding '`s`' spacing is subsequently reinstated.

If you specify more than one '`s`' and/or '`o`' for a column, the last one prevails. If you don't specify any, the previous '`s`' specification carries over. If there is no previous '`s`', you get the default for ⟨value$_{ICS}$⟩.

A table has white space before the first column and after the last one. An '`s`' or '`o`' placed before the first '`|`' in the table format applies to the white space before the first column. An '`s`' or '`o`' placed before the last '`|`' in the table format applies to the white space after the last column. These exterior white spaces are spanned by the lines `\_` draws across the table. To keep

---

[7] `\omit` is TEX's primitive for omitting a column format.

[8] This rule is overly restrictive. You can in fact use '`n`' or '`N`' with a font selection key (the numeric key must come first), but you should be aware that TABLE will base its calculation of the field width(s) for the numeric format on the font in effect when the format section is read, rather than on the selected font. If those two fonts aren't very different, the alignment will probably come out O.K.; if it doesn't, use '`~`'s (and the technique of Section 4.15, if necessary).

[9] The ⟨Spec$_{ICS}$⟩ and ⟨Value$_{ICS}$⟩ notation is explained near the end of the Section 2.

these lines from being "too long", TABLE automatically reduces the width of exterior white space to half its nominal value. To see what is meant by "too long", work Exercise 35 below.

When a table is expanded or contracted horizontally, it is the white spaces that stretch or shrink, not the columns. Stretching and shrinking take place in proportion to nominal values: an '`s6`' space will always be twice as wide as an '`s3`' space. (Expansion is possible only if TABLE's inter-column space unit has a stretch component; contraction is possible only if that unit has a shrink component. The default unit has both stretch and shrink components.)

Exercise 34. Suppose the defaults of Section 2 are in force. What inter-column space is specified by: (1) '`s1`'; (2) '`s`'; (3) '`o4`'; (4) '`s(.25in)`'?

Exercise 35. Reset the AT&T COMMON STOCK table using the format '`s6 | cs3 | c | cs6 |`', which puts $3 = \frac{1}{2}6$ units of white space before the first column, 3 units between the first and second column and between the second and third column, and $3 = \frac{1}{2}6$ units after the last column. How does the resulting table compare to the original one?

Exercise 36. Suppose TABLE's inter-column space unit is $1/8$ inch and its inter-column space factor is 3. What inter-column spacing results from these formats:

(1) '`| | | |`'; (2) '`s4| | | |`';
(3) '`o0| | | |`'; (4) '`|s2| |s6|o4|o2| |s0|`'?

3.1.6. KERN KEYS — '`i`', '`j`', '`k`'

A kern is a chunk of space that can't stretch or shrink. '`k`⟨spec$_K$⟩' makes its column wider by placing `k`erns of width ⟨value$_K$⟩ before and after each entry. '`i`⟨spec$_K$⟩' and '`j`⟨spec$_K$⟩' are similar, except that '`i`' only places its kern *before* each entry, while '`j`' only places its kern *after* each entry. Note that the kern keys add space *to* a column, while the inter-column space keys put space *between* columns; to appreciate the distinction, work Exercise 38 below.

If you specify several '`i`'s, '`j`'s, and/or '`k`'s for the same column, the effects are cumulative. For example, '`k1k1`' and '`i1j2i1`' are each equivalent to '`k2`'.

Exercise 37. Assume the defaults of Section 2 are in force. What are the width of the kerns specified by: (1) '`k1`'; (2) '`k`'; (3) '`i3`'; (4) '`j(.5cm)`'?

Exercise 38. '`o`u `j0`' and '`o0 j`u' both visually separate the current column from the next by u units of space, but not in the same way (here $u =$ `1`, or `2`, or `3`, etc.). What are the differences?

Example[10] The open-form of the AT&T COMMON STOCK table illustrated a use for '`k`'. '`i`' can be used in conjunction with TABLE's `\BackSpace` command

[10] This material can be skipped on first reading.

to indent column entries, as in the following SIDE EFFECTS table, excerpted from *The New England Journal of Medicine* **317** (1987), 413:

Percentages of Patients Reporting Side Effects

SYMPTOM	PLACEBO ($N = 23$)	COP 1 ($N = 25$)
Local		
Soreness	35	92
Itching	22	64
Swelling	17	88
Other		
Headache	39	32
Nausea	17	24
Vomiting	4	4

This is produced in part by

```
\BeginTable
  \def\BS{\BackSpace}
  \BeginFormat
s8| li2              | c          | c          | .
  " \BS2 Local       "                         \\+30
  " Soreness         " 35         " 92         " \\
  " Itching          " 22         " 64         " \\
  " \BS2 Other       "                         \\+30
  " Headache         " 39         " 32         " \\
  " Nausea           " 17         " 24         " \\
\EndTable
```

The '`i2`' format spaces each entry in the first column over 2 units; '`\BackSpace 2`' undoes this indentation by moving back 2 units. (In general, `\BackSpace` takes $\langle \text{Spec}_K \rangle$ for its argument.)

Exercise 39. Code the column headings in the SIDE EFFECTS table.

Exercise 40. Use '`j`' and `\BackSpace` to code the PECULIAR ALIGNMENT table to the right.

Here's a peculiar alignment you will probably never use.

Example In Section 1 you learned how to align the third column of the AT&T COMMON STOCK table using '~'s. Exercise 16 asked you to do the alignment with '`n`'. It's worth considering yet another approach because of the lessons to be learned from it. The idea is to split the column in two at the decimal points, coding it as

```
\BeginTable \def\C{\JustCenter} \BeginFormat
  | ro0         | l            | .  \_
  | \use2 \JustCenter Dividend | \\ \_
  |        \$2  "  .60         | \\ \_
  |          2  "  .70         | \\ \_
\EndTable
```

and so on. But this code produces

Dividend
$2.60
2.70

which is not what's wanted. The problem is that the word 'Dividend' is wider than the numeric values. In any situation like this, where some spanning element is over-wide, TEX's convention is to allocate the excess width to the right-most column being spanned. That's what happened here — the '`.60`' column ended up wider than the '`$2`' column. This imbalance can be corrected by using '`i`' and '`j`' to increase the width of the numeric entries to the point where 'Dividends' is no longer over-wide. With the format changed to '`| ro0i2 | lj2 |`' you get the desired result:

Dividend
$2.60
2.70

Exercise 41. Use '`k`' and '`r`' to set the second column of the WORLD POPULATION table so that the column heading is centered over the population figures.

3.1.7. COLUMN WIDTH KEY — '`w`'

'`w`⟨spec$_{CW}$⟩' forces the width of its column to be at least ⟨value$_{CW}$⟩. The column will be wider than this amount only if some entry is. For example, the format '`| cw(1in) | cw(1in) |`' specifies two centered columns, each *at least* 1 inch wide. If none of the column entries is wider than this, each column will be *exactly* 1 inch wide. You can use this technique to set up several columns of equal width.

'`w`' can also be used in the form '`w[`⟨sample entry⟩`]`'. After all the keys for the column have been read, '⟨sample entry⟩' is typeset according to the stipulated column format and the width of the result is used in place of ⟨value$_{CW}$⟩. For example, '`lmw[a+b]`' specifies a left-adjusted math-mode column with a width at least that of the expression '$a + b$'.

If you make more than one '`w`' specification for a given column, the last one prevails.

Exercise 42. Suppose the defaults of Section 2 are in force. What minimum column widths are specified by: (1) '`w1`'; (2) '`w`'; (3) '`w5`'; (4) '`w(2.5cm)`'; (5) '`Bw[width]`'?

3.1.8. PARAGRAPH MODE KEY — '`p`'

The key '`p`⟨spec$_{CW}$⟩' sets each entry in its column in what may be loosely called a paragraph mode with a line length of ⟨value$_{CW}$⟩. (The TEXnical terms for "paragraph mode" and "line length" are "internal vertical mode" and "`\hsize`".) A '`p`' entry can be a single line, a whole paragraph, or even several paragraphs, with line breaks chosen automatically by TEX. It can contain displayed equations, hanging indentation, and other such vertical mode material. In short, you can think of it as a mini-page.

TABLE places the upper (respectively, lower) half of one its standard struts on the top (respectively, bottom) line of each '`p`' entry. These half-struts ensure an even vertical spacing between consecutive '`p`' entries; they also separate '`p`' entries from horizontal lines in the table.[11]

By default, '`p`' entries are set ragged-right because paragraphs in tables are typically narrow, and narrow paragraphs generally look best set in that style. Moreover, first lines of paragraphs are not indented. You can, however, choose any style you want by placing the instruction

`\EveryTableParBox={`⟨style commands⟩`}`

in your input file. Here ⟨style commands⟩ specifies commands for TEX to read before it works on every '`p`' entry. To get ordinary right-justified paragraphs, leave ⟨style commands⟩ blank. To get such paragraphs without the usual paragraph indentation, specify ⟨style commands⟩ as '`\parindent=0pt`'. To set really narrow paragraphs like those in the SHORT STORY example in Chapter 6 of *The TEXbook*, specify ⟨style commands⟩ as, say,

'`\tolerance=10000 \hbadness=10000 `'.

These commands instruct TEX not to complain about any under- or over-full boxes as it goes about right-justifying lines. The blank after '`\hbadness=10000`' is important; don't omit it (see Exercise 44 below).

If you place the `\EveryTableParbox` instruction in the prologue to a table, it will apply just to that table. If you place the instruction outside a table, it will apply to all subsequent tables.

Exercise 43. Suppose the defaults of Section 2 are in force. What line sizes are specified by: (1) '`p1`'; (2) '`p`'; (3) '`p30`'; (4) '`p(3in)`'?

Exercise 44. B. L. User entered '`\EveryTableParBox={\parindent=0pt \tolerance=10000 \hbadness=10000}`' and everything worked just fine until TABLE typeset a '`p`' entry that began '`4 score and 7 years ...`' — the '4' disappeared! Explain why.

Exercise 45. What paragraph style is specified by

```
'\EveryTableParBox={%
  \noindent
  \hangafter=1
  \hangindent=\parindent}'?
```

Exercise 46. Guess how TABLE specifies `\EveryTableParBox` to set up its default paragraph style.

Exercise 47. Set the NEW YORK AREA ROCKS table using right-adjusted paragraphs with no indentation on the top line. For simplicity, don't switch to nine point type after the first two lines of the table, as the original does.

---

[11] `\OpenUp`*hd* increases the height of the top half-strut by *h* strut units and the depth of the bottom half-strut by *d* strut units.

Exercise 48. Set the following CLINICAL CRITERIA table, excerpted from *The New England Journal of Medicine* **317** (1987), 399. Specify eight point type for the body of the table, nine point for the heading. (TEX has an easier time composing narrow paragraphs when they're set in small type.)

Clinical Criteria for Abnormal Organ Function in the 27 Patients

	14 SURVIVORS	13 WHO DIED
Cardiovascular: dopamine therapy for mean arterial pressure >45 mm Hg in the absence of hypovolemia (pulmonary artery wedge pressure >6 mm Hg)	3	6
Renal: serum creatinine >300 μmol/liter in preceding 24 hr	6	6
Gastrointestinal bleeding: fresh blood from a nasogastric tube and/or melena or fresh blood from rectum, with a fall in hemoglobin of >2 g/dl	1	2

3.1.9. VERTICAL LINE KEYS — '|', '\|'

The '|'s in the format section are actually keys, their primary function being to show which columns the other keys apply to. They have, however, another function — they govern the thickness of the vertical lines in the table. When the format keys for a given column and the one immediately following it are separated by '|⟨spec$_{LT}$⟩', any vertical rule drawn in the table between these two columns will have a thickness of ⟨value$_{LT}$⟩.[12]

Exercise 49. Suppose the defaults of Section 2 are in force. How thick are the vertical lines in the table when the format is '`|(1pt) c | c |0 c |1 c |2 c |3 c |4`'?

The[13] ruled tables which TABLE constructs look best when drawn with solid lines, and you are strongly encouraged to stick with them. TABLE will, however, begrudgingly let you specify other kinds of rules by using the format key

'`\|{`⟨alternate vertical rule⟩`}`'

in place of the usual '|'. Here ⟨alternate vertical rule⟩ has to be expressed in the form of what Chapter 22 of *The TEXbook* calls a template. For example, to specify the double vrule ||, enter ⟨alternate vertical rule⟩ as

'`\vrule \hskip 2pt \vrule #`',

or, somewhat more meaningfully, as

'`\span\DoubleVrule`',

[12] You can vary the thickness of vertical lines on a row-by-row basis. See the `\|` command in Section 4.5.

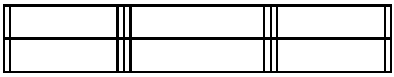
[13] This material can be skipped on first reading.

having previously defined `\DoubleVrule` by

'`\def\DoubleVrule{\vrule \hskip 2pt \vrule ##}`'.

(In general, when you store a template in a macro you have to enter '`#`' as '`##`' and you have to type `\span`[14] before the macro when you invoke it.)

Exercise 50. Code the following PIN-STRIPED table:



3.1.10. "DO-IT-YOURSELF" KEYS — '`a`', '`b`', '`\{`'

From time to time, you may want to specify a column format that can't be handled with the previous keys. TABLE's "do-it-yourself" keys '`a`', '`b`', and '`\{`' allow you to do just that.

'`b{`⟨stuff⟩`}`' sets things up so that TEX reads ⟨stuff⟩ `b`efore each column entry. '`a{`⟨stuff⟩`}`' is similar, specifying ⟨stuff⟩ for TEX to read `a`fter each column entry. For example, '`b{$} a{$}`' duplicates the function of TABLE's math-mode key '`m`' by placing '`$`'s before and after each entry.

If you specify several '`a`'s and/or '`b`'s for the same column, the results are cumulative "from the inside out". For example, '`b`⟨stuff$_1$⟩ `b`⟨stuff$_2$⟩ `b`⟨stuff$_3$⟩' is equivalent to '`b`⟨stuff$_3$stuff$_2$stuff$_1$⟩', whereas '`a`⟨stuff$_1$⟩ `a`⟨stuff$_2$⟩ `a`⟨stuff$_3$⟩' is equivalent to '`a`⟨stuff$_1$stuff$_2$stuff$_3$⟩'.

In the preceding discussion, ⟨stuff⟩ can be any string of tokens (i.e., characters and control sequences) that is properly balanced with respect to braces. Thus, ⟨stuff⟩ could be '`{}`', but not '`{`' or '`}`'.

The key '`\{`' puts braces around each entry. '`\{`' can be used with '`a`' and '`b`'. For example, the key sequence '`\{ b{$\displaystyle} a{$}`' first encloses each entry in braces (via '`\{`'), then places '`$\displaystyle`' before the bracketed entries (via '`b`'), and finally places '`$`' after the bracketed entries (via '`a`'); the net result duplicates the function of TABLE's '`M`' key by placing '`$\displaystyle{`' before, and '`}$`' after, each entry. In general, `\{`'s braces will enclose the ⟨stuff⟩ inserted by previous '`a`'s and '`b`'s, but not that inserted by subsequent '`a`'s and '`b`'s. For example, the key sequence '`b`⟨stuff$_1$⟩ `a`⟨stuff$_2$⟩ `\{ b`⟨stuff$_3$⟩ `a`⟨stuff$_4$⟩ `\{ b`⟨stuff$_5$⟩' places '⟨stuff⟩$_5$`{`⟨stuff⟩$_3$`{`⟨stuff⟩$_1$' before, and '⟨stuff⟩$_2$`}`⟨stuff⟩$_4$`}`' after, each entry.

Exercise 51. Use '`a`' and '`b`' to duplicate the functions of: (1) the key '`B`', which puts '`\bf`' before each entry; (2) the key '`\m`', which puts '`${}`' before, and '`$`' after, each entry; (3) the key '`\M`', which puts '`$\displaystyle{{}`' before, and '`}$`' after, each entry. ('`\m`' and '`\M`' also imply '`l`', but we're ignoring that for the purposes of this exercise.)

Exercise 52. Show how to use '`a`' and '`b`' to place '`\EndTableParbox`' after, and '`\BeginTableParbox{3in}`' before, each entry.

---

[14] `\span` is a TEX primitive; see the bottom of page 238 of *The TEXbook*.

Exercise 53. Section 3.1.2 stated that "if you make more than one font selection for a column, the first one prevails." Explain this rule, given that TABLE defines '`B`', '`I`', etc., in terms of '`b`'.

3.1.11. CONVENIENCE KEYS — '`.`', '`*`'

You can use the key '`.`' as a convenient substitute for `\EndFormat`. Only a "free-standing" '`.`' will end a format; the '`.`'s in specifications like '`n2.3`', '`N[0.00]`', '`w(.5in)`', and '`p(3.25in)`' don't count.

TABLE's '`*`' key makes it easy to specify a repeating sequence of keys: '`*{`*n*`}{`⟨keys⟩`}`' is equivalent to entering ⟨keys⟩ *n* times in a row. If the repeat count *n* is a single digit, you can omit the braces around it. ⟨keys⟩ can contain additional '`*`' expressions. For example, '`*4{c|*3{r|} }`' expands to

'`c|r|r|r| c|r|r|r| c|r|r|r| c|r|r|r| `'.

Exercise 54. What's the easy way to enter

'`\BeginFormat s4| rmo0|\m | rmo0|\m | rmo0|\m | \EndFormat`'?

