Typesetting at the CWI — Part 2

by Jaap Akkerhuis

Handling of tabular material
Although it is possible to do this "by hand", the easiest way of handling tabular material is using the TBL [1] preprocessor. This preprocessor turns a simple description of a table into TROFF commands.
I will illustrate TBL with some examples. The TBL preprocessor will only process lines between the delimiters, .TS and .TE, and so, in general, a table will look like

```
.TS
options ;
format .
data
.TE
```

The symbol ⓣ in the input represents a tab character.
The first example is a three part table, with the first items centered, and the rest of the items left adjusted.

Input:

```
.TS
box;
c c c
l l l.
Language ⓣ Authors ⓣ Runs on

Fortran ⓣ Many ⓣ Almost anything
PL/I ⓣ IBM ⓣ 360/370
C ⓣ BTL ⓣ 11/45,H6000,370
BLISS ⓣ Carnegie-Mellon ⓣ PDP-10,11
IDS ⓣ Honeywell ⓣ H6000
Pascal ⓣ Stanford ⓣ 370
.TE
```

The first part of this article appeared in CWI Newsletter no. 3 (June 1984).

Output:

Language	Authors	Runs on
Fortran	Many	Almost anything
PL/I	IBM	360/370
C	BTL	11/45,H6000,370
BLISS	Carnegie-Mellon	PDP-10,11
IDS	Honeywell	H6000
Pascal	Stanford	370

The following table has all entries boxed, and the first entry centered and spanned. The complete table is expanded over the available line length as well.

Input:

```
.TS
expand allbox;
c c c c c c c c c
n n n n n n n n n.
Type ⓣ Vf ⓣ If ⓣ Wa ⓣ Wg2 ⓣ Ea ⓣ Eg2 ⓣ Wo ⓣ Mhz
807 ⓣ 6.3 ⓣ .9 ⓣ 30 ⓣ 3.5 ⓣ 750 ⓣ 300 ⓣ 50 ⓣ 60
813 ⓣ 10 ⓣ 5 ⓣ 100 ⓣ 22 ⓣ 2000 ⓣ 400 ⓣ 260 ⓣ 30
815 ⓣ 6.3 ⓣ 1.6 ⓣ 25 ⓣ 4 ⓣ 500 ⓣ 200 ⓣ 56 ⓣ 125
829A ⓣ 6.3 ⓣ 2.5 ⓣ 40 ⓣ 7 ⓣ 750 ⓣ 240 ⓣ 87 ⓣ 200
832A ⓣ 6.3 ⓣ 1.6 ⓣ 15 ⓣ 5 ⓣ 750 ⓣ 250 ⓣ 26 ⓣ 200
```

Output:

Type	Vf	If	Wa	Wg2	Ea	Eg2	Wo	Mhz
807	6.3	.9	30	3.5	750	300	50	60
813	10	5	100	22	2000	400	260	30
815	6.3	1.6	25	4	500	200	56	125
829A	6.3	2.5	40	7	750	240	87	200
832A	6.3	1.6	15	5	750	250	26	200

The next table is more complicated. To get the complicated header right, it uses the `.T&` (table continue) command. This table is boxed again, and has some entries boxed, by using the | specifier. The = in the data part of the table denotes a double line over the full width of the column. A _ specifies a single line. The `\^` specifies a vertical aligned column. Note the difference between this table and the "all boxed" one in the previous example.

Input:

```
.TS
box;
cfB s s s s s.
Ranges of Typical Commercial Spark Stations
_
.T&
c | c s s s | c
c | c s s s | c
c | c | c s s | c
c | c | c s s | c
c | c | c | c | c | c.
 ⓣ Range in nautical miles ⓣ Wave-
Power ⓣ _ ⓣ \^
\^ⓣ Over sea ⓣ Over Land ⓣ Length
required ⓣ \^ⓣ _ ⓣ \^
\^ⓣ \^ⓣ Flat ⓣ Hilly ⓣ Mountainous ⓣ (metres)
=
.sp 0.5
.T&
n | n | n | n | n | n.
300 watts ⓣ 100 ⓣ 77 ⓣ 30 ⓣ 13 ⓣ 300
\^ⓣ 100 ⓣ 95 ⓣ 73 ⓣ 52 ⓣ 1200
1\(12 kw ⓣ 220 ⓣ 170 ⓣ 67 ⓣ 28 ⓣ 300
\^ⓣ 220 ⓣ 210 ⓣ 160 ⓣ 115 ⓣ 1200
3 kw ⓣ 280 ⓣ 220 ⓣ 84 ⓣ 36 ⓣ 300
\^ⓣ 280 ⓣ 270 ⓣ 200 ⓣ 145 ⓣ 1200
5 kw ⓣ 340 ⓣ 260 ⓣ 100 ⓣ 43 ⓣ 300
\^ⓣ 340 ⓣ 325 ⓣ 240 ⓣ 175 ⓣ 1200
10 kw ⓣ 470 ⓣ 360 ⓣ 138 ⓣ 59 ⓣ 300
\^ⓣ 470 ⓣ 450 ⓣ 330 ⓣ 240 ⓣ 1200
.nr x \n(.n-7n \" Remember position
.TE
.ta \nxuL             \" set a tab
 ⓣ \fI(Sankey)\fP     \" goto tab and print
```

