

**1989 Conference Proceedings**

TEX Users Group  
Tenth Annual Meeting  
Stanford, August 20–23, 1989

# **TUGBOAT**

COMMUNICATIONS OF THE TEX USERS GROUP

TUGBOAT EDITOR BARBARA BEETON

PROCEEDINGS EDITOR CHRISTINA THIELE

VOLUME 10, NUMBER 4

PROVIDENCE

DECEMBER 1989

RHODE ISLAND

U.S.A.

## Production Notes

An interesting range of programs, fonts and output devices were used to produce this issue of *TUGboat*. It is hoped that the information provided here is complete, and that it gives the reader a good sampling of what can be done with *TeX*, non-*TeX* files, and printers, both laser and typesetter. A number of figures have been reduced, in order to fit on the page. Phototypesetting services for most of the articles produced by an APS-μ5 were provided by Computer Composition Corporation.

### ■ Apple LaserWriter II:

- Henderson, *Introduction to METAFONT*.

### ■ Apple LaserWriter II NT:

- Benson et al, *Inserts in a Multiple-Column Format*.
- Hosek, *Design of Oriental Characters with METAFONT*.

### ■ Apple LaserWriter Plus:

- Greene, *TEXcreation — Playing Games with *TeX*'s Mind*.

### ■ APS-μ5:

- Batzinger, *Thai Languages and METAFONT*.
- Billawala, *Opening Pandora's Box*, using her newly designed Pandora fonts. The figures were reproduced from her book *Metamarks* (1989).
- Cheswick, *A Permuted Index for *TeX* and LATEX*.
- Clark, *Olde Worlde *TeX**.
- Conrad, *Fine Typesetting with *TeX* Using Native Autologic Fonts*. Samples 1–4, which compare laser proof copies to typeset copies were done on an Apple LaserWriter Plus PostScript device and an APS-μ5, respectively.
- Díaz, **TeX* in México*. Figures were produced using a printer with a Canon SX engine.
- Doob, *Of the Computer Scientist, by the Computer Scientist, for the Computer Scientist*.
- Hamilton, *Mastering *TeX* with Templates*.
- Haskell et al, **TeX* for 30,000*.
- Hoenig, *Fractal Images with *TeX**. Figures were produced on a Hewlett-Packard LaserJet Series II.
- Hoover, *Using WordPerfect 5.0 to Create *TeX* and LATEX Documents*.
- Knuth, *The Errors of *TeX**.
- Kubek, **TeX* for the Word Processing Operator*.
- Latterner and Woolf, **TeX* at Mathematical Reviews*.
- McClure, **TeX* Macros for COBOL Syntax Diagrams*.
- Mittelbach and Schöpf, *With LATEX into the Nineties*.
- Rattey-Hicks, **TeX* and Its Versatility in Office Production*. Figures in Appendix B were produced on a Hewlett-Packard LaserJet Series II.
- Younigen et al, *Migration from Computer Modern Fonts to Times Fonts*.

### ■ HP LaserJet Series II with QMS JetScript board:

- Olejniczak-Burkert, *texpic — Design and Implementation of a Picture Graphics Language in *TeX* à la pic*.

### ■ IMAGEN ImageStation:

- Abbott, *The UKTEX Archive at the University of Aston*.

### ■ Linotype L200P PostScript (1270dpi):

- Hobby, *A METAFONT-like System with PostScript Output*.

### ■ Linotronic 300:

- Riley and Halverson, *Creating an Efficient and Workable PC Interface for *TeX**. Some font substitution was done: *cmbxti10*, *logo8*, and *logo10* were replaced by Times Roman Bold Italic, Avant Garde Bold 7pt, and Avant Garde Bold 9pt, respectively.

### ■ QMS810 PS:

- Renfrow, *Methodologies for Preparing and Integrating PostScript Graphics*.

### ■ Variyper VT600:

- Pind, *Lexicography in Iceland*. The dictionary sample sheet was done on a Linotronic 300.
- Sydoriak, *LATEX Memos and Letters*.

## Other Conference Proceedings

*Proceedings of the First European Conference on *TeX* for Scientific Documentation*. Dario Lucarella, ed. Reading, Mass.: Addison-Wesley, 1985. [16–17 May 1985, Como, Italy.]

*Proceedings of the Second European Conference on *TeX* for Scientific Documentation*. Jacques Désarménien, ed. Berlin: Springer-Verlag, 1986. [19–21 June 1986, Strasbourg, France.]

*Conference Proceedings: *TeX* Users Group Eighth Annual Meeting*. Dean Guenther, ed. *TEXniques* No. 5. Providence, Rhode Island: *TeX* Users Group, 1988. [24–26 August 1987, University of Washington, Seattle, Washington.]

*Conference Proceedings: *TeX* Users Group Ninth Annual Meeting*. Christina Thiele, ed. *TEXniques* No. 7. Providence, Rhode Island: *TeX* Users Group, 1988. [22–24 August 1988, McGill University, Montréal, Canada.]

*TEX88 Conference Proceedings*. Malcolm Clark, ed. Chichester, England: Ellis Horwood, 1989. [18–20 July 1988, Exeter University, Exeter, England.]

## Trademark Information

*TeX*, *AMS-TeX* are trademarks of the American Mathematical Society. *METAFONT* is a trademark of Addison-Wesley Inc. All other brand and product names are registered trademarks or trademarks of their respective holders.

## **Editor's Introduction**

Welcome to the *Proceedings* issue of *TUGboat*, vol. 10, no. 4—*TUGboat* is now a quarterly. Formerly published in the *TEXniques* series, the *Proceedings* will now be available to the entire TUG community, part of your regular membership.

This year's Tenth Anniversary Meeting at Stanford was a smashing success, with the highest attendance of any meeting in TUG's ten-year history: some 235 individuals attended, representing countries and interests from around the world (a List of Participants is included after the articles).

What better way to remind us how much has happened over the past decade than to have the Keynote Address delivered by Prof. Donald E. Knuth himself. How ironic, in the midst of celebrating our prowess with *TEX*, for the designer to describe, with more than the occasional poke at himself, the various categories of errors, as he and *TEX* gradually got onto speaking terms with one another. It was a delightful morning we spent, listening to a quiet analysis of where and how things went wrong in the beginning.

With 30 presentations (up from 24 last year), there was much to choose from, with several papers devoted to *LATeX*, and a healthy dose of *METAfont* too. Previous years have seen interest focused rather heavily on *TEX* alone; to see so many *LATeX* and *METAfont* papers was indeed a pleasure.

Recent meetings have shown a move away from the notion of *TEX* as an all-purpose program, towards the more pragmatic idea that *TEX* has the flexibility to integrate with and complement other programs, each one used for what it does best. This year saw a continuation of this, in discussions on graphics programs, and pre-processors. Extending *TEX*'s capabilities will probably be the main direction of effort in our second decade.

Another aspect to *TEX* which has been emerging over the past several meetings is that of the "new user" of *TEX* these days. It seems to me that here too we are seeing something new for the next decade: *TEX* has gone from the playground into the real world. And in the real world of applications for *TEX* and its integration into larger processes, the users are not always going to be computer literate, or even interested in becoming so. They have a job to do, and must use *TEX* to do it. Support staff, office personnel, one's colleagues and the like, these are the people who want to use *TEX* but also want to be spared the theory (a sacrilegious thought for some). Training and motivating them, helping them

along, that is an area which should concern those of us who "know how to use *TEX*".

The amount of teaching material available is still quite small, a fact which seriously hampers good use of all three programs. However, scattered throughout these articles are references to many documents available, including several from Europe, written in a variety of languages.

The internationalization of *TEX*—and our increasing awareness of this fact—was another noteworthy element at this year's meeting. Not simply because of the presence of several European coordinators and presenters, but also in the fact that we learned of the many user groups and annual meetings, with quite an extensive collection of documents on *TEX* and *LATeX*.

There is a great deal of experience, whether hard won or achieved accidentally, described in these pages; there is food for thought for both the *TEXpert* and the neophyte; and macros for everyone's needs. On this 10th Anniversary Meeting of the *TEX* Users Group, they are a celebration of the many achievements and results due to the work begun by Prof. Knuth a decade ago (more or less).

And so, the overall impression I have of the 1989 Annual Meeting of the *TEX* Users Group is that we did indeed celebrate a decade of accomplishments, but we also were in a way rejuvenated, with ideas and suggestions and examples of where the new challenges are not just being found, but are being attacked, and solutions being proposed. *TEX* is here to stay—now we must look towards extending it, stretching it, integrating it into the larger picture. I truly look forward to the second decade, and wonder what we will have to see at our 20th Anniversary Meeting.

And now to the articles.

Christina Thiele  
Carleton University

**1989 Program Committee.** The 1989 Annual Meeting was organised by the following people, under the most able guidance of Dean Guenther, Program Coordinator.

- Dean Guenther (Washington State University, Pullman, Washington)
- Hope Hamilton (National Center for Atmospheric Research, Boulder, Colorado)
- Doug Henderson (Blue Sky Research, Portland, Oregon)
- Christina Thiele (Carleton University, Ottawa, Canada)



TEX USERS GROUP  
STANFORD UNIVERSITY  
STANFORD, CALIFORNIA  
AUGUST 20-23, 1989

# Introduction to METAFONT

DOUG HENDERSON

Blue Sky Research  
534 SW Third Avenue  
Portland, Oregon 97204

## ABSTRACT

The purpose of this “Introduction to METAFONT” talk is to give a small amount of historical background on what METAFONT is, to introduce a few key concepts and METAFONT commands, and to go over a few more complicated examples and commands. It is beyond the scope of this twenty-minute talk to explain how METAFONT works in detail, but I hope you find METAFONT as interesting as I do, and I hope that I do not verbally wander off on you — at least, not too far.

### 1. What is METAFONT?

METAFONT is a very powerful tool for producing fonts. Created in 1981 by Prof. Donald E. Knuth, it has undergone quite a few changes to bring it to its current state. Prof. Knuth needed to create the TeX typesetting program/language to be able to create the beautiful math which he was familiar with in his *Art of Computer Programming* books, and METAFONT is the companion program which creates typefaces for TeX to use. TeX can be labeled a markup language, since one embeds control sequences in a document, and TeX processes the file accordingly. METAFONT is similar in that it too has an extremely powerful language, but with METAFONT, the user specifies commands which direct METAFONT to place strokes of an electronic pen on a “digital canvas”. We will be exploring some of the basic METAFONT commands to get a better understanding of these concepts.

### 2. Coordinate System

METAFONT works in the cartesian coordinate system. This means that positive coordinates are found above and to the right of the 0,0 point, which is known as the origin. Fig. 1 shows a representation of METAFONT’s cartesian coordinate system. Most METAFONT characters are drawn in the top right quadrant (A, where  $x$  and  $y$  are positive), but characters such as a lowercase g, j, or y have descenders, which extend below the baseline. B represents the baseline of a TeX Font Metrics or `tfm` box. For TeX to be able to use characters that METAFONT creates, it needs to know certain things, such as how wide, high and deep characters are, in order to place one character box next to another. This information is kept in the `tfm` file.

Let’s look at a few characters and their `tfm` boxes, to see how they fit in METAFONT’s coordinate system. Fig. 2 shows the uppercase letter W: (w) indicates the width of the `tfm` box, (h) is the height, and (d) is the depth of the box. Fig. 3 shows another character, the lowercase letter g, which has a non-zero depth value, and we see that it has a descender, which goes below the baseline. We can also see some labels inside the character; these are called control points.

#### 2.1 Control Points

Control points tell METAFONT where to draw, or, more accurately, where to have the digital pen pass through, leaving a wake of ink. Here is one way to assign a value to control point 1:<sup>1</sup>

```
x1=10;  
y1=25;
```

<sup>1</sup> Semicolons are used to separate METAFONT statements.

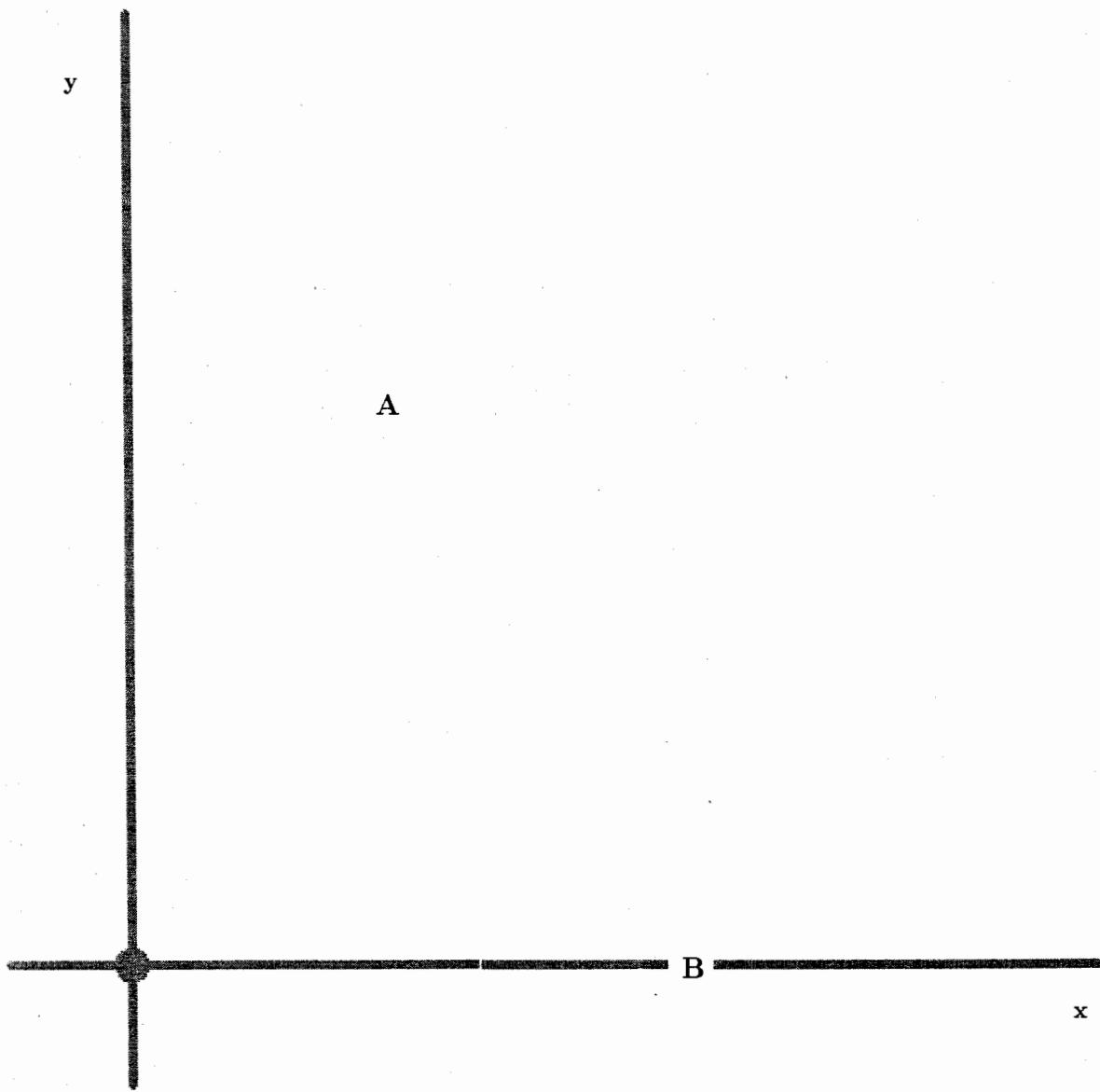


Figure 1: The Cartesian Coordinate System

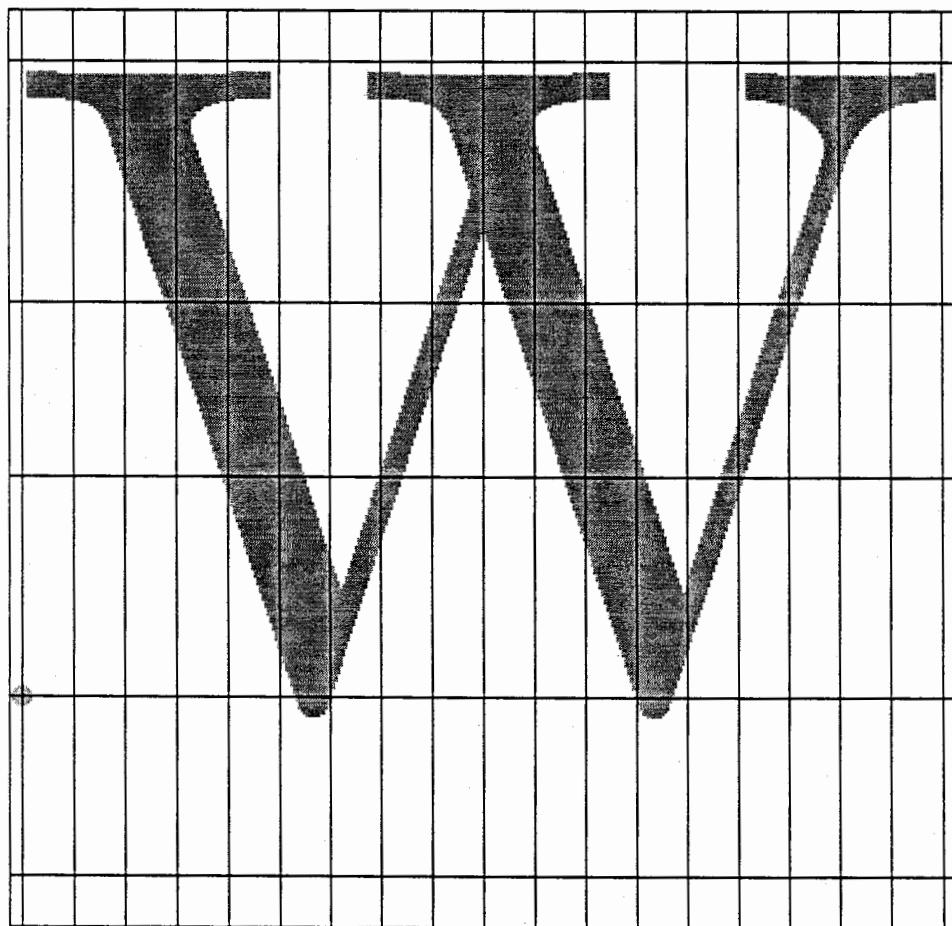


Figure 2: The capital letter **W**

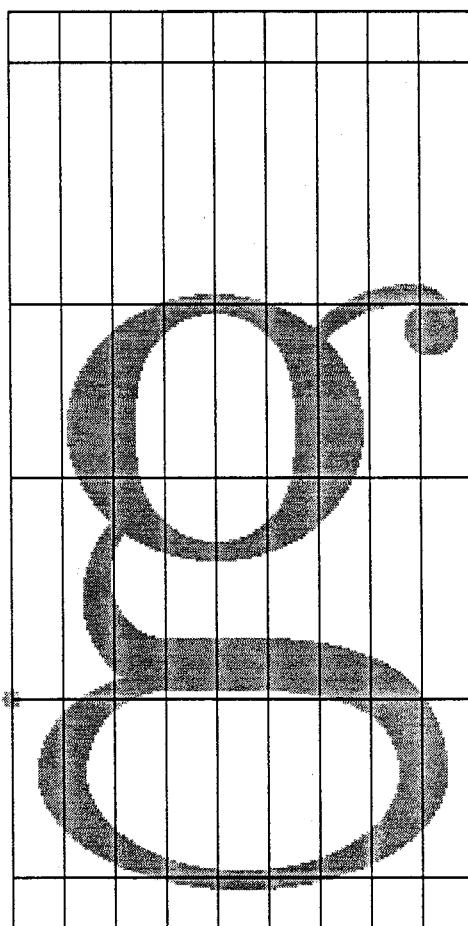


Figure 3: The lowercase letter **g**

We can also assign the same values to the same control point with a single statement like this:

```
(x1,y1)=(10,25);
```

or alternately, a pair of variables can be assigned with a *z*-point notation where *z* represents an *x-y* pair. It is sort of a shorthand method for describing a **coordinate pair**. It looks like this:

```
z1=(10,25);
```

All three statements just shown are equivalent.