Exercise 55. What mistake did B. L. User make when he abbreviated

'`c | c | c | c | c | c | c | c | c | c | c | `'

to '`*11{c | }`'?

3.2. DEFINING NEW FORMAT KEYS

With TABLE's key system, you're not limited to just the built-in keys. Rather, you can use TABLE's `\NewFormatKey` command to define new keys as you see fit. The name of a new key can be any single character[15] or control sequence[16] that's not already in use as a key.

One of the simplest uses of `\NewFormatKey` is to define a key which stands for a string of some other keys. Consider, for example, the format

```
s4| rmo0|\m | rmo0|\m | rmo0|\m |
```

for the PAIRED EQUATIONS table (Exercise 11). The key sequence '`rmo0|\m`' occurs here three times, and would occur many times over if you were to set many tables like that one. It would be convenient to be able to specify this sequence with a single keystroke. The command

```
\NewFormatKey q{\ReadFormatKeys rmo0|\m}
```

---

[15] Actually, there are a few exceptions — characters with category code 0, 1, 2, 5, 9, 10, 14, or 15 can't be used as keys. Under the conventions of Plain TEX, this excludes '`\`', '`{`', '`}`', '`^^M`', '`␣`', and '`%`'.

[16] A control sequence key can have the same name as a TEX primitive or macro without causing any problem. For example, there's no conflict with TABLE's use of '`\{`' as a key and Plain TEX's definition of '`\{`' as a macro.

makes this possible by defining the key '`q`' (for equations) to be an abbreviation for '`rmo0|\m`'. Here's how '`q`' works: The command `\BeginFormat` instructs TABLE to start reading the keys in the format section, processing each in turn. When TABLE reaches a '`q`', it passes control to TEX to process the text '`\ReadFormatKeys rmo0|\m`'. The `\ReadFormatKeys` command instructs TEX to tell TABLE to resume reading keys. The next keys TABLE sees are '`r`', '`m`', '`o`', '`|`', and '`\m`' (the '`0`' is an argument to '`o`'); then TABLE sees whatever key follows '`q`' on the format line. With '`q`' defined as above, the format for the PAIRED EQUATIONS table simplifies to '`s4|q|q|q|`'.

Exercise 56. Explain how the key '`9`' on page 12 works.

Exercise 57. How many '`|`'s are there in the format '`s4|q|q|q|`'?

Exercise 58. Show how to define the key '`Q`' to be equivalent to '`s4|q|q|q|`'. Use '`Q`' to answer Exercise 54.

Exercise 59. Show how to define the key '`\left`' to be a synonym for '`l`'.

Exercise 60. Guess how TABLE defines the font-selection key '`B`' in terms of the "primitive" do-it-yourself key '`b`'.

Exercise 61. Guess how TABLE defines the math key '`\m`' in terms of '`l`', '`b`', and '`m`'.

Keys can have arguments. For example,[17]

```
\NewFormatKey K#1{\ReadFormatKeys b{\kern #1} a{\kern #1}}
```

defines a key '`K`' which duplicates part of the function of TABLE's built-in kern key '`k`', in that '`K{`⟨dimen⟩`}`' places a kern of width ⟨dimen⟩ before and after each entry in its column.

For a more involved example, suppose you have macros `\viiipt`, ..., `\xiipt` that switch to eight point, ..., twelve point type, and you want to define a Point-size key '`P`' such that '`P{8}`' will set each entry in its column in eight point type, '`P{9}`' will set each entry in its column in nine point type, and so on. The following definition would do the job:

```
\NewFormatKey P#1{%
  \ifnum #1=8
    \def\SetSize{\ReadFormatKeys b{\viiipt}}
  \else
    \ifnum #1=9
      \def\SetSize{\ReadFormatKeys b{\xipt}}
    \else
      \ifnum #1=10
        \def\SetSize{\ReadFormatKeys b{\xpt}}
```

[17] The rest of this section presumes some familiarity with TEX's macro facilities, described in Chapter 20 of *The TEXbook*.

```
      \else
        \ifnum #1=11
          \def\SetSize{\ReadFormatKeys b{\xipt}}
        \else
          \ifnum #1=12
            \def\SetSize{\ReadFormatKeys b{\xiipt}}
          \else
            \message{Point size #1 unavailable; using xpt}
            \def\SetSize{\ReadFormatKeys b{\xpt}}
          \fi
        \fi
      \fi
    \fi
  \fi
  \SetSize}
```

The `\ifnum` clauses compare '`P`'s argument to 8, 9, ..., 12 in order to determine the appropriate argument for '`b`'.

Exercise 62. What size type would you get from '`P{14}`'?

Exercise 63. Define a key '`C`' such that '`C{`⟨dimen⟩`}`' would place the commands

```
$\vcenter \bgroup \normalbaselines
  \parindent=0pt \hsize=⟨dimen⟩ \strut
```

before, and the commands

```
\strut \egroup $
```

after, each entry in its column.

Exercise 64. (For TEXperts) Guess how TABLE's repeat key '`*`' is defined.

The general form of the `\NewFormatKey` command is

`\NewFormatKey`⟨key⟩⟨parameter text⟩`{`⟨replacement text⟩`}`

where the ⟨parameter text⟩ and ⟨replacement text⟩ are governed by exactly the same rules that apply in connection with TEX's `\def` command; these rules are set out on pages 203–204 of *The TEXbook* and will not be repeated here. After ⟨key⟩ has been defined, it functions as follows: Whenever TABLE encounters ⟨key⟩ during its scan of a format section, it passes control to TEX, which processes the ⟨replacement text⟩, taking the ⟨parameter text⟩ into account. Sooner or later, control must be passed back to TABLE via the `\ReadFormatKeys` command. The complexity of the keys you can define in this way is limited only by your skill at writing TEX macros.

If you define a key in the prologue to a table, the key will apply just to the format for that table. If, however, you define a key outside a table, the key can be used to format any subsequent table.

3.3. FORMAT DIAGNOSTICS

TABLE gives you the opportunity to see exactly how it's interpreting your format specifications. This feature is useful, for example, if you've put together a new column format and you want to check if it's being implemented as you expected. Or perhaps some table isn't coming out as intended due to a misunderstanding on your part about what some format key does; you may be able to spot the source of the trouble by examining TABLE's diagnostic output.

That output consists of the column templates for the `\halign` that will be used to construct your table. `\halign`s and templates are explained in Chapter 22 of *The TEXbook*. Pages 235–238 are especially pertinent to the present discussion, and you might want to review those pages before reading further.

To get a listing of the templates TABLE constructs for a table, place the command

```
\TracingFormats=1
```

in the prologue to that table. (If you specify '`\TracingFormats=1`' outside a table, you'll get format listings for all subsequent tables.) The listing is written to both your terminal and log file. Three such listings are exhibited below to give you some experience in reading them.

First, here (slightly edited) is what '`\TracingFormats=1`' produces for the format '`| c | c | c |`' of the AT&T COMMON STOCK table:

```
TABLE FORMAT
Column: Template
 *c: ##\tabskip 0pt
  r: \hfil \vrule \!thWidth 2\!taLTU ##\hfil
       \tabskip 7.5pt plus 1.5fil minus 3.75pt
 1c: \hfil ##\hfil
  r: \hfil \vrule \!thWidth 2\!taLTU ##\hfil
 2c: \hfil ##\hfil
  r: \hfil \vrule \!thWidth 2\!taLTU ##\hfil
 3c: \hfil ##\hfil
  r: \hfil \vrule \!thWidth 2\!taLTU ##\hfil
       \tabskip 0pt
 *c: ##\tabskip 0pt
```

There are more templates here than you might have expected because this table not only has three *data* columns containing the entries 'Year', 'Price', 'Dividend', etc., but it also has four *rule* columns holding the vertical lines surrounding the data columns. Moreover, TABLE places an extra data column at each side of a table; nothing is ever (intentionally) entered in these columns, but their presence simplifies TABLE's internal logic. Each of the various data and rule columns has a template.

The templates for the three main data columns are labeled '`1c`', '`2c`', and '`3c`'. Each template is '`\hfil ##\hfil`', corresponding to the centering key '`c`'. (Actually, each template has only one '`#`' sign, as TEX requires; the

listing has '`##`' because the templates are written out using TEX's `\write` command, and `\write` doubles each '`#`' in its argument.)

The templates for the four rule columns are labeled simply '`r`'. '`\hfil \vrule \!thWidth 2\!taLTU ##\hfil`' is TABLE's slightly cryptic way of writing '`\hfil \vrule width 2\LineThicknessUnit #\hfil`'. (The cryptic form uses less of TEX's memory and takes up less space in the format listing.) These templates were generated by the '`|`' keys in the format; the '2' multiplying '`\!taLTU`' is the default value of `\LineThicknessFactor`.

The template for the first rule column sets TEX's tabskip glue to 7.5 pt plus 1.5 fil minus 3.75 pt. This is half of TABLE's default

(inter-column space factor) × (inter-column space unit)
= 3 × (.5 em plus 1 fil minus .25 em),

for inter-column space (with 1 em = 10 pt for ten point roman type).

The tabskip glue in the template for the first dummy data column (labeled '`*c`') is the default value of a special glue, `\LeftTabskip`, that hasn't been mentioned yet. Similarly, the tabskip glue in the template for the last rule column is the default value of `\RightTabskip`. These special glues are discussed in Section 5.1.

Next, here's the listing '`\TracingFormats=1`' produces from the format

```
s4| rmo0 | \m | rmo0 | \m | rmo0 | \m |
```

for the PAIRED EQUATIONS table:

```
TABLE FORMAT
Column: Template
 *c: ##\tabskip 0pt
  r: \hfil \vrule \!thWidth 2\!taLTU ##\hfil
       \tabskip 7.5pt plus 1.5fil minus 3.75pt
       \tabskip 10pt plus 2fil minus 5pt
 1c: \hfil $##$\tabskip 0pt
  r: \hfil \vrule \!thWidth 2\!taLTU ##\hfil
 2c: ${}##\tabskip 10pt plus 2fil minus 5pt$\hfil
  r: \hfil \vrule \!thWidth 2\!taLTU ##\hfil
 3c: \hfil $##$\tabskip 0pt
  r: \hfil \vrule \!thWidth 2\!taLTU ##\hfil
 4c: ${}##\tabskip 10pt plus 2fil minus 5pt$\hfil
  r: \hfil \vrule \!thWidth 2\!taLTU ##\hfil
 5c: \hfil $##$\tabskip 0pt
  r: \hfil \vrule \!thWidth 2\!taLTU ##\hfil
 6c: ${}##\tabskip 10pt plus 2fil minus 5pt$\hfil
  r: \hfil \vrule \!thWidth 2\!taLTU ##\hfil
       \tabskip 0pt
 *c: ##\tabskip 0pt
```

Notice that the rule column templates are still present, even though no vertical lines were drawn in this table; TABLE always alternates data columns with rule columns. The first rule column template contains two tabskip specifications;

the first is TABLE's default, the second came from the key '`s4`'. If a template contains more than one tabskip specification, TEX uses the last one; this is how TABLE's default gets overriden. In the template for column `1c`, the `\hfil` came from the key '`r`', the '`$`'s from the key '`m`', and the '`\tabskip 0pt`' from '`o0`'; TEX removes tabskip specifications after it has read them, so the final template for this column will be just '`\hfil $##$`'. The template for column `2c` contains the tabskip specification TABLE automatically put there to restore the tabskip glue cancelled by '`o0`'; the rest of that template, to wit '`${}#$\hfil`', came from the key '`\m`'.

Exercise 65. TABLE always halves the inter-column space specified by the keys '`s`' and '`o`' when it sets TEX's tabskip glue. Guess why.

Finally, here's the listing '`\TracingFormats=1`' produces for the format

```
\|{\span\D} cw5 \|{\span\T} cw7 \|{\span\T} cw5 \|{\span\D}
```

for the PIN-STRIPED table of Exercise 50, with the macros `\D` and `\T` being defined by

```
\def\D{\vrule \hskip 2pt \vrule ##}
\def\T{\vrule \hskip 2pt \vrule \hskip 2pt \vrule ##}
```

(TEX replaces the '`##`'s in these macros by '`#`' when it reads them):

```
TABLE FORMAT
Column: Template
 *c: ##\tabskip 0pt
  r: \span \D \tabskip 7.5pt plus 1.5fil minus 3.75pt
 1c: \hfil ##\hfil
  w: 25pt
  r: \span \T
 2c: \hfil ##\hfil
  w: 35pt
  r: \span \T
 3c: \hfil ##\hfil
  w: 25pt
  r: \span \D \tabskip 0pt
 *c: ##\tabskip 0pt
```

The `\span` commands in the rule column templates tell TEX to expand `\D` and `\T` when it's reading those templates. The `w` lines following `1c`, `2c`, and `3c` in the listing show the minimum column widths specified by the '`w`' keys in the format; 35 pt is 7 times TABLE's default column width unit of 0.5 em (= 5 pt for ten-point type).

There's a stronger form of the tracing command: '`\TracingFormats=2`' gives you not only the column templates but also the `\halign` preamble constructed from them. (There is no new information in the preamble, though.)

You can also trace TABLE's scan of format keys. '`\TracingKeys=1`' writes a message to your terminal and your log file each time you define a new format

key. '`\TracingKeys=2`' not only reports new keys, but also reports each time a key is referenced. (Some keys invoke other keys, so you'll get reports about more than just the keys you specified in your format). This feature is useful if you want to know exactly where TABLE is working on your format when some particular error message arises.

Here is the output from '`\TracingKeys=2`' for the PIN-STRIPED table:

```
KEY: "\|"
KEY: "c"
KEY: "\LeftGlue"
KEY: "\RightGlue"
KEY: "w"
KEY: "\|"
KEY: "c"
KEY: "\LeftGlue"
KEY: "\RightGlue"
KEY: "w"
KEY: "\|"
KEY: "c"
KEY: "\LeftGlue"
KEY: "\RightGlue"
KEY: "w"
KEY: "\|"
```

`\LeftGlue` and `\RightGlue` are built-in keys that weren't mentioned before because you really don't need to know about them. To settle your curiosity, though, '`\LeftGlue{`⟨glue⟩`}`' specifies the ⟨glue⟩ to be placed at the extreme left of a template and, similarly, '`\RightGlue{`⟨glue⟩`}`' specifies the ⟨glue⟩ to be placed at the extreme right of a template. TABLE defines the key '`c`' as '`\LeftGlue{\hfil}\RightGlue{\hfil}`'.

Exercise 66. Guess how TABLE defines '`l`' and '`r`'.

4. COMMANDS

This section lists all of TABLE's commands, grouped by function. As was the case with TABLE's format keys, most of the commands are amply illustrated in Section 1; the examples here are limited to features that were previously only lightly touched on, if at all. Before proceeding, you should review the material on TABLE's quantum systems in Section 2. The command summary in Appendix C is convenient for quick reference.

4.1. BEGINNING AND ENDING A TABLE

The commands `\BeginTable` and `\EndTable` delineate the table environment. `\BeginTable` has several options — namely, `[c]`, `[t]`, `[b]`, and `[u]` — that are discussed in Section 5.1. Some of TABLE's commands, `\-` and `\\` in particular, have their stated meaning only within the table environment. Appendix C details which commands are of limited scope and which are not.

4.2. BEGINNING AND ENDING A FORMAT SECTION

The commands `\BeginFormat` and `\EndFormat` delineate the format section of a table. TABLE accepts the format key '`.`' as a substitute for `\EndFormat`.

4.3. DEFINING NEW FORMAT KEYS

The command `\NewFormatKey` defines a new format key. This command and its companion, `\ReadFormatKeys`, are described in detail in Section 3.2.

4.4. ENDING AN INPUT ROW

The command `\\` ends an input row. Ordinarily `\\` inserts a strut into the table row, so as to maintain an even spacing of the rows of the table. You can "zero out" this strut by suffixing `\\` with '`0`': no strut is inserted in a table row terminated by '`\\0`'. This feature is typically used in conjunction with the line drawing commands `\-` and `\=`. Moreover, you can increase the height and depth of the usual strut by h and d strut units, respectively, by suffixing `\\` with '`+{`h`}{`d`}`'. Here h and d must be integers (positive, negative, or zero); if either is a single digit, the braces surrounding it may be omitted. TABLE's strut unit is discussed in Sections 4.9 and 4.11.

Exercise 67. What's the difference between '`\\0`' and '`\\+00`'?

Exercise 68. Hoping to decrease the height and depth of the usual end-of-row strut by 2 strut units and 3 points, respectively, B. L. User entered '`\\-2(-3pt)`'. What mistakes did he make?

4.5. SEPARATING ENTRIES

The commands '`|`' and '`"`' are used to separate the entries on each row of a table. Moreover, a '`|`' or '`"`' must precede the first and follow the last entries on each row.

When a '`|`' is used to separate two entries, a vertical line is drawn across the row midway between the columns containing those entries. The breadth of this line is determined by the line-thickness specification carried by the corresponding '`|`' in the table format; for example, if the format has '`|4`', the breadth of the line will be four times TABLE's line-thickness unit. If, however, the format has a '`\|`' instead of a '`|`', the ⟨alternate vertical rule⟩ specified by '`\|`' is used instead of the usual solid line. When a '`"`' is used to separate two entries, nothing is drawn across the row between the columns containing those entries. Similar remarks apply to '`|`'s and '`"`'s at the ends of the row.