Output:

Ranges of Typical Commercial Spark Stations					
Power required	Range in nautical miles				Wave-length (metres)
	Over sea	Over Land			
		Flat	Hilly	Mountainous	
300 watts	100	77	30	13	300
	100	95	73	52	1200
1½ kw	220	170	67	28	300
	220	210	160	115	1200
3 kw	280	220	84	36	300
	280	270	200	145	1200
5 kw	340	260	100	43	300
	340	325	240	175	1200
10 kw	470	360	138	59	300
	470	450	330	240	1200

(Sankey)

A more exotic use of TBL is to generate a picture with it, as in the next example. The **ce0** stands for **c**entered **e**qualwidth column with space of **0** ens* between the columns. Note that the complete table is centered on the page as well.

Input:

```
.TS
center;
ce0 ce0 ce0 ce0 ce0 ce0 ce0 ce0 ce0 ce0 ce0 ce.
- Ⓣ - Ⓣ - Ⓣ - Ⓣ - Ⓣ - Ⓣ - Ⓣ - Ⓣ - Ⓣ - Ⓣ - Ⓣ -
| Ⓣ . Ⓣ . Ⓣ . Ⓣ . Ⓣ . Ⓣ . Ⓣ . Ⓣ . Ⓣ . Ⓣ . Ⓣ +
| Ⓣ . Ⓣ . Ⓣ @ Ⓣ . Ⓣ . Ⓣ . Ⓣ . Ⓣ ] Ⓣ . Ⓣ . Ⓣ |
| Ⓣ . Ⓣ . Ⓣ . Ⓣ . Ⓣ B Ⓣ . Ⓣ . Ⓣ . Ⓣ . Ⓣ . Ⓣ |
| Ⓣ . Ⓣ . Ⓣ . Ⓣ . Ⓣ . Ⓣ . Ⓣ . Ⓣ . Ⓣ . Ⓣ . Ⓣ |
- Ⓣ - Ⓣ - Ⓣ - Ⓣ - Ⓣ + Ⓣ - Ⓣ - Ⓣ - Ⓣ - Ⓣ - Ⓣ -
.TE
```

* An en is the width on the character n in the current font.

Output:

```
- - - - - - - - - - - - -
| . . . . . . . . . . . +
| . . @ . . . . ] . . |
| . . . . B . . . . . |
| . . . . . . . . . . . |
- - - - - - + - - - - - -
```

Handling of mathematics material
Mathematics material is handled by the EQN [2] preprocessor. Just as the TBL preprocessor it arranges low level TROFF commands.
The input resembles a programming language which allows specification of the equation to be typeset by describing the desired output in words. It is claimed that it is easy to use and one can learn to typeset an in-line formula like $\int_0^1 \sin \pi x \, dx$ or displayed equations like

$$D_n = \frac{1}{n} \sum_{k=1}^{n} \left| \exp \left[\frac{k}{n} 2\pi i \right] - 1 \right| =$$

$$= \frac{1}{n} \sum_{k=1}^{n} \left| \left[\cos \frac{2\pi k}{n} + i \sin \frac{2\pi k}{n} \right] - 1 \right| =$$

$$= \frac{1}{n} \sum_{k=1}^{n} \left[2 - 2\cos \frac{2\pi k}{n} \right]^{\frac{1}{2}} = \frac{2}{n} \sum_{k=1}^{n} \sin \frac{\pi k}{n}$$

in an hour or so.

EQN will process everything in the input between `.EQ`, `.EN` and pairs of dollar signs (`$`), the most used *delimiters*. The `.EQ` and `.EN` with their optional arguments are copied through untouched.

The *—ms*-package will center the given equation. A `.EQ L` will make the displayed equation a left justified block, and `.EN I` somewhat indented. An optional third argument gives the equation number placed towards the right margin.

```
.EQ I (3a).
a = b + c
.EN
```

$$a = b + c \qquad (3a).$$

Note that this is not a feature of EQN, but of the macro package actually used. So if you want to have the equation number in front, you have to rewrite the `.EQ` macro.