Now let's define some more control points and see what happens when we try to draw something with the **draw** command. Here is one way to define some control points:

```
y1=25;  
z2=(75,50);  
x3=100; x1=y3=y5=10;  
z4=(120,-20);  
x5=150;
```

Notice that the *x1*, *y3* and *y5* values have all been assigned in one statement as being 10, and that the *z2* and *z4* control points were assigned in a single statement. By combining the *x-y* assignment and the *z* assignment methods, we can save quite a bit of typing and also make it clear to METAFONT the relationship between our control points at the same time.

Here is a simple **draw** statement to help us see the path we have defined (after we have started up the demo on the Macintosh, that is):

```
virmf2 &cm \mode=proof; screenstrokes; input tugcon  
draw z1..z2..z3..z4..z5;
```

Figure 4 illustrates the path that results after executing the **draw** statement. So we can get a better feel for what METAFONT is doing, let's look at the individual control points along the curve we just drew (Fig. 5):

```
lose_control(1,2,3,4,5)
```

This macro was one that I created for this conference so it would punch holes in the path of the previous **draw** command, and we could better see how control points are used.

Of course, we don't need to say **draw z1..z2..z3..z4..z5**; with the control points ordered sequentially from 1 to 5; we can also draw starting and ending at any defined control point. For instance, if we said:

```
clearit;3  
draw z1..z3..z4..z5..z2;
```

instead of our previous order, we would get the shape shown in (Fig. 6)

After we expose the control points, Fig. 7 shows the results of:

```
lose_control(1,3,4,5,2);
```

We can see by the control points inside our drawn path that the curve starts at point 1, proceeds down to 3, then curves nicely around to 4 and 5, and ends up at point number 2. METAFONT draws nice curves through these points, and in order to continue smoothly to the next point, it needs to swing out a little ways after passing through a control point. The way that METAFONT makes these pleasing curves is internal; all you need to do is specify the control points to draw through and it does the rest for you. You can change how METAFONT draws curves with special curve modifying commands and we might explore a few of these later. For now, let's look at how METAFONT draws curves.

## 2.2 Curves

When METAFONT draws a curve, it uses something we can simply call "the four-point method". If we have four control points (Fig. 8):

<sup>2</sup> *inimf* is known as the initialization version of METAFONT, *virmf* is the production version.

<sup>3</sup> The **clearit**; statement is one which we use to erase the previous picture that METAFONT was saving for us so we can draw again.

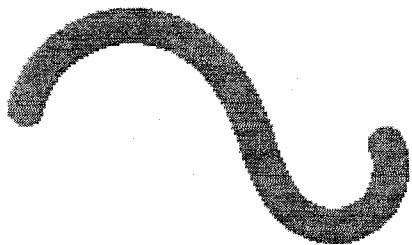


Figure 4: Curve 1 2 3 4 5 with default pen

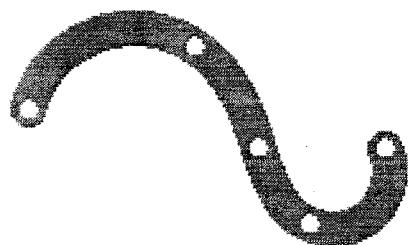


Figure 5: Curve 1 2 3 4 5 with exposed control points

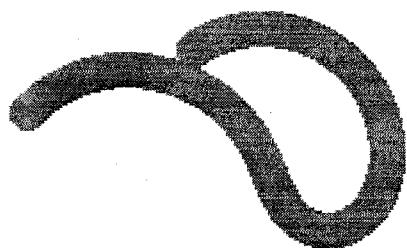


Figure 6: Curve 1 3 4 5 2 with default pen

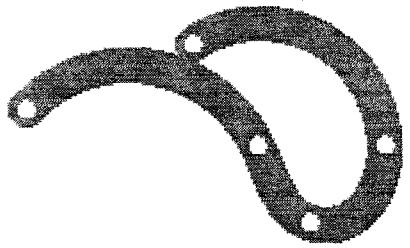


Figure 7: Curve 1 3 4 5 2 with exposed control points

Figs. 4-7: Curves and Control Points

```
z1=(35,100); z2=(60,10); z3=(200,10); z4=(225,75);
```

the curve that METAFONT would draw is found by repeated mid-point calculations, as in Fig. 9.

A more technical name for the curve defined by METAFONT is a Bézier cubic. What METAFONT does for us is take the original control points we supply and add other control points of its own, as we see in Fig. 10. Then it refines the curve between the "scaffolding", as Knuth calls it, until the curve is left on the innermost path between midpoints, which is what we see in Fig. 11. METAFONT then discards the scaffolding and draws a nice curve which is inside the scaffolding.

These are the basics for drawing curves, using this four-point refinement method. There are also other commands which affect how the scaffolding is built. For instance, there are commands which can create more tension in the curve, such as in Fig. 12, or more curling of curves at the endpoints (Fig. 13).

The degree to which you can manipulate METAFONT curves is really quite astounding. Unfortunately, there is not enough time to go into all the ways to generate different curves with METAFONT.

### 2.3 Pens

Another interesting concept is that of a METAFONT pen. A good way to view the sizes and strokes we use to draw with METAFONT is to think of them as being produced by nibs of different pens (because, in fact, they are). Until now, we have only used one pen type for our examples and since we didn't specify, METAFONT provided us with a default pen. Let's look at some different pen types and how to use them.

Before you start drawing with a pen, you generally have to pick it up first, and here is how we tell METAFONT to do just that:

```
pickup pencircle;
```

In addition to a circular nibbed pen, there are a few other pen types that METAFONT knows about (through definition in the plain base file). They are:

```
pensquare  
penspeck  
penrazor
```

`penspeck` and `penrazor` are special-purpose in nature; `penspeck` is used in the `drawdot` macro, and `penrazor`, as the name implies, is a razor-thin pen (one pixel). `pencircle` and `pensquare` perform mostly as you would expect of pens with such names. Let's look at how we can specify different pen nibs via some examples:

```
% clear drawing board, but not control points  
clearit;  
% pickup a pen to draw with  
pickup pencircle;  
% and draw !  
draw z1..z2..z3..z4..z5;
```

As we see, this is the pen we used before (the default pen). Let's look at a few ways to "build" some pens for METAFONT to use. One way to change our pen is by scaling it to the size desired. There are three scaling commands: `scaled`, `xscaled`, and `yscaled`. Here is a command which scales a pen to nearly one tenth point size:

```
clearit;  
pickup pencircle scaled .1pt;  
draw z1..z2..z3..z4..z5;
```

Notice the size difference from the last pen we used. This pen (Fig. 14a) is much smaller than our default in Fig. 4, which was approximately .4pt.

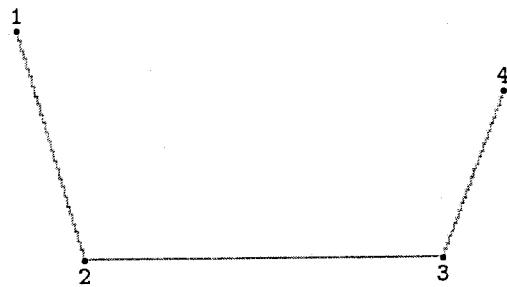


Figure 8: Four-point method 1 2 3 4 draw

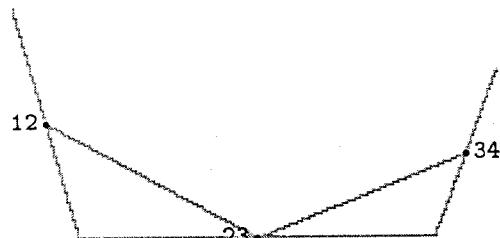


Figure 9: Four-point method 12 23 34 draw

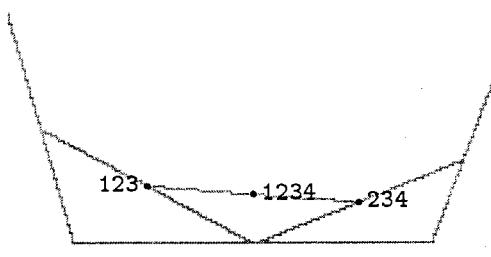


Figure 10: Four-point method 123 1234 234 draw

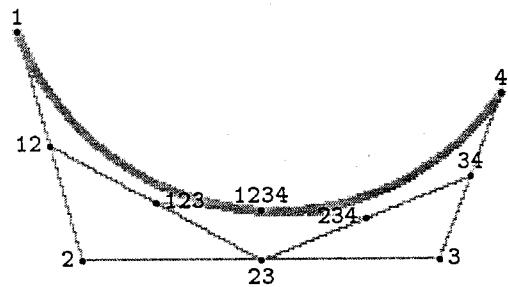


Figure 11: Four-point method 1 1234 4 draw



Figure 12: Four-point method 1 tension 2 1234 tension 2 4 draw



Figure 13: Four-point method 1 curl infinity 1234 curl infinity 4 draw

Figs. 8–13: Four Point Method

Figure 14a: Pen = .1pt

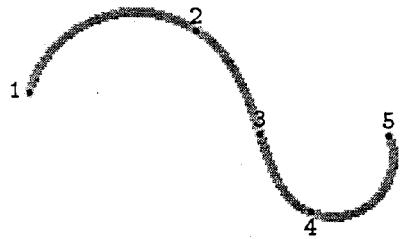


Figure 14b: Curve with pen = .1pt

Figure 15a: Pen = xscale=.6pt, yscale=.2pt

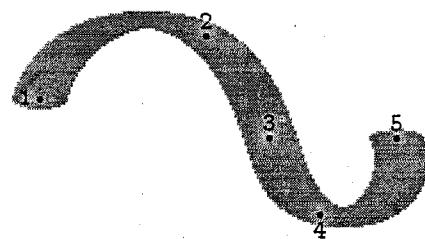


Figure 15b: Curve with pen = xscale=.6pt, yscale=.2pt

Figure 16a: Pen = xscale.2pt, yscale.6pt

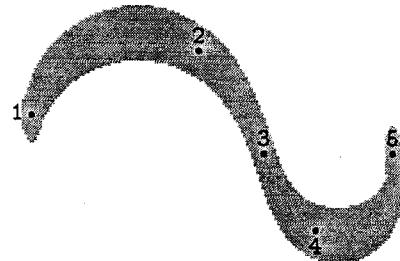


Figure 16b: Curve with pen = xscale=.2pt, yscale=.6pt

Figure 17a: Pen = xscale.2pt, yscale.6pt,  
rotate 32 degrees

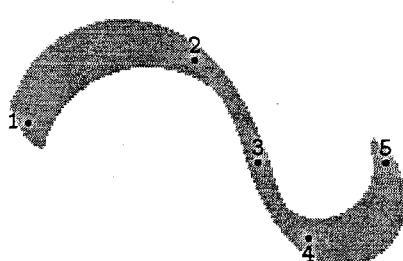


Figure 17b: Curve with pen = xscale.2pt, yscale.6pt,  
rotate 32 degrees

Figs. 14-17: Pen Building

The following command introduces the *x* and *y* scaling operation:<sup>4</sup>

```
clearit;
pickup pencircle xscaled .6pt yscaled .2pt;
drawem;
```

In Fig. 15a, we can see the pen nib is wide and short, since the xscaling is greater than the yscaling. If we switch the *x* and *y* scaling, like this:

```
clearit;
pickup pencircle xscaled .2pt yscaled .6pt;
drawem;
```

we get the results shown in Fig. 16a, where the nib is thin and tall, since the yscaling is greater than the xscaling.

Another parameter of control that one has for pen manipulation, aside from scaling, is **rotation**. Here is a sample (Fig. 17a) that has the pen rotated 32 degrees with the same xscaling and yscaling as the previous example (*x*=.2; *y*=.6):

```
clearit;
pickup pencircle xscaled .2pt yscaled .6pt rotated 32;
drawem;
```

This last pen seems to emulate a calligraphic pen, with the rotation acting as the angle of a pen being held by a hand.

This concludes the section on METAFONT commands. Now I will attempt to give a brief history of METAFONT, and then show a little of the power behind METAFONT.

### 3. METAFONT — Evolution of a Program

Originally, METAFONT had only 28 parameters which described the small pieces which make up a complete character. After working with Hermann Zapf, Mathew Carter, Charles Bigelow and Kris Holmes (receiving much feedback from them all in 1981), Knuth worked very hard at bettering his original typefaces. Then, in April 1982, Richard Southall came to Stanford and helped make extensive changes to the Computer Modern programs (especially the sans serif letters). This resulted in the refinement of the METAFONT language and brought the number of parameters to 45. Although small refinements occasionally surface in the Computer Modern typefaces, they remain today steady and stable with the total number of parameters at 62 as there have been since 1985 (see Appendix A for a list of the parameters for **cmr10**).

So one of the key ideas behind a METAFONT is that there are a large number of parameters to describe what a character looks like. By varying these parameters, we can see how different typefaces are created. Let's look at some differences in parameters by viewing the result of a test file named **6test**, which uses six different sets of parameters to create six variations of the same character.

```
virmf & \mode=proof; mag=.33; screenchars; input 6test
```

As we can see in Fig. 18, there are six different characters that have been generated on the screen. They are: **cmr10** or Computer Modern Roman (top left), **cmss10** or Computer Modern Sans Serif (top middle), **cmtt10** or Computer Modern Typewriter (top right), **cmb10** or Computer Modern Roman Bold (bottom left), **cmbx10** or Computer Modern Roman Bold Extended (bottom middle), and **cmti10** or Computer Modern Text Italic (bottom right).

### 4. A Word About Computer Modern

The Computer Modern typefaces, 75 in all, comprise the effort put forth by Knuth to create the spirit of the typeface Monotype Modern 8A, which has traditionally been used to print textbooks of all sorts, including Knuth's first two volumes of *The Art of Computer Programming*. If one doesn't like Computer Modern, I believe it is because one doesn't like Monotype Modern 8A, not because Knuth made a poor rendition of same.

---

<sup>4</sup> I was lazy and didn't want to make too many typos in my demo, so I created the macro **drawem**, which draws from control points 1 to 2 to 3 to 4 to 5.

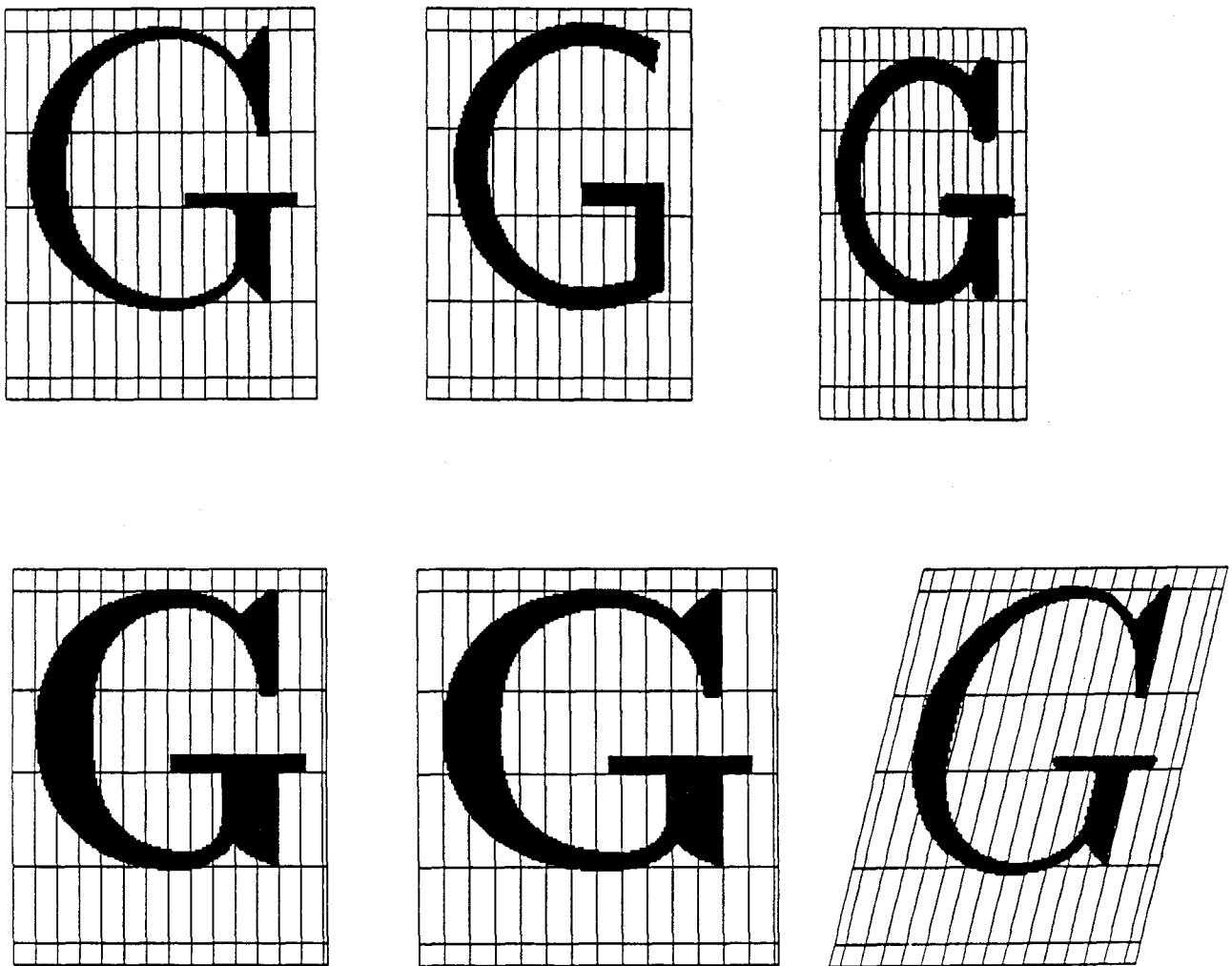


Figure 18: Results of the `6test` file

An extreme example of this misunderstanding that I have encountered, occurred when someone waved an Epson dot matrix folio in my face, and exclaimed "this is ugly". Well, looking at it, I had to agree. And when this person further claimed that the sad looking characters on the page didn't look anything like Times Roman (obviously what he expected) I also had to agree. I think this misunderstanding is common. People often view Computer Modern and say to themselves "Why didn't Knuth typeset his first volumes in Garamond or Palatino?", just wishing that Computer Modern was actually one of those, or perhaps Times Roman (these fonts happen to be in vogue now). Well, he didn't and they are not. They should not be compared in this apples-are-better-than-oranges way. Besides, Garamond and Palatino fonts are proprietary fonts; source files would certainly not be available — as they are for the CM fonts — for users to modify and alter and re-shape.

The challenge which lies ahead is for brave souls to create new typefaces, or adapt classic typefaces to satisfy the TUG community. We can all start with some understanding of typography, and the basics of METAFONT, and go from there.