To typeset the character '`|`' in a table, enter '`\VBar`'; to typeset '`"`', enter '`\DQuote`'.

Exercise 69. Consider the following code:

```
\BeginTable
  \def\DVR{\hfil \vrule \hskip 2pt \vrule ##\hfil}
  \BeginFormat
  |  c  |  c  |3 c  |(1pt) c \|{\span\DVR} c \|{\span\DVR} . \_
  "  1  |  2  "  3  |    4    "       5       | \\ \_
\EndTable
```

What is drawn across the first (and only) row: (1) before the 1st column; (2) between the 1st and 2nd columns; (3) between the 2nd and 3rd columns; (4) between the 3rd and 4th columns; (5) between the 4th and 5th columns; (6) after the 5th column?

The[18] command `\|`[19] can be used in place of '`|`' and '`"`' as a column separator. `\|` provides considerable flexibility as to what is placed between columns in a table row:

(1) '`\|`⟨spec$_{LT}$⟩' draws a vertical line of breadth ⟨value$_{LT}$⟩ across the row. Remember that ⟨spec$_{LT}$⟩ stands for either a blank, an unsigned integer, or '`(`⟨dimen⟩`)`'.

(2) If `\|` is immediately followed by a '`*`', a user-specified "pseudo vrule" is drawn across the row. This pseudo vrule is defined by the parameterless macro `\PseudoVrule`. For example

```
\def\PseudoVrule{\hfil \vrule \hskip2pt \vrule \hfil}
```

defines `\PseudoVrule` to be the "double vrule" ||.

(3) If `\|` is immediately followed by `{`⟨tokens⟩`}`, the text defined by those tokens is used in place of the usual vertical line. If only one token is involved, the

---

[18] This materical can be skipped on first reading.

[19] Don't confuse the *command* `\|` (which is used as a column separator in the rows of the table) with the *key* '`\|`' (which is used in the table format).

braces surrounding it may be omitted. To place a '`*`' between columns, however, you must specify '`\|{*}`', since '`\|*`' is by rule 2 equivalent to '`\|\PseudoVrule`'.

These rules have two important consequences: (1) Something, possibly a blank, must intervene between `\|` and the following entry (otherwise the first token of the entry will be placed between the columns involved). (2) Something, possibly a blank, must intervene between two consecutive `\|` commands. The various uses of `\|` are illustrated by the code

```
\BeginTable
  \def\PseudoVrule{\hfil \vrule \hskip 2pt \vrule \hfil}
  \def\Vdots{$\vcenter{\baselineskip=4pt \lineskiplimit=0pt
    \hbox{.}\hbox{.}\hbox{.}}$}
  \BeginFormat
  | w4  |  w4  |  w4   |   w4   |   w4    | .  \_
  |  A \|   B \|*  C  \|{*} D  \|\Vdots E | \\ \_
  |  1 \|6  2 \|*  3  \|{*} 4  \|\Vdots 5 | \\ \_
\EndTable
```

which produces the ALTERNATE VERTICAL RULES table

A	B	C	*	D	⋮	E
1	2	3	*	4	⋮	5

4.6. SPANNING COLUMNS

The `\use` command is used to span columns: '`\use{`*c*`}`' instructs TABLE to utilize the space of the next *c* columns (including the intervening white space) and the format of the last of those columns. If *c* is a single digit, the braces around it may be omitted. `\use`, if present, must come first in an entry.

Exercise 70. What unexpected result did B. L. User get when he entered '`\use11 This text spans eleven columns.`'?

Exercise 71. In the table defined by

```
\BeginTable
  \BeginFormat
  | cB | rI | lR | cS | rT | .
  " 1  "  2 "  3 "  4 "  5 " \\
  "    " \use3 Span   "    " \\
\EndTable
```

which columns are spanned by the word "Span", and how is that word formatted?

When a spanning entry is too wide to fit within the specified columns, the width of the last column is automatically increased to accommodate the entry. If this is not to your liking, you can (1) use the kern keys '`i`', '`j`', '`k`' to increase the width of the columns being spanned, as explained in Section 3.1.6, and/or (2) use the inter-column space keys '`s`' and '`o`' to increase the space between the columns being spanned.

Exercise 72. Explain how to put half an inch of white-space between the 3rd, 4th, and 5th columns of a table.

4.7. DRAWING HORIZONTAL LINES

The command `\_`⟨spec$_{LT}$⟩ draws a horizontal line of breadth ⟨value$_{LT}$⟩ across the entire table; this line will join up with any vertical lines drawn at the edges of the table. `\_` can't be used as a column entry; it must immediately follow either `\EndFormat`, a `\\` command, or a `\Vspace` command.[20]

The command `\-`⟨spec$_{LT}$⟩ draws a horizontal line of breadth ⟨value$_{LT}$⟩ across the column in which it appears. The line lies along the baseline of the row and is exactly as wide as the column.

The command `\=`⟨spec$_{LT}$⟩ is like `\-`⟨spec$_{LT}$⟩ in that it draws a line of breadth ⟨value$_{LT}$⟩ across its column, but it is different in these respects: (1) A `\=` line extends half way into adjacent inter-column white space so as to join up with any neighboring vertical lines; (2) If that white space is expanded or compressed by any of the commands in Section 4.12, a `\=` line will lengthen or shorten accordingly; (3) '`&`'s must be used in place of the usual '`|`'s or '`"`'s as the column separators on each side of `\=`.

Both `\-` and `\=` are ordinarily used in rows that are terminated by `\\0`. When preceded by `\use{`c`}`, `\-` and `\=` lines span the next c columns.

Exercise 73. Show how to use `\=` to achieve the effect of `\_`.

Exercise 74. Code the following HORIZONTAL LINES table, making the lines in the frame twice as thick as the other lines:

xxxxx	
	xxxxx

Exercise 75. Stretch the preceding table half an inch horizontally by placing the command '`\WidenTableBy{.5in}`' (see Section 4.12) in the prologue. How does the new table compare to the original?

4.8. REFORMATTING ENTRIES

Some entries (such as column headings) in a table may need to be set in a different format than that specified for their column. TABLE provides several commands for reformatting an entry.

The commands `\Left{`⟨entry⟩`}`, `\Right{`⟨entry⟩`}`, and `\Center{`⟨entry⟩`}` place ⟨entry⟩ into its column flush left, flush right, or centered, respectively,

[20] Or TEX's primitive `\noalign{...}` command (see page 237 of *The TEXbook*). `\noalign` itself can be used immediately after either `\EndFormat`, a `\\` command, a `\_` command, a `\Vspace` command, or another `\noalign` command.

regardless of whether the column format specifies '`l`', '`r`', or '`c`'. The remaining keys, if any, in the column format apply without change. For example, '`\Right{\alpha + \beta}`' will position '$\alpha + \beta$' flush right in a column with a '`cm`' format. Braces must be used unless ⟨entry⟩ is a single character or control sequence.

The commands `\JustLeft`, `\JustRight`, and `\JustCenter` are like `\Left`, `\Right`, and `\Center` in the way that they position an entry, but they totally ignore the format specified for the column. For example, '`\JustLeft Heading`' will place the word 'Heading' flush left in its column, regardless of what the column format may stipulate. The `\Just...` commands do not require braces; indeed, they automatically apply to the entire entry, which they must precede. All six repositioning commands can follow `\use`.

TABLE's most versatile (but slowest) reformatting command is `\ReFormat`, which takes the form

'`\ReFormat[`⟨format keys⟩`]{`⟨entry⟩`}`'.

This command formats ⟨entry⟩ according to ⟨format keys⟩, just as though ⟨format keys⟩ had been specified for the column format.[21] For example, the instruction '`\ReFormat[rm]{\alpha+\beta}`' places $\alpha + \beta$ flush right in its column, regardless of what the column format stipulates. `\ReFormat` can be used with `\use`, which must come first. TABLE provides

'`\Use{`c`}[`⟨keys⟩`]{`⟨entry⟩`}`'

as a convenient substitute for '`\use{`c`}\ReFormat[`⟨keys⟩`]{`⟨entry⟩`}`'. (`\Use` is similar to, but more versatile than, LaTeX's `\multicolumn` command.)

Exercise 76. Give several ways to place the word 'Heading' in italics flush left in a column with a '`cB`' format.

4.9. MAKING STRUTS

Struts play an important role in the construction of a table. TABLE packs the rows of a table tightly together so that the bottom of each row just touches the top of the row beneath it. This is done so that the vertical lines which the '`|`' command draws across a row will join up with corresponding lines in neighboring rows. Struts are needed to keep the rows evenly spaced and to insert some white space between them. TABLE's end-of-row command `\\` automatically inserts a strut into its row. In some circumstances you may want to employ a strut that is higher and/or deeper than usual. This section on TABLE's strut-making commands tells you how to do that.

TABLE maintains a quantum system for struts somewhat like the quantum systems for line thickness, inter-column space, etc., described in Section 2. There is a *strut unit*, specified by the parameter `\StrutUnit`, and *strut height*

[21] ⟨format keys⟩, however, can't contain these keys: '`w`', '`s`', '`o`', '`|`', '`\|`', '`.`', or '`]`'.

and *depth factors*, specified by the parameters `\StrutHeightFactor` and `\StrutDepthFactor`. The strut that `\\` automatically inserts in its row has height (strut height factor)×(strut unit) and depth (strut depth factor)×(strut unit). The defaults for `\StrutHeightFactor` and `\StrutDepthFactor` are 8 and 3, respectively. The default for `\StrutUnit` is $^{12}/_{11}$ point,[22] so that the extent (i.e., height plus depth) of TABLE's usual strut is 12 $\left(= (8+3) \times {}^{12}/_{11}\right)$ points, the normal distance between baselines for ten point type.

You may change the values of TABLE's strut parameters as you see fit, subject to the factor parameters being non-negative integers and the unit parameter being a dimension. For example, to make TABLE's strut have the height and depth of the strut produced by TEX's `\strut` command (8.5 points and 3.5 points, respectively), you could specify

'`\StrutUnit=.5pt \StrutHeightFactor=17 \StrutDepthFactor=7`'.

If you place these commands in the prologue to a table, they will apply just to that table. If, however, you place these commands outside a table, they will apply to all subsequent tables.

Exercise 77. Set the RANDOM table

```
\BeginTable
  \BeginFormat
  |   c   |   c    |  c   |  c   |   cm    | .
  \_
  | abcde " ABCED  " 1234 " ([]) " 2^2_3   | \\
  \_
  | leth  " gjqpy  " !@*+ " ':;? " e^{-a}  | \\
  \_
  | ()     "( )    " ...  " ---  " a_{jik} | \\
  \_
\EndTable
```

using several different settings of the strut parameters, keeping the extent of the strut at 12 points. Try strut height-to-depth ratios of 9 to 4, 8 to 3 (TABLE's choice), 8.5 to 3.5 (TEX's choice), and 7 to 3 (LATEX's choice). Which result do you like best?

To increase the height and depth of the strut for a particular row by h and d strut units, suffix the `\\` command for that row by '`+{`h`}{`d`}`'. To increase the height and depth of the struts for all the rows in a table, place the command

`\OpenUp{`h`}{`d`}`

---

[22] This is a simplification. The default strut unit is actually one-eleventh of TEX's `\normalbaselineskip` parameter, so that the extent of TABLE's strut is the normal distance between baselines, no matter what the size of the type involved. Remember, that if you change the type size, you must subsequently specify `\NormalTableUnits`.

in the prologue to that table. In both situations, h and d must be integers (positive, negative, or zero); if either h or d is a single digit, the braces surrounding it may be omitted. The effects of `\OpenUp` and `\\+` are cumulative. For example, if you were to specify '`\OpenUp{`h_1`}{`d_1`}`' and '`\OpenUp{`h_2`}{`d_2`}`' in the prologue to a table, and '`\\+{`h_3`}{`d_3`}`' for a particular row, the strut for that row would be higher than usual by $h_1+h_2+h_3$ strut units and deeper than usual by $d_1+d_2+d_3$ strut units.

Exercise 78. Assuming TABLE's defaults are in force, what suffix to `\\` would produce a strut that is: (1) higher and deeper than usual by one strut unit each; (2) less high and less deep than usual by one strut unit each; (3) exactly one strut unit high and deep?

Exercise 79. The '`\OpenUp11`' command in the prologue to the RECIPE table puts 2 strut units of space between each pair of rows in that table. The table, however, would look better if the extra space between the first two rows were eliminated. Explain how to remove that space.

Exercise 80. Guess how TABLE defines `\OpenUp` in terms of the strut height and depth factors.

TABLE doesn't insert a strut into a row terminated by '`\\0`'. Such a row has the height of the tallest (respectively, depth of the deepest) item on it.

If you like, you can place a strut of your own making in a row; remember, though, to cancel TABLE's strut with '`\\0`' if your strut is smaller than TABLE's. You can manufacture a strut with the command

`\MakeStrut{`H`}{`D`}`

which makes a strut of height H and depth D, H and D being dimensions. For example, '`\MakeStrut{2pt}{0pt}`' makes a strut 2 points in height and 0 points in depth. Note that `\MakeStrut` doesn't use TABLE's strut quantum system.

Exercise 81. Show how to produce the following LADDER-LIKE table:

Exercise 82. Explain how to use `\MakeStrut` to make a row of a table at least 15 points high and 10 points deep.

TABLE has two other strut-making commands.

`\StandardTableStrut`

produces the strut used by the end-of-row command `\\`, and

`\AugmentedTableStrut{`η`}{`δ`}`

produces a strut that is higher than that strut by η strut units and deeper than it by δ strut units; here η and δ can be any fixed point numbers (positive, negative, or zero). For example, '`\AugmentedTableStrut{1.5}{2.5}`'

produces a strut that is 1.5 strut units higher and 2.5 strut units deeper than TABLE's standard strut.

Exercise 83. Guess how TABLE defines `\StandardTableStrut` in terms of `\MakeStrut`.

Exercise 84. Explain how to use `\AugmentedTableStrut` to achieve the effect of `\\+{`h`}{`d`}`.

Sometimes an entry is larger than TABLE's standard strut. You could instruct TABLE to leave some space above and below such an entry by using `\\+{`h`}{`d`}`, but you'd have to guess at what h and d to use. You can avoid the guesswork by using TABLE's `\Enlarge` command instead of `\\+`:

`\Enlarge{`H`}{`D`}{`⟨entry⟩`}`

typesets ⟨entry⟩ using the format for its column, increasing ⟨entry⟩'s natural height and depth by the dimensions H and D, respectively. The command

`\enlarge{`η`}{`δ`}{`⟨entry⟩`}`

is similar, but uses increments of η and δ strut units; here η and δ can be any fixed point numbers (positive, negative, or zero). If η is a single digit, the braces around it can be omitted; the same goes for δ. For example, to instruct TABLE to leave 3 points of space above and below a large integral sign in a column having a '`cM`' format, specify '`\Enlarge{3pt}{3pt}{\int}`'. To use 3 strut units instead of 3 points, specify '`\enlarge33{\int}`'.

Exercise 85. Set the SPECIAL FUNCTIONS table, using `\enlarge33` instead of `\OpenUp`. How does the result compare to the original?

Exercise 86. Guess how TABLE defines `\enlarge` in terms of `\Enlarge`.

4.10. RAISING AND LOWERING ENTRIES

TABLE doesn't provide a true vertical spanning operation comparable to `\use`. You can, however, do some makeshift vertical spanning — such as raising or lowering an entry a half line — that often accomplishes the same goal as the real thing.

The command

`\Raise`⟨spec$_{RL}$⟩`{`⟨entry⟩`}`

raises ⟨entry⟩ by ⟨value$_{RL}$⟩ and, similarly, the command

`\Lower`⟨spec$_{RL}$⟩`{`⟨entry⟩`}`

lowers ⟨entry⟩ by ⟨value$_{RL}$⟩. Here the subscript "*RL*" is a mnemonic for "Raise or Lower", and the various options for ⟨spec$_{RL}$⟩ and ⟨value$_{RL}$⟩ are as follows:

⟨spec$_{RL}$⟩	⟨value$_{RL}$⟩
⟨null⟩ or a string of one or more blanks	a half-line*
an unsigned integer i	i multiples of `\StrutUnit`
`(`⟨dimen⟩`)`	⟨dimen⟩

* $\frac{1}{2}$(`\StrutHeightFactor` + `\StrutDepthFactor`) × `\StrutUnit`

Except for the built-in default of a half-line for ⟨value$_{RL}$⟩ in the absence of an explicit declaration for ⟨spec$_{RL}$⟩, ⟨spec$_{RL}$⟩ and ⟨value$_{RL}$⟩ operate on the same principles as do ⟨spec$_{LT}$⟩ and ⟨value$_{LT}$⟩, ⟨spec$_{ICS}$⟩ and ⟨value$_{ICS}$⟩, etc. For example, '`\Lower{Heading}`' drops 'Heading' by a half-line, '`\Raise2{Item}`' lifts 'Item' up 2 strut units, and '`\Lower(6pt){Item}`' and '`\Raise(-6pt){Item}`' each lower 'Item' by 6 points.