One of the problems of the beginning user is that input spaces are nearly always ignored, so the following different input lines

```
x=y+z

x = y + z

x    =    y
    + z
```

will all result in the same output:

$$x = y + z$$

The (input) spaces are only used as obligatory keyword delimiters, so `$x  sub2$` will give you *xsub* 2 instead of x_2.

However, if you want to have spaces in the output, the ˜ character will give you a space and the . half a space. So

```
x˜=˜y˜+˜z
```

will look like

$$x \ = \ y \ + \ z.$$

Subscripts and superscripts, special names and brackets

As you will have noticed before, subscripts are easily generated by the `sub` command. Superscripts are made in the same way, so

```
x sup 2 + y sub k
```

gives

$$x^2 + y_k$$

Of course, more complicated matter can be treated in the same way:

```
e sup {i pi sup {rho +1}}
```

$$e^{i\pi^{\rho+1}}$$

The braces `{}` are used here to group things together. Everything between them will be treated by EQN as a block. Also `pi` and `rho` are translated into the corresponding Greek character. To get the braces itself one uses `{` and to get large braces in the output, the `left` and `right` constructs are used. See the examples in the following paragraph.

Fractions and roots

In the following example, you see the use of big brackets and the way fractions are made with the `over` construct.

```
left { a over b + 1 right }
 ~=~ left ( c over d right )
 + left [ e right ]
```

$$\left\{ \frac{a}{b} + 1 \right\} = \left(\frac{c}{d} \right) + \left[e \right]$$

Square roots are made just as easily

```
sqrt a+b
```

$$\sqrt{a+b}$$

One should avoid big square roots, they don't look nice.

```
sqrt {a sup 2 over b sub 2}
```

$$\sqrt{\frac{a^2}{b_2}}$$

One can better rearrange the formula so as to get

$$(a^2 / b_2)^{½}$$

This is done by

```
(a sup 2 /b sub 2 ) sup half
```

Large operators and matrices

Large operators like integrals and summations are made straightforwardly as in:

```
prod from { i != j } to n
     { ( 1-x sub i x sub j sup -1 ) } sup k
```

$$\prod_{i \neq j}^{n} (1 - x_i x_j^{-1})^k$$

A matrix is made by

```
matrix {
  ccol { x sub i above y sub i }
  ccol { x sup 2 above y sup 2 }
  }
```

which results in

x_i x^2

y_i y^2

There are more ways of placing things on top of each other, and also for letting parts of an equation line up with each other. Also, font changes can take place on demand.

As a last example, here follows the input of the big display from the beginning. Note the use of defining macros with `define name @ macro body @`.

```
.EQ I
define T1 @ 1 over n sum from { k = 1 } to n @
D sub n ~ mark = ~  T1 left | ~ exp
     left ( k over n ~ 2 pi i  right ) ~-1 right | ~=
.EN
.EQ I
lineup = ~ T1 left | ~ left ( cos { 2 pi k } over n
     ~ + ~ i ~ sin { 2 pi k } over n right )
     ~ -1 right | ~=
.EN
.EQ I
lineup = ~ T1 left ( 2 ~ - ~ 2 cos  {2 pi k } over n
     right ) sup { 1 over 2 } ~=~ 2 over n
     sum from { k = 1 } to n sin { pi k } over n
.EN
```

Bibliographic references

An application of inverted indices on the UNIX system is the refer [3] preprocessor. This program makes it possible to give an imprecise citation in the text, which is translated into a proper reference. The database of citations searched may be tailored to each system, and individual users may specify their own citation files.

For example, the reference for TEXTSCHAAF was specified by

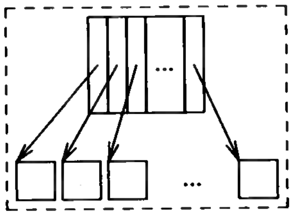
```
. . .
handled by the EQN
.[
tekstschaaf Grune
.]
preprocessor
. . .
```

There are various styles in which the references can appear, for instance, in a footnote, or in a list which can be sorted on various items at the end of the paper (as done here). Also the style of the labels used can be varied. Although refer provides the necessary information to do this, the actual implementation of how the output looks, as usual, is dependent on the macros package actually used.

Handling of graphics material

Typesetters are not really made for this, but it is sometimes possible to turn them into drawing machines. For drawing pictures there exists the PIC [4] preprocessor. It implements a language of the same name for specifying simple figures. The basic objects in PIC are boxes, circles, ellipses, lines, arrows, arcs, spline curves, and text. These may be placed anywhere, at positions specified absolutely or in terms of previous objects. Just like the other preprocessors, PIC only looks at the part between certain macros, `.PS` and `.PE`.

I will not explain all the details of the language in this article but I hope the next example, which has been taken straight out of the user manual, will give a general idea of how pictures are made.