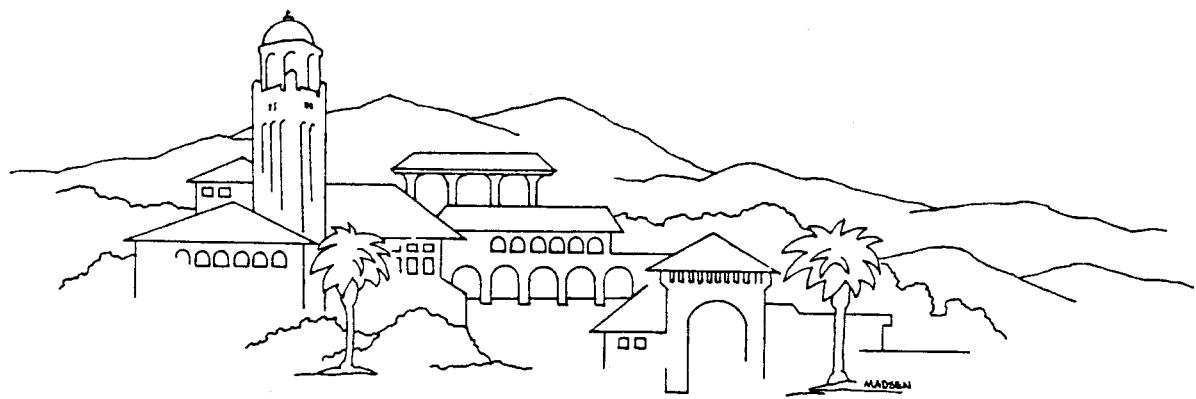
Any volunteers?

### Bibliography

- Knuth, Donald, E. *The Art of Computer Programming*, vols. 1-3 Reading Mass.: Addison Wesley, 1968, 1973
- Knuth, Donald, E. *The METAFONTbook Computers and Typesetting*, Vol. C. Reading, Mass.: Addison Wesley, 1986.
- Knuth, Donald E. *METAFONT: The Program. Computers and Typesetting*, Vol. D. Reading, Mass.: Addison-Wesley, 1986.
- Knuth, Donald E. *Computer Modern Typefaces. Computers and Typesetting*, Vol. E. Reading, Mass.: Addison-Wesley, 1986.

## Appendix A

```
% This is CMR10.MF in text format, as of Mar 31, 1986.  
% Computer Modern Roman 10 point  
if unknown cmbase: input cmbase fi  
font_identifier:="CMR"; font_size 10pt#;  
u#:=20/36pt#; % unit width  
width_adj#:=0pt#; % width adjustment for certain characters  
serif_fit#:=0pt#; % extra sidebar near lowercase serifs  
cap_serif_fit#:=5/36pt#; % extra sidebar near uppercase serifs  
letter_fit#:=Opt#; % extra space added to all sidebars  
body_height#:=270/36pt#; % height of tallest characters  
asc_height#:=250/36pt#; % height of lowercase ascenders  
cap_height#:=246/36pt#; % height of caps  
fig_height#:=232/36pt#; % height of numerals  
xheight#:=155/36pt#; % height of lowercase without ascenders  
math_axis#:=90/36pt#; % axis of symmetry for math symbols  
bar_height#:=87/36pt#; % height of crossbar in lowercase e  
comma_depth#:=-70/36pt#; % depth of comma below baseline  
desc_depth#:=70/36pt#; % depth of lowercase descenders  
crisp#:=Opt#; % diameter of serif corners  
tiny#:=8/36pt#; % diameter of rounded corners  
fine#:=7/36pt#; % diameter of sharply rounded corners  
thin_join#:=7/36pt#; % width of extrafine details  
hair#:=9/36pt#; % lowercase hairline breadth  
stem#:=25/36pt#; % lowercase stem breadth  
curve#:=30/36pt#; % lowercase curve breadth  
ess#:=27/36pt#; % breadth in middle of lowercase s  
flare#:=33/36pt#; % diameter of bulbs or breadth of terminals  
dot_size#:=38/36pt#; % diameter of dots  
cap_hair#:=11/36pt#; % uppercase hairline breadth  
cap_stem#:=32/36pt#; % uppercase stem breadth  
cap_curve#:=37/36pt#; % uppercase curve breadth  
cap_ess#:=35/36pt#; % breadth in middle of uppercase s  
rule_thickness#:=.4pt#; % thickness of lines in math symbols  
dish#:=1/36pt#; % amount erased at top or bottom of serifs  
bracket#:=20/36pt#; % vertical distance from serif base to tangent  
jut#:=28/36pt#; % protrusion of lowercase serifs  
cap_jut#:=37/36pt#; % protrusion of uppercase serifs  
beak_jut#:=10/36pt#; % horizontal protrusion of beak serifs  
beak#:=70/36pt#; % vertical protrusion of beak serifs  
vair#:=8/36pt#; % vertical diameter of hairlines  
notch_cut#:=10pt#; % maximum breadth above or below notches  
bar#:=11/36pt#; % lowercase bar thickness  
slab#:=11/36pt#; % serif and arm thickness  
cap_bar#:=11/36pt#; % uppercase bar thickness  
cap_band#:=11/36pt#; % uppercase thickness above/below lobes  
cap_notch_cut#:=10pt#; % max breadth above/below uppercase notches  
serif_drop#:=4/36pt#; % vertical drop of sloped serifs  
stem_corr#:=1/36pt#; % for small refinements of stem breadth  
vair_corr#:=1/36pt#; % for small refinements of hairline height  
apex_corr#:=0pt#; % extra width at diagonal junctions  
o#:=8/36pt#; % amount of overshoot for curves  
apex_o#:=8/36pt#; % amount of overshoot for diagonal junctions  
slant:=0; % tilt ratio (delta x/delta y)  
fudge:=1; % factor applied to weights of heavy characters  
math_spread:=0; % extra openness of math symbols  
supereness:=1/sqrt2; % parameter for superellipses  
superpull:=1/6; % extra openness inside bowls  
beak_darkness:=11/30; % fraction of triangle inside beak serifs  
ligs:=2; % level of ligatures to be included  
square_dots:=false; % should dots be square?  
hefty:=false; % should we try hard not to be overweight?  
serifs:=true; % should serifs and bulbs be attached?  
monospace:=false; % should all characters have the same width?  
variant_g:=false; % should an italic-style g be used?  
low_asterisk:=false; % should the asterisk be centered at the axis?  
math_fitting:=false; % should math-mode spacing be used?  
generate roman % switch to the driver file
```



**T<sub>E</sub>X Users Group Meeting and Short Course  
Stanford University, July 25–30, 1982**

# Opening Pandora's Box

NEENIE BILLAWALA

841 Stendhal Lane  
Cupertino, California 95014  
sun!metamarks!nb

## ABSTRACT

The motivation behind the Pandora project was to use METAFONT as a design tool rather than as a production tool in the creation of a typeface.

Pandora was developed by using generalized descriptions of the visual relationships between parts of characters, characters in a font, and fonts in a typeface family.

Pandora transforms from one font in her family to the next through different parameter settings applied to a single framework. It is not important that all variations look good, rather that a reasonable set can be found within the framework. A rich description allows a designer to quickly look at a number of possibilities.

## 1. METAFONT

### 1.1 Background

When I first heard about METAFONT in 1980, the idea seemed intriguing — using computers to design type. In 1984 when actually faced with the first generation of SUN computers, an experimental V operating system with a cryptic boot command, and version 0.\*\* of METAFONT, I was not sure what it could really do — or rather, what I could convince it to do. The idea of designing type that was modifiable by the proverbial “turning of knobs” was seductive. And the idea that you, the designer, could choose those knobs was even more so.

It was not very confidence-inspiring to know that I had no computer experience and there was no manual yet to use as a guide. However, I did have some ideas of what I would like METAFONT to do and lots of help from the authors of the language.

Designing begins with ideas of how to solve a problem. What is the problem, what tools can be used to help solve the problem, how are those tools best used, what palette of solutions presents itself?

The problem of type design involves identifying the requirements of a design. Such requirements are outlined in a document called a *design brief*. For what purpose is the design being considered? Is it a text or a display face, or intended primarily for use as symbols? What visual flavor is it to have? What feeling does it convey? Will it have a single weight or will a family of weights need to be considered? Type is intended to be reproduced without deviation within a given range of tolerance. So, which technologies will be used in reproducing it? Additionally for METAFONT — what kind of flexibility do you want to build into the design to make it adaptable to different marking engines or different resolutions?

### 1.2 Drawing

Drawing type by hand is a special skill that takes years to develop, and even then, only a handful of people are successful at it. Having an eye and a sense of what belongs in a design takes another talent. Sometimes that talent combines in one person, but often it is the collaboration of these two types of skills that results in type.

Drawing type with METAFONT, a tool that can draw a precise line and a pleasing curve, takes time to learn also. Instead of the hand-eye coordination that a calligrapher or a type designer develops, you need to develop a feel for mind-eye coordination. You need to be able to translate

visual shapes into the mathematical constructs that METAFONT understands. A steady hand is no longer needed as METAFONT will do the drawing for you. Though, an understanding of the problem and a sensitive eye is still required.

METAFONT uses pens to draw — and it has a broad notion of pens. These pens can have an arbitrary shape and size and inclination. They may even have no thickness, so that an outline can be filled. They relate to pens that we use for drawing, making the marks of a tip or nib that follows a path (Figure 1). However, the path traced out by the human hand will be much different from that traced out by a mathematical equation.

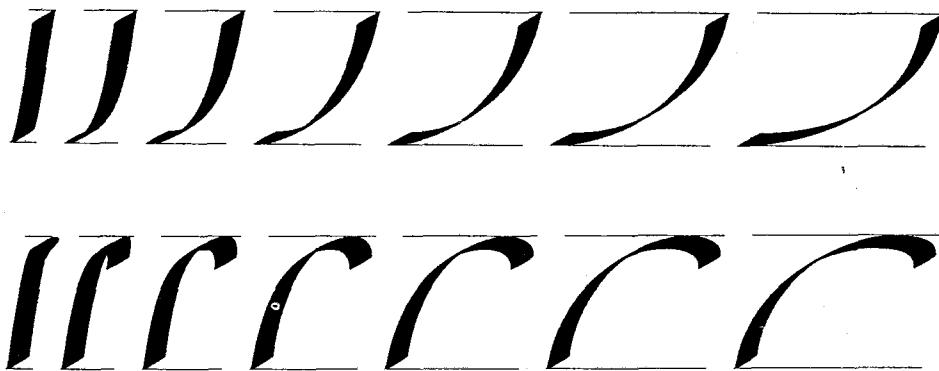


Figure 1: Pen strokes defined by two (top) or three (bottom) control points

A calligrapher's pen creates marks that are a combination of the shape of the nib, the consistency of the ink, the texture of the writing surface and subtle changes in pen pressure and pen angle. Such variety is difficult to recreate with METAFONT. While METAFONT can easily change pen nibs and pen angles, the "written" result is consistent — there is no fluctuation for writing surface or ink or pressure. This is not to say that one result is more or less beautiful than the other, just that it is not the same.

### 1.3 Designing

What makes METAFONT special is the fact that you can actually design with it. You can go through the same trial and refinement process of the traditional type design method — only the drawing tool looks a bit different.

METAFONT's strengths lie in its flexibility. The value of a flexible description becomes apparent as the technology changes. I remember a few years back when a 200 dpi XGP printer was being retired in the Computer Science department at Stanford. There were some 500 hand-tuned fonts that had been created for this printer and that represented man-years of work. Professor Art Samuel tackled the task of making these fonts work with the newer generation of 300 dpi laser printers. He developed an ingenious program that considered the pixel patterns, alternating rows of pixels and neighboring pixels, and then automatically converted a 200 dpi pattern to a 300 dpi pattern. In some ways the results were successful — previously created documents could still use the same fonts and they had about the same look as the 200 dpi printer. Unfortunately, the method could not take advantage of the higher resolution to improve the look of the fonts. Had those fonts been in a METAFONT format, it would have been a small matter to regenerate them at a higher resolution.

There are times when it might be useful to include more than resolution information into a typeface. You might have a printer where one-pixel lines are too thin. It might be a write-white printer or perhaps a high resolution imagesetter. It might be convenient to be able to set a minimum pixel thickness of 1 or 2 or even 5 as the resolutions increase. It might be useful to thicken up a design if you are going to print in reverse or increase all serif lengths by an arbitrary amount. Perhaps you just want a slightly different design. It might be that none of the above applies and other considerations will arise in the future.

Once you learn how to transfer your ideas, you can embed more information into the character than could ever be guessed from a single drawing. Though it's possible to use METAFONT in a form of digital tracing, the potential richness of a flexible description would be lost.

METAFONT's value as a design tool is carried out by the fact that you can draw any number of iterations, changing one or many things in order to find a design solution. The drawing can be done more quickly than by hand, allowing the designer to consider many more possibilities and directions. With the aid of a printer to test variations and compare the results, you can draw and experiment.

## 2. Metamarks

In some ways type design lends itself well to description. The visual consistency that defines a typeface can be described in a program. Traits that are shared among characters — vertical dimensions, stem thicknesses, terminal treatments — can also share an analytical relationship.

The question is one of how to translate those visual relationships into language METAFONT understands. Words convey ideas, concepts and information well, but they are often lacking when it comes to describing something visual. It is really true that "one picture is worth a thousand words".

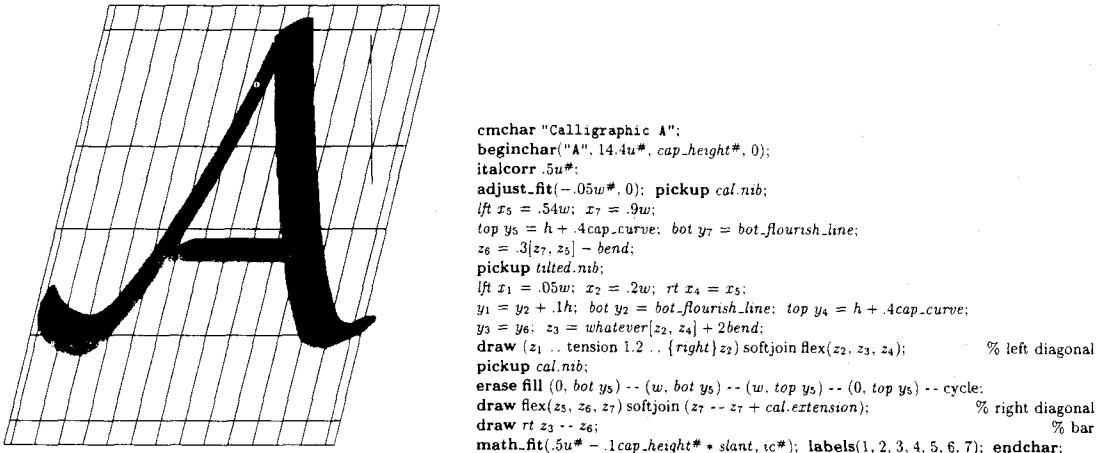


Figure 2: Sample of METAFONT character output and source file (Computer Modern)

Mathematics has long been used as a tool to link the visual with the symbolic. METAFONT is an algebraic language. Algebraic equations describe the relationships you want to keep (Figure 2). Cubic Bézier curves are then drawn according to these relationships (Figure 3).

Certain relationships are very useful to maintain with type. The xheight of the typeface should be the same for all lowercase letters, just as the height of capitals should be the same. Sometimes all the numerals should have the same width so that they will align. With a drawn representation of a set of characters, measurements need to be taken to ensure consistency. A change to the xheight requires careful re-drawing of all the characters. However, should the xheight be a parameter that is shared by all characters, then a change to this parameter will have a global effect.

The first thing is to find a relationship in related parts of characters, and the next step is to look at a number of variations. It is a matter of experimentation and experience that will lead to discovery of those relationships that work well together. The visual consistency found in the characters of a typeface led to *Metamarks: Preliminary Studies for a Pandora's Box of Shapes*, an exploration of the related but not identical shapes that are found in several characters.

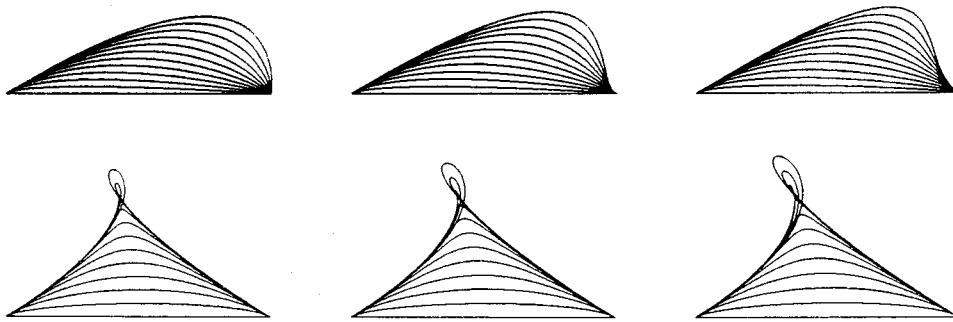


Figure 3: A series of Bézier curves, showing the general effect of control point placement

Part of the discovery of how METAFONT works lies in understanding its logic. You may understand how the curves are drawn and how parameters work, but you may still be surprised at the result. Since you can build a description, you must also be aware of the relationships you are building. You will need to think clearly about how the inter-relationships of the parameters will affect what you build.

Though everything is written down explicitly enough for a computer to understand, the specifications may not show the whole intent of the designer. It is virtually impossible to predict all outcomes for all cases, but with trials and error messages, METAFONT will direct you.

### 3. Pandora

Pandora started out as an experiment. Could this new tool and some wild ideas really combine into something that overlapped with the spirit of good type design? Was it possible to take the sense of traditional type design and apply it to the digital technology? Did the new format present options that were previously unavailable? Could the general requirements of functionality for type be satisfied? Generally, the answers turned out to be yes.

Pandora was the result of a *meta*-design brief that developed as I learned more about METAFONT, type and technology. Other than the idea that Pandora would be a family of roman-based forms and that text faces would be the main focus, there were no *specific* requirements to fulfill. No style had to be matched, no vertical dimensions maintained and no single marking engine was targeted. Pandora's was a *meta*-design brief, because many potential briefs were considered.

Mostly, Pandora is a family of text typefaces. I could see a relationship shared among text typefaces that was not as clear in display typefaces; for example, Bodoni and Helvetica are much closer than Calypso and Stop (Figure 4).

However, in place of the specifics that make up a traditional design brief, Pandora has parameters. Rather than have a specific *xheight*, Pandora has a parameter called *xheight*. Instead of a single stem weight, there are several parameters for setting stem thicknesses. The general slant of the typeface or obliqueness is also a changeable value. Pandora is special, because of her versatility.

#### 3.1 Meta-Design

In meta-designing a typeface family, you need to think about the visual “essence” of a character. What makes a character recognizable? Is there an objective ideal shape? Do some cultures read letters better when they are of one style and others understand them better when they are of another? How much of what we consider to be clear and beautiful text comes from what we have grown up with? Is Univers really a universal typeface?

An essence cannot be defined by a single shape, but some of it is in every letter that we read. It is this indescribable essence that makes people from around the world understand the wide variation in handwriting and typefaces. Today we have a hard time reading the texts that were popular 500 years ago and printed in a blackletter type. Is it because we are not used to them or are they intrinsically harder to read?