Exercise 87. Suppose TABLE's strut unit is .5 point and the strut height and depth factors are 17 and 7, respectively. How far would 'Item' be lowered by: (1) '`\Lower {Item}`'; (2) `\Lower4{Item}`'; (3) '`\Lower24{\Item}`'; (4) '`\Raise{Item}`'; (5) '`\Lower(.1in){Item}`'?

`\Raise` and `\Lower` differ from TEX's `\raise` and `\lower` primitives not only in regard to the ⟨spec$_{RL}$⟩ option, but also in two other important respects. First, `\Raise` and `\Lower` typeset ⟨entry⟩ in the format for its column.[23] For example, '`\Lower{\alpha:\beta}`' will lower '$\alpha : \beta$' by a half-line in a math-mode column. Second, `\Raise` and `\Lower` lie to TEX by telling it that the typeset entry has height and depth zero, regardless of what the true height and depth are. Usually this has the effect one wants; for example,

```
\BeginTable
  \BeginFormat
  |    c       |           c                  | .     \_
  | A two line | \Lower{A one line heading} | \\+20
  | heading    |                              | \\+02 \_
\EndTable
```

produces the GOOD HEADINGS table

A two line heading	A one line heading

rather than

A two line heading	A one line heading

[23] `\Raise` and `\Lower` don't work with '`n`', '`N`', and '`p`' formats.

as it would if TABLE didn't lie to TEX about the depth of the lowered entry. You can, however, run into trouble raising or lowering an entry that is unusually high or deep; for example,

```
\BeginTable
\BeginFormat
  | c        | cM                | .  \_
  | A big    | \Lower{\int_0^1} | \\
  | integral |                   | \\ \_
\EndTable
```

produces the BIG INTEGRAL table

A big integral	$\int_0^1$

— there isn't enough space above and below the integral sign.

Exercise 88. Suggest a way to fix the spacing problem in the BIG INTEGRAL table above.

TABLE[24] has a `\Smash` command that works just like TEX's `\smash`, except that `\Smash` centers its argument vertically before setting its height and depth to zero. `\Smash` can be used together with `\\0` as an alternative to `\Raise` and `\Lower`. For example,

```
\BeginTable
  \BeginFormat
  |    c       |           c                          | . \_
  | A two line |                                      | \\+20
  |            | \Smash{\strut A one line heading} | \\0
  | heading    |                                      | \\+02 \_
\EndTable
```

reproduces the GOOD HEADINGS table:

A two line heading	A one line heading

Exercise 89. Solve Exercise 88 using `\Smash`.

4.11. SETTING THE DEFAULTS FOR TABLE'S UNIT PARAMETERS

The command `\NormalTableUnits` re-establishes the defaults for the unit parameters in TABLE's quantum systems for line thickness, inter-column space, column width, kern size, and strut size. This section explains how you can change those defaults.

To begin with, you need to know how `\NormalTableUnits` works. The macros say

```
\def\NormalTableUnits{%
  \LineThicknessUnit    = \the\NormalLTU
```

[24] This material can be skipped on first reading.

```
\InterColumnSpaceUnit = \the\NormalICSU
\ColumnWidthUnit      = \the\NormalCWU
\KernUnit             = \the\NormalKU
\StrutUnit            = \the\NormalSU},
```

with the token-list registers `\NormalLTU`, `\NormalICSU`, etc., being defined by

```
\NormalLTU  = {1in \divide \LineThicknessUnit by 300 }
\NormalICSU = {.5em plus 1fil minus .25em}
\NormalCWU  = {.5em}
\NormalKU   = {.5em}
\NormalSU   = {\normalbaselineskip \divide \StrutUnit by 11 }
```

Remember that the construction '`\the`⟨token-list register⟩' extracts the contents of ⟨token-list register⟩. Thus, for example, TEX reads

'`\LineThicknessUnit = \the\NormalLTU`'

as

'`\LineThicknessUnit = 1in \divide \LineThicknessUnit by 300 `';

these instructions set TABLE's line-thickness unit to $1/300$ inch — the width of one pixel on a 300 dot per inch laser printer. In effect, `\NormalLTU` contains a recipe for the line-thickness parameter. To change the default for that parameter, you just change the recipe. For example, to set the default to $1/240''$, enter

'`\NormalLTU = {1in \divide \LineThicknessUnit by 240 }`'.

The same idea applies to the other four parameters. For example, to set the default for TABLE's strut unit to one-twelfth of the normal distance between the baselines of consecutive lines of type, enter

'`\NormalSU ={\normalbaselineskip \divide \StrutUnit by 12 }`'.

TABLE's default for the line-thickness unit is set for a 300 dot per inch printer. If your printer has a different resolution, you should change `\NormalLTU` accordingly. The defaults for the other units are purposely coded in terms of quantities that depend on the size of the type in use when the `\NormalTableUnits` command is given.

Exercise 90. Show how to change the default for: (1) TABLE's line-thickness unit to $1/600''$; (2) the inter-column space unit to twice the width of a digit; (3) the column width unit to one centimeter; (4) the kern unit to a tenth of an inch; and (5) the strut unit to one twenty-fourth of the normal distance between baselines.

Exercise 91. Guess why TABLE doesn't have a `\NormalTableFactors` command akin to `\NormalTableUnits`.

4.12. STRETCHING OR COMPRESSING A TABLE HORIZONTALLY

The command

`\SetTableToWidth{`⟨dimen⟩`}`

will stretch or compress a table horizontally so as to be exactly ⟨dimen⟩ wide. For example, '`\SetTableToWidth{\hsize}`' makes a table exactly as wide as the current line length. This particular form of the command is so common that TABLE provides the abbreviation

`\Expand`

for it. The related command

`\WidenTableBy{`⟨dimen⟩`}`

will widen a table by ⟨dimen⟩. (⟨dimen⟩ may be negative, in which case the table is narrowed.) For example, '`\WidenTableBy{1in}`' will stretch a table by one inch. If you specify `\SetTableToWidth` or `\WidenTableBy` in the prologue to a table, the command will apply just to that table. If, however, you place either of these commands outside a table, it will apply to all subsequent tables until overriden. To cancel the effect of such a command, enter

'`\WidenTableBy{0pt}`'.[25]

When a table is stretched or compressed, it is the white space between columns that stretches or compresses, not the columns themselves. TEX will complain if you try to push the white space beyond the limits of its elasticity.[26] Lines drawn by `\_` and `\=`, but not by `\-`, lengthen or shorten to match changes in the white space. For example, here is the open form of the AT&T COMMON STOCK TABLE widened by half an inch:

AT&T Common Stock

Year	*Price*	*Dividend*
1971	41–54	$2.60
2	41–54	2.70
3	46–55	2.87
4	40–53	3.24
5	45–52	3.40
6	51–59	.95*

* (first quarter only)

and here expanded to the full width of the page:

[25] Or use TEX's grouping mechanism; see Chapter 5 of *The TEXbook*.

[26] With TABLE's default inter-column space unit in effect, white space can stretch indefinitely, but can compress to only half its natural width.

AT&T Common Stock

Year	*Price*	*Dividend*
1971	41–54	$2.60
2	41–54	2.70
3	46–55	2.87
4	40–53	3.24
5	45–52	3.40
6	51–59	.95*

* (first quarter only)

Exercise 92. State the commands that were used to widen/expand the two preceding tables.

The command

```
\LongLines
```

makes the lines drawn by `\_` extend all the way across the page, without changing the usual inter-column white space. For example, here is the AT&T COMMON STOCK table once again, this time with `\LongLines` specified in the prologue:

AT&T Common Stock

Year	*Price*	*Dividend*
1971	41–54	$2.60
2	41–54	2.70
3	46–55	2.87
4	40–53	3.24
5	45–52	3.40
6	51–59	.95*

* (first quarter only)

`\LongLines` shouldn't be used with tables having vertical lines at the edges because `\_` lines extend beyond the edge lines, making such tables look funny.

Exercise 93. Set the STATISTICS table using `\Expand` instead of `\LongLines`. How does the result compare to the original?

Exercise 94. Predict what B. L. User got from this code:

```
\BeginTable
  \LongLines
  \BeginFormat
  | w | w | .  \_
  | 1 | 2 | \\ \_
  | 3 | 4 | \\ \_
  | 5 | 6 | \\ \-
\EndTable
```

4.13. USING PARAGRAPHS AS ENTRIES

Recall that entries in a '`p`' column are set in the "paragraph mode" described in Section 3.1.8. You can set an entry in a non-'`p`' column in this same mode by beginning the entry with the command

`\BeginTableParBox{`⟨line length⟩`}`

and ending it with the command

`\EndTableParBox`

Here ⟨line length⟩ must be specified as a dimension, e.g. '`3in`'. The style commands specified by `\EveryTableParBox` will apply, just as they do to the entries in a '`p`' column.

Exercise 95. Code the following SINGLE-CELLED table. (Enter '`\BeginTableParBox`' as '`{\tt \string \BeginTableParBox}`'.)

This entry was set using `\BeginTableParBox` and `\EndTableParBox`. The line length is 2.8 inches. Lines are set ragged-right because that's the default style `\EveryTableParBox` specifies.

Exercise 96. Explain how to right-justify the lines in the table in the preceding exercise.

4.14. INSERTING VERTICAL SPACE IN A TABLE

The command

`\Vspace`⟨spec$_{VS}$⟩

inserts a gap of ⟨value$_{VS}$⟩ between the rows of a table. Here the subscript "*VS*" is a mnemonic for "Vertical Space", and the various options for ⟨spec$_{VS}$⟩ and ⟨value$_{VS}$⟩ are as follows:

⟨spec$_{VS}$⟩	⟨value$_{VS}$⟩
⟨null⟩	`\VspaceFactor` multiples of `\StrutUnit`
an unsigned integer i	i multiples of `\StrutUnit`
`(`⟨glue⟩`)`	⟨glue⟩

`\VspaceFactor` defaults to 2. For example, '`\Vspace`' inserts a gap of 2 strut units, '`\Vspace3`' inserts a gap of 3 strut units, and '`\Vspace(3pt plus 2pt minus 1pt)`' inserts a gap of 3 points that can stretch and shrink a bit. Vertical lines are *not* drawn across the gaps inserted by `\Vspace`. This point is illustrated in the table above, where the code '`\_ \Vspace \_`' was used to

create the double horizontal line. `\Vspace` can not be used as a table entry; it must immediately follow `\EndFormat`, a `\\` command, a `\_` command, or another `\Vspace` command.[27]

Exercise 97. Suppose TABLE's strut unit is 1 point. How broad are the gaps inserted by: (1) '`\Vspace1`'; (2) '`\Vspace`'; (3) '`\Vspace12`'; (4) '`\Vspace (.25in)`'?

Exercise 98. Reset the open form of the AT&T COMMON STOCK table using `\Vspace` instead of `\\+`.

Exercise 99. Solve Exercise 79 using `\Vspace`.

4.15. SPACING FORWARD AND BACKWARD

TABLE defines '`~`' to be a non-printing character having the width of a digit (i.e., half an em). As illustrated by the AT&T COMMON STOCK table, '`~`' can be used to align numeric entries involving varying numbers of digits.

It[28] is worth taking a look at how TABLE implements '`~`'. The macros say

```
\catcode`\~=\active
\def~{\kern.5em}
```

These instructions make '`~`' an active character whose definition[29] is, in lay terms, "insert a blank space having the width of a digit". Whenever TEX encounters an active character, it replaces the character by its current definition. Thus when TEX comes across a '`~`' in a table, it effectively treats that '`~`' as a non-printing digit.

Suppose now that you wanted to make '`;`' behave in a table as a non-printing character having the width of a comma. You could do that by placing the instructions

```
\newdimen\CommaWidth
{\catcode`\;=\active
\gdef\MakeNonprintingComma{%
  \setbox0 =\hbox{,}
  \CommaWidth=\wd0
  \catcode`\;=\active
  \def;{\kern\CommaWidth}}}
```

near the start of your input file and by specifying '`\MakeNonprintingComma`' in the prologue of each table[30] where you want '`;`' to have its special meaning. For

---

[27] Or TEX's primitive `\noalign{...}` command.

[28] This material can be skipped on first reading.

[29] TABLE's catcoding and definition of '`~`' apply only within the table environment. Plain TEX makes '`~`' active too, but gives it a somewhat different definition.

[30] Don't specify `\MakeNonprintingComma` outside a table, unless you want TEX to treat all subsequent semi-colons in this special way.

example, the first column of the WORD USAGE table could be coded as follows:

```
\BeginTable
  \MakeNonprintingComma
  \BeginFormat
    |               c             | .
    " \it Number of words         " \\
    " \it used exactly $n$ times  " \\
    "             14,376          " \\+30
    "             ~4,343          " \\
    "             ~2,292          " \\
    "             ~~;~~7          " \\
    "             ~~;~~5          " \\
\EndTable
```

The old way of entering this data was simpler. Nonetheless, the new technique can be put to good use in situations where TABLE's '`n`' format is inconvenient or inapplicable (for instance, when entries need to be aligned on a symbol other than a decimal point).

Exercise 100. Show how to make '`:`' into a non-printing character having the width of a decimal point. Use this special '`:`' to set the STACK of numbers in Exercise 8.

The command

`\BackSpace`⟨spec$_K$⟩

inserts a kern of width $-$⟨value$_K$⟩, thereby effectively backspacing by ⟨value$_K$⟩. Because TEX ignores blanks after a control sequence, one or more blanks may intervene between '`\BackSpace`' and '⟨spec$_K$⟩'. For example: (1) '`\BackSpace (.25in)`' backspaces by a quarter inch; (2) '`\BackSpace 12`' backspaces by 12 kern units; (3) '`\BackSpace 1 2`' backspaces by 1 kern unit and prints a '`2`'; and (4) '`\BackSpace A`' backspaces by `\KernFactor` kern units and prints an '`A`'. Section 3.1.6 illustrates the use of `\BackSpace` in conjunction with the kern keys '`i`' and '`j`'.

Exercise 101. Suppose TABLE's kern unit and factor have the default values specified in the QUANTUM SYSTEMS table in Section 2. What size kerns are produced by the four `\BackSpace` commands in the preceding paragraph?

4.16. ACTIVATING '|' AND '"'

TABLE activates the characters '`|`' and '`"`' in the table environment and defines the activated characters so as to serve as column separators. Because of TEX's rules for tokenization,[31] only new '`|`' s and '`"`'s are made active; '`|`'s and '`"`'s that TEX has already seen — for example, as the argument to a macro, such

[31] See the top of page 39 in *The TEXbook*.

as Plain TEX's `\centerline` — retain the category code they received when TEX first saw them. Inactive '`|`'s and '`"`'s won't separate entries.

The main point to remember is that the rows of a table shouldn't be included in the argument to a macro. This is an easy rule to abide by if you adopt the habit of using '`\Begin` ... `\End`' structures.

You could, of course, rule out the possibility of an improper tokenization of '`|`' and '`"`' in tables by making these characters active throughout your document. TABLE's command

```
\ActivateBarAndQuote
```

activates '`|`' and '`"`' and defines[32] the activated characters so that outside of tables they behave like the corresponding inactive characters. You should, however, use `\ActivateBarAndQuote` with discretion, since it introduces some other, admittedly minor, problems. For example, TEX won't interpret hex numbers properly because the '`"`' in TEX's hexadecimal notation must have category code 12, whereas active characters have category code 13. The author's advice is to activate '`|`' and '`"`' only if you have to.[33]

4.17. USING THE TABLE LOGO

Just as `\TeX` and `\LaTeX` produce the TEX and LATEX logos, so the command

```
\TaBlE
```

produces the TABLE logo.

[32] If one of these characters is already active, its definition isn't changed.

[33] This manual was written without ever invoking `\ActivateBarAndQuote`.

5. MISCELLANEOUS TOPICS

This section deals with several unrelated topics. The most important of these is the directions given in Section 5.1 on how to position tables in the page layout. Section 5.2 discusses TABLE's `\EveryTable` parameter, by means of which you can specify commands that will affect every subsequent table. Section 5.3 discusses TABLE's error messages concerning the improper use of format keys. Finally, Section 5.4 discusses those commands that don't have the same meaning in the table environment as they do in Plain TEX.

5.1. POSITIONING TABLES IN THE PAGE LAYOUT

The simplest way to position a table in the page layout is to use the construction

```
$$\BeginTable
  ⋮
  \EndTable$$
```

This centers the table horizontally and inserts some space above and below it. Most of the tables in this manual up to now have been displayed in this manner.

The '`$$`...`$$`' construction positions a table at the point where it is specified. Alternatively, you can allow a table to float to where there's room for it by making use of Plain TEX's `\midinsert`, `\topinsert`, and/or `\pageinsert` commands. Here's the sample code for a `\midinsert`:

```
\midinsert
  \hbox to \hsize \bgroup
    \hss
    \BeginTable
    ⋮
    \EndTable
    \hss
  \egroup
\endinsert
```

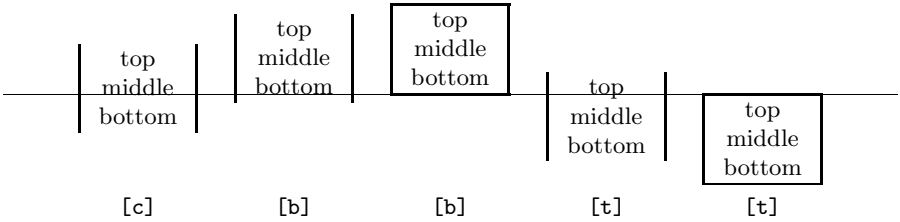
This centers the table horizontally. To position the table flush left, omit the first `\hss`. To position the table flush right, omit the second `\hss`. For reasons explained in Section 4.16, you shouldn't try to use Plain TEX's `\centerline`, `\leftline`, or `\rightline` macros to effect such positioning.

