AT&T Bell Laboratories
Murray Hill, New Jersey 07974

Computing Science Technical Report No. 97

A Typesetter-independent TROFF

Brian W. Kernighan

Revised, March, 1982

A Typesetter-independent TROFF

Brian W. Kernighan

AT&T Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

Although TROFF has been the mainstay of document preparation at Bell Labs for several years, it has heretofore been very dependent on one particular typesetter, the Graphic Systems CAT.

This paper describes conversion of TROFF to deal with a wide class of typesetters.

Some of these typesetters provide many more facilities than the CAT does. Typical extra features include more sizes and fonts, larger alphabets, and the ability to create new characters and to draw graphical objects. The paper describes the enhancements that permit TROFF to take advantage of some of these capabilities as well.

Revised, March, 1982

A Typesetter-independent TROFF

Brian W. Kernighan

AT&T Bell Laboratories
Murray Hill, New Jersey 07974

1. A Bit of History

> ''I will be speaking today about work in progress, instead of completed research; this was not my original intention when I chose the subject of this lecture, but the fact is I couldn't get my computer programs working in time.''
>
> Donald E. Knuth[1]

The TROFF text formatter[2] was originally written by the late Joe Ossanna in about 1973, in assembly language for the PDP-11. (NROFF, which drives terminals instead of a typesetter, is essentially identical to TROFF; we will use ''TROFF'' as a generic term henceforth.) It was rewritten in C around 1975, and underwent slow but steady evolution until Ossanna's death late in 1977.

In spite of some obvious deficiencies — a rebarbative input syntax, mysterious and undocumented properties in some areas, and a voracious appetite for computer resources (especially when used with macro packages and preprocessors like EQN and TBL) — TROFF has been the basis of document preparation at Bell Labs for some years, and is likely to remain so for years to come.

Early in 1979, the Computing Science Research Center decided to acquire a new typesetter, primarily because of our interests in typesetting graphics. At the same time, the Murray Hill Computer Center began to investigate the possibility of replacing their family of aging CAT's with a new, high-performance typesetter, simply to keep up with their rapidly expanding load.

My first thought (a thought shared by many others) was that this would be a glorious opportunity to replace TROFF with a new formatting language: better designed, easier to work with, and of course much faster. This remains a desirable goal, but, after quite a bit of thought spread over several years, I am still not really much closer to a better design, let alone an implementation. Furthermore, a great deal of software depends on TROFF — the preprocessors, the macro packages, and of course all of their documentation and our accumulated expertise. Tossing this aside is not something to be done lightly.

Accordingly, in the spring of 1979, I set about to modify TROFF so that it would run henceforth without change on a variety of typesetters. The ground rule was that TROFF should retain its current specifications, so that existing software like EQN, TBL and the macro packages would continue to work with it.

Since much of the rest of this paper is encrusted with details that could appeal only to maintainers or masochists, I will give here a brief summary of what has been done. Non-specialists can stop reading at the end of the section.

TROFF is highly dependent on the Graphic Systems CAT typesetter, not just in details of code but also in many aspects of its design. The language design issues have been largely ignored (few are truly fundamental), while the code has been modified so that dependencies are either eliminated or at least parameterized.

TROFF originally had parameters of the typesetter compiled into the code, often in non-obvious ways. The new version reads a parameter file each time it is invoked, to set values for machine resolution, legal sizes, fonts and characters, character widths and the like.

TROFF output used to be binary device codes specific to the CAT and arcane beyond description. The output of the new version is ASCII characters in a simple and (I hope) universal language that describes where each character is to be placed and in what size and font. A post-processor must be written for each typesetter to convert this typesetter-independent language into specific codes for that typesetter. Post-processors currently exist for the CAT, the Mergenthaler Linotron 202, the Autologic APS-5, the Tektronix 4014 terminal, The Imagen Canon laser printer, Versatec printers, and a bit-map terminal. New ones can generally be written in less than a day; they share much of their code with previous ones.

The new output language contains information that is not readily identifiable in the older output. Most notably, the beginning of each page and line is marked, so post-processors can do device-specific optimizations such as sorting the data vertically or printing it boustrophedonically, independently of TROFF.