ABCDEFHIJKLMNOPQRSTUVWXYZ  
vwxyzABCDEFHIJKLMNOP

CALYPSO

ABCDEFGHIJKLMNOPQRS  
TUVWXYZ

STOP

ABCDEFGHIJKLMNOPQRSTUVWXYZ  
abcdefghijklmnopqrstuvwxyz

HELVETICA LIGHT

ABCDEFGHIJKLMNOPQRSTUVWXYZ  
abcdefghijklmnopqrstuvwxyz  
1234567890

MONOTYPE BODONI

Figure 4: Contrast between display and text fonts

With Pandora I tried to capture some of this essence. I tried to construct the characters and the relationships that would preserve it. It is important to emphasize the features of each character that make it readily recognizable. Not that every variation would be a beautiful or appropriate solution, but that a set of reasonable solutions could be found. I did not look for the one and only solution, or an all-encompassing one — just a set that would work reasonably well and be flexible.

You can set relationships that should be preserved, and put in as much or as little control as you want. Sometimes it is interesting to give METAFONT a simple general guide and see what kinds of shapes can be created. It may lead you onto a path that you might not otherwise have considered. You might make a typeface that includes a certain amount of randomness. Or one that has several variations of the same character. It would be possible to create a set of three (or more) related interchangeable fonts and then perhaps set each third word with characters from each for the three fonts. Or you might want more control and place more conditions on your characters. For example, I have made the joining point of the arch in the lowercase *n* be on the stem somewhere between the top of any serif that there may be and the bottom part of the thickness of the arch. This avoids problems and fills the condition of most useful arches.

In looking for underlying relationships, does everyone think differently or is there something “objective” about the relationships? Separately, both Don Knuth and I came up with essentially the same serif scheme, though we never discussed it. On the other hand, there are many differences in the approach to Computer Modern and to Pandora. It would be interesting to see the results of giving the same set of characters to ten different designers, with the same instructions to fill out the family. Would there be ten different results? Would they all share some of the underlying concepts of change? In fact, Figure 5 shows the results of having 26 different designers create two characters (one uppercase, one lowercase), with the same vertical dimensions. However, it takes more than mathematical relations to make a uniform typeface.

The underlying concept will not apply to every kind of letter in every language. It is good judgement that decides the balance between a fair amount of flexibility and creating a separate basic form. Pandora and much of the Computer Modern family are good examples of roman forms. Other sets of basic forms will work for gothic characters, Chinese, Tamil and so on.

A B C D E F G H I J K K L M  
 N O P Q R S T T U V W X Y Z  
 a b c d e f g h i j k l m  
 n o p q r s t u v w x y z

Figure 5: Results from 26 designers given only a few design parameters (Knuth 1984:106)

### 3.2 A Design Approach

Many of our text types come with a history whose origins begin with the written letter. It is a natural assumption when looking at a letter, to think that its inspiration might be a pen. Indeed, my first trials with METAFONT consisted of letters created by penstrokes. These characters tended to have a calligraphic flavor, but were somewhat limited.

However, type is created for a specific technology. Type does not exist without the means to create it. It is in the re-working of the forms from the hand-written inspiration that the deviations in shape occur — deviations that are no longer easily recreated by the penstroke.

For this reason, designing with penstrokes, whereby most of an unmodified METAFONT pen-stroke defines a character, can be limiting. It is useful in some cases and necessary in others. For the most part, however, I found it much more flexible to design with an outline, or a penstroke of no thickness.

The next step is to identify explicitly all those things that are important in the design. Vertical dimensions are an excellent example of this. You may know that you want lowercase characters to share an xheight, but you may not know what that value is to be. It may be between 50–55% of the point size, or it may be dependent on the height of capital letters, or it may be dependent on some other consideration. Rather than give an explicit xheight value in character description, you make the value of the xheight a parameter. Then as that parameter changes, all the places where the xheight is used will be changed.

The same applies for the height of capitals and all other shared vertical dimensions. When we talk about the height of a character, we are talking about the visual height of a character, not necessarily the mathematically measured value from top to bottom.

There are a few typical optical illusions that need to be considered. Look at the case where a circle, a square, and a triangle all share the same vertical and horizontal dimensions (Figure 6). The shapes all have a different visual weight. This is natural if you think about the area encompassed by each. Though vertical dimensions are equal, the circle and the triangle appear a bit shorter and narrower than the square. It shows that by at least one mathematical calculation, the “same value” does not result in the same visual result.

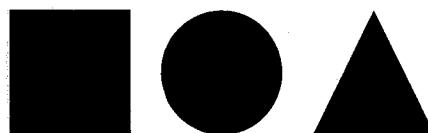


Figure 6: Optical illusion: square, circle and triangle with same vertical dimensions

These optical illusions carry over into type design. If the curved and pointed characters, such as an *O* and *A*, are to have the same visual vertical height as other characters, they may actually have to be a bit bigger in the vertical dimension. Those characters that have a strong

horizontal feature, such as an *E* or a *T*, may have to be slightly shorter. One way to make the necessary adjustment is to build in another parameter that is dependent on the vertical height and perhaps on the resolution also.

Other features of a character combine to create a visual total. The inner shape or *counter* of a character plays a critical role. The proportion of thick and thin in a letter can combine to change the visual height and width. The example from *Metamarks* (Figure 7) gives a clear demonstration. Equal horizontal and vertical weights may give the illusion that one is heavier than the other. Unequal stem widths may actually look better. Typically weight is either added or sculpted away in order to achieve a pleasant and functional visual balance.

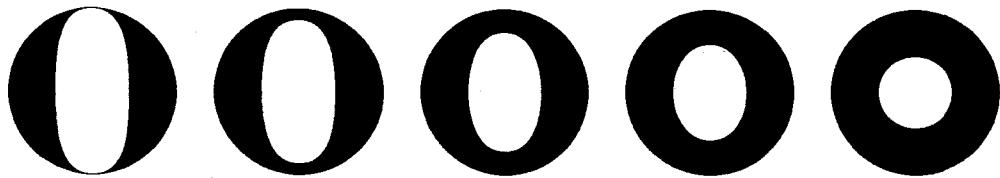


Figure 7: Circular shapes with variation in vertical stroke weight

Part of the design process remains the same. There are still the same problem solving aspects and visual considerations in the result. However, METAFONT does tend to change the approach. Rather than start with specifics, general parameters are built and visual relationships that have mathematical counterparts are constructed.

### 3.3 Parameters

Pandora relies on over one hundred parameters to define her shape. Some define dimensional limits, others relate to specific parts of characters, and some affect every character. The first group of parameters has values that are independent; the next group has values that are mostly dependent on the first group.

The independent parameters include common vertical values and thicknesses of stems and bowls. The *designsize* is set to the point size of the font. Other values, though independent, are implicitly dependent on the *designsize* or on each other. Typically the height of capitals or *cap* height is less than the *designsize*. An *xheight* is usually smaller than the *cap* height. Stem weights are usually less than bowl weights.

The dependent parameters are generally those that relate to parts of characters. These parameters are used in the macros that are then used to build the actual character descriptions. For example, there are a number of parameters that determine the shape of serifs and terminals. The *serif\_thickness* is dependent on the *xheight*. It cannot be greater than half of the *xheight*, otherwise a character with two serifs would disappear. The *terminal\_thickness* is related to the *serif\_thickness*, and in Pandora they are set equal.

In addition to the serifs and terminals, there are parameters for arms, arches, bowls, circular shapes, junctures and notches. There are also settings for punctuation, accents, width, obliqueness, softness of the corners, and a number of other features.

However, the parameters are useless unless they have values. It is somewhat like working backward. You know you will need to have an *xheight*, but unlike a traditional method of working, you might not know how high it is until you have fit in the rest of the design. By defining the parameters, you will have necessarily thought about what each is meant to affect and you will have a starting point as to what value to give the parameters. In order to see an instance of the character or typeface, you will need to give an arbitrary value to each of the parameters. Not every possibility will work, but neither will every hand-drawn shape. It is more important to find a range of parameter settings that work together reasonably.

### 3.4 A Rich Description

Instructions for a letter to behave under certain conditions can be embedded into the character description. Algebraic equations specify conditions; booleans and conditional statements are

used to make decisions. If all stems are to have the same number of pixels, then give them all the same value, and make sure they all begin on a pixel boundary. If point A is halfway between points B and C, METAFONT's way of saying this is  $A = .5[B, C]$ . In Pandora, there are alternate character possibilities for some of the lowercase letters. An upright style may use one variation, whereas an italicized version might use another (Figure 8).

Since the glue that ties the METAFONT family together is the underlying concept, this means that *fixed width* and *proportional* characters can share a relationship. Ideas of typeface consistency remain the same — stem widths are the same or similar, there is the notion of an xheight and cap height and so on. The major difference is that the widths have a special requirement that they all be the same. How do you fit the letter *m* into the same space as the letter *i*? Since letters were not meant to all have the same width, it is almost like trying to fit them into a straight-jacket. Variations can be made that might stand out in a normal text, such as adding or lengthening a serif in the *i* in a face that is otherwise sans serif.

Unlike previous technologies, there is no longer a need for different weights in a family to retain the same widths. Widths can vary according to the space they need to maintain the proper visual relationships. It may be more difficult to give a good description of what that relationship should be.

METAFONT also introduces a new feature not available with drawings — program bugs. They result from settings in the program that conflict with the way METAFONT draws. Sometimes the shape is affected, other times not. This is one of the trade-offs of having the ability to consider many possibilities.

Because you have to describe key relationships that are to be maintained, the METAFONT description gives more than a visual relationship. In fact, though related by METAFONT program and concept, the shapes of characters and the typefaces themselves may no longer be related in the traditional sense. METAFONT changes the meaning of a typeface family.

### 3.5 The Pandora Family of Typefaces

Pandora became an example of a flexible typeface family. It might only be the initiated who will recognize this, as the relationship is no longer purely visual. Many parameters or hooks have been built in and some have been used. Some of the parameters have limits placed on them, in the hopes of keeping the results reasonable. However, these too can be altered.

The inspiration for making a fixed-width typeface came out of a desire to print the METAFONT programs in Pandora and have the programs align as they did on the screen. It was a self-referential task — creating programs for characters so that the programs that created the characters could be printed.

The current family consists of 8 styles — serif and sans serif versions of a regular and a bold weight, a fixed pitch style, and slanted versions of each. The character sets match those of Computer Modern for the *typewriter* and *roman* styles. This means that the typefaces can be used interchangeably in terms of characters, though the look of the two families remains quite different.

## 4. Conclusion

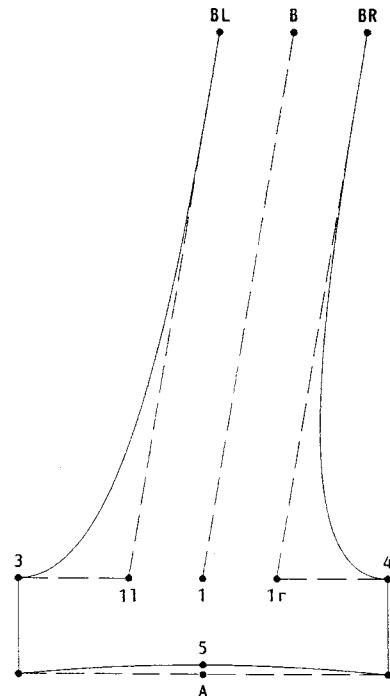
Tools like METAFONT make it very easy to generate and disseminate printable characters. No longer is a lengthy apprenticeship needed in order to have access to the tools that once were among a kingdom's most valued riches. Limited access had the advantage of a certain amount of quality control. The amateur was not likely to be in a position to do much damage, so to speak. On the other hand, accessibility spawns creativity and the sharing of information.

## Bibliography

Billawala, Neenie. *Metamarks: Preliminary Studies for a Pandora's Box of Shapes*. Dept. of Computer Science, Stanford University. Report No. STAN-CS-89-1256.

Knuth, Donald E. "A Course on METAFONT Programming." *TUGboat* 5(2):105–118, 1984.

Knuth, Donald E. *Computer Modern Typefaces. Computers and Typesetting* Vol. E. Reading, Mass.: Addison-Wesley, 1986.



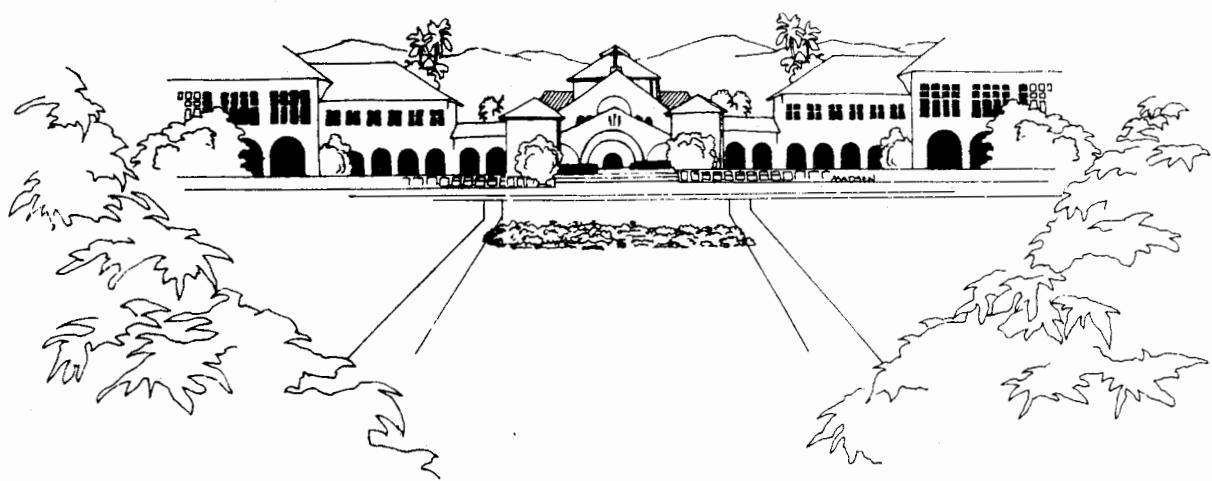
3.12 Serifs are parts of a character that often protrude from the stems of letters such as 'l', 'm', or 'n'. They may be very subtle, showing only as a slight bowing of a vertical stem; or they may be very prominent. The placement of a serif in a character usually has some historical relationship to how the character was drawn or written.

For our purposes, serifs have the following properties: They are always connected to a stem, even if that stem has no thickness. They have a left part and a right part, with both parts combining to form the full serif. They have a horizontal base, and a bracket area that connects the stem to the base. We may also take a small amount out of the bottom of the base — Matthew Carter has suggested that this be called "entasis." Midbracket pull is created by adding a point in the bracketed region through which the serif path must pass. This extra point lies somewhere on the line that protrudes from each bracket area.

Bracketing can also be specified by choosing control points on the base and the stem.

The general serif illustrated here shows the key points constructed by the program on the opposite page.

Figure 8: Serifs: base and stem variations (top); general description (bottom)



AMS-TEX82 Users Course and TEX Users Group Meeting

Stanford University, July 11-15, 1983

Terman Engineering Center Auditorium

# Fractal Images with T<sub>E</sub>X

ALAN HOENIG

Department of Mathematics  
John Jay College of Criminal Justice  
[use mailing address below]:  
17 Bay Avenue  
Huntington, NY 11743

## ABSTRACT

This article is an additional demonstration that T<sub>E</sub>X can create non-textual images. Using halftone fonts created by Knuth, T<sub>E</sub>X can typeset fractal images. The images are somewhat coarse-grained; this is due to the limitations on the number of characters per page that can be typeset by T<sub>E</sub>X.

Sometimes one tires of the printed word, and at such black times, it's nice to know that T<sub>E</sub>X has its non-textual uses as well. Knuth [7] has previously demonstrated the possibility of typesetting halftone images using special halftone fonts. Shortly thereafter, Clark [2] was the first (apparently) to use these fonts and comment in writing on the experience. Their work is fascinating, but the drawback to the casual user of T<sub>E</sub>X who wants to participate in these experiments lies with specialized equipment one needs to generate the halftone data. I will discuss ways of using purely mathematical rules to generate images of beauty and realism.

### 1. Fractals

In the past 20 years or so, mathematicians have discovered new families of objects, shapes of great mystery, beauty, and intrigue. There is mystery and intrigue because, despite the simple mathematics used to generate them, they have lain undiscovered all this time. These shapes have become known as *fractals*. A precise definition of this term is out of place here; see [8]. It suffices to explain, I think, that they involve the disclosure of patterns of extraordinary complexity where one would naïvely expect to encounter chaos or monotony.

We set the stage with a recipe for generating a simple fractal pattern. Let  $z_i$  represent point  $i$  on the complex plane. We use the METAFONT mediation function to denote some other point lying some portion of the way between two given points. If  $z = t[z_0, z_1]$ , then  $z$  is the point lying on the line connecting  $z_0$  to  $z_1$  and located a fraction  $t$  of the way between  $z_0$  and  $z_1$ .

Begin by locating the three corners of a non-degenerate triangle, and by selecting a random point  $z_0$  lying somewhere within the triangle. Draw this point. Now:

1. Select one of the vertices of the triangle at random.<sup>1</sup> Let  $z_v$  be the chosen vertex.
2. Construct the midpoint  $z$  between  $z_0$  and  $z_v$ ; that is,

$$z = 1/2[z_0, z_v]$$

3. Draw this point. Re-label this point  $z_0$ , and return to step 1.
4. Continue this iteration until several tens of thousands of points have been generated and drawn.

What pattern will appear on the page? I am not ashamed that my own intuition "assures" me that a uniform spread of dots will appear. Nothing prepares me for what really appears.

<sup>1</sup> A roll of a single die makes a good random vertex generator. Choose vertex 1 if 1 or 2 show on the die; choose the second vertex if 3 or 4 show on the die; choose the third vertex otherwise.

But before displaying this pattern, let me make a connection between this procedure and  $\text{\TeX}$  or METAFONT. The mathematical procedure that creates the midpoint is a special kind of object called an *affine transformation*. These are precisely the tools within METAFONT for rotating, shifting, stretching, and shrinking graphic objects. As a first attempt, we might try to create a single character using METAFONT to plot lots and lots of points. My success with METAFONT has been limited, largely because great quantities of dots cause METAFONT to exceed its capacity.

$\text{\TeX}$  provides an altogether more satisfactory approach. The idea is to use one of the halftone fonts suggested and demonstrated by Knuth [7]. Here's how to proceed.

We use an external computer program to generate the data. First, we translate the above algorithm into a high-level language (I used QuickBASIC or FORTRAN). The computer program should ask the computer to maintain a large  $2 \times 2$  array, the four corners of which represent the corners of our diagram. Then, each time the computer generates a random point, it must determine which spot in the array best corresponds to the location of the dot on the page. Increment the value of this element in the array by 1. Do this several thousand, or tens of thousands, of times.

When done with this series of iterations, the program will spit this information out to a disk file. Instead of spewing rows of whole numbers, we ask the computer to spread the values in some way between the character **0** and **p**.<sup>2</sup> There are 64 characters lying between **0** and **p** in the ASCII convention, and Knuth's halftone characters occupy these positions. Next, print out these characters to the disk file. This creates an ASCII file which we feed into  $\text{\TeX}$ . Of course, we need to make some adjustments so  $\text{\TeX}$  processes the file properly — suppress interlineskipping and indentation, invoke the halftone font, and some other things. Here is what you see when you typeset this pattern (Fig. 1). Benoit Mandelbrot, the mathematician who is the father of things fractal, calls it the Sierpinski carpet.