This picture was generated with the following input:

```
     .PS
1    h = .5i
2    dh = .02i
3    dw = .1i
4    [
5        Ptr: [
6            boxht = h; boxwid = dw
7            A: box
8            B: box
9            C: box
10           box wid 2*boxwid "..."
11           D: box
12       ]
13       Block: [
14           boxht = 2*dw; boxwid = 2*dw
15           movewid = 2*dh
16           A: box; move
17           B: box; move
18           C: box; move
19           box invis "..." wid 2*boxwid; move
20           D: box
21       ] with .t at Ptr.s - (0,h/2)
22       arrow from Ptr.A to Block.A.nw
23       arrow from Ptr.B to Block.B.nw
24       arrow from Ptr.C to Block.C.nw
25       arrow from Ptr.D to Block.D.nw
26   ]
27   box dashed ht last [].ht+dw wid last [].wid+dw at last []
     .PE
```

The line numbers don't belong to the input, they have been inserted for the sake of the following explanation only.

The first three lines give values to later used constants. These values are in inches, but could have been scaled to centimetres if the `scale=2.54` had been issued first.

Any sequence of PIC statements may be enclosed in brackets `[...]` to form a block, which can than be treated as a single object, just as the braces are used for grouping with EQN. At line 4 a big block starts, consisting of two other blocks made again with the square brackets and some arrows.

These two blocks are labeled with `Ptr` and `Block`, which allows them to be referenced later by name. The labels refer to the center of the blocks.

The block `Ptr` is made out of several boxes which will get their sizes specified in line 6. At line 10 a box is specified twice the size of the default box (in the current block) with a text consisting of periods in the middle of it.

The second block `Block` is specified in a similar fashion. Note that on line 19 an invisible box is specified, which contains `...` as text.

Line 21 will place the block `Block` with its top (`.t`), at the South of block Ptr (`Ptr.s`), moved over the vector `(0,h/2)` downwards.

Line 22 specifies that an arrow be drawn from the center of box `A` in block `Ptr` to the North-West corner of the box `A` in block `Block`.

At the last line (27), a box is specified with a dashed line style and a height (`ht`) of the height of the last block plus something extra, (`[].ht+dw`). Also, the width is specified and finally this box is placed on top of the previous (conceptual) block, with the centers aligned (`at last []`).

More pictures

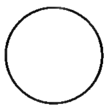
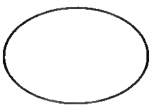
Yet another preprocessor is IDEAL [5]. It implements a language for describing pictures with the same name. It has similar capabilities as PIC.

Some conclusions

The UNIX typesettings tools produce a fine quality of printed material with a relatively small claim on the financial and manpower budget of a medium scale research institute like the CWI.

What really made the system a success is its modularity and flexibility. Separate modules are used for separate functions which implies that if you are not using a certain function, you leave out the module, and so you won't get a surprising effect when you accidentally trigger a function in the system you are not aware of.

An advantage of this separated functionality is that the next complicated looking output can easily be made. Mathematics as well as graphics are used inside the following table.

Text	Equation	Graphics
circle	$x^2 + y^2 = r^2$	
ellipse	$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$	

To have a text processing system as described here around, has its influence on the people dealing with it. Although the quality of the printed material is superior to what was produced before, there tends to be more criticism of it than before. Partly this criticism is justified. There are some defects in the system that need to be changed. The programs currently in use have actually never officially been released for use in a production environment. There is still work going on to make them more stable and to increase the quality of the output.

Also people tend to have difficulties proofreading material to be typeset. A lot of the time there are complaints about the appearance of the text, while the contents should be checked. Some people have strong objections to checking the contents of a text when the output is produced by simulating the typesetter on a cheaper raster style device. I guess that this is the price to pay for "in house" text processing.

Acknowledgements
I would like to thank the editors for commenting on this paper, their suggestions for improving my English and some of the used material.

References

[1] M. E. Lesk, "TBL—A program to format tables", *UNIX Programmer's manual* **2**, section 10, 1979.
[2] B. W. Kernighan, L. L. Cherry, "A System for Typesetting Mathematics", *CACM*, **51**(1975), 151-156.
[3] M. E. Lesk, "Some applications of inverted indexes on the UNIX system", *UNIX Programmer's manual* **2**, section 11, 1979.
[4] Brian W. Kernighan, *PIC — A Crude Graphics Language for Typesetting*, Computing Science Technical Report 85, Murray Hill, NJ, (revised edition), 1982. Also in: *SIGPLAN Symposium on Text Manipulation,* Portland, Oregon, 1981.
[5] Christopher J. van Wyk, "A Graphics Typesetting Language", *SIGPLAN Symposium on Text Manipulation,* Portland, Oregon, 1981.