Since actual output is done by a post-processor, not TROFF, new capabilities for graphics have been easy to add. TROFF now recognizes commands for drawing diagonal lines, circles, ellipses, circular arcs, and quadratic B-splines; these are used in the PIC[3] and IDEAL[4] languages.

A number of limitations have been eased or eliminated. A document may have an arbitrary number of fonts on any page (if the output device permits it, of course). Fonts may be accessed merely by naming them; ''mounting'' is no longer necessary. Character height and s\a*nt* may be set independently of width.

The new TROFF is about 1000 bytes larger in instruction space and 13000 bytes larger in data space (thus guaranteeing that it will not run on PDP-11/40 style machines). It runs about as fast as the original version, though a simple improvement that I made could be retrofitted into the earlier version to keep it about 20% faster. The post-processors are not included in these time comparisons; they typically take 10-20% of the TROFF time.

2. Typesetter Dependencies

TROFF turns out to be surprisingly dependent on the Graphic Systems CAT, not just in the code but in its design.

Some of the design dependencies are pretty obvious. For example, the CAT provides four fonts (of 102 characters each) and 15 sizes. The specific sizes are wired into the syntax of the language: since the CAT has no sizes larger than 36 points, `\s46` can be uniquely decoded as a 4-point 6, while `\s36` is simply a switch into size 36.

TROFF makes the assumption that there are four fonts, three of which are physically isomorphic (that is, the same characters appear in the same position in each) and one ''special'' font that is logically a part of each of the others. The reserved font name `S` finds its way into several commands and receives special treatment in a variety of contexts.

Some commands have the basic resolution of the CAT wired into their definition; for example, the units of the `.ss` command (which sets the size of the inter-word spacing) are 36th's of an em, because the CAT typesetter itself works in those units.

Some command line options reflect idiosyncrasies of the CAT. For example, the option `-p` requests that the output all be printed in one size; since the CAT is excruciatingly slow at changing point sizes, this prints an approximation to final output comparatively quickly. The `-g` option causes font information to appear in the output file for the benefit of the operations staff at the Murray Hill Computer Center.

At the same time, there are myriad places where the characteristics of the CAT are an integral part of the code for TROFF. Some of these are quite evident; others are subtle indeed.

The most obvious instance is the internal encoding of a character. Within TROFF, objects are passed around as 16 bit quantities. There are two fundamental objects — printable characters and motions. An object looks like this:

1	4	2	1	8
m	s	f	z	c

If the `m` bit is a 1, the object represents a motion; if it is zero, the object is something to be printed.

In that case, `s` is the size (actually an index into a table of legal sizes), `f` is the font, `z` is the zero-motion bit (i.e., no space after printing), and `c` is the character. If the high order bit of `c` is set, the character is to be looked up in a table of special names (for example, `0200` is the hyphen `\(hy`); otherwise it is ASCII. Furthermore, if `c` is less than octal 40 or greater than octal 370, the character is actually some encoded control function or very special character such as `\e` or `\{`.

Clearly the tight packing makes it utterly impossible to add another size or font — there are no bits left over. It also implies limits on the number of characters in a font and on the number of special names (names of the form `\(xx`).

Motions are encoded as

1	1	1	13
m	v	n	mag

`m` is the ''motion bit'', which is 1 for a motion, `v` is 1 for a vertical motion, and `n` is 1 for a negative motion. The remaining 13 bits give the magnitude. Since there are only 13 bits, the maximum amount of motion is 8191 machine units. With the CAT's resolution of 432 units per inch, this is a generous 19 inches. But for the Linotron 202 (resolution 972/inch), it is only 8.5 inches.

Within the code, certain character values are used in tests and assignments without identification. For instance the octal value `0200` is used (without identification) as a hyphen. But of course the mask `0200` also occurs many times. As might be imagined, it takes some study to determine whether any particular `0200` is a hyphen or a mask.

The basic resolution of the CAT is 432 units per inch horizontally and 144 vertically. Character widths are given as the number of units at size 6 points. There are 72 points in an inch. Thus the program contains as magic numbers every factor of 432, and ±1 from each factor as well.

Finally, TROFF is simply a big program (at least by my standards) — about 7000 lines of virtually uncommented C. (I am indebted to Lorinda Cherry for a new version of the C beautifier that made the code legible, if not comprehensible.) It was implemented before the recent additions to C, so there are no `typedef`'s to distinguish among the various kinds of integers, relatively few macros with arguments, no internal `static` variables, and a startling number of global variables with two-character names.