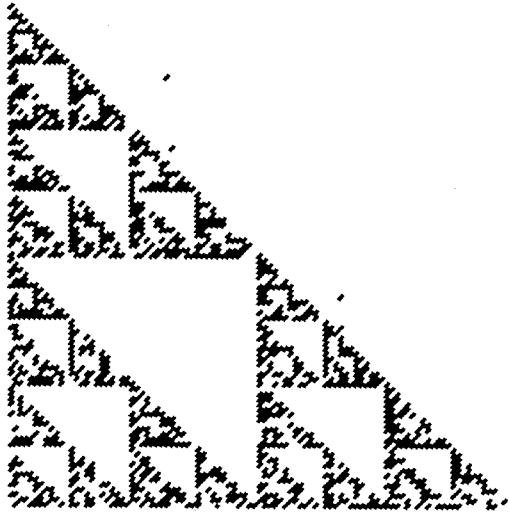


Fig. 1: Typesetting the Sierpinski carpet

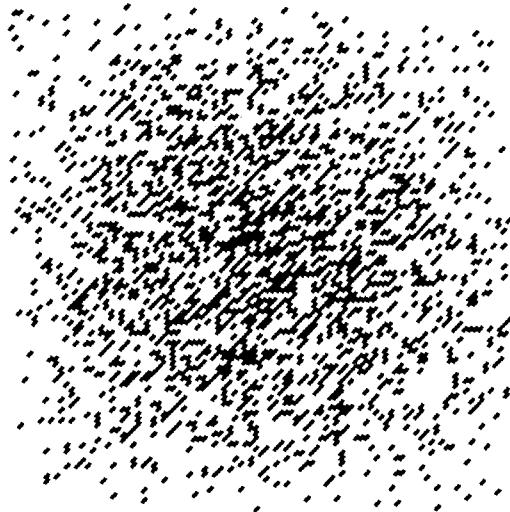


Fig. 2: A 4-cornered carpet: first attempt (unsuccessful)

The coarseness of this carpet's weave is due to at least two factors. There are various upper limits on the number of characters that  $\text{\TeX}$  can typeset on a given page. Remember, each dot in a halftone is a separate character as far as  $\text{\TeX}$  is concerned, and a single image involves far more characters than are usual on any single page of typeset matter. A greater dot density would be more pleasing. Furthermore, to enhance the contrast in these few figures, only two distinct characters were used in the typesetting, **0** if a pixel never contained any point, or **p** otherwise.

---

<sup>2</sup> Because of the peculiarities of human perception, it probably shouldn't be a linear spread; see the helpful figures on page 137 of reference [7] for insight into this problem.

87	22
69	12
22	22
12	12
12	12
22	22
22	22
22	22
22	22

22	12
22	22
22	22
22	22
22	22

22	22
22	22
22	22
22	22
22	22

Fig. 3: A 4-cornered carpet: second attempt (successful)      Fig. 4: A pentagonal Sierpinski carpet

## 2. Fractal Botany

Consider  $n$  transformations, and a set of probabilities  $p_1$  through  $p_n$ , so that  $\sum_{i=1}^n p_i = 1$ . We will create a sequence of points, place them in an array, and then typeset this array as we did the Sierpinski carpets. The sequence begins with a single point, and successive points are found by transforming a predecessor point by one of the  $n$  transforms. Any of these transforms are chosen randomly, the only constraint on this random choice being that in the long run, the  $i^{\text{th}}$  transform should be chosen a fraction  $p_i$  of the time. A deep theorem guarantees that it is possible to choose the transforms and the probabilities so that the resulting pattern resembles real objects. These objects may appear uncannily real, perhaps because these images seem to repeat certain patterns at ever decreasing levels of scale, like Russian *matrushka* dolls. One example is the fern shown in Fig. 5. Some computer scientists find these images exciting; this technique holds promise that the large amounts of information one might think necessary to create images at a convincing level of detail can be easily summarized by some relatively small group of transformations.



Fig. 5: A fractal fern

Fig. 6: A fractal, though stemless, fern

This fern is an outgrowth of the application of four transformations. Each transform  $w_i$  is determined by six constants. Let the six constants for transform  $w_i$  be  $a_i$  through  $f_i$ , and if  $z = (x, y)$ , then the transformed point  $w_i(z)$  is

$$w_i(z) = \begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \end{pmatrix}$$

The following table shows the values for the fractal fern and the associated probabilities for each transform  $w_i$ :<sup>3</sup>

$w$	$a$	$b$	$c$	$d$	$e$	$f$	$p$
1	0	0	0	0.16	0	0	0.01
2	0.85	0.04	-0.04	0.85	0	1.60	0.85
3	0.20	-0.26	0.23	0.22	0	1.60	0.07
4	-0.15	0.28	0.26	0.24	0	0.44	0.07

I was especially intrigued by the first probability,  $p_1 = .01$ . It's hard to imagine what the contribution of  $w_1$  can be to the final image since it is applied so seldom. It is easy to investigate by re-running the program with  $p_1 = 0$  and  $p_2 = .86$  (so the sum of the probabilities continues to equal 1). In the absence of  $w_1$ , we grow a peculiar stemless fern (Fig. 6).

Barnsley [1, p. 87] discusses this material in great detail, and displays some extraordinary examples of the fractalist's art. This book displays several striking plates — subjects include fields of sunflowers, Alaskan huskies, a young Bolivian woman, and scenes from the Black Forest. It would have been nice to use these transforms to typeset pictures, but the data is proprietary.

### 3. Mandelbrot and Julia Sets

There is a final group of fractal images we can generate. If we imagine that our page represents the complex plane, then complex points  $c$  within a so-called *Mandelbrot set* are those which lead to sequences of points which remain bounded. The sequence of points stem from a repeated iteration of a simple non-linear equation:

$$\begin{aligned} z_0 &= 0 \\ z_n &= z_{n-1}^2 + c \end{aligned}$$

that is, the sequence  $0, c, c^2 + c, (c^2 + c)^2 + c$ , and so on. For some values of  $c$ , these sequences eventually generate elements which lie ever farther from the origin of the page. For some other values of  $c$ , no matter how many terms we examine, they remain within commuting distance of the origin. It is these latter points which comprise the Mandelbrot set.<sup>4</sup>

Using the apparatus available for all these projects — auxiliary program to generate data and prepare a *TEX* source file, typesetting using special halftone fonts, and final previewing or printing — leads to some interesting results. The shape of the Mandelbrot set surpasses rational expectation (Fig. 7).

The fuzzy outline of this set is not an artifact of the conversion to *TEX* halftones, and we can prove this by magnifying the image near the boundary of the set. The image fails to exhibit any further smoothness no matter how much the magnification is increased. Increasing levels of magnification reveal ever-increasing levels of complex patterns (Fig. 8). The patterns appear to me to be quite striking, but judge for yourself.

When I began playing with this material, I used *METAFONT* to generate the numerical data. Runs used to take eight hours! I have since learned how to speed up my calculations by a factor of about 24.

Pictures of related objects, *Julia sets*, deserve mention. Recall that in the preparation of the Mandelbrot set, we varied  $c$ . The sequence always began with  $z_0 = 0$ . To generate a Julia set, follow

<sup>3</sup> These values originally appear in [1].

<sup>4</sup> By the way, concepts such as "complex numbers," "sequences," and "distances from the origin" are not complex concepts at all. The reader is urged to seek out references [3]–[5] for an excellent introduction to these concepts and their connection to Mandelbrot and Julia sets.

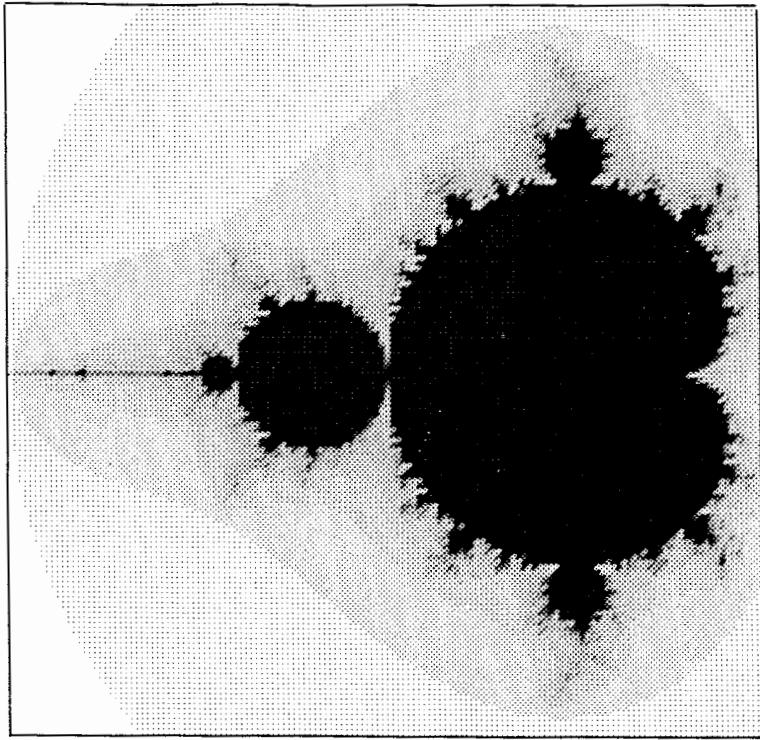


Fig. 7: The Mandelbrot set

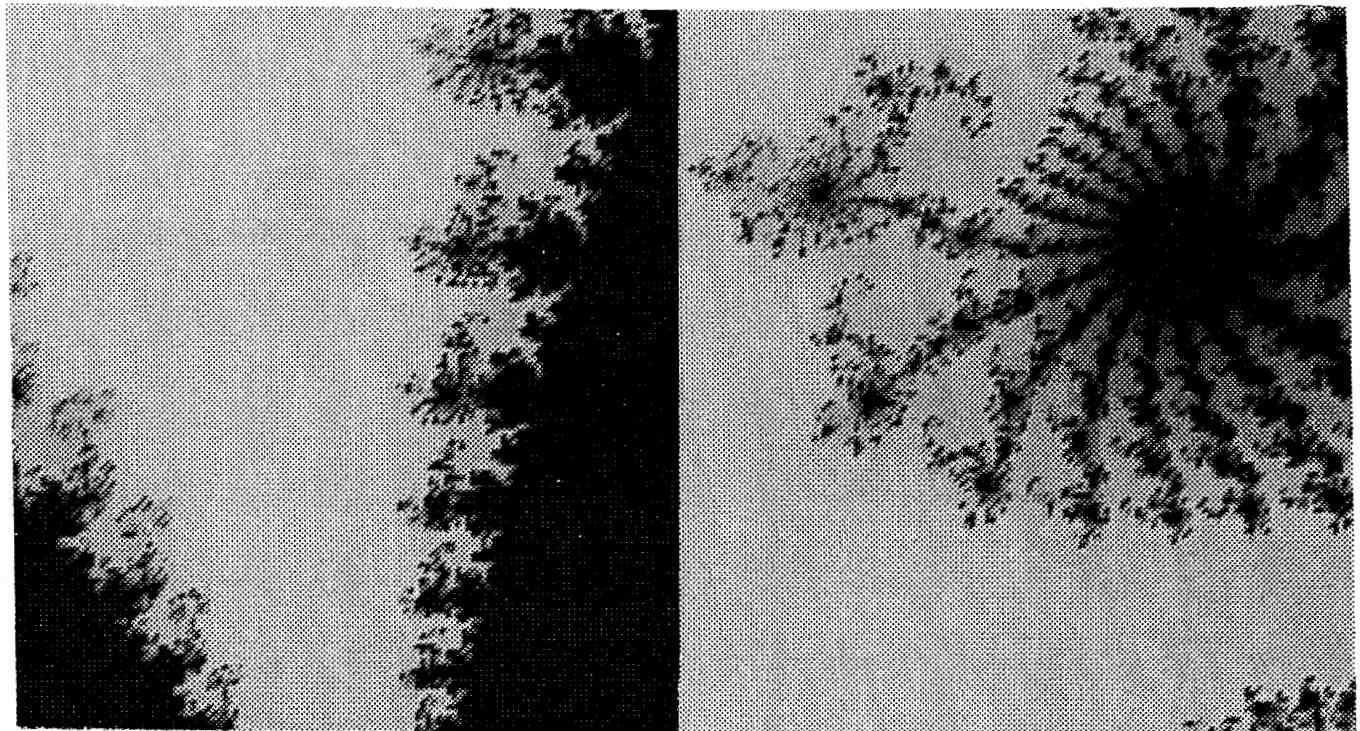


Fig. 8: The Mandelbrot set: several magnifications

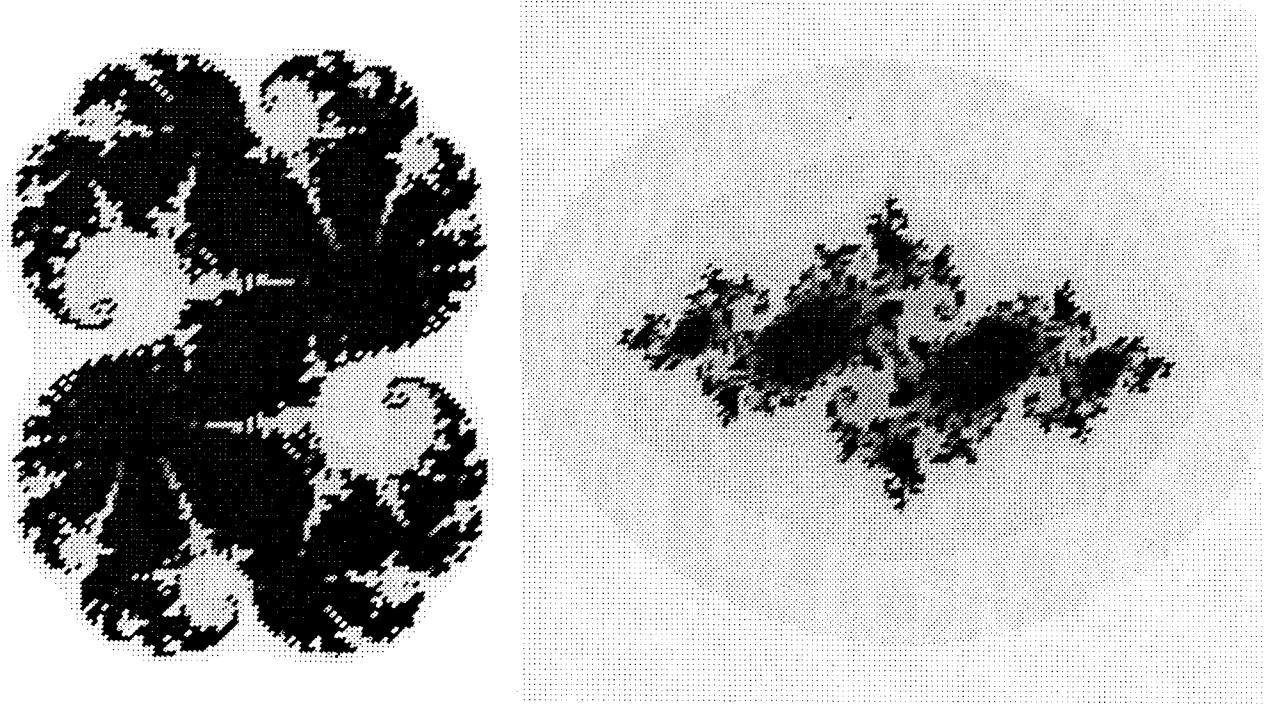


Fig. 9: Julia sets set with  $\text{\TeX}$

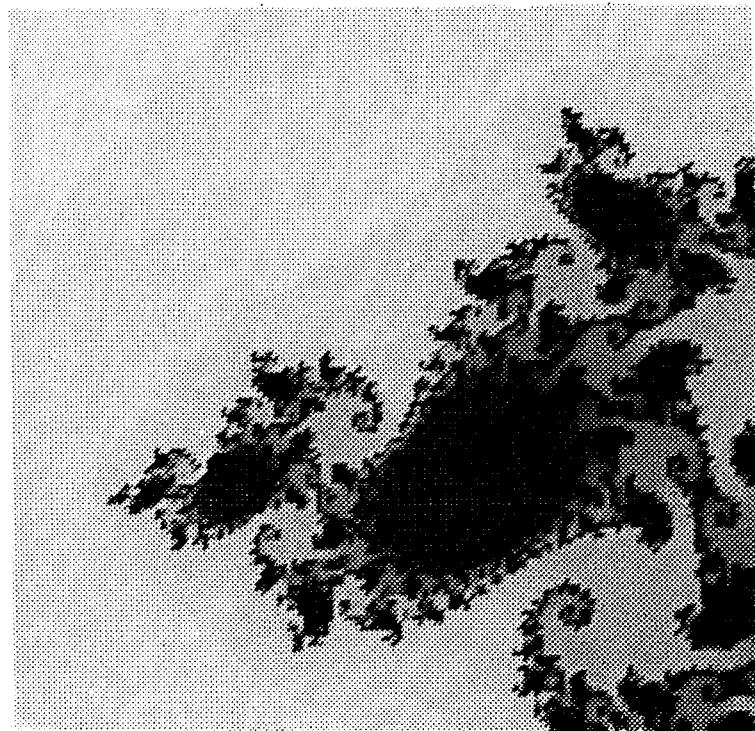


Fig. 10: Detail from a Julia set

the algorithm for the Mandelbrot set with a few modifications. In this case, fix a value of  $c$  and let the choice of  $z_0$  range over the complex plain. The points of the Julia set consist of those  $z_0$  for which the sequence  $z_0, z_1 = z_0^2 + c, z_2 = z_1^2 + c, \dots$  remains bounded. Julia sets display their own good looks (Fig. 9).

As with the Mandelbrot set, magnifying the detail of a Julia set reveals new levels of intricate filigree (Fig. 10).

Of course, there are other kinds of halftone fonts that will suffice when typesetting these pictures. Knuth has suggested several of them [7], and their uses give interesting new looks to these pictures (Fig. 11).

Knuth has observed that if you enjoy “fooling around making pictures” and if you have suitable fonts, then “ $\text{\TeX}$  will be a source of endless frustration/amusement for you, because almost anything is possible . . .” [6, p. 389]. METAFONT is now as widely available as  $\text{\TeX}$ , so it’s easy to create special purpose fonts. I encourage readers to have as much fun making pictures as I have had.

## Bibliography

- [1] Barnsley, Michael. *Fractals Everywhere*. New York: Academic Press, 1988.
- [2] Clark, Adrian F. “Halftone Output from  $\text{\TeX}$ .” *TUGboat* 8:270–274, 1987.
- [3] Dewdney, A.K. “Computer Recreations.” *Scientific American* 253, 2:16–24, August, 1985.
- [4] Dewdney, A.K. “Computer Recreations.” *Scientific American* 257, 5:140–145, November, 1987.
- [5] Dewdney, A.K. “Computer Recreations.” *Scientific American* 260, 2:108–111, February, 1989.
- [6] Knuth, Donald E. *The \TeXbook*. Reading, MA: Addison-Wesley, 1984.
- [7] Knuth, Donald E. “Fonts for Digital Halftones.” *TUGboat* 8:135–160, 1987.
- [8] Mandelbrot, Benoit. *The Fractal Geometry of Nature*. New York: W.H. Freeman and Co., 1983.