Exercise 102. Define macros `\BeginInsertedTable` and `\EndInsertedTable` by means of which the above construction could be simplified to

```
\BeginInsertedTable
⋮
\EndInsertedTable
```

Exercise 103. Define macros `\BeginIndentedTable` and `\EndIndentedTable` which will indent a table by the current paragraph indentation and insert some space above and below the table.

You can place a table in line with other text, just as though the table were a big character. To do so, though, you need to know what the baseline of a table is. TABLE ordinarily places the baseline of a table at the vertical midpoint of the table. The suffix '`[b]`' to `\BeginTable` lowers the baseline of the table to the baseline of the bottom row. If, however, a horizontal line is drawn across the bottom of the table, `\BeginTable[b]` will set the baseline of the table at the very bottom of the table. Similarly, the suffix '`[t]`' to `\BeginTable` raises the baseline of the table to the baseline of the top row. If, however, a horizontal line is drawn across the top of the table, `\BeginTable[t]` will set the baseline of the table at the very top of the table. `\BeginTable[c]` is equivalent to `\BeginTable`. The various possibilities are illustrated in the COMMON BASELINES table below:



The line extending across the page is the common baseline of the five tables, which were set with the indicated suffixes.

Exercise 104. Code the following tables, paying attention to the level of their baselines:

(1) $\boxed{\begin{array}{c}\text{top}\\ \text{middle}\\ \text{bottom}\end{array}}$; (2) $\begin{array}{c}\alpha+\beta\\ \gamma+\delta+\epsilon\end{array}$; (3) $\begin{array}{c}\text{top}\\ \text{bottom}\end{array}$; (4) $\boxed{\begin{array}{c}\text{top}\\ \text{bottom}\end{array}}$.

TABLE is recursive, so that tables can be placed inside tables inside tables inside tables ... . For example, the COMMON BASELINES table above was produced by

```
$$\hbox to \hsize{\ninepoint \NormalTableUnits
    % Draw common baseline for the tables
    \rlap{\vrule height .5\LineThicknessUnit
                depth  .5\LineThicknessUnit width \hsize}%
```

```
% Overlay the tables
\hss
\BeginTable[t]
  \def\rows{%
    | top    | \\
    | middle | \\
    | bottom | \\ }
  \def\A#1{%
    \BeginTable[#1] \BeginFormat |c|. \rows \EndTable}
  \def\B#1{%
    \BeginTable[#1] \BeginFormat |c|. \_ \rows \_ \EndTable}
  \def\P#1{\tt [#1]}
  \LineThicknessFactor=3
  \BeginFormat | c | c | c | c | c |.
  " \A c " \A b " \B b " \A t " \B t " \\
  " \P c " \P b " \P b " \P t " \P t " \\+30
\EndTable
\hss}$$
```

TABLE ordinarily places a table in an invisible box[34] that can't be split across pages. The suffix `[u]` to `\BeginTable` instructs TABLE to create an "unboxed" table that can extend over more than one page. Don't try to use the `[u]` option to create a table that would fill several pages of a book; such a huge table would exceed TEX's memory.

By default, a `[u]` table is positioned flush left. To center such a table, specify

```
\LeftTabskip=0pt plus 1fill
\RightTabskip=0pt plus 1fill
\Expand
```

in the prologue to the table. (Those three lines of code are exactly what `\LongLines` stands for, so you can enter just `\LongLines` instead.) To position a table flush right, omit the line '`\RightTabskip=0pt plus 1fill`'.

`\LeftTabskip` is a glue TABLE places to the extreme left of each table; `\RightTabskip` is similar, but is placed at the extreme right of each table. These glues behave like the glues that make up the inter-column white space inserted by the keys '`s`' and '`o`'; in particular, they are stretched and/or compressed by `\SetTableToWidth` and `\WidenTableBy`. Both glues default to zero and so normally have no effect. Non-zero values of `\LeftTabskip` and `\RightTabskip` present a small complication, in that `\_` lines span these glues and so extend beyond the "natural" edges of a table;[35] to draw horizontal lines that stop at the edges of the table, it is necessary to use `\use` and `\=` instead of `\_`; see Exercise 73.

---

[34] TEXnically speaking, an `\hbox`.

[35] See Exercise 94 for an example.

Exercise 105. Code the open form of the AT&T STOCK TABLE so that the table can split across pages.

5.2. SPECIFYING COMMANDS THAT AFFECT EVERY TABLE

TABLE places the ⟨commands⟩ specified by

`\EveryTable={`⟨commands⟩`}`

at the start of the prologue of every table following the `\EveryTable` specification. For example, if you have a `\ixpt` macro that sets up nine point type, you could specify that every subsequent table be set in that size type by entering '`\EveryTable={\ixpt \NormalTableUnits}`'. To expand every subsequent table to the full width of the page, enter '`\EveryTable={\Expand}`'. To cancel the effect of a prior `\EveryTable` specification, enter

'`\EveryTable={}`'.[36]

Exercise 106. Explain how to make the command `\C` an abbreviation for `\JustCenter` within every table, without precluding the use of `\C` for other purposes outside of tables.

5.3. ERROR MESSAGES CONCERNING FORMAT KEYS

TABLE will complain if you: (1) try to use a format key that hasn't been defined; (2) try to use `\NewFormatKey` to define a format key that's already in use; or (3) try to use an inadmissible format key in conjunction with the `\ReFormat` command. TABLE's complaint takes the form of an error message that is written to your terminal and log file. The message begins '`TABLE error`' and contains a brief description of the error. If you type '`H` ⟨carriage return⟩' in response to an error message, TABLE will give you a longer description of the error and suggest what to do next. In some cases, TABLE will prompt you to enter a replacement for the offending key. TABLE will signal that it's ready to receive the replacement key by displaying the line

`\!tkReplacement=`

on your terminal; respond with

'⟨replacement key⟩ ⟨carriage return⟩'.

Don't try to enter a replacement key under any other circumstances; you will only make matters worse.

Other mistakes that you make may elicit an error message from TEX. For example, if you forget to end an input row with `\\`, TEX will come to a halt, displaying the error message '`Extra alignment tab has been changed to`

[36] Or make use of TEX's grouping mechanism; see Chapter 5 of *The TEXbook*.

`\cr.`' You can recover gracefully from this error by responding '`I&`'. Unfortunately, some of TEX's error messages may not make sense to you because they refer to commands in TABLE's innards with which you are not familiar. By looking at the bottom line of TEX's error message you can at least see at what point in your input file the error occurred; you can then check the syntax of the faulty input line(s) to try to spot what caused the problem.

5.4. POTENTIAL CONFLICTS WITH PLAIN TEX

You should be aware that the characters '`|`', '`"`', and '`~`', and the control sequences `\|`, `\-`, `\=`, and `\_` have different meanings in the table environment than they do in Plain TEX. As mentioned in Section 4.5, you can specify the characters '`|`' and '`"`' in a table by `\Vbar` and `\DQuote`, respectively. TEX interprets '`~`' as a "tie" and `\-` as a discretionary hyphen. These two constructs are needed only for vertical mode material, where they influence TEX's choice of line breaks; TABLE automatically restores TEX's meanings of '`~`' and `\-` for every table entry set in the "paragraph mode" stipulated by the format key '`p`' and/or the command `\BeginTableParBox`. In math mode, TEX interprets '`|`' and `\|` as delimiters; you can recover TEX's meanings by using `\vert` in place of '`|`' and `\Vert` in place of `\|`. Finally, TEX interprets `\=` as a macron accent and `\_` as an underscore character. These two constructs are so seldom used in tables that TABLE makes no provision for restoring TEX's meanings within a table. You can do that yourself, however, by placing the commands

```
\let\MacronAccent = \=
\let\UnderScore = \_
```

outside a table; in subsequent tables you could then use `\MacronAccent` in place of `\=` and `\UnderScore` in place of `\_`.

Exercise 107. What is produced by the following code?

```
\let\MA = \=
\let\US = \_
\BeginTable
  \BeginFormat
  |  cT      | p(.5in)      |  cm           |  c          | .  \_
  |  \VBar   | A~half\-inch | \vert x \vert | \MA o       | \\
  "          |    \-        |               & \=          & \\0
  |  \DQuote |              | \Vert x \Vert | net\US gain | \\ \_
\EndTable
```

5.5. FINAL EXERCISES

Exercise 108. Code the BUDGET TRANSFERS table in the preface.

Exercise 109. Code the SUMMARY OF PARAMETERS table in Appendix D, up to the `\ColumnWidthUnit` entry.

Exercise 110. Code the FONT LAYOUT table on the cover. (That table is adapted from the table on page 427 of *The TEXbook*.) The octal and hexadecimal numbers in the margins of the table are conveniently specified by the macros `\O` and `\X` defined by

```
\def\O#1{\hbox{\rm\'{}\kern-.2em\it#1\/\kern.05em}}
\def\X#1{\hbox{\rm\H{}\tt#1}}
```

For example, '`\O{05x}`' produces '*´05x*', and '`\X E`' produces '˝E'. Enter the characters in the body of the table simply as '`\:`', the macro `\:` being defined by

```
\def\:{\enlarge11{\char\count255}%
  \global\advance\count255 by 1 }
```

Place the definitions of `\O`, `\X`, and `\:` in the prologue to the table, along with the initialization statement '`\global \count255=0 `'.

APPENDICES

A. ANSWERS TO ALL THE EXERCISES

1. Input:

```
\BeginTable
  \def\AD{\sevenrm\ A.D.}
  \def\BC{\sevenrm\ B.C.}
  \def\C{\JustCenter}
  \BeginFormat
  |     r     |          r          |
  \EndFormat
  \_
  |  \C Year  |  World Population   |  \\
  \_
  |  8000\BC  |        5,000,000    |  \\
  |    50\AD  |      200,000,000    |  \\
  |  1650\AD  |      500,000,000    |  \\
  |  1850\AD  |    1,000,000,000    |  \\
  |  1945\AD  |    2,300,000,000    |  \\
  |  1980\AD  |    4,400,000,000    |  \\
  \_
\EndTable
```

Output:

Year	World Population
8000 B.C.	5,000,000
50 A.D.	200,000,000
1650 A.D.	500,000,000
1850 A.D.	1,000,000,000
1945 A.D.	2,300,000,000
1980 A.D.	4,400,000,000

The spacing about the horizontal lines is tighter, and less aesthetically pleasing. (Overall, the table above is smaller than the one in the introduction because this appendix is set in nine point type.)

2. Input:

```
\BeginTable
  \def\AD{\sevenrm\ A.D.}
  \def\BC{\sevenrm\ B.C.}
  \BeginFormat
  |     c     |
  \EndFormat
  \_
  |   Year    |  \\+22
  \_
  |  8000\BC  |  \\+20
  |  ~~50\AD  |  \\
  |  1650\AD  |  \\
  |  1850\AD  |  \\
  |  1945\AD  |  \\
  |  1980\AD  |  \\+02
  \_
\EndTable
```

Output:

Year
8000 B.C.
50 A.D.
1650 A.D.
1850 A.D.
1945 A.D.
1980 A.D.

3. Input:

```
\BeginTable
  \def\AD{\sevenrm\ A.D.}
  \def\BC{\sevenrm\ B.C.}
  \BeginFormat
  |     ck    |          rk           |
  \EndFormat
  \_
  " \sl Year  " \sl World Population  "  \\+22
  "  \-       " \-                    "  \\0
  "  8000\BC  "        5,000,000      "  \\+20
  "  ~~50\AD  "      200,000,000      "  \\
  "  1650\AD  "      500,000,000      "  \\
  "  1850\AD  "    1,000,000,000      "  \\
  "  1945\AD  "    2,300,000,000      "  \\
  "  1980\AD  "    4,400,000,000      "  \\+02
  \_
\EndTable
```

Output:

Year	*World Population*
8000 B.C.	5,000,000
50 A.D.	200,000,000
1650 A.D.	500,000,000
1850 A.D.	1,000,000,000
1945 A.D.	2,300,000,000
1980 A.D.	4,400,000,000

4. This:

AT&T Common Stock

Year	*Price*	*Dividend*
1971	41–54	$2.60
2	41–54	2.70
3	46–55	2.87
4	40–53	3.24
5	45–52	3.40
6	51–59	.95*

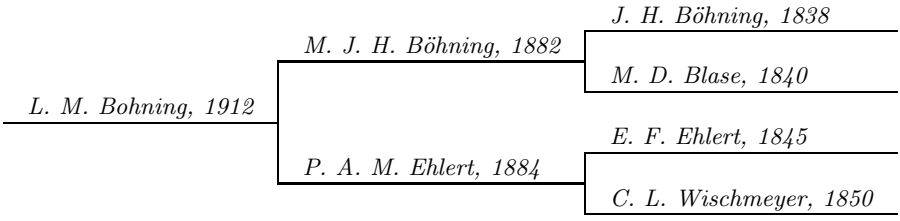
* (first quarter only)

On rows that contain only horizontal line(s), you will usually want to "zero out" the strut which `\\` automatically inserts by suffixing '`\\`' by `0`.

5. The author set the MORTALITY table with

```
\BeginTable
  \BeginFormat
  |     r      |   c   |   c   |  r  |
  \EndFormat
  "            " \use2 control "     " \\
  "            " alive " dead  "     " \\+02
  "            & \use2 \=      &     " \\0
  "   low risk |   5   |  ~3   |  8  " \\
  "            & \use3 \=            & \\0
  "  high risk |   4   |  26   | 30  " \\
  "            & \use3 \=            & \\0
  "            "   9   |  29   | 39  " \\
\EndTable
```

6. With the format changed from '`| l | l | l |`' to '`| lI | lI | lI |`', the FAMILY TREE table is somewhat more regal looking:



7. Input:

```
\BeginTable
  \def\C{\JustCenter}
  \BeginFormat
  | cn[0,000,000,000]   |
  \EndFormat
  \_
  | \C World Population |  \\+22
  \_
  |        5,000,000    |  \\+20
  |      200,000,000    |  \\
  |      500,000,000    |  \\
  |    1,000,000,000    |  \\
  |    2,300,000,000    |  \\
  |    4,400,000,000    |  \\+02
  \_
\EndTable
```

Output:

World Population
5,000,000
200,000,000
500,000,000
1,000,000,000
2,300,000,000
4,400,000,000

8. The author set the STACK of numbers with

```
\BeginTable
  \BeginFormat
  | cn[00.0] |
  \EndFormat
  \_
  |    46    | \\+20
  |    23    | \\
  |    15    | \\
  |     6.5  | \\
  |     2.1  | \\+02
  \_
\EndTable
```

9. With '`+00.00`' replaced by '`0.00`' in the format, the SOPORIFICS table in abbreviated form looks like this:

Patient	*A*	*B*	*Difference*
1	+0.7	+1.9	+1.2
2	−1.6	+0.8	+2.4
3	−0.2	+1.1	+1.3
4	−1.2	+0.1	+1.3
5	−0.1	−0.1	0
10	+2.0	+3.4	+1.4
Mean	+0.75	+2.33	+1.58

The '`N`' entries don't appear to be centered in their columns because only the last row uses 3 digits.

10. The author set the 2-BY-2 table with

```
\BeginTable
  \def\y#1#2{y_{#1#2}}
  \BeginFormat
  |   cm    |   cm    |   cm    |
  \EndFormat
  "  \y11   "   \y12  |   n_1 " \\
  "  \y21   "   \y22  |   n_2 " \\+03
  \_
  "   m     "   n-m   |   n   " \\+10
\EndTable
```

11. The author set the PAIRED EQUATIONS table with

```
\BeginTable
  \OpenUp11
  \BeginFormat
s4| rmo0 | \m           | rmo0 | \m           | rmo0 | \m      |
  \EndFormat
  " V_i  " =v_i-q_iv_j, " X_i  " =x_i-q_ix_j, " U_i  " =u_i,
                                    \qquad\hbox{for $i\ne j$}; " \\
  " V_j  " =v_j,        " X_j  " =x_j,        " U_j  " =u_j+
                                          \sum_{i\ne j}q_iu_i. " \\
\EndTable
```

12. The author set the ANSWER SHEET with

```
\BeginTable
  \BeginFormat
  | cmI          | cIw(1.25in) | cIw(2in)       | .
  " $Quantity$  " Estimate    " Standard Error " \\+02 \_
  | \mu_U       |             |                | \\+77 \_
  | \mu_G       |             |                | \\+77 \_
  | \mu_U-\mu_G |             |                | \\+77 \_
\EndTable
```

The coding of the first column deserves discussion because it shows how to make convenient use of a little-known TEXnicality. Most of the entries for the first column are in math mode, so it's desirable to specify '`m`' format. One has, however, to get out of math mode to set the word 'Quantity'. Because '`m`' places each entry between '`$`'s, and because TEX treats '`$$`' as an empty math formula when it's working in internal horizontal mode (as it is when it's setting table entries), just putting '`$`'s around '`Quantity`' in the input file does the trick simply and neatly.