None of these remarks should be taken as denigrating Ossanna's accomplishment with TROFF. It has proven a remarkably robust tool, taking unbelievable abuse from a variety of preprocessors and being forced into uses that were never conceived of in the original design, all with considerable grace under fire.

3. Modifying TROFF

The first step was to widen the 16-bit internal representation of a character to 32 bits, to accommodate more sizes and fonts. The current representation is

1	7	8	1	15
z	s	f	m	c

If `m` is 1, bits 16 and 17 are `v` and `n`. Access to this representation is entirely through macros; for example, a macro called `cbits` fetches the character bits, another called `setsfbits` sets the size and font bits, and so on.

This stage took several weeks of meticulous checking, since it was necessary to examine every integer constant, variable and function in the program to decide whether it was being used to store an internal character. These are now all identified and `typedef`'d for future reference.

Widening 16 bits to 32 turns out to be quite costly on the PDP-11, since the program must process `long` integers instead of short ones. The result is approximately a 25% increase in program size and perhaps 25% increase in run time. Furthermore the temporary file in which TROFF keeps its macro and string definitions doubles in size (to 256k bytes).

Note that at this stage the program has not been changed in any fundamental way — it still generates CAT output, so a bit-for-bit regression test against the original version can be performed after each change. This proved to be very important for maintaining sanity in both program and programmer.

4. Dynamic Machine Parameters

The next step was to find all the the numbers in the program that depend on the CAT and replace them by variables. This also contributed marginally to slower execution, since many values and expressions that were constants now became variables.

With parameters identified as such, the next step was to make it possible to load a description of the typesetter each time TROFF is run, rather than creating a compiled version for each typesetter. Accordingly, a set of description files was designed and created for each typesetter and each font. The description really comes in two pieces — a table describing parameters of the machine, and a table of widths for each font. To illustrate, here is the parameter file for the CAT:

```
# Graphic Systems CAT-4
res 432
hor 1
vert 3
unitwidth 6
sizes 6 7 8 9 10 11 12 14 16 18 20 22 24 28 36 0
fonts 4 R I B S
charset
\|  \^  \-  \_
hy  bu  sq  em  ru  14  12  34  mi  fi  fl  ff  Fi  Fl  de  dg  sc  fm  aa  ga
ul  sl  *a  *b  *g  *d  *e  *z  *y  *h  *i  *k  *l  *m  *n  *c  *o  *p  *r  *s
*t  *u  *f  *x  *q  *w  *A  *B  *G  *D  *E  *Z  *Y  *H  *I  *K  *L  *M  *N  *C
*O  *P  *R  *S  *T  *U  *F  *X  *Q  *W  sr  ts  rn  >=  <=  ==  ~=  ap  !=  ->
<-  ua  da  eq  mu  di  +-  cu  ca  sb  sp  ib  ip  if  pd  gr  no  is  pt  es
mo  pl  rg  co  br  ct  dd  rh  lh  **  bs  or  ci  lt  lb  rt  rb  lk  rk  bv
lf  rf  lc  rc
```

A # introduces a comment. `res` is the machine resolution in units per inch. `hor` and `vert` give the minimum number of machine units that it is possible to move in the corresponding direction. `unitwidth` specifies the point size at which the character widths map directly into machine units. `sizes` lists the set of legal point sizes, terminated by a zero. `fonts` lists the default set of fonts (which can be overridden by subsequent `.fp` commands). `charset` introduces the set of legitimate special names (names of the form `\(xx`, including some special cases like `\|`).