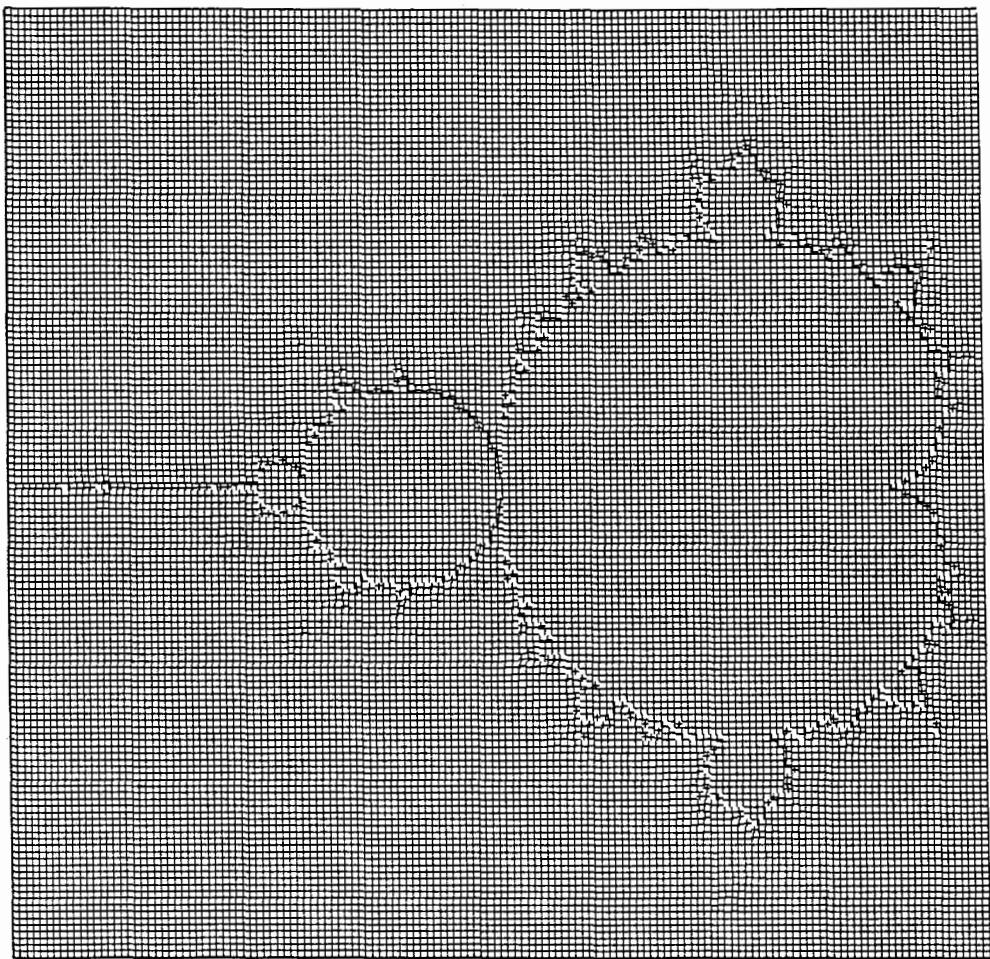


Fig. 11: New looks to the Mandelbrot set

# Design of Oriental Characters with METAFONT

DON HOSEK

Platt Campus Center  
Harvey Mudd College  
Claremont, CA 91711  
`dhosek@cymir`  
`dhosek@jarthur.claremont.edu`

## ABSTRACT

The goals of a meta-font can perhaps be best realized in the design of Oriental characters. These characters, unlike Western alphabets, are composed of a finite number of identifiable components. For example, the Kanji characters used by Japanese are each composed of a number of radicals which are then composed of a set of strokes. Variations in the size and appearance of these elements have a certain degree of regularity to them.

The Quixote Oriental Fonts Project (QOFP)<sup>1</sup> has two goals: the first is to simplify the creation of the large number of characters required by languages such as Chinese and Japanese by making the top-level description of the characters as simple as possible. Ideally the program for a single Kanji character would be composed entirely of names of radicals and mnemonic names for their placement. No coördinates would appear at all.

The second goal is to pave the way for the creation of new families of Oriental typefaces by reducing the required work to that of re-designing the component strokes of a character (and possibly making minor changes to the programs for radicals) rather than requiring the designer to modify thousands of individual character programs.

### 1. Introduction

Before considering the details of QOFP, it is worthwhile to consider the question of why an American with little or no knowledge of *any* Oriental language would take on a project as large as that entailed in the design of character sets for Chinese, Japanese, and Korean (there are a total of nearly twenty thousand characters involved in this endeavor).

The main reason relates to my interest in the philosophy of METAFONT. In Knuth (1982), the sentiment is expressed that ultimately we might hope to find the "essence" of a letter such as A and be able to express all possible typefaces by changing the values of various parameters. However, the long evolution of Western letterforms makes this a difficult process.<sup>2</sup> It seemed to me, however, that the appearance of the Kanji characters used in Oriental typefaces was ideal for implementation in METAFONT. One almost wonders if the originators of the Oriental characters foresaw the creation of a system such as METAFONT.

While it is difficult to define the "essence" of the letter "A", this is not so big a problem with the typical Kanji character. Each character is not defined so much by its total appearance as it is defined by the combined appearance of the strokes that comprise it.

My earliest efforts to create a Kanji font were frightfully bad.<sup>3</sup> For one thing, my lack of knowledge of Chinese or Japanese caused me to make many invalid simplifications of certain characters and to introduce unnecessary complications. After showing some of the early samples to some friends studying Japanese, I received many helpful suggestions. One of these was to define characters in terms of their

<sup>1</sup>The reason for choosing this name should be apparent to anyone who thinks about it.

<sup>2</sup>Modern Western typefaces have a variety of influences in their form stemming from the different manners in which they have been produced over the centuries.

<sup>3</sup>They are now hidden away in archival storage and I refuse to show them to anyone.

radicals rather than their strokes. This has two main advantages: first, it dramatically simplifies the character programs and second, it makes it far easier to correct my mistakes as I go along.

## 2. The Plan of Attack

Rather than rush into the character design on my second try, I spent more time on developing support code for the characters and came up with what might best be described as an “object-oriented metafont.” The main principle behind this technique is to reduce the amount of information any particular portion of the code needs to “know” about the remainder of the system. For example, a typical character program might look like the following:

```
beginjchar(">", "=")(">", "")(UNDEFINED);
  triplet(1, 2, 3, 4, 5, 6);
  nich(1, 2);
  nich(3, 4);
  nich(5, 6);
endjchar;
```

In this character program, not a single explicit coördinate is specified. For that matter, there is not even any need for the program to be aware of the shapes of the radicals used either. The program for  $\text{H}$  is identical to  $\text{H}$  with the sole exception that *nichi* is replaced with *kuchi*. Note also that the dimensions of the character itself are not given, but rather are specified elsewhere.

These two items are one of the primary things setting off QOFP from the work done by Guoan and Hobby in the old METAFONT (see Guoan and Hobby 1984). While they also took the approach of calling subroutines to put strokes and radicals together into characters, they additionally specified all dimensions of the character and internal coördinates explicitly, in this manner limiting the possibilities of the meta-ness of their font.<sup>4</sup>

### 2.1 How Meta- is my Font?

In keeping with my philosophy of object-oriented METAFONT, I decided to determine parameters for the font in a manner similar to that described by John Sauter (1986) for the Computer Modern fonts. A top-level input file would contain only the bare minimum specifications that define that typeface:

```
if unknown qjbase: input qjbase fi                                % Make sure qjbase is present.
font_identifier := "QKJM"; font_size 10pt#;
font_coding_scheme "JIS";
driverfile "qkanji";
input bbqkjm                                         % Generate QKJM at 10pt.
```

Through some fortunate coincidences, the syntax of the opening file has developed some pleasant regularity. Any declaration in which  $:=$  appears is of no importance in the generation of the font while the remainder of the declarations are used in producing the font.

Numerous behind-the-scenes macros are used in QOFP to convert information from the human-readable format in which it is input into something that will be more useful for the system. For example, the `begin_jchar` macro examines the value given by `font_coding_scheme` to determine which pair of character codes given (if any) should be used in determining the final code used. The use of these codes is not simply confined to a basic mapping of the two-byte code to METAFONT `charcode` and `charext`: for example, if the `font_coding_scheme` is set to “*jTeX JIS*”, the code will be automatically re-mapped into the subfont divisions used in *jTeX* (see Saito 1987). This partially explains the use of `default_coding_scheme` in the above example. Since `qkjm10` divided into subfonts is still `qkjm10`, it makes sense to re-use that part of the code with the following file used as a top-level input file:

<sup>4</sup>They are to be forgiven for this, however, since they were among the first pioneers of METAFONT. Knuth does similar things with his character programs in Computer Modern (1986); for example, it is not possible, without changing the character program itself, to create a Century Schoolbook version of the  $\mathcal{E}$  ligature from the CM description of it due to the hard-coding of such things as the height of the crossbar of the “E”.

```
font_coding_scheme "jTeX JIS";
choose_subfont 8;
input qkjm10
```

which will generate the subfont indicated by `\ja` in Saito's *jTeX*. Note that `font_coding_scheme` was modified to not change the coding scheme if one is already in effect (which makes this particular application practicable).

By organizing the top-level input files in this fashion, it should be a fairly simple procedure to generate any font in a given family with minimal effort.

One thing worth noting is the use of "design-sized" fonts in this system. In a letter from Edgar Cooke, I was informed that at present, this practice does not occur in Japanese typography. This is doubtless due to the monumental effort that would be required to do such a task. However, just because this is not a current practice doesn't mean that it shouldn't be done. Only experience will tell whether the ability to have an Oriental font tailored to a specific typesize will indicate whether such a practice is worthwhile.

## 2.2 Choosing Radical Placements

Perhaps the most revolutionary aspect of QOFP is the fact that no coördinates are specified explicitly in the character program. Instead, as demonstrated above, symbolic names are given for the placement of the radicals in the characters. A cursory examination of any Oriental code table will indicate that radicals come together to form characters in a relatively small number of positional combinations. By exploiting this, characters can be easily generated. At present, I am still experimenting with the potentials of this technique, so I am unable to include any examples of the lowest level code involved.<sup>5</sup> The big obstacles in this approach are designing radical programs to be general enough to fit together in the different placement combinations, and prudently choosing the coördinates to be generated for the different placement codes.

### *Korean*

The Hangul alphabet of Korean is of particular interest in this project since Hangul characters are subject to the same sort of regular placement rules as Kanji are, but to an even greater extent. It may be possible to even simplify Hangul character programs to the point where all that needs be specified for any character is the letters which compose it (assuming that the organization of Hangul characters in the Korean character set is reasonably algorithmic). I hope to have some samples of this available for display at the conference in August.

## 3. Plans for the Future

Ultimately, QOFP will result in at least three, possibly more, families of typefaces for the Japanese, Chinese, and Korean national character sets. Once the basic routines are working reliably, creation of a single character can be accomplished in five minutes or less (depending on how familiar the individual inputting the character definition is with the names and use of the METAFONT macros involved). Even after the project is completed, this will be a useful feature since the assorted national character sets omit many thousands of uncommon Kanji from their coding which could possibly be necessary for certain documents. Ideally, the code will be distributed freely, but economic circumstances may make this impractical.

---

<sup>5</sup> A printed copy of all METAFONT code written to date will be available for perusal at the conference.

## Appendix: Oriental T<sub>E</sub>X

QOFP is currently *not* concerned with the problems of special versions of T<sub>E</sub>X for Oriental processing or with solving the Kanji input problem. However, I have collected a few thoughts on the problem of creating a suitable Oriental T<sub>E</sub>X processor and a corresponding environment for dvi output:

- A “big” T<sub>E</sub>X implementation which uses 64-bit words rather than 32-bit words could be used for Oriental processing (the main memory array does not necessarily need to be increased to greater than 64,000 words if this would cause problems in a low-memory environment). If the word size is 64 bits, then the size of a quarterword would then be 16 bits, which would allow for character codes adequately large enough for two-byte character sets.
- A dvi driver for an Oriental language which uses non-printer-resident fonts should only download the characters in the font actually used. It’s a good idea to remember (for all dvi drivers, actually) that the character codes used by T<sub>E</sub>X do not necessarily need to correspond to those used in the printer. For example, the fact that T<sub>E</sub>X accesses some character at character code 255 does not mean that that character must be accessed as character 255 when it is used on the printer (this is especially important for those output devices which have a limit on the size of a character set less than 256).

## Bibliography

- Fenn, C.H. *The Five Thousand Dictionary*. Cambridge, Mass.: Harvard University Press, 1955.
- Guoan, Gu and John Hobby. “A Chinese Meta-Font.” *TUGboat* 5:119–136, 1984.
- Knuth, Donald E. “The Concept of a Meta-Font.” *Visible Language* 16:3–27, 1982.
- Knuth, Donald E. *Computer Modern Typefaces*. Reading, Mass.: Addison-Wesley, 1986.
- Rose-Innes, Arthur. *Beginners’ Dictionary of Chinese-Japanese Characters*. New York: Dover Books, 1977.
- Saito, Yasuki. “Report on jT<sub>E</sub>X: A Japanese T<sub>E</sub>X.” *TUGboat* 8:103–116, 1987.
- Sauter, John. “Building Computer Modern Fonts.” *TUGboat* 7:151–152, 1986.
- Tobin, Georgia K.M. “Designing for Low-Res Devices.” *TUGboat* 9:126–128, 1988.

# Thai Languages and METAFONT

BOB BATZINGER

Bob Batzinger  
UBS Tech Support Center  
Box 116  
Chiang Mai MAI 50000  
Thailand

## ABSTRACT

The Thai languages poses new challenges for typesetting with **TEX**. The high degree of curvature in the script requires special attention to the discreteness of the **METAFONT** generated fonts. Different traditional styles of the Thai font have radically different loops. While diacritical marks of the Thai script could be handled by methods already published, the lack of white space between words requires a new approach. Since many lines are longer than the width of a page, intercharacter glue is required. This special glue must be able to center some diacritical marks over the expandable intercharacter space. Thai typesetting tradition requires line breaks only between words instead of between syllables. This paper will describe the methods and auxiliary programs we have used to adapt **TEX** for use with Thai text.<sup>1</sup>

---

<sup>1</sup> Due to travel difficulties, the final version of this paper will not be included in the Proceedings. However, it will be published in a forthcoming issue of *TUGboat*, as soon as it becomes available -Ed.



**T<sub>E</sub>X Users Group**

**Stanford University, August 13–24, 1984**

**Terman Engineering Center Auditorium and The Graduate School of Business**

# A METAFONT-like System with PostScript Output

JOHN D. HOBBY

AT&T Bell Laboratories  
600 Mountain Avenue  
Murray Hill, NJ 07974  
[hobby@research.att.com](mailto:hobby@research.att.com)

## ABSTRACT

The MetaPost system implements a language very much like METAFONT except that the output is expressed with cubic splines and PostScript commands rather than in METAFONT's raster-oriented Generic Font Format. It is a powerful language for expressing figures for documents printed on PostScript printers, and it can also be used for creating PostScript fonts.

The data types are mostly the same as in METAFONT, except that pictures represent a continuous version of what is scan-converted in order to create METAFONT's pictures. Some raster-oriented METAFONT primitives are removed and primitives for expressing PostScript concepts are added. Facilities are also included for adding text to pictures. This should make it convenient for figures to include labels that match the typography of the rest of the document.

## 1. Introduction

In addition to being a font-making tool, METAFONT is also a powerful graphics language. The only problem is that METAFONT produces raster output which is not very suitable for applications other than font making. This paper describes a tool for applying METAFONT's power as a graphics language to applications where PostScript output is more appropriate.

A less ambitious approach to the problem due to Leslie Carr had some success but ran into practical and theoretical difficulties [2]. Some work by Shimon Yanai avoids many of these problems by using a slightly altered version of METAFONT [7]. The work discussed here includes a number of important features that Yanai does not implement. It should avoid most of the difficulties encountered in earlier work, but one practical problem remains: PostScript character definitions based on METAFONT programs turn out to be rather large. Because of this, we concentrate on turning METAFONT into a system for typesetting graphics analogous to Brian Kernighan's *pic* [3,4].

Even with bitmap output, METAFONT has already found some use as a figure-drawing tool. METAFONT can be used to create a special font that contains one character for each figure in the document, and this font can be used to print all the figures. This works reasonably well for some output devices, but it does require working with characters that may be several inches wide. Another drawback is that it is difficult to create figures that contain text as well as graphics.

This paper describes a variant of METAFONT that is currently under development. The new system is called MetaPost. It processes a language very similar to METAFONT, but it produces PostScript programs instead of a *gf* file. The new output medium allows MetaPost to be used as a figure-drawing tool without dealing with enormous characters. It also facilitates MetaPost commands for integrating text and graphics.

MetaPost produces a sequence of PostScript programs that need to be merged with the rest of the document before being sent to a PostScript printer. If the document is written in TeX, then the dvi-to-PostScript translator should do the merging as indicated by Figure 1. If the TeX document uses downloaded fonts, then the translator must be modified to scan the included PostScript programs and ensure that any required characters get downloaded properly. Of course, the output of MetaPost

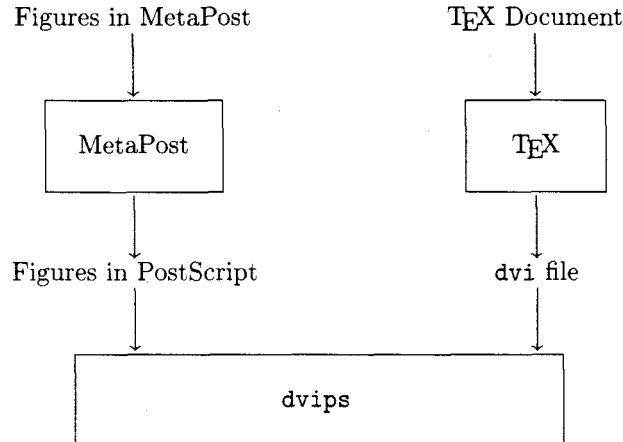


Fig. 1: A diagram of the processing for a *TeX* document with figures in MetaPost

will be designed to make this scanning process as easy as possible.

The description of the language in Section 2 assumes familiarity with *The METAFONTbook* and concentrates on the differences between METAFONT and MetaPost.

Section 3 describes two possible extensions. The first is the use of a preprocessor to allow text included in MetaPost pictures to be coded in *TeX*. The second possible extension is a feature to allow resolution-dependent operations such as those that Knuth discusses in Chapter 24 of *The METAFONTbook* and uses in Computer Modern.

Finally, Section 4 contains a few concluding remarks.

## 2. The Language

Since MetaPost is essentially an altered version of METAFONT, it is easiest to describe it by comparing it to METAFONT. We therefore assume some familiarity with METAFONT and describe the alterations necessary to deal with continuous rather than discrete output.

One important difference is that while METAFONT primitives work in units of pixels, MetaPost uses points, as *TeX* does. Since MetaPost is intended to be used with a macro package analogous to *plain.mf*, some of the differences in the primitives can be shielded from the user by suitable adjustments to the standard macro package.

### 2.1 Pens

Of METAFONT's eight data types, boolean, numeric, pair, path, string, and transform have no relation to the discrete raster and can be used in MetaPost as they are in METAFONT. The other two types are pen and picture. They both describe concepts that are useful in MetaPost, but their meaning is altered to eliminate their discrete flavor.