13. The entries for the first two rows start with '`\C`', which was defined in the prologue of the table to be an abbreviation for '`\JustCenter`'. '`\JustCenter`' discards the stipulated format for the column in question and simply centers the rest of the entry. The first two rows of the table are thus set in the ten point type which was in effect when the table began.

14. The author set the RECURRENCE CRITERIA table with

```
\BeginTable
  \OpenUp22
  \ninepoint
  \BeginFormat
  | lp(70pt) | lp(170pt) | lp(40pt) |
  \EndFormat
  \_
  | \it Classification | \it Criterion | \it Reference | \\
  \_3
  | $X$ is positive recurrent if
    | $I$ is finite.
      | Theorem 3.15 | \\
  \_
  | $X$ is positive recurrent if and only if
    | the equations
      $$u_k = \sum_{j\in I} u_jp_{jk},\quad k\in I,$$
      have a non-trivial nonnegative summable solution.
      | Theorem 4.2 | \\
  \_
  | $X$ is recurrent (positive or null) if and only if
    | for some arbitrary $i_0\in I$, the equations
      $$x_i = \sum_{j\not=i_0} p_{ij}x_j,\quad i\not=i_0,$$
      have no bounded non-trivial solution.
      | Theorem 5.14 | \\
  \_
\EndTable
```

Note that it is not necessary to line up the column separators '`|`'.

15. '`\NewFormatKey m{\ReadFormatKeys b{$} a{$}}`'.

16. This code will set the 1st and 3rd columns of the "closed" version of the AT&T COMMON STOCK table:

```
\BeginTable
  \def\C{\JustCenter}
  \BeginFormat
  | cn[0000] |   cn[\$0.00]  |
  \EndFormat
  \_
  | \C Year  | \C Dividend   |  \\  \_
  |   1971   |  \$2.60       |  \\  \_
  |      2   |    2.70       |  \\  \_
  |      3   |    2.87       |  \\  \_
  |      4   |    3.24       |  \\  \_
  |      5   |    3.40       |  \\  \_
  |      6   |     .95\rlap* |  \\  \_
\EndTable
```

17. The author set the COMPOSITION OF FOODS table with

```
\BeginTable
  \def\Pro{\Lower{Protein}}  \def\Fat{\Lower{Fat}}
  \def\Food{\JustCenter \Raise2{Food}}
  \BeginFormat
  | l          |   c   |   c   |     c    | .
  \_
  | \use4 \bf Composition of Foods        | \\+22
  \_
  |            | \use3 Percent by Weight  | \\+22
  |            & \use 3 \=                & \\0
  |   \Food    | \Pro  | \Fat  |  Carbo-  | \\+20
  |            |       |       |  hydrate | \\+02
  \_
  | Apples     | ~~.4  |  ~.5  |   13.0   | \\+20
  | Halibut    | 18.4  |  5.2  |          | \\
  | Lima beans | ~7.5  |  ~.8  |   22.0   | \\
  | Milk       | ~3.3  |  4.0  |   ~5.0   | \\
  | Mushrooms  | ~3.5  |  ~.4  |   ~6.0   | \\
  | Rye bread  | ~9.0  |  ~.6  |   52.7   | \\+02
  \_
\EndTable
```

18. The author set the LIZARDS table with

```
\BeginTable
\LongLines \NewFormatKey L{\ReadFormatKeys cn[00]} \def\U{\use3 \H}
\def\H{\JustCenter\it} \def\G{\H G} \def\O{\H O} \def\T{\H Total\/}
\BeginFormat
| lIo5 | cm   | cmo5 | L  | L |Lo5|  L | L |Lo5|  L | L | L | .
\_4
"      "      "      " \use9 \H Time of Day                " \\+22
"      "      "      " \use9 \-                            " \\0
"      "      "      " \U Early   " \U Mid-day " \U Late    " \\+22
"      "      "      " \use3 \-   " \use3 \-   " \use3 \-   " \\0
" \H S " \H D " \H H " \G "\O "\T " \G "\O "\T " \G "\O "\T " \\+22
\_
" Sunny" \le2 "  <5  " 20 " 2 "22 "  8 " 1 " 9 "  4 " 4 " 8 " \\+20
"      "      " \ge5 " 13 " 0 "13 "  8 " 0 " 8 " 12 " 0 "12 " \\
"      "  >2  "  <5  "  8 " 3 "11 "  4 " 1 " 5 "  5 " 3 " 8 " \\
"      "      " \ge5 "  6 " 0 " 6 "  0 " 0 " 0 "  1 " 1 " 2 " \\
" Shady" \le2 "  <5  " 34 "11 "45 " 69 "20 "89 " 18 "10 "28 " \\+20
"      "      " \ge5 " 31 " 5 "36 " 55 " 4 "59 " 13 " 3 "16 " \\
"      "  >2  "  <5  " 17 "15 "32 " 60 "32 "92 "  8 " 8 "16 " \\
"      "      " \ge5 " 12 " 1 "13 " 21 " 5 "26 "  4 " 4 " 8 " \\+02
\_4
" \use{12} \JustCenter {\it S}, sunny/shady;
   {\it D}, diameter (in); {\it H}, height (ft);
   {\it G, grahami\/}; {\it O, opalinus\/}.                " \\+20
\EndTable
```

(Do these lizards prefer: sunny or shady sites; high or low sites; small- or large-diameter sites? Does time-of-day or species make any difference?)

19. The author set the start of the FORMAT KEY USAGE table with

```
\BeginTable
  \Expand
  \def\m{\tt\string\m}
  \def\M{\tt\string\M}
  \def\CI{\JustCenter \it}
  \NewFormatKey X{\ReadFormatKeys cT|1 }
  \BeginFormat
s1| lb{\csc} |cn[00]|X X X X X X X X  X X  X X X X X X X X cT|2 .
  "          "  {}  & \use{19} \=                              &\\0
  "          "  {}  | \use{19} \it Format Key                  |\\+11
  \_
  | \CI Sample table
        |\CI Page\/ |b|B|c|I|k|l|m|\m|M|\M|n|N|o|p|r|s|T|w|.| \\+11
  \_
  | 2-by-2      | 8 | | |X| | | |X|  | |  | | | | | | | | |\\ \_1
\EndTable
```

20. The author set the start of the COMMAND SUMMARY table with

```
\BeginTable
  \ninepoint \Expand
  \EveryTableParBox={\noindent \hangafter=1 \hangindent=\parindent}
  \BeginFormat
  | l             | lp(3.5in)   | .
  " \it Command   " \JustCenter \it Usage " \\+02
  \_
  | \tt \VBar     | End entry, with trailing vertical line. | \\+20
  | \tt \DQuote   | End entry, without trailing vertical line. | \\
```

'|' and '"' couldn't be entered directly in the first column because TABLE interprets these characters as column separators. '`\VBar`' and '`\DQuote`' are macros which retain the original meanings of '|' and '"', respectively. The '`\EveryTableParBox`' specification set up the hanging indentation used in the second column; see Section 3.1.8.

21. Commands: '`\InterColumnSpaceUnit=2mm  \InterColumnSpaceFactor=5`'. Gaps: (1) 2 millimeters; (2) 1 centimeter; (3) 1.6 centimeters.

22. (1) $1/300''$ = .241 pt — the width of one pixel on a 300 dot per inch laser printer; (2) $2/300''$; (3) $4/300''$; (4) $20/300''$. A `\_20` line looks like this: ▬▬▬. Stay with the small multiples. (1′) .4 pt; (2′) .4 pt; (3′) 1.6 pt; (4′) 8 pt.

23. (1) the width of 1 digit; (2) the width of 10 digits; (3) the width of 30 digits. (1′) half an inch; (2′) one inch; (3′) 15 inches (way too big).

24. (1) the width of a digit; (2) same as (1); (3) the width of 4 digits. (1′) $1/10''$; (2′) $3/10''$; (3′) $4/10''$.

25. '`k-2`' is invalid because '`-2`' is not a nonnegative integer. '`k(-2\KernUnit)`' is the proper way to specify a negative kern two units wide.

26. (2) This is a borderline case. '`p`' alone is a valid specification; it instructs TABLE to use the default number of multiples of the column width unit. But after TABLE has read '`p`', it will go on to report '`4`' as an "invalid format key". (3) 4.5 is not a whole number. (7) TEX insists that you type '`in`' to specify inches. (8) Like (7). (9) TABLE has no default for '`()`'; you have to specify a value.

27. One more than the number of columns in the table.

28. **Boldface**. 7 point roman.

29. (1) $\frac{1}{2}$; (2) $\dfrac{1}{2}$.

30. This fragment

Name	*Definition*
Gamma	$\Gamma(z)=\int_0^\infty t^{z-1}e^{-t}\,dt$
Sine	$\sin(x)=\dfrac{1}{2i}(e^{ix}-e^{-ix})$

of the SPECIAL FUNCTIONS table shows the result of changing `\M` to '`lM`' in the format: there's not enough space in front of the equal signs.

31. The author set the TABLETTE with

```
\BeginTable
  \BeginFormat
  | n2.3  | .
  \_
  | 25    | \\+20
  | 6.25  | \\
  |  .125 | \\+02
  \_
\EndTable
```

32. (1) He should have specified '`n`' instead of '`N`', because of the comma in '1,234.2'. (2) He should have entered the sample entry as '`[0,000.000]`'; '`4.3`' doesn't leave room for a comma. (3) He didn't type '`\JustCenter`' in front of '`My Data`' in order to escape from the numeric format. (4) He didn't put a blank between '`23.815`' and the trailing column separator '`"`'. Of the four errors, the last is by far the most serious; it provokes an error message from TEX. (To make his table as small as the one in the text, he'd also have to specify '`\sevenrm \StrutUnit=.7\StrutUnit`' in the prologue; this, however, is a side issue.)

33. (1) After the '2' (implicit); (2) between the '8' and the '2' (explicit); (3) after the '2' (implicit); (4) after the 'C' (implicit); (5) between the 'B' and the 'C' (explicit). As the last two cases illustrate, '`n`' and '`N`' entries don't have to be numbers.

34. (1) .5 em plus 1 fil minus .25 em; (2) 1.5 em plus 3 fil minus .75 em; (3) 2 em plus 4 fil minus 1 em; (4) 1/4 inch. The last specification is rigid; it can't stretch or shrink.

35. This fragment

AT&T Common Stock		
Year	Price	Dividend
1971	41–54	$2.60
2	41–54	2.70

of the table shows the result of changing the format from '`| c | c | c |`' to '`s6 | cs3 | c | cs6 |`'. The table looks funny because the two inner vertical lines bisect the inner inter-column white spaces, while the two outer vertical lines are drawn along the edges of the outer white spaces. Note, however, that if the inner vertical lines are removed, the table looks O.K. in the new format:

AT&T Common Stock		
Year	Price	Dividend
1971	41–54	$2.60
2	41–54	2.70

but funny in the original format:

AT&T Common Stock		
Year	Price	Dividend
1971	41–54	$2.60
2	41–54	2.70

36. (1) This format specifies three centered columns. The default spacing of 3/8″ applies between the first and second column and between the second and third column. Half the default, or 3/16″, applies before the first column and after the third one. This situation may be illustrated schematically as

$$3/16'' \;\|_1\; 3/8'' \;\|_2\; 3/8'' \;\|_3\; 3/16'',$$

'$\|_i$' designating the i^{th}column. Using the same notation, the answers to the other parts are:

(2) $1/4'' \;\|_1\; 1/2'' \;\|_2\; 1/2'' \;\|_3\; 1/4''$;
(3) $0'' \;\|_1\; 3/8'' \;\|_2\; 3/8'' \;\|_3\; 3/16''$;
(4) $3/16'' \;\|_1\; 1/4'' \;\|_2\; 1/4'' \;\|_3\; 3/4'' \;\|_4\; 1/2'' \;\|_5\; 1/4'' \;\|_6\; 3/4'' \;\|_7\; 0''$.

37. (1) .5 em; (2) .5 em; (3) 1.5 em; (4) half a centimeter.

38. (1) An '`o`*u*' space is typically a rubber space (it can stretch or shrink), whereas a '`j`*u*' space is always rigid. (2) The units may be different — '`o`*u*' specifies *u* inter-column space units, whereas '`j`*u*' specifies *u* kern units. (3) A `\-` line will extend across a '`j`*u*' space, because that space is considered to be part of its column; `\-` will not, however, bridge an '`o`*u*' space. (4) A vertical line will bisect an '`o`*u*' space, but will be drawn along the right-hand edge of a '`j`*u*' space.

39. The author set the beginning of the SIDE EFFECTS table with

```
\BeginTable
  \def\BS{\BackSpace}
  \BeginFormat
s8| li2               | c            | c        | .
  " \use3 \ss Percentages of Patients
       Reporting Side Effects                    " \\+03
  \_3
  "                   " \csc Placebo " \csc Cop " \\+30
  " \BS2 \csc Symptom "   $(N=23)$   " $(N=25)$ " \\
  " \BS2 Local        "                           \\+30
```

the control sequences `\ss` and `\csc` being defined to select sans serif and CAPS AND SMALL CAPS type, respectively.

40. The author set the PECULIAR ALIGNMENT table with

```
\BeginTable
   \def\BS{\BackSpace}
   \BeginFormat
   | rj2                   | .
   \_
   | Here's a peculiar\BS2 | \\+20
   | alignment you         | \\
   | will probably         | \\
   | never use.            | \\+02
   \_
\EndTable
```

41. **Input:**

```
\BeginTable
  \def\C{\JustCenter}
  \BeginFormat
  |        rk2          | .      \_
  | \C World Population |  \\+22 \_
  |    2,300,000,000    |  \\+20
  |    4,400,000,000    |  \\+02 \_
\EndTable
```

Output:

World Population
2,300,000,000
4,400,000,000

42. (1) .5 em; (2) 5 em; (3) 2.5 em; (4) 2.5 cm; (5) the width of the expression '**width**'.

43. (1) .5 em — *impossibly* narrow; (2) 5 em (≐ 3/4″ for ten point type) — much too narrow; (3) 15 em (≐ 2″ for ten point type) — barely adequate; (4) three inches. If you plan to make heavy use of '`p`', you'd be best off resetting TABLE's column width unit to a more convenient value, say half an inch; then, for example, '`p5`' will specify a line length of 2.5″.

44. TABLE places the `\EveryTableParBox` commands immediately before each '`p`' entry. The result in this case was '... `\hbadness=100004 score and 7` ...', which instructed TEX to set `\hbadness` to 100004 and begin the paragraph with 'score'.

45. This style is suitable for a single right-justified paragraph where the top line starts flush left and subsequent lines are indented by the usual paragraph indent. (See page 102 of *The TEXbook*.)

46. The macros include the instruction

```
\EveryTableParBox={%
  \parindent=0pt
  \raggedright
  \rightskip=0pt plus 4em
  \relax}
```

The `\rightskip` specifies a stretchier glue than Plain TEX's `\raggedright` command calls for. The extra stretch leads to fewer hyphenated words and fewer "over-full box" messages from TEX. To see why the '`\relax`' is important, work Exercise 27.4 in *The TEXbook*.

47. Input:

```
\BeginTable
  \EveryTableParBox={\noindent \tolerance=10000 \hbadness=10000 }
  \def\C{\JustCenter}   \OpenUp11
  \BeginFormat
  | l           | lp(1.5in)       | lp(1.5in)      | .
  \_
  | \use3 \C \it New York Area Rocks                | \\ \_
  | \C Era      | \C Formation    | \C Age (years) | \\ \_
```

and so on: the rest of the code is identical to that used before.

Output (in ten point type):

New York Area Rocks		
Era	Formation	Age (years)
Precambrian	Reading Prong	>1 billion
Paleozoic	Manhattan Prong	400 million
Mesozoic	Newark Basin, including Stockton, Lockatong, and Brunswick formations; also Watchungs and Palisades.	200 million
Cenozoic	Coastal Plain	On Long Island 30,000 years; Cretaceous sediments redeposited by recent glaciation.

48. The author set the CLINICAL CRITERIA table with

```
\BeginTable
  \eightpoint % (From Appendix E of the The TeXbook)
  \NormalTableUnits
  \EveryTableParBox={%
    \raggedright \hangafter=1 \hangindent=1em \noindent}
  \OpenUp04 \LongLines
  \BeginFormat
  | lp(2.5in)                                  | c | c | .
  " \use3 \ninepoint\ss Clinical Criteria for Abnormal
      Organ Function in the 27 Patients                  " \\
  \_3
  " " \csc 14 Survivors " \csc 13 Who Died               " \\+40
  " {\bf Cardiovascular:} dopamine therapy for mean arterial
    pressure $>$45 mm Hg in the absence of hypovolemia
    (pulmonary artery wedge pressure $>$6 mm Hg) " 3 " 6 " \\
  " {\bf Renal:} serum creatinine $>$300\ $\mu$mol/liter
    in preceding 24 hr                           " 6 " 6 " \\
  " {\bf Gastrointestinal bleeding:} fresh blood from a
    nasogastric tube and/or melena or fresh blood from rectum,
    with a fall in hemoglobin of $>$2 g/dl       " 1 " 2 " \\
  \_
\EndTable
```

49.