For comparison, here is the description file for the Mergenthaler Linotron 202. The 202 has many more sizes and fonts than the CAT, and quite a few more characters as well. (The 202 actually permits nearly 250 sizes; no use has yet been found for most of them.)

```
# Mergenthaler Linotron 202
fonts 10 R I B BI H HB HK PO CH S
sizes 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
   21 22 23 24 25 26 27 28 29 30 32 34 36 38 40
   45 50 55 60 66 72 78 84 90 96 102 108 0
res 972
hor 1
vert 2
unitwidth 4
paperwidth 7500
charset
\|  \^  \-  \_
**  *C  *D  *F  *G  *H  *L  *P  *Q  *S  *W  *a  *b  *c  *d  *e  *f  *g  *h
*i  *k  *l  *m  *n  *p  *q  *r  *s  *t  *w  *x  *y  *z  +-  ->  <-  <=
==  >=  L.  Sl  al  aa  ap  b0  br  bs  bu  bv  ca
cd  ci  co  ct  cu  dd  de  dg  di  em  eq  es  fe  fm  ga  gr  hy
ib  if  ip  is  l.  lh  ma  mi  mo  mu  no  or  pd  pl  pp  pt  rg  rh  ru
sb  sc  sl  sp  sq  sr  tm  tp  ts  ~=  ~~  ul  rn  en
lf  rf  lc  rc  lt  rt  lb  rb  lk  rk  !=  ua  da  12  fa  te  ma  fe
hc  ..  ob  bx  *o  *u  b9  14  34  ss  vr
```

`paperwidth` specifies the maximum width of paper in units, overriding the default 7¾".

In addition to this description file (one per typesetter), there is one file per font that lists properties of that font. Here is part of the description of Times Roman for the CAT:

```
# Times Roman for CAT-4
name R
internalname 1
ligatures ff fi fl ffi ffl 0
charset
\|      6       0       0
\^      3       0       0
a       17      0       025
b       20      2       012
c       16      0       027
d       20      2       011
e       18      0       031
f       13      2       014
...
!       12      2       0145
&       28      2       050
(       16      2       0132
)       16      2       0133
*       16      0       0122
+       36      0       0143
,       12      0       047
hy      13      0       040     hyphen
-       "       =hy
\-      36      0       0123
.       10      0       044
de      15      0       0136    degree
dg      20      0       0137    dagger
fm      8       0       0150
rg      20      0       0141
co      20      0       0153
ct      19      0       0127
...
```

The `name` is the external TROFF name, one or two characters, as used in `.ft` and `\f` commands. The internal name is not used by TROFF itself, but is necessary for the postprocessors. If the font has ligatures, they are listed. It is possible to set the width of the space for each font separately with the `spacewidth` command, not illustrated here. There is also a keyword `special` to indicate that the font is a ''special'' font — one that is to be searched if the regular font does not contain the character requested. There is no limit on the number of special fonts, but they should be listed last in the `fonts` part of the description file.

The four columns of data are the character name, its width, its ascender/descender information (1 → descender, 2 → ascender, 3 → both), and the actual typesetter code required to print it. When the point size is `unitwidth`, the width is the character width in machine units. If the name is a single character, it is taken simply as a normal ASCII character. Two or more characters indicate a name of the form `\(xx`; there are a handful of historical exceptions like `\-`. A width of """" `"  "  "` indicates that the character is a synonym for the immediately preceding character, as in `-` and `hy` above. Comments may follow the four data fields.

There are significant advantages to having the font and typesetter descriptions merely text files that can be edited easily, but it is too time-consuming to load all this ASCII information each time TROFF is invoked. Thus a separate program called `makedev` is used to compile it into a binary file that can be read by TROFF in a single read. When TROFF is invoked, an argument of the form `-Txxx` tells it to load the description file for typesetter `xxx` from a standard directory.

Descriptions for the default fonts are compiled into the description file; if a new font is requested by a `.fp` command, its description data replaces the original values. The `.ft` command has been modified so that if the requested font is not currently ''mounted'', its description data will be placed in the hidden font position 0. Since there is only one such position, each new non-standard font overlays the previous one. This mechanism is intended to make occasional use of non-standard fonts easy; the `.fp` mechanism remains necessary for other purposes.

The notion of special fonts has been generalized somewhat — rather than a single special font, there can be several. The algorithm currently used is to search for each character on the current font; if it is not found there, then the special fonts are searched in order (as given in the description file). Other than that, the treatment of special fonts is essentially unchanged.

At this stage in the modifications to TROFF, all internal arithmetic is done in terms of the resolution of the specified typesetter. The character set and character widths are defined by the values loaded from the font description. Indeed, the mapping from character name to what gets printed is determined by this table — fonts are no longer all isomorphic.