In the case of pens, this means that it is not appropriate for MetaPost to convert elliptical pens into polygons as METAFONT does [5, pp. 148–149].<sup>1</sup> A better strategy for MetaPost is to treat elliptical pens as ideal ellipses so that appropriate PostScript commands can be given when drawing with them. For example, in the case of a circular pen,

```
draw quartercircle scaled 200 withpen pencircle scaled 10
```

might be rendered in PostScript (Fig. 2) as

```

newpath
0 0 100 0 90 arc
10 setlinewidth 1 setlinecap stroke

```

---

<sup>1</sup> Ideally, the PostScript interpreter should do this.

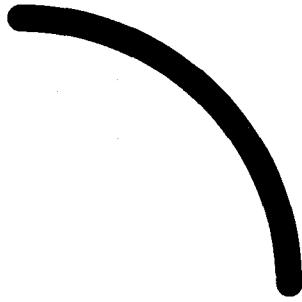


Fig. 2: The result of a `draw` command with `pencircle scaled 10`

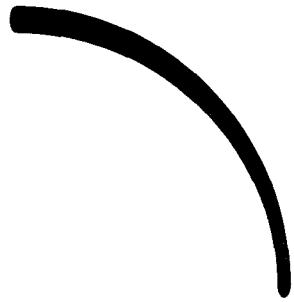


Fig. 3: The result of a `draw` command with `pencircle xscaled 5 yscaled 10`

Using an elliptical pen of a more general shape,

```
draw quartercircle scaled 200 withpen pencircle xscaled 5 yscaled 10
```

can be rendered in PostScript (Fig. 3) as:

```
newpath
0 0 100 0 90 arc
1 2 scale
5 setlinewidth 1 setlinecap stroke
```

METAFONT also allows polygonal pens to be constructed via the `makepen` operator or with the `pensquare` macro from `plain.mf`. MetaPost needs to treat such pens as polygons and implement them via PostScript's `fill` operator. For example,

```
draw quartercircle scaled 200 withpen pensquare scaled 10
```

might be rendered in PostScript (Fig. 4) as

```
newpath
105 -5 moveto
5 5 100 0 90 arc
-5 105 lineto
-5 -5 100 90 0 arcn
closepath fill
```

As explained in [6, part 24], METAFONT already implements drawing with polygonal pens by using the outline representation. MetaPost does the same thing except it turns the outline into PostScript commands instead of using it to control scan conversion routines.

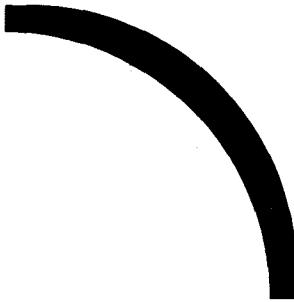


Fig. 4: The result of a `draw` command with `pensquare scaled 10`

## 2.2 Pictures

The picture data type loses its discrete flavor in MetaPost, and this affects the set of operations allowed on pictures. MetaPost pictures function mainly as repositories for the results of `draw` commands and differ from METAFONT pictures in that they do not get digitized until after they have been translated into PostScript and read by the PostScript interpreter. Thus MetaPost does not have METAFONT's restriction on the types of transformations that can be applied to pictures (see [5, pp. 117, 144]). Another consequence of the non-discrete nature of MetaPost pictures is the loss of the `totalweight` operator.

One of the more fundamental differences between METAFONT pictures and MetaPost pictures is in how pixel values are interpreted. In a METAFONT picture, each pixel has a numeric weight, `draw` commands add to the weights, and pixels with positive weights are considered to be black. In contrast, MetaPost adopts PostScript's view of the world where pixels have colors and drawing operators assign new values to the colors of affected pixels. Thus while the `unfill` macro from `plain.mf` is implemented with the `cull` command, the analogous operation in MetaPost is accomplished by filling `withcolor white`.<sup>2</sup>

An important feature of MetaPost pictures is that they can contain textual labels. The text can come from any font for which there is a `tfm` file available, but the PostScript version of the picture assumes that the font is known to the printer and is scaled to match the design size in the `tfm` file. If the picture is to be included in a `TEX` document, then support code is needed in the dvi-to-PostScript translator in order to ensure that these conditions are satisfied.

The syntax for generating textual labels involves a single operator `infont`:

```
<picture secondary> —> <string primary> infont <string primary>
```

This produces a picture that contains the given string set in the given font at the design size specified in the `tfm` file with the reference point of the first character at  $(0, 0)$ . The result can then be transformed as desired and added to a picture variable, as in:

```
addto currentpicture also "label" infont "cmr10" scaled 1.2 shifted (100,100)
```

Operators for finding the bounding box of a textual label facilitate placing the label and ensuring that there is enough space for it. This information is available to MetaPost from reading the `tfm` file and can be accessed via the following operators:

```
<pair primary> —> llcorner <picture primary>  
| ulcorner <picture primary>  
| lrcorner <picture primary>  
| urcorner <picture primary>
```

These operators find the bounding box of any picture whether or not it is the result of `infont`. This includes all elements of the picture, even those filled `withcolor white`.

---

<sup>2</sup> METAFONT's `cull` command is not implemented in MetaPost because it deals with pixel weights and therefore does not fit into PostScript's view of the world.

### 2.3 Drawing Commands

Drawing commands in MetaPost must differ from those in METAFONT in order to take advantage of the features of PostScript, but the overall syntax is similar except for the new features. Thus MetaPost has METAFONT's primitive commands

```
addto <picture variable> contour <path expression> <with list>
addto <picture variable> doublepath <path expression> <with list>
```

and the corresponding macros

```
fill <path expression> <with list>
draw <path expression> <with list>
```

where a <with list> can be empty or contain various kinds of clauses. New kinds of clauses that provide access to PostScript features are explained below.

MetaPost interprets **addto** commands and **fill** and **draw** macros as assigning a new color to some region of a picture, and this region is determined according to PostScript's non-zero winding number rule [1, Section 4.6]. Since the **safefill** macro defined in *The METAFONTbook* uses the same rule, MetaPost behaves roughly as METAFONT would if all calls to the **fill** macros were replaced by calls to **safefill** (see [5, p. 121]).

If the <with list> contains a

```
withpen <pen expression>
```

clause, then the affected region is enlarged to include everything swept out by the pen.

The color to be assigned to the affected region is given by a

```
withcolor <color expression>
```

clause. For this purpose, there is a three-component numeric type **color** normally accessed via pre-defined constants **white**, **black**, **red**, **green**, and **blue**. Colors can be added together, multiplied by numeric constants or used in mediation expressions. For example:

```
0.3black + 0.7white and 0.7[black,white]
```

are the same shade of gray. Of course **red**, **green**, and **blue** will seldom be used as long as most PostScript printers can only handle black and white or use halftoning to render shades of gray.

Another important feature of PostScript is the ability to draw dashed lines. This is accessed by giving a

```
dashed <picture expression>
```

clause in a <with list> and using the picture expression to specify the pattern of dashes desired. For example, the picture

```
begingroup clearit;
  draw (0,0)..(6,0);
  draw (14,0)..(20,0);
  currrentpicture endgroup
```

illustrated in Figure 5a may be used in a **dashed** clause to specify the dash pattern in Figure 5b, created by laying copies of Figure 5a end to end.

— —

Fig. 5a: A dash pattern

— — — — —

Fig. 5b: A line created five from copies of it

This dash pattern is specified by the PostScript command

```
[12 8] 6 setdash
```

In general, **dashed**  $\langle$ picture expression $\rangle$  means that the pattern of dashes is what would be produced by laying copies of  $\langle$ picture expression $\rangle$  end to end. This ignores features of  $\langle$ picture expression $\rangle$  such as color and line width since the only purpose is to tell MetaPost what argument to give to PostScript's **setdash** operator.

To provide access to PostScript's **clip** operator, there is a primitive drawing command

```
clip <picture variable> to <path expression>
```

and a macro

```
clipto <path expression>
```

equivalent to

```
clip currentpicture to <path expression>
```

The region affected by the **clip** command is determined by the usual non-zero winding number rule, but instead of being filled with black or some other color, the clipping region is treated as a window, and all parts of the picture falling outside of the window are removed.

### 3. Possible Extensions

The MetaPost language described in Section 2 is basically a version of METAFONT with raster-oriented features removed and features added to provide access to PostScript primitives. A heavy reliance on Knuth's public-domain code for METAFONT should make this project manageable. This section describes other features that might not be included in the initial version of the language because either they are difficult to implement or they require a substantial amount of external software.

#### 3.1 TEX Text in Pictures

Section 2.2 described an **infont** operator that could be used to add textual labels to pictures. While this is fine for simple applications, it makes no provision for mathematical typesetting or even interword spaces. Thus words need to be positioned individually, and math formulas require piecing together many characters from different fonts in a complicated fashion.

These difficulties could be avoided by having a preprocessor that reads a MetaPost input file with TEX commands interspersed, and outputs an identical file with the TEX commands replaced by sequences of **addto** commands involving **infont** operations. This would produce an ordinary MetaPost input file that could then be processed in the usual way.

The preprocessor would work by running the interspersed commands through TEX and then extracting the character placement information from the **dvi** file. To find the picture expression corresponding to the TEX commands

```
$\displaystyle{\sqrt{3a+b \over ac}}$
```

the preprocessor could use TEX to obtain a **dvi** file containing this equation on a page by itself. The preprocessor could then scan the **dvi** file to find the coordinates of each character and each horizontal or vertical rule on the page. It is then a simple matter for the preprocessor to create an appropriate MetaPost picture expression. In the above example, this leads to the following picture expression:

```
def addalso = addto currentpicture also enddef;
begingroup
  save currentpicture;
  picture currentpicture;
  clearit;
  addalso "r" infont "cmex10" shifted (0,15.96);
  fill unitsquare xscaled 29.2 yscaled 0.4 shifted (10,15.96);
  addalso "3" infont "cmr10" shifted (11.2,6.77);
  addalso "a" infont "cmmi10" shifted (16.2,6.77);
  addalso "+" infont "cmr10" shifted (23.71,6.77);
  addalso "b" infont "cmmi10" shifted (33.71,6.77);
  fill unitsquare xscaled 26.8 yscaled 0.4 shifted (11.2,2.61);
```

```

addalso "ac" infont "cmmi10" shifted (19.79,-6.86);
currentpicture
endgroup

```

This MetaPost picture contains all the components of the formula exactly as TeX would typeset them, except that the horizontal rules have been replaced by calls to the `fill` macro. If the required fonts are available, the formula should look as though typeset by TeX, except that it may be difficult to ensure that the rounding to pixel units is done according to the rules found in `dvitype`.

### 3.2 Pixel Rounding

MetaPost has not been designed to deal with discrete pixels because its PostScript output is continuous in nature. However, the PostScript interpreter does produce pixel output and it may need help if it is to do a good job. In order to provide this help, a PostScript program occasionally needs to transform coordinates into what the PostScript manuals call *device space* to facilitate pixel-oriented rounding operations.

Techniques for coping with discreteness are discussed in Chapter 24 of *The METAFONTbook*. Since many of them are specific to font making, they are most applicable when MetaPost is being used to create a PostScript font. As Leslie Carr concluded in [2], this seems to be impractical at present because of the large size of a PostScript font description. However, this problem might be alleviated by future advances in PostScript interpreters combined with techniques for simplifying the character descriptions by degrading the outlines slightly.

Regardless of the application, it is worth considering how to implement something like the `round` macro from `plain.mf`. When applied to a pair, the `round` macro finds the nearest pixel corner. The following PostScript commands perform the same operation on a coordinate pair at the top of the operand stack:

```
itransform round exch round exch transform
```

The problem in using this is that MetaPost would not know the value of the pair computed by the `round` macro. All it would have would be a string of PostScript commands for computing it. Thus all subsequent operations involving the pair would also have to be maintained as sequences of PostScript commands. For instance, the result of

```
round(3,2) + (4,1)
```

might be the following PostScript commands:

```
3 2 itransform round exch round exch transform
1 add exch 4 add exch
```

Note that the coordinates (3,2) and (4,1) are in units of points, not pixels.

Not all operations on a pair such as `round(3,2)` can easily be done in PostScript. For example, in a path expression such as

```
round(3,2){up}..(7,3)
```

MetaPost computes control points according to the rules on pages 130–132 of *The METAFONTbook*. Rather than attempt to do this with PostScript commands, it seems more natural to start by converting

```
(3,2){up}..(7,3)
```

into

```
(3,2) .. controls (3,4.2122) and (5.95897,4.95193) .. (7,3)
```

as usual. It is then a relatively simple matter to give PostScript commands that compute control points near (3,4.2122) and (5.95897,4.95193) based on the assumption that the curve begins at the rounded version of (3,2).

Generalizing from this example, it would be nice to be able to specify a PostScript procedure and a MetaPost macro that computes an approximation. For the above example, the PostScript procedure is

```
itransform round exch round exch transform
```

and the MetaPost macro is

```
def round primary p = p enddef
```

Although in this case, the macro operates on a pair and returns a pair, the basic idea works for other combinations of types. The only requirement is that the arguments and results be of types that can be manipulated in PostScript. There is no formal requirement about how well the result of the MetaPost macro should approximate the result of the PostScript procedure. It just needs to be good enough to achieve reasonable results when used for things like path construction. Anything comparable to the half-pixel accuracy in the above example should be sufficient.

It remains to be tested in practice, but it seems very promising to be able to delay some computations until the PostScript interpreter is run on MetaPost's output. It would certainly be very flexible and would impose no hard and fast limitations on the techniques to be used to cope with the discreteness of the pixel grid.

#### 4. Conclusion

We have outlined a language based on METAFONT with modifications to make it more suited to PostScript output. Except for the extensions discussed in Section 3, the implementation should be a relatively straightforward modification to the public domain code for METAFONT.

The language is designed for generating graphics such as figures for technical papers, but it could be used for almost any application involving graphics and the generation of PostScript programs. Because the language is so much like METAFONT at the user level, it is tempting to consider taking existing METAFONT programs for a typeface such as Computer Modern and rewriting some of the macros so that the programs could be processed by MetaPost.

#### Bibliography

- [1] Adobe Systems Incorporated. *PostScript Language Reference Manual*. Reading, Mass.: Addison-Wesley, 1986.
- [2] Carr, Leslie. "Of METAFONT and PostScript." | *TeXniques* 5:141–152, 1988.
- [3] Kernighan, Brian W. "PIC—A Language for Typesetting Graphics." *Software Practice and Experience* 12(1):1–21, 1982.
- [4] Kernighan, Brian W. *PIC—A Graphics Language for Typesetting*. *Computing Science Technical Report* 116. Murray Hill, New Jersey: AT&T Bell Laboratories, 1984.
- [5] Knuth, Donald E. *The METAFONTbook*. Reading, Mass.: Addison-Wesley, 1984.
- [6] Knuth, Donald E. *METAFONT: The Program*. *Computers and Typesetting*, Vol. D. Reading, Mass.: Addison-Wesley, 1986.
- [7] Yanai, Shimon. *Environment for Translating METAFONT to PostScript*. M.Sc. Thesis. Faculty of Computer Science, Technion: Haifa, Israel, 1989.

# Migration from Computer Modern Fonts to Times Fonts

R.E. YOUNGEN, W.B. WOOLF, AND D.C. LATTERMNER

American Mathematical Society  
201 Charles St.  
Providence, RI 02904  
[rey@math.ams.com](mailto:rey@math.ams.com)

*Mathematical Reviews*  
P.O. Box 8604  
Ann Arbor, MI 48107-8604  
[wbw@math.ams.com](mailto:wbw@math.ams.com)  
[dcl@math.ams.com](mailto:dcl@math.ams.com)

## ABSTRACT

After several years of publishing in  $\text{\TeX}$  and (Almost-)Computer Modern, the American Mathematical Society has switched to  $\text{\TeX}$  and the Autologic Times family of fonts. This paper discusses the steps taken by the AMS to access Autologic's proprietary typefaces through  $\text{\TeX}$  and some of the difficulties encountered, particularly the problems associated with spacing in the math italic font. Although the procedures described are specific to the Autologic fonts, the principles outlined are applicable to other non-METAFONT fonts on other typesetters or printers.

### 1. Introduction and Background

In the early 1980s, the American Mathematical Society (AMS) and Mathematical Reviews (MR) (a division of the AMS which produces the secondary journal *Mathematical Reviews*<sup>1</sup>) were using a proprietary computer-driven typesetting system called STI (from Science Typographers Inc.) to typeset their publications, utilizing the Times fonts available on STI's Harris Fototronic typesetters. Around 1983, the AMS began using  $\text{\TeX}$  and the Almost-Computer Modern (AM) typeface for a small portion of regular journal production. For several years papers typeset with  $\text{\TeX}$  in AM fonts and those typeset with STI were incorporated into the same journals. This was true of the MR journal and *Current Mathematical Publications* (CMP) — MR's early awareness journal — as well: several of the indexes to these publications were set in  $\text{\TeX}$  while the main body was set in STI.

From 1983 to 1987, the percentage of papers typeset in  $\text{\TeX}$  at the AMS gradually increased, to the point that by mid-1987, STI had been completely phased out and 100% of the AMS's books and journals were being typeset in  $\text{\TeX}$ . Starting with the January 1985 issue of MR and Vol. 17, no. 1 of CMP, MR switched to  $\text{\TeX}$  exclusively. A history of MR and the reasons for that switch to  $\text{\TeX}$  (including the resulting efficiencies in multiple output formats from the common database and in user control of the journal design) are described in another paper presented to this conference.<sup>2</sup>

Although the AMS was a strong supporter of the  $\text{\TeX}$  project from its inception, a major drawback in using  $\text{\TeX}$  in production was the severe limitation on choice of typeface (only AM, and later CM, were available). There remained in the minds of several managers of MR and the AMS the concern that the AM fonts did not provide quite the right match of aesthetic qualities with efficient space utilization.

<sup>1</sup> A secondary journal carries abstracts and/or reviews (with bibliographic information) of articles published in primary journals and collections, as well as of monographs.

<sup>2</sup> "TeX at *Mathematical Reviews*" by Latterner and Woolf.

This concern (coupled perhaps with a bit of nostalgia) became the impetus for the eventual return to publishing in Times.

## 2. The Switch Back to Times

When the AMS purchased an Autologic APS-μ5 typesetting machine in the spring of 1987, the entire family of Autologic Times fonts was acquired. Negotiations with Autologic led to an agreement allowing the AMS to obtain character widths for the Autologic fonts, as well as raster versions of the fonts, from which low-resolution fonts could be created.<sup>3</sup> Using the character width information, `tfm` files could be created for use with `TEX`, and the low-resolution versions of the Autologic fonts could be converted to a standard `METAFONT` output format (`pxl` format) for use on the laser proofing devices in the AMS headquarters office in Providence, Rhode Island, and the MR office in Ann Arbor, Michigan.

Coincidentally, MR was reviewing the typographic design of the MR journal, with Richard F. Southall serving as a consultant. Fortunately, Southall was at home in both the `TEX/METAFONT` environment and in the typographic considerations implicit in adapting the Autologic Times family to being driven by `TEX`. The team assembled for the conversion was led by Southall; major contributions were made by Ron Whitney of the AMS Providence staff, Dan Latterner of the Ann Arbor (MR) office, and (later) by Ralph Youngen of the Providence staff.

## 3. Creation of Times Text Fonts

A number of problems had to be overcome in order to create Autologic Times fonts for use with `TEX`: (1) Autologic fonts did not contain all the characters found in their Computer Modern counterparts; (2) characters within an Autologic font were ordered differently than `TEX` fonts; (3) the Autologic fonts needed to be “tuned” for use with `TEX` (i.e., kerning information had to be provided to `TEX`).