Location of vertical line	*Thickness*
Before the 1st column	1 point
Between the 1st and 2nd column	$^{2}/_{300}''$
Between the 2nd and 3rd column	$^{0}/_{300}''$ (invisible)
Between the 3rd and 4th column	$^{1}/_{300}''$
Between the 4th and 5th column	$^{2}/_{300}''$
Between the 5th and 6th column	$^{3}/_{300}''$
After the 6th column	$^{4}/_{300}''$

50. The author set the PIN-STRIPED table with

```
\BeginTable
  \def\D{\vrule \hskip 2pt \vrule ##} % Double
  \def\T{\vrule \hskip 2pt \vrule \hskip 2pt \vrule ##} % Triple
  \BeginFormat
  \|{\span\D} cw5 \|{\span\T} cw7 \|{\span\T} cw5 \|{\span\D}.
  \_
  |                |               |               | \\
  \_
  |                |               |               | \\
  \_
\EndTable
```

51. (1) '`b{\bf}`'; (2) '`b{${}} a{$}`'; (3) '`\{ b{$\displaystyle{}} a{$}`'.

52. '`b{\BeginTableParbox{3in}} a{\EndTableParbox}`'. '`p(3in)`' does the same thing with less typing.

53. Here's an example: '`BI`' is equivalent to '`b{\bf}b{\it}`', which in turn is equivalent to '`b{\it \bf}`' because of the "from the inside out" rule. '`BI`' thus places '`\it \bf`' before each entry. The effect is the same as if only '`\bf`' had been specified.

54. '`\BeginFormat s4|*3{rmo0|\m|}.`'.

55. He forgot to enclose the repeat count in braces; he should have entered '`*{11}{c | }`'.

56. When TABLE reaches '`9`' during its scan of the format keys, it passes control to TEX to read the text '`\ReadFormatKeys b{\ninepoint}`'. `\ReadFormatKeys` passes control back to TABLE. TABLE reads the key '`b`', which sets things up so that TEX will place the command `\ninepoint` in front of each entry in the column involved. TABLE then goes on to read the key following '`9`'.

57. Seven — each '`q`' contributes one '`|`'.

58. With '`Q`' defined by '`\NewFormatKey Q{\ReadFormatKeys s4|q|q|q|}`', the answer to Exercise 54 becomes '`\BeginFormat Q.`'.

59. '`\NewFormatKey\left{\ReadFormatKeys l}`'.

60. '`\NewFormatKey B{\ReadFormatKeys b{\bf}}`'.

61. '`\NewFormatKey \m{\ReadFormatKeys l b{{}} m}`'.

62. Ten point. The message 'Point size 14 unavailable; using xpt' would also show up on your terminal and in your log file.

63.
```
\def\BeginVerticallyCenteredParagraph#1{%
  $\vcenter \bgroup \normalbaselines
  \parindent=0pt \hsize=#1 \strut}
\def\EndVerticallyCenteredParagraph{%
  \strut \egroup $}
\NewFormatKey C#1{%
  b{\BeginVerticallyCenteredParagraph{#1}}
  a{\EndVerticallyCenteredParagraph}}
```

64.
```
\NewFormatKey *#1#2{%
  \count0 =#1
  \toks0 ={}
  \loop
    \ifnum \count0 >0
    \toks0 =\expandafter{\the\toks0 #2}
    \advance \count0 by -1
  \repeat
  \expandafter\ReadFormatKeys\the\toks0 }
```

65. '`s`' and '`o`' specify the white space w between consecutive data columns. Because that white space is actually composed of the equal white spaces w_l and w_r to the left and right of the intervening rule column, it is necessary to set TEX's tabskip glue to $w_l = w/2$.

66. '`l`' is defined as '`\LeftGlue{} \RightGlue{\hfil}`'; '`r`' is defined as '`\LeftGlue {\hfil} \RightGlue{}`'.

67. '`\\0`' removes the usual strut from a table row. '`\\+00`' adds 0 strut units to the height and depth of that strut, leaving it unchanged.

68. First of all, the '`(...)`' option of TABLE's quantum systems can't be used with `\\` (or with any of the strut commands). Assuming that a decrease of 3 strut units would do just as well as the desired decrease of 3 points, B. L. User should have entered '`\\+{-2}{-3}`'. Remember that the suffix '`+`' to `\\` must be used to "add" something to the usual end-of-row strut.

69. (1) Nothing; (2) a vertical line of breadth 2`\LineThicknessUnit`; (3) nothing; (4) a vertical line of breadth 1 point; (5) nothing; (6) the "double vrule" ‖. With TABLE's default line-thickness unit in force, the table looks like this:

1	2	3	4	5 ‖

70. He got "1 This text spans eleven columns.", all in one column. He should have specified '`\use{11}`'.

71. A centered slanted "*Span*" spans columns 2, 3, and 4.

72. Specify '`o(.5in)`' in the formats for the 3rd and 4th columns. Alternatively, you could specify '`s(.5in)`' in the format for the 3rd column and restore the previous inter-column space by an appropriate '`s`' specification in the format for the 5th column.

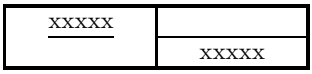
73. For a table with n columns, '`&\use{`n`}\=&`' is equivalent to `\_` (provided TABLE's `\LeftTabskip` and `\RightTabskip` glues have their default values of 0 pt).

74. The author set the HORIZONTAL LINES table with

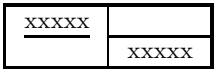
```
\BeginTable
  \BeginFormat
  |4   c  |2  c   |4 .
  \_4
  | xxxxx |       | \\
  | \-2   & \=2   & \\0
  |       | xxxxx | \\
  \_4
\EndTable
```

'`|2`', '`\-2`', and '`\=2`' could be shortened to '`|`', '`\-`', and '`\=`', respectively, since 2 is the default for TABLE's line-thickness factor.

75. With the command '`\WidenTableBy{.5in}`' in the prologue, the HORIZONTAL LINES table becomes



For comparison, here is the original table:



Note that in the new version of the table, the inter-column spaces are wider, and so is the line drawn by `\=` across the second column. The line drawn by `\_` across the first column has, however, exactly the same length in both tables.

76. Here are some possibilities: (1) '`\Left{\it Heading}`'. (2) '`\JustLeft \it Heading`'. (3) '`\ReFormat [lI] {Heading}`'. (4) '`\Use1 [lI] {Heading}`'. And here are two somewhat more TEXnical solutions: (5) '`\it Heading \hfill`' (this works because the `\hfill` overpowers the `\hfil` TABLE uses to position entries). (6) '`\omit \it Heading`' (this works because `\omit` is TEX's primitive for omitting column formats and because TEX places alignment entries flush left by default).

77. First, here is the RANDOM table set with a strut height-to-depth ratio of 9 to 4 (the author specified

```
\StrutUnit=12pt \divide\StrutUnit by 13
\StrutHeightFactor=9 \StrutDepthFactor=4
```

in the prologue to the table):

abcde	ABCED	1234	([])	2_3^2
leth	gjqpy	!@*+	';;?	e^{-a}
()	( )	...	—	a_{jik}

Here the strut height-to-depth ratio is 8.5 to 3.5:

abcde	ABCED	1234	([])	2_3^2
leth	gjqpy	!@*+	';;?	e^{-a}
()	( )	...	—	a_{jik}

Here the ratio is 8 to 3:

abcde	ABCED	1234	([])	2_3^2
leth	gjqpy	!@*+	';;?	e^{-a}
()	( )	...	—	a_{jik}

Here the ratio is 7 to 3:

abcde	ABCED	1234	([])	2_3^2
leth	gjqpy	!@*+	';;?	e^{-a}
()	( )	...	—	a_{jik}

If you prefer a strut height-to-depth ratio other than 8 to 3, you can override TABLE's defaults by placing your choices for the strut parameters at the start of your input file (but after the point at which the TABLE macros are loaded).

78. (1) '`+11`'; (2) '`+{-1}{-1}`'; (3) '`+{-7}{-2}`'. (Increases to the height and depth of a strut are more common than decreases, so TABLE's `\\+` command was written to make increases easier to code.)

79. Code the first two rows as

```
" \sl American " \sl French     " \sl Cooking  " \\+0{-1}
" \sl Chicken  " \sl Connection " \sl Methods  " \\+{-1}3
```

80. TABLE defines `\OpenUp` by:

```
\def\OpenUp#1#2{%
  \advance \StrutHeightFactor by #1
  \advance \StrutDepthFactor  by #2\relax}
```

81. The author set the LADDER-LIKE table with

```
\BeginTable
  \def\MS{\MakeStrut{2pt}{0pt}}
  \BeginFormat
  | w5 | w5 | w5 | w5 | w5 | .
  \_
  | \MS|    |    |    |    | \\0
  \_
  |    |    |    |    |    | \\
  \_
  " \MS"    "    "    "    " \\0
  \_
\EndTable
```

Note that the rows containing the `\MS` command are exactly 2 points high and 0 points deep because TABLE's standard strut was zeroed out for these rows.

82. Include the command '`\MakeStrut{15pt}{10pt}`' in some entry for the row.

83. '`\StandardTableStrut`' is short for

'`\MakeStrut{\StrutHeightFactor\StrutUnit}`
`  {\StrutDepthFactor\StrutUnit}`'.

84. Include the command '`\AugmentedTableStrut{`h`}{`d`}`' in some entry. (This is, in fact, essentially what TABLE does.)

85. Input:

```
\BeginTable
\def\erf{\mathop{\rm erf}}
\def\E{\enlarge33}
\BeginFormat
|4 l     |  r M o0   | \M                                  |4
\EndFormat
\_4
|\it Name| \use2 \JustCenter \it Definition        | \\+44
\_
| Gamma  | \Gamma(z) "=\E{\int_0^\infty t^{z-1}e^{-t}\,dt}| \\ \_
| Sine   | \sin(x)   "=\E{{1\over 2i}(e^{ix} - e^{-ix})}  | \\ \_
| Error  | \erf(z)   "=\E{{2\over \sqrt\pi} \int_0^z
                          e^{-z^2}\,dz}                  | \\ \_
| Bessel | J_0(z)    "=\E{{1\over \pi} \int_0^\pi
                          \cos(z\sin \theta)\,d\theta}   | \\ \_
| Zeta   | \zeta(s)  "=\E{\sum_{k=1}^\infty k^{-s}
                                      \quad (\Re s>1)} | \\
\_4
\EndTable
```

Output:

Name	*Definition*
Gamma	$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t}\,dt$
Sine	$\sin(x) = \frac{1}{2i}(e^{ix} - e^{-ix})$
Error	$\operatorname{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-z^2}\,dz$
Bessel	$J_0(z) = \frac{1}{\pi} \int_0^\pi \cos(z \sin\theta)\,d\theta$
Zeta	$\zeta(s) = \sum_{k=1}^\infty k^{-s} \quad (\Re s > 1)$

The spacing of the rows isn't as uniform as in the original table.

86. TABLE defines `\enlarge` by

'`\def\enlarge#1#2{\Enlarge{#1\StrutUnit}{#2\StrutUnit}}`'.

Remember that `\enlarge` uses TABLE's strut unit, but `\Enlarge` does not; `\Enlarge` is more versatile than `\enlarge`, but typically requires a few more keystrokes to specify.

87. (1) One half-line, i.e., $\frac{1}{2}((17+7) \times .5\,\text{pt}) = 6$ points. (2) $4 \times .5\,\text{pt} = 2$ points. (3) 12 points. (4) -6 points. (5) $\frac{1}{10}''$.

88. Here is one possibility:

```
\BeginTable
  \BeginFormat
  | c          | cM                  | .      \_
  | A big      | \Lower{\int_0^1}    | \\+30
  | integral   |                     | \\+03 \_
\EndTable
```

produces

A big integral	$\int_0^1$

89. Input:

```
\BeginTable
  \BeginFormat
  | c         | cM                | .      \_
  | A big     |                   | \\+30
  |           | \Smash{\int_0^1}  | \\0
  | integral  |                   | \\+03 \_
\EndTable
```

Output:

A big integral	$\int_0^1$

90. (1) '`\NormalLTU={1in \divide \LineThicknessUnit by 600 }`';
(2) '`\NormalICSU ={1em}`' (this is a bad choice, because it doesn't let the inter-column white space stretch or shrink);
(3) '`\NormalCWU ={1cm}`';
(4) '`\NormalKU ={.1in}`';
(5) '`\NormalSU ={\normalbaselineskip \divide \StrutUnit by 24 }`'.

91. TABLE doesn't have a `\NormalTableFactors` command because the factor parameters typically don't change throughout a document. In contrast, the unit parameters may change many times over (once for each shift in the size of type).

92. The author specified '`\WidenTableBy{.5in}`' in the prologue to the first table, and '`\Expand`' in the prologue to the second table. The rest of the code is the same as before.

93. With `\LongLines` replaced by `\Expand` in the prologue, the STATISTICS table becomes:

Estimate	*Standard Error*	*Correlations*		
$\hat{\beta}_1 = -1.61$	0.38	1.0		
$\hat{\beta}_2 = -4.97$	0.47	0.53	1.0	
$\hat{\beta}_3 = -3.32$	0.43	0.57	0.66	1.0

For comparison, here is the original version of the table:

Estimate	*Standard Error*	*Correlations*		
$\hat{\beta}_1 = -1.61$	0.38	1.0		
$\hat{\beta}_2 = -4.97$	0.47	0.53	1.0	
$\hat{\beta}_3 = -3.32$	0.43	0.57	0.66	1.0

The inter-column white space is stretched in the new version of the table, but not in the original version.

94. This is what he got:

	1	2	
	3	4	
	5	6	

95. The author set the SINGLE-CELLED table with

```
\BeginTable
  \OpenUp22
  \BeginFormat | |. \_
  |\BeginTableParBox{2.8in}This entry was set using {\tt \string
    \BeginTableParBox} and {\tt \string \EndTableParBox}. The line
    length is 2.8 inches. Lines are set ragged-right because that's
    the default style {\tt \string \EveryTableParBox} specifies.%
  \EndTableParBox | \\ \_
\EndTable
```

96. Specify '`\EveryTableParBox={\noindent \tolerance=1000 }`' in the prologue to the table. ('`\tolerance=1000 `' lets TEX stretch the inter-word glue more than it normally would.)

97. (1) 1 point; (2) 2 points; (3) 12 points; (4) 1/4″.

98. Input:

```
\BeginTable
  \def\L{\JustLeft}
  \BeginFormat
  |    ck      |    ck       |      ck         |
  \EndFormat
  \_
  \Vspace
  "  \use3  \it AT\&T Common Stock            "  \\
  \Vspace
  "  \use3  \-                                "  \\0
  \Vspace
  "  \it Year  "  \it Price  "  \it Dividend   "  \\
  \Vspace
  "  \-        "  \-         "   \-            "  \\0
  \Vspace
  "  1971      "  41--54     "   \$2.60        "  \\
  "  ~~~2      "  41--54     "    ~2.70        "  \\
  "  ~~~3      "  46--55     "    ~2.87        "  \\
  "  ~~~4      "  40--53     "    ~3.24        "  \\
  "  ~~~5      "  45--52     "    ~3.40        "  \\
  "  ~~~6      "  51--59     "    ~~.95\rlap*  "  \\
  \Vspace \_ \Vspace
  "  \use3 \L * (first quarter only)          "  \\
\EndTable
```

Output:

AT&T Common Stock

Year	*Price*	*Dividend*
1971	41–54	$2.60
2	41–54	2.70
3	46–55	2.87
4	40–53	3.24
5	45–52	3.40
6	51–59	.95*

* (first quarter only)

99. Place the instruction '`\Vspace(-2\StrutUnit)`' between the code for the first row and the code for the second row. ('`\Vspace-2`' doesn't work.)

100. With the instructions

```
\newdimen\PeriodWidth
{\catcode`\:=\active
\def\MakeNonprintingPeriod{%
  \setbox0 =\hbox{.}
  \PeriodWidth=\wd0
  \catcode`\:=\active
  \def:{\kern\PeriodWidth}}}
```

placed near the start of your input file, the STACK

```
46
23
15
 6.5
 2.1
```

of numbers in Exercise 8 would result from

```
\BeginTable
  \MakeNonprintingPeriod
  \BeginFormat
  |    c     | .
  \_
  |    46:~  | \\+20
  |    23:~  | \\
  |    15:~  | \\
  |    ~6.5  | \\
  |    ~2.1  | \\+02
  \_
\EndTable
```

101. '`\BackSpace (.25in)`' inserts a kern of width $-1/4''$. '`\BackSpace 12`' inserts a kern of width -6 em. '`\BackSpace 1 2`' and '`\BackSpace A`' each insert a kern of width $-.5$ em.

102.
```
\def\BeginInsertedTable{%
  \midinsert
    \hbox to \hsize \bgroup
      \hss
      \BeginTable}
\def\EndInsertedTable{%
      \EndTable
      \hss
    \egroup
  \endinsert}
```

103. Here's one solution:

```
\def\BeginIndentedTable{%
  $$\hbox to \hsize \bgroup
      \hskip\parindent
      \BeginTable}
\def\EndIndentedTable{%
      \EndTable
      \hss
    \egroup$$}
```

104. (1) This is a case for `\BeginTable[c]`:

```
\BeginTable[c] \BeginFormat | c | . \_
  | top    | \\
  | middle | \\
  | bottom | \\ \_
\EndTable
```

`\BeginTable` could be used in place of `\BeginTable[c]`.