Although it is certainly convenient for testing during program development, it is not clear that loading the font information for the typesetter each time TROFF is invoked is the right way to operate in an environment that supports only a single kind of typesetter. In a production mode it might be desirable to compile in the default information for the standard typesetter.

5. Output Language

The final step is to modify the output of TROFF so that it is typesetter-independent. The new version of TROFF produces output that is not intended to go directly to a typesetter. Rather, it is more or less independent of any typesetter, except that the numbers in it have been computed on the basis of the resolution specified in the description file for the intended typesetter.

The output language is simple:

```
sn       size in points
fn       font as number from 1 to n
cx       ASCII character x
Cxy      character \(xy; terminate xy by white space
Hn       go to absolute horizontal position n. (n > 0)
Vn       go to absolute vertical position n (down is positive)
hn       go n units horizontally (to the right; n > 0)
vn       go n units vertically (down; n > 0)
nnc      move right nn, then print c (nn is exactly 2 digits!)
nb a     end of line (information only -- no action needed)
         b = space before line, a = after
w        paddable word space (information only -- no action needed)
pn       new page n begins -- set V to 0
x ...\n  device control functions
D ...\n  drawing functions (graphics)
```

Encoding small horizontal motions followed by a character as *nnc* shrinks the output file size by about 35% and run-time by about 15%.

The device control and graphics commands are intended as open-ended families, to be expanded as needed.

```
x i      init
x T s    name of typesetter is s
x r n h v  resolution is n/inch, h = minimum horizontal motion, v = min vert
x p      pause (can restart)
x s      stop -- done forever
x t      generate trailer
x f n s  font position n contains font s
x H n    set character height to n
x S n    set slant to n
```

Subcommands like "`i`" are often spelled out like "`init`".

The drawing functions are

```
Dl dh dv        draw line from current position by dh dv
Dc d            draw circle of diameter d with left side here
De d1 d2        draw ellipse of diameters d1 d2
Da dh1 dv1 dh2 dv2
                draw arc from current position to dh1+dh2 dv1+dv2,
                center at dh1 dv1 from current position
D~ dh1 dv1 dh2 dv2 ...
                draw B-spline from current position to dh1 dv1,
                then to dh2 dv2, then to ...
```

In all of these, *dh dv* is an increment on the current horizontal and vertical position, with down and right positive.

Blanks, tabs and newlines may occur as separators in the input, and are mandatory to separate constructions that would otherwise be confused.

To illustrate, the following is the output from the input

```
hello
.br
.ps 20
.ft H
goodbye
```

using `-Tcat`:

```
x T cat
x res 432 1 3
x init
x font 1 R
x font 2 I
x font 3 B
x font 4 S
V0
p1
s10
f1
H416
V72
ch
35e30l17l17on72 0
x font 0 H
f1
H416
f0
s20
V144
cg
70o70o70d73b73y67en72 0
x trailer
V4752
x stop
```

Here is the same output from the 202. Notice how the numbers are more than twice as big, reflecting the higher resolution of the 202. Also notice that the font `H` is a standard font on the 202 so no special loading is needed to switch to it.

```
x T 202
x res 972 1 2
x init
x font 1 R
x font 2 I
x font 3 B
x font 4 BI
x font 5 H
x font 6 HB
x font 7 HK
x font 8 PO
x font 9 CH
x font 10 S
V0
p1
s10
f1
H936
V156
ch
73e63l35l35on156 0
H936
f5
s20
V312
cg
h150co
h150co
h150cd
h150cb
h150cy
h135ce
n156 0
x trailer
V10692
x stop
```

The output is guaranteed to be ASCII, and thus amenable to processing by all the normal Unix tools. The language is simple — it is straightforward to write a prototype driver for a particular typesetter, especially when one can steal an existing one as a model, although it may take some effort to make one of production quality. This form also demystifies TROFF output and makes it possible for anyone to write other programs to process or generate it.

On the other hand, it is about twice as voluminous as the output intended for the CAT. Cleverness (or sacrificing ASCII-ness) could bring it down to not much more than one byte per character printed, but so far I have not felt this to be very important.

There is not really much to say about the post-processors. D202 drives the 202, DCAT drives the CAT (although not very efficiently — the vital boustrophedon and size-sorting features are not there), DAPS drives the APS-5, DCAN drives the Canon, and TC (notice the parallel name structure) drives the Tektronix 4014. There are also rough and ready drivers for other display terminals.

Since it is easy to identify page and line boundaries, some drivers are able to offer useful features that were not feasible with the older TROFF. For example, TC permits the user to ask for specific pages by number, and to skip back and forth in the document. D202 displays the output page number on the operator's control panel and will also stop at specified pages if requested. An experimental postprocessor called DSORT sorts the intermediate language by vertical position, to minimize vertical motion; it is intended for printing complicated PIC diagrams and for circuit diagrams produced by PLTROFF. (PLTROFF converts the standard Unix plot language into TROFF commands.) All of the post-processors can scan their input relatively quickly to print only selected pages.

6. New Things

During the process of making these changes to TROFF, I have made some changes to the program that are visible to users (thus violating the avowed goal of keeping it compatible).

Some changes are simply the easing of restrictions. For example, there is no longer any limit on the number of fonts that can occur on a page (if one's typesetter is up to it), nor does the `.fp` command that requests ''mounting'' a new font have to occur only at the beginning. Furthermore, a font may be accessed merely by naming it — the command

```
\f(ARBell Laboratories\fP
```

will produce

Bell Laboratories

(The font AR is Avant-Garde Book. Obviously two-letter font names will have to go.)

Character sets can be sensibly defined since they no longer have to all be isomorphic. Thus, for example, on the 202 the ''printout'' font

```
This one
```

includes a complete ASCII alphabet. All of the characters have the same width, even the word space (defined by the `spacewidth` attribute in the font description file). Thus this font is suitable for printing program listings without any special precautions. On the other hand, some fonts are rather small, containing only a handful of characters.

Larger and smaller point sizes are available, although there are still some limitations. For example, `\s72` is still parsed as a 7 point 2, since changing this would affect at least EQN and probably other programs as well. (Sadly, EQN had to be changed anyway — it also accepts an option of the form `-Txxx`.) But `.ps 72` works fine: each input size is mapped into the closest legal size for the current typesetter. Fractional point sizes are not allowed and may never be.

Character height and slant may be set independently of character width, for those typesetters that allow these operations. `\H'n'` sets the character height to `n` points, `\H'±n'` sets it to `±n` from the current point size, and `\H'0'` restores it to normal height. `\S'n'` sets the slant to `n` degrees positive or negative; if `n` is zero, slanting is turned off.

Some obsolete commands have been eliminated (e.g., `.fz`, `.li`; the command line options for constant-size printing and suppressing boustrophedon; all code related to GCOS).

A new command `.sy` has been added to permit calling another program from within TROFF:

`.sy` *command line*

causes *command line* to be executed. The output is *not* automatically collected anywhere, but the new number register `\n($$` (the process id of the TROFF process) can be used to create unique file names to be picked up with subsequent `.so` commands. The built-in string `\*(.T` contains the name of the current typesetter obtained from the `-T` argument or its default.

7. Graphics Commands

The most significant new facility is the ability to draw simple graphical objects — diagonal lines, circles, ellipses, arcs, and splines — in TROFF.

The new graphical commands are

```
\D'l dh dv'     draw line from current position by dh, dv
\D'c d'         draw circle of diameter d with left side at current position
\D'e d1 d2'     draw ellipse of diameters d1 d2
\D'a dh1 dv1 dh2 dv2'
                draw arc from current position to dh1+dh2 dv1+dv2,
                with center at dh1 dv1 from current position
\D'~ dh1 dv1 dh2 dv2 ...'
                draw B-spline from current position by dh1 dv1
                then by dh2 dv2, then by dh2, dv2, then ...
```

For example, the input `\D'e0.2i 0.1i'` draws the ellipse, and the input `\D'l.2i -.1i'\D'l.1i .1i'` will draw the line. The position after a graphical object has been drawn is at its ''end'', where for circles and ellipses, the end is at the right side. As with other commands, default units are ems horizontally and line spaces vertically.

Realistically, these commands are not intended for direct use, but for preprocessors like PIC and IDEAL.

The output generated by these commands is shown in the discussion of the TROFF output language in an earlier section. Rather than drawing the shape in TROFF, a command with the proper parameters is passed through to the device post-processor, which does whatever it can with the request.