(2) This is another case for `\BeginTable[c]`:

```
\BeginTable[c] \BeginFormat | cm | .
  " \alpha + \beta             " \\
  " \gamma + \delta + \epsilon " \\
\EndTable
```

Actually, structures such as this one are more efficiently programmed in terms of macros, such as Plain TEX's `\eqalign`, especially designed for the purpose. Using `\eqalign`, the code would be

```
$\eqalign{\alpha + \beta \cr \gamma + \delta + \epsilon \cr}$
```

(3) This is a case for `\BeginTable[t]`:

```
\BeginTable[t] \BeginFormat | c | .
  " top    " \\
  " bottom " \\
\EndTable
```

(4) Here '`\BeginTable[t]`' alone won't work because it would place the baseline of the table at the top of the table. To make the baseline of this table the baseline of the top row, it's necessary to `\raise` the table. Since the line drawn across the top of the table is `\StrutHeightFactor` strut units above the baseline of the top row, the following code does the job:

```
\dimen0 = \StrutHeightFactor\StrutUnit
\raise \dimen0 \hbox{%
  \BeginTable[t] \BeginFormat | c | . \_
    | top    | \\
    | bottom | \\ \_
  \EndTable}
```

105.
```
$$\BeginTable[u]
  \LongLines
  \def\L{\JustLeft}
  \BeginFormat
  |    ck      |    ck       |      ck          | .
  &  \use3 \=                                    &  \\0
  "  \use3  \it AT\&T Common Stock               "  \\+22
  "  ~~~6      "  51--59     "    ~~.95\rlap*    "  \\+02
  &  \use3 \=                                    &  \\0
  "  \use3 \L * (first quarter only)             "  \\+20
\EndTable$$
```
(The omitted code is the same as before.)

106. Enter '`\EveryTable{\def\C{\JustCenter}}`' before your first table.

107. This:

|	A half-inch	$\|x\|$	$\bar{o}$
"		$\|\|x\|\|$	net_gain

108. The author set the BUDGET TRANSFERS table with

```
\BeginTable
  \def\C{\JustCenter}
  \BeginFormat
  |    l          |   c       |   c   |   cm          | . \_
  | \use4 \C 1970 Federal Budget Transfers            | \\+20
  | \use4 \C (in billions of dollars)                 | \\+02
  \_3
  |\C\Lower{State}| Taxes     | Money | \C\Lower{Net} | \\+20
  |               | Collected | Spent |               | \\+02
  \_
  | New York      | 22.91     | 21.35 | -1.56         | \\+20
  | New Jersey    | ~8.33     | ~6.96 | -1.37         | \\
  | Connecticut   | ~4.12     | ~3.10 | -1.02         | \\
  | Maine         | ~0.74     | ~0.67 | -0.07         | \\
  | California    | 22.29     | 22.42 | +0.13         | \\
  | New Mexico    | ~0.70     | ~1.49 | +0.79         | \\
  | Georgia       | ~3.30     | ~4.28 | +0.98         | \\
  | Mississippi   | ~1.15     | ~2.32 | +1.17         | \\
  | Texas         | ~9.33     | 11.13 | +1.80         | \\+02
  \_
\EndTable
```

109. The author set the start of the SUMMARY OF PARAMETERS table with

```
\BeginTable
  \Expand
  \OpenUp11
  \def\C{\JustCenter}
  \def\E{\\ \_1 }
  \BeginFormat
  | lT                        |1 Tp(2in)    |1 c       |.
  \_
  | \it Parameter             | \it Default |\it Page\/| \\+22
  \_
  " \use3 \C \it Quantum System Parameters             " \\+53
  \_
  | \string\ColumnWidthFactor | 10          | 16--17   | \E
  | \string\ColumnWidthUnit   | .5em        | 16--17   | \E
```

110. The author set the FONT LAYOUT table with

```
\BeginTable
  \OpenUp11
  \LineThicknessFactor=1
  \eightpoint % From Appendix E of the TeXBook
  \NormalTableUnits \rm
  \def\O#1{\hbox{\rm\'{}\kern-.2em\it#1\/\kern.05em}}
  \def\X#1{\hbox{\rm\H{}\tt#1}}
  \global\count255=0
  \def\:{\enlarge11{\char\count255}%
    \global\advance\count255 by 1 }
  \def\EL#1{\\ &\use9 \=& \Smash{\strut\X{#1x}} \\0 }
  \def\OL{\\ \_}
  \BeginFormat
    |  c      |  c |  c |  c |  c |  c |  c |  c |  c |  .
    "         | \O0| \O1| \O2| \O3| \O4| \O5| \O6| \O7| \\+11 \_
    " \O{00x} | \: | \: | \: | \: | \: | \: | \: | \: | \EL0
    " \O{01x} | \: | \: | \: | \: | \: | \: | \: | \: | \OL
    " \O{02x} | \: | \: | \: | \: | \: | \: | \: | \: | \EL1
    " \O{03x} | \: | \: | \: | \: | \: | \: | \: | \: | \OL
    " \O{04x} | \: | \: | \: | \: | \: | \: | \: | \: | \EL2
    " \O{05x} | \: | \: | \: | \: | \: | \: | \: | \: | \OL
    " \O{06x} | \: | \: | \: | \: | \: | \: | \: | \: | \EL3
    " \O{07x} | \: | \: | \: | \: | \: | \: | \: | \: | \OL
    " \O{10x} | \: | \: | \: | \: | \: | \: | \: | \: | \EL4
    " \O{11x} | \: | \: | \: | \: | \: | \: | \: | \: | \OL
    " \O{12x} | \: | \: | \: | \: | \: | \: | \: | \: | \EL5
    " \O{13x} | \: | \: | \: | \: | \: | \: | \: | \: | \OL
    " \O{14x} | \: | \: | \: | \: | \: | \: | \: | \: | \EL6
    " \O{15x} | \: | \: | \: | \: | \: | \: | \: | \: | \OL
    " \O{16x} | \: | \: | \: | \: | \: | \: | \: | \: | \EL7
    " \O{17x} | \: | \: | \: | \: | \: | \: | \: | \: | \OL
    "         |\X 8|\X 9|\X A|\X B|\X C|\X D|\X E|\X F| \\+11
  \EndTable
```

The TEXbook's recipe for `\:` repositions entries that are unusally high or deep; expressed in terms of TABLE's parameters, *The TEXbook*'s definition would read

```
\def\:{%
  \setbox0=\hbox{\enlarge11{\char\count255}}%
  \ifdim\ht0>\StrutHeightFactor\StrutUnit
    \Reposition
  \else
    \ifdim\dp0>\StrutDepthFactor\StrutUnit
      \Reposition
    \fi
  \fi
  \box0
  \global\advance\count255 by 1 }
```

```
\def\Reposition{%
  \setbox0=\hbox{$\vcenter{%
    \kern\StrutUnit
    \box0
    \kern\StrutUnit}$}}
```

B. SUMMARY OF BUILT-IN KEYS

Each of TABLE's built-in keys is listed below in alphabetical order along with a brief description and a page reference for more details.

Key	*Meaning*	*Page*
`*{`*n*`}{`⟨keys⟩`}`	Repeat ⟨keys⟩ *n* times.	29
`.`	Abbreviation for `\EndFormat`.	29
`a{`⟨tokens⟩`}`	Place ⟨tokens⟩ after each entry.	28
`B`	Use boldface type.	19
`b{`⟨tokens⟩`}`	Place ⟨tokens⟩ before each entry.	28–29
`c`	Center entries.	19
`f`⟨font⟩	Use ⟨font⟩ type.	19–20
`I`	Use italic type.	19
`i`⟨spec$_K$⟩	Place kern of size ⟨value$_K$⟩ before each entry.	23–25
`j`⟨spec$_K$⟩	Place kern of size ⟨value$_K$⟩ after each entry.	23–25
`k`⟨spec$_K$⟩	Place kern of size ⟨value$_K$⟩ before and after each entry.	23–25
`l`	Set entries flush left.	19
`\M`	Use variant of display-style math mode.	20
`M`	Use display-style math mode.	20
`\m`	Use variant of math mode.	20
`m`	Use math mode.	20
`N`⟨field width⟩	Use numeric (math) mode.	20–22
`n`⟨field width⟩	Use numeric (non-math) mode.	20–22
`o`⟨spec$_{ICS}$⟩	Set inter-column space to ⟨value$_{ICS}$⟩, only to the right of the current column.	22–23
`p`⟨spec$_{CW}$⟩	Use paragraph mode with a line length of ⟨value$_{CW}$⟩.	25–27
`R`	Use roman type.	19
`r`	Set entries flush right.	19
`S`	Use slanted type.	19
`s`⟨spec$_{ICS}$⟩	Set subsequent inter-column space to ⟨value$_{ICS}$⟩.	22–23
`T`	Use typewriter type.	19
`w`⟨spec$_{CW}$⟩	Set minimum column width to ⟨value$_{CW}$⟩.	25
`\{`	Enclose entries in `{}`'s.	28
`|{`⟨alternate vertical line⟩`}`	Column separator specifying user-defined vertical line.	27–28
`|`⟨spec$_{LT}$⟩	Column separator specifying vertical lines of breadth ⟨value$_{LT}$⟩.	27

C. SUMMARY OF COMMANDS

Each of TABLE's commands is listed alphabetically below, along with a brief description and a page reference for more details. Commands marked with a '⋆' have their stated meaning only in the table environment.

Command		*Meaning*	*Page*
`"`	⋆	End entry without a trailing vertical line.	37
`\-`⟨spec$_{LT}$⟩	⋆	Draw horizontal line of breadth ⟨value$_{LT}$⟩ exactly the width of the current column.	39
`\=`⟨spec$_{LT}$⟩	⋆	Draw horizontal line of breadth ⟨value$_{LT}$⟩ across current column and half-way into adjacent inter-column white space.	39
`\\`	⋆	End row with a standard strut.	36
`\\+{`h`}{`d`}`	⋆	End row with a strut that is h strut units higher and d strut units deeper than usual.	36
`\\0`	⋆	End row, omitting the standard strut.	36
`\_`⟨spec$_{LT}$⟩	⋆	Draw horizontal line of breadth ⟨value$_{LT}$⟩ across the table.	39
`|*`	⋆	Draw a "pseudo-vrule" across the row between columns.	37–38
`|`⟨spec$_{LT}$⟩	⋆	Draw a vertical line of breadth ⟨value$_{LT}$⟩ across the row between columns.	37–38
`|{`⟨tokens⟩`}`	⋆	Place ⟨tokens⟩ between columns.	37–38
`|`	⋆	End entry with a trailing vertical rule.	37
`~`	⋆	Typeset a non-printing digit.	50
`\ActivateBarAndQuote`		Make '`|`' and '`"`' active characters.	52
`\AugmentedTableStrut {`η`}{`δ`}`		Make strut that is η strut units higher and δ strut units deeper than a standard strut.	42–43
`\BackSpace`⟨spec$_K$⟩		Insert a kern of width $-$⟨value$_K$⟩.	51
`\BeginFormat`		Begin format section.	36
`\BeginTable`		Start table environment.	36
`\BeginTableParBox {`⟨line length⟩`}`		Begin entry to be set in paragraph mode with a line length of ⟨line length⟩.	49
`\Center{`⟨entry⟩`}`	⋆	Center ⟨entry⟩ and use the rest of the column format.	39–40
`\DQuote`		Typeset the character '`"`'.	37
`\EndFormat`		End format section.	36
`\EndTable`		End table environment.	36
`\EndTableParBox`		End paragraph mode entry.	49
`\Enlarge{`H`}{`D`} {`⟨entry⟩`}`		Typeset ⟨entry⟩ in the format for its column, increasing the natural height and depth of the result by the dimensions H and D, respectively.	43

Command		*Meaning*	*Page*
`\enlarge{`η`}{`δ`}{`⟨entry⟩`}`		`\Enlarge{`$\eta \times$ strut unit`}{`$\delta \times$ strut unit`}{`⟨entry⟩`}`	43
`\Expand`		Stretch table to full width of the page.	47–48
`\JustCenter`		Center entry and omit the column format.	40
`\JustLeft`		Position entry flush left and omit the column format.	40
`\JustRight`		Position entry flush right and omit the column format.	40
`\Left{`⟨entry⟩`}`	⋆	Position ⟨entry⟩ flush left and use the rest of the column format.	39–40
`\LongLines`		Extend `\_` lines across the entire page.	48
`\Lower`⟨spec$_{RL}$⟩`{`⟨entry⟩`}`		Lower ⟨entry⟩ by ⟨value$_{RL}$⟩.	43–45
`\MakeStrut{`H`}{`D`}`		Make strut having the dimensions H and D for its height and depth, respectively.	42
`\NewFormatKey`		Define new format key.	36
`\NormalTableUnits`		Reestablish the default units for TABLE's quantum systems.	45–46
`\OpenUp{`h`}{`d`}`		Add h strut units to the height and d strut units to the depth of the usual end-of-row strut.	41–42
`\Raise`⟨spec$_{RL}$⟩`{`⟨entry⟩`}`		Raise ⟨entry⟩ by ⟨value$_{RL}$⟩.	43–45
`\ReadFormatKeys`		Go back to reading format keys.	36
`\ReFormat[`⟨format keys⟩`] {`⟨entry⟩`}`		Set ⟨entry⟩ according to ⟨format keys⟩, ignoring the current column format.	40
`\Right{`⟨entry⟩`}`	⋆	Position ⟨entry⟩ flush right and use the rest of the column format.	39–40
`\SetTableToWidth {`⟨dimen⟩`}`		Make table exactly ⟨dimen⟩ wide.	47
`\Smash{`⟨entry⟩`}`		Typeset ⟨entry⟩ so as to be centered vertically about the baseline, but declare the height and depth of the result to be zero.	45
`\StandardTableStrut`		Make standard end-of-row strut.	42
`\TaBlE`		TABLE logo.	52
`\use{`c`}`	⋆	Use the space of the next c columns and the format of the last of those columns.	38
`\Use{`c`}[`⟨keys⟩`]{`⟨entry⟩`}`	⋆	`\use{`c`}\ReFormat[`⟨keys⟩`]{`⟨entry⟩`}`.	40
`\VBar`		Typeset the character '|'.	37
`\Vspace`⟨spec$_{VS}$⟩		Insert a gap of ⟨value$_{VS}$⟩ between rows.	49–50
`\WidenTableBy{`⟨dimen⟩`}`		Widen table horizontally by ⟨dimen⟩.	47

D. SUMMARY OF PARAMETERS

This appendix lists each of TABLE's parameters along with its default value and a page reference for more details.

Parameter	*Default*	*Page*
	Quantum System Parameters	
`\ColumnWidthFactor`	`10`	16–17
`\ColumnWidthUnit`	`.5em`	16–17
`\InterColumnSpaceFactor`	`3`	16–17
`\InterColumnSpaceUnit`	`.5em plus 1fil minus .25em`	16–17
`\KernFactor`	`1`	16–17
`\KernUnit`	`.5em`	16–17
`\LineThicknessFactor`	`2`	16–17
`\LineThicknessUnit`	`.00333in`	16–17
`\StrutDepthFactor`	`3`	40–41
`\StrutHeightFactor`	`8`	40–41
`\StrutUnit`	`1.0909pt`	40–41
`\VspaceFactor`	`2`	49
	Recipes for default quantum system units	
`\NormalCWU`	`{.5em}`	45–46
`\NormalICSU`	`{.5em plus 1fil minus .25em}`	45–46
`\NormalKU`	`{.5em}`	45–46
`\NormalLTU`	`{1in \divide \LineThicknessUnit by 300 }`	45–46
`\NormalSU`	`{\normalbaselineskip \divide \StrutUnit by 11 }`	45–46
	Miscellaneous parameters	
`\EveryTable`	{⟨null⟩}	56
`\EveryTableParBox`	`{\parindent=0pt \raggedright \rightskip=0pt plus 4em \relax}`	26
`\LeftTabskip`	`0pt`	55
`\PseudoVrule`	⟨null⟩	37–38
`\RightTabskip`	`0pt`	55
`\TracingFormats`	`0`	32–34
`\TracingKeys`	`0`	34–35

You should change the '`300`' in the `\NormalLTU` recipe to the number of pixels per inch on your printer and you should change the '`11`' in the `\NormalSU` recipe to the sum of your defaults for the strut height and depth factors.

E. BIBLIOGRAPHY

Efron, B. and Thisted, R. A. (1976). Estimating the Number of Unseen Species: How Many Words Did Shakespeare Know? *Biometrika* **63** 435–447.

Ferguson, Michael J. (1986). Table Making — the INRSTEX Method. *TEXniques* 2, TEX Users Group, Providence.

Knuth, Donald E. (1984). *The TEXbook*. Addison-Wesley, Reading.

Lamport, Leslie (1986). LATEX*: A Document Preparation System*. Addison-Wesley, Reading.

Lesk, M. E. (1984). Tbl — A Program to Format Tables. Reprinted in *Unix User's Manual: Supplementary Documents*. University of California, Berkeley.

Schoener, T. W. (1970). Nonsynchronous spatial overlap of lizards in patchy habitats. *Ecology* **51** 408–418.

Student (W. S. Gosset) (1908). On the probable error of a mean. *Biometrika* **6** 1–25.

F. INDEX

Underlined page numbers give the main source of information about whatever is being indexed. Pages with *slanted* numbers contain example(s) of the use of the concept in question.