8. Loose Ends

This version of TROFF has been in use since September of 1979. Most of our experience with it has been on the 202 and Tektronix scopes, but the CAT and APS-5 drivers have been exercised to some degree.

As mentioned, there are some obvious things that could be improved or that remain to be added. For instance, TROFF's bracket-building function `\b` really ought to be implemented by postprocessor, so that typesetters like the 202 can draw a character like { directly instead of synthesizing it from pieces. It also appears to be necessary to add ''modes'' to permit graphical objects to be dotted, dashed, etc.

Efficiency is always a problem, and is likely to be so forever, especially with the proliferation of preprocessors generating ever more complicated input. Small improvements (perhaps 10 percent) can be had from artifices like better table searching. Placing the temporary file upon which TROFF stores macro definitions in memory is good for another 10-20 percent on machines like the VAX that have enough memory. But no order-of-magnitude speedup is likely without a gross revision of the basic design.

It is clear that TROFF ought to be replaced by something better, but, as I said above, it is far from clear how to do the job a lot better. So for better or worse, TROFF is likely to be with us for a long time.

Acknowledgements

I am deeply indebted to Ken Thompson and Joe Condon, without whose efforts in taming the Mergenthaler Linotron 202 the TROFF modifications discussed here would be irrelevant. Thompson also provided most of the character-generating software. I am also grateful to Chris Van Wyk for the line and circle drawing algorithms and to Theo Pavlidis for the spline algorithm used by all the postprocessors.

1. Donald E. Knuth, ''Mathematical typography,'' *Bulletin (New Series) of the American Mathematical Society* **1**(2) (1979).

2. J. F. Ossanna, ''NROFF/TROFF User's Manual,'' Comp. Sci. Tech. Rep. 54, Bell Laboratories, Murray Hill, NJ (October 1976).

3. Brian W. Kernighan, ''PIC — A Crude Graphics Language for Typesetting,'' Comp. Sci. Tech. Rep. 85, Bell Laboratories, Murray Hill, NJ. Also in SIGPLAN Symposium on Text

Manipulation, Portland, June 1981.

4. Christopher J. Van Wyk, ''A Graphics Typesetting Language,'' SIGPLAN Symposium on Text Manipulation, Portland (June, 1981).

Appendix: Summary of Language Changes

This appendix enumerates the changes to the TROFF language since the last printing of the manual.

Command line arguments

The argument `-Txxx` loads parameters and character definitions for typesetter `xxx`, which at the moment is typically one of `202`, `aps` or `cat`.

`-Fxxx` causes font information to be loaded from directory `xxx` instead of the default `/usr/lib/font/dev202`.

Graphics commands

As described in section 7.

Other new commands

`.sy` *commandline* executes the command, then returns. Output is not captured anyplace.

`.cf` *file* copies *file* into the TROFF output file at this point, uninterpreted. Havoc ensues unless the motions in the file restore current horizontal and vertical position. This command hasn't been used much, and is probably a bad idea anyway.

`.pi` *program* (pipe the output into *program*) now works in TROFF as well as NROFF, since it makes somewhat more sense to allow it.

`\H'n'` sets the character height to `n` points. A height of the form $\pm n$ is an increment on the current point size; a height of zero restores the height to the point size.

`\S'n'` sets the slant to `n` degrees. *n* may be negative.

The number register `$$` contains the process id of the TROFF process.

The string `.T` contains the name of the current typesetter (e.g., `202`, `aps`, `cat`).

The `.fp` command accepts a third argument that causes the data for the font to be loaded from that directory. This has been added as a first step to allowing dynamic character definitions.

The `.ft` command causes the named font to be loaded on font position 0 (which is in all other ways inaccessible) if the font exists and is not currently mounted by default or by a `.fp` command. The font must be still or again in position 0 when the line is printed.

Transparent mode (`\!`) has been fixed so that transparent output actually appears in the output; thus special commands can be passed through to postprocessors by witchcraft like

```
.if "\*(.T"202" \!x ...
```

(If this makes no sense to you, you shouldn't be using it anyway.)

Deletions

The `.fz` and `.li` commands are no more. The `-p`, `-g` and `+n` command line arguments have also been eliminated, as has the `hp` number register.