To overcome these problems, a utility called `APStoPXL`, written by ArborText of Ann Arbor, Michigan was used. `APStoPXL` allows for the creation of “composite” or “virtual” fonts, which are fonts that are composed of characters from several different fonts. This composite font scheme solved each of the problems listed above by: (1) allowing characters to be imported from other Autologic fonts or Computer Modern fonts; (2) providing a method to establish the correct correspondence between Autologic character positions and `TEX` character positions; and (3) providing a mechanism for kerning the Autologic fonts when used with `TEX`.

`APStoPXL` calls for the creation of a `map` file for each composite font. Each line in the `map` file contains information for a single character in the composite font. If a character is to be imported from another font, the name of the `pxl` file containing that character is given. The composite fonts created at the AMS were primarily composed of characters from one of the Autologic Times fonts, but characters were also imported from other fonts (Autologic and Computer Modern) to fill in the gaps. Autologic Times Roman, for example, does not contain the uppercase Greek characters found in `cmr10`, so these had to be imported from the Autologic Times Greek. Other characters, such as some of the accents and the dotless `j`, were brought in from `cmr10`. Ordering of the characters in the font was accomplished simply by listing in the `map` file the Autologic or imported characters in the order expected by `TEX`. Fine-tuning the inter-character spacing for the Autologic Times fonts required major efforts. The procedure used was essentially an iterative one in which the left and right sidebearings (the amount of space to the left and right edges of the “bounding box” for a character) were increased a small amount at a time and tested with all character combinations until suitable values were found. Although this process was somewhat automated, it was nonetheless time-consuming, since every combination of characters had to be viewed by human eyes.

`APStoPXL` takes as input the `map` file for the font and an Autologic Edscan-format file (which contains raster images of the Autologic fonts) and produces a standard `METAFONT` `pxl` file, as well as a `pl` file and an `xpl` file. The `pxl` file can be converted to a `gf` or `pk` format file for use with a standard laser proofing device. The `pl` (Property List) file can be run through the standard `TEX` utility `PLtoTF` to produce a `tfm` file. Finally, the `xpl` file (eXtended Property List) is run through another ArborText

---

<sup>3</sup> Unfortunately, to the best of our knowledge, the AMS is the only site that has been able to obtain raster versions of Autologic fonts.

utility called PLFONT to produce an **amf** file (Autologic Metrics File). The **amf** file is similar to a **tfm** file except that it contains device-specific information for every character in the font. This **amf** file is read by the device driver, DVIAPS; its purpose is to assign **TeX** character calls in the **dvi** file to the correct font and character position on the Autologic typesetter. Figure 1 illustrates this entire procedure.

The key to this system lies in the differences between the **pl** file and the **xpl** file. A character entry in the **xpl** file may contain any of the following device-specific parameters which control exactly how that character is to be typeset:

Parameter	Meaning
DEVWD	Specifies the width of the character.
DEVMAG	Adjusts the horizontal placement of the character.
DEVYOFFSET	Adjusts the vertical placement of the character.
DEVFONT	Specifies the font in which the character is to be found.
DEVCHAR	Specifies the character position in the DEVFONT.
DEVXWID	Specifies the amount to magnify the character horizontally.
DEVYHGT	Specifies the amount to magnify the character vertically.

The following excerpts from the **pl** file and the **xpl** file for the AMS's Autologic Times Math Italic font show the addition of device-specific information to the **xpl** file:

<pre>(CHARACTER O 17   (CHARWD R 0.46273)   (CHARHT R 0.495769)   (CHARDP R 0.021397)   (CHARIC R 0.075)   (COMMENT     (KRN O 177 R 0.055555)   ) ) (CHARACTER O 20   (CHARWD R 0.5)   (CHARHT R 0.745255)   (CHARDP R 0.18)   (CHARIC R 0.055)   (COMMENT     (KRN O 177 R 0.083333)   ) )</pre>	<pre>(CHARACTER O 17   (CHARWD R 0.4627304)   (CHARHT R 0.4957685)   (CHARDP R 0.0213966)   (CHARIC R 0.0749998)   (COMMENT     (KRN O 177 R 0.0555553)   ) ) (DEVFONT D 28801) (DEVCHAR D 16) (DEVWD R 0.4674005) (DEVXMAG R 1.1400003) (DEVY MAG R 1.1400003) ) (CHARACTER O 20   (CHARWD R 0.5)   (CHARHT R 0.7452554)   (CHARDP R 0.1800003)   (CHARIC R 0.0550003)   (COMMENT     (KRN O 177 R 0.0833330)   ) ) (DEVFONT D 14201) (DEVCHAR D 32) (DEVWD R 0.5100002) (DEVXOFFSET R -0.0100002) )</pre>
pl sample	xpl sample

METAFONT font designers have the ability to adjust the placement of each character within the **tfm** boundary. When typesetting with a manufacturer's native fonts, however, the placement of the character within its bounding box is not normally adjustable. However, using the parameters described above it is possible to attain complete control over how characters are to be typeset in relation to one another.

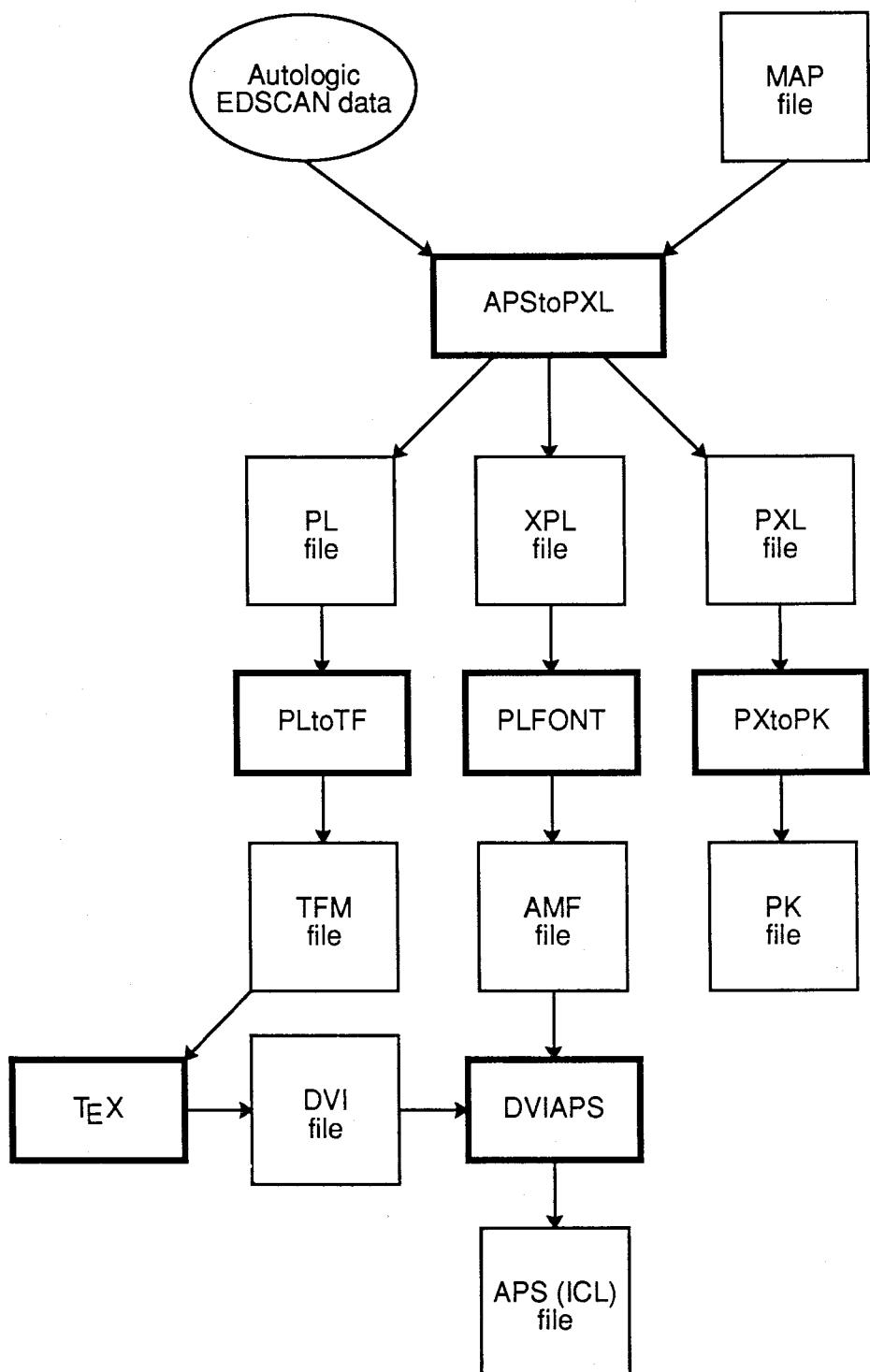


Figure 1: Overview of creation of Autologic fonts for use with  $\text{\TeX}$

## 4. Creation of a Times Math Italic Font

The AMS and MR had two options for a math italic font to use with the Autologic Times text fonts: simply use the Computer Modern math italic (`cmmi10`) or create a new Times math italic. Because of the significant differences in the style and weights of CM and Autologic Times, as well as the fact that Times had a greater x-height than CM, the creation of a Times math italic font was essential. At the same time, however, the difficulties of creating a math italic font were clear. The proper tests to run and parameters to change in the process of creating a math italic font were not as straightforward as they were for the creation of the text fonts, since `TeX` follows a different set of rules in math mode than it does in text mode. Italic correction, in particular, plays a very different role in a math italic font, since italic correction is added between adjacent characters from a math italic font.<sup>4</sup> In addition, the interaction of a math italic font with characters from the variety of different fonts that `TeX` processes in math mode (such as symbol, extension, and roman) has to be taken into account.

### 4.1 Spacing of Adjacent Math Italic Characters

To illustrate the difficulties involved, suppose that the spacing of the math italic character combination  $xy$  needs to be increased. There are four different ways that this could be accomplished: (1) increase the right sidebearing on the  $x$ , (2) increase the left sidebearing on the  $y$ , (3) insert a kern into the ligtable for the  $xy$  pair, or (4) increase the italic correction on the  $x$ . Note that all but option (4) would also apply for normal text fonts.

To determine which of these options to use, it is necessary first to decide whether the space should be added to the  $x$  or the  $y$  on a global basis, or whether this particular character combination is a special case that should be handled with a kern. For example, if other character combinations with the  $x$  also look too close, then space should probably be added to the right of the  $x$ . However, either option (1) or option (4) above would accomplish this in a math italic font.

To choose between these two options, it is necessary to look at how superscripts interact with the  $x$  character. This is because italic correction is added before a superscript is placed on a math italic character in math mode. If superscript placement on the whole looks good with  $x$ , then option (1) — increasing the right sidebearing on the  $x$  — ought to be done. If, however, superscripts are also too tight, then using option (4) — increasing the italic correction on the  $x$  — would solve both problems.

### 4.2 Spacing between a Math Italic Character and a Character from Another Font

Aside from math italic, `TeX` frequently uses characters from the symbol fonts and the math extension fonts when constructing math formulas. When the AMS and MR decided to create a Times math italic font, it was also decided that the symbol font `cmsy` and the math extension font `cmex` would be used in conjunction with the Times math italic. While the difference in x-height and weight between the Computer Modern characters from these two fonts and those of the Autologic Times fonts was still apparent, it was not considered to be as much a problem, since the symbol and extension characters are already quite different from text characters. In practical terms, there was actually no other choice: Autologic did not provide a set of math characters that covered the range of mathematical characters required to typeset mathematics. The decision to use `cmsy` and `cmex`, however, meant that if spacing problems arose between a Times math italic character and a character from one of these fonts, the Times math italic character had to be changed, and not the `cm` character.

Since kerns cannot be used between characters of different fonts, this means that the only options available are to change either the right sidebearing or the italic correction of the math italic character.<sup>5</sup>

Figure 2 shows a comparison of mathematics typeset using all Computer Modern fonts, versus mathematics typeset with the Computer Modern symbol and extension fonts and Times math italic.

<sup>4</sup> Note that if a kern is also specified between two characters in a math italic font, then both the kern and the italic correction are added between adjacent characters.

<sup>5</sup> Italic correction will still work since it is also applied when `TeX` switches from math italic to another font.

$$x \times y \quad A \subseteq B \quad t \rightarrow \infty$$

$$\int_0^\infty \frac{t - ib}{t^2 + b^2} e^{iat} dt = e^{ab} E_1(ab), \quad a, b > 0.$$

$$\pi(n) = \sum_{m=2}^n \left[ \left( \sum_{k=1}^{m-1} \lfloor (m/k)/\lceil m/k \rceil \rfloor \right)^{-1} \right]$$

Figure 2a: Math output with CM fonts

$$x \times y \quad A \subseteq B \quad t \rightarrow \infty$$

$$\int_0^\infty \frac{t - ib}{t^2 + b^2} e^{iat} dt = e^{ab} E_1(ab), \quad a, b > 0.$$

$$\pi(n) = \sum_{m=2}^n \left[ \left( \sum_{k=1}^{m-1} \lfloor (m/k)/\lceil m/k \rceil \rfloor \right)^{-1} \right]$$

Figure 2b: Math output with Times fonts

#### 4.3 Other considerations

The inherent problem in adjusting the spacing of a math italic font is that there is an order of magnitude of difference between the number of visual spacings to adjust and the number of parameters available to make these adjustments. Every character can potentially be typeset against every other character either as adjacent characters, or with one or the other in the superscript or subscript position. This leads to a large number of possible character combinations with relatively few parameters to make the necessary adjustments.

As a result, changes to a single parameter cause visual changes to a great number of spacings, as in the case where adjusting the italic correction on the  $x$  to obtain more spacing between the  $xy$  pair would also adjust the space between every character combination using  $x$  on the left and all superscripts typeset on the  $x$ . Therefore, a constant game of compromise must be employed when adjusting any of the parameters.

The tendency may be to use kerns too frequently to solve specific spacing problems. This approach is valid, except for the fact that each font in TeX can only have up to 255 LIG and KRN commands in the ligtable. Thus, careful decisions must be made to classify every spacing problem as either a "global" problem with a single character, or a "specific" problem with a pair of characters.

### 5. Conclusion

While the difficulties of training TeX to set non-METAFONT fonts are great (though not insurmountable), the results are certainly worth the effort. Other Autologic fonts that were purchased by the AMS have since been made available to TeX, and the procedure is becoming easier with more experience. The appearance of many of the AMS books and journals and of MR is, in the opinion of many involved, greatly improved. The purely aesthetic question of which typeface is more attractive is too subjective for quantitative evaluation, and, in the case of the MR journal, separating the contribution of macro design changes from those of changes in typeface is difficult. The reader is invited to make her/his own choices: Figure 3 displays the same MR review set in Almost-Computer Modern using the old design format and in Autologic Times using the new design format.

### 6. Update

During the TUG conference sometime after this paper had been presented, Prof. Knuth endorsed the use of the `xpl/amf` system, and challenged implementors of `dvi` drivers to utilize the design. David Rodgers, president of ArborText, also announced that he would make the specifications of the `xpl` and `amf` format available to the TeX community in a future issue of *TUGboat*.

## 60J Markov processes

Knuth, D. E. (1-STF-C)

86c:60112

An algorithm for Brownian zeroes. (German summary)

*Computing* 33 (1984), no. 1, 89–94.

Let  $B(t)$  be the Brownian motion on  $[0, 1]$  starting at  $a \neq 0$ , and  $B^{(n)}(t)$  its polygonal approximation, formed by setting  $B^{(n)}(t) = B(t)$  for  $t = i/2^n$ ,  $i = 0, 1, \dots, 2^n$ , and interpolating linearly elsewhere. This paper is concerned with an efficient algorithm for simulating the number and positions of the zero crossings of  $B(n)$ , without necessarily simulating  $B^{(n)}$  itself. Briefly, the idea is as follows: divide  $[0, 1]$  into two equal parts, and “decide”, with the appropriate hitting probability, whether  $B$  might have a zero crossing in either. An empty interval is ignored. If there is a zero crossing, divide that interval into two equal parts, and proceed as before, until the required level of resolution is reached. Simulation of  $B$  at each interval end point that enters into this procedure is required. The algorithm requires  $O(2^{n/2})$  simulations of Gaussian variates, as opposed to  $O(2^n)$  required by the obvious direct method of first simulating all of  $B^{(n)}$ . Robert J. Adler (Haifa)

Figure 3a: Sample review from *Mathematical Reviews* set in Almost Computer Modern

## 60J Markov processes

86c:60112 60J65 65C10

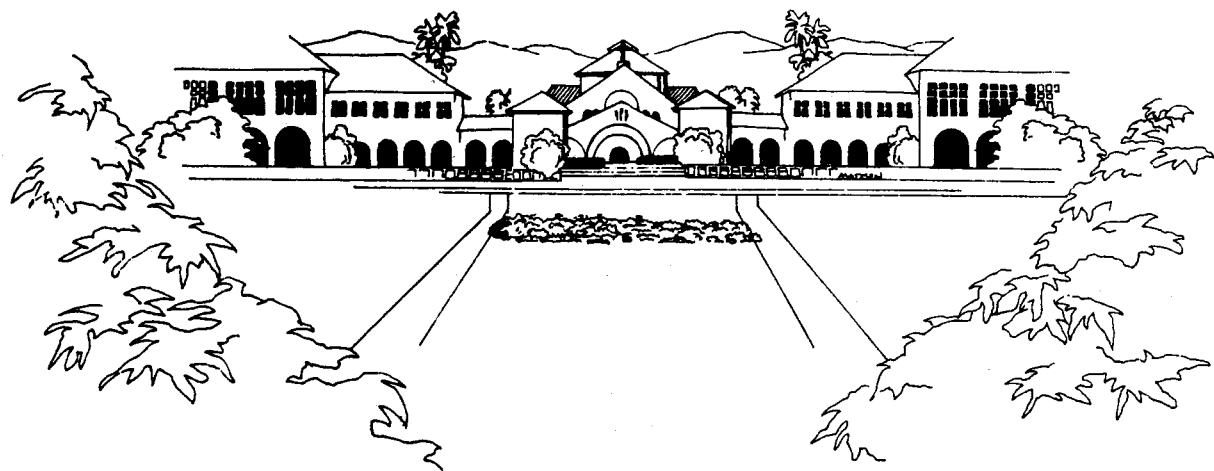
Knuth, D. E. (1-STC-C)

An algorithm for Brownian zeroes. (German summary)

*Computing* 33 (1984), no. 1, 89–94.

Let  $B(t)$  be the Brownian motion on  $[0, 1]$  starting at  $a \neq 0$ , and  $B^{(n)}(t)$  its polygonal approximation, formed by setting  $B^{(n)}(t) = B(t)$  for  $t = i/2^n$ ,  $i = 0, 1, \dots, 2^n$ , and interpolating linearly elsewhere. This paper is concerned with an efficient algorithm for simulating the number and positions of the zero crossings of  $B(n)$ , without necessarily simulating  $B^{(n)}$  itself. Briefly, the idea is as follows: divide  $[0, 1]$  into two equal parts, and “decide”, with the appropriate hitting probability, whether  $B$  might have a zero crossing in either. An empty interval is ignored. If there is a zero crossing, divide that interval into two equal parts, and proceed as before, until the required level of resolution is reached. Simulation of  $B$  at each interval end point that enters into this procedure is required. The algorithm requires  $O(2^{n/2})$  simulations of Gaussian variates, as opposed to  $O(2^n)$  required by the obvious direct method of first simulating all of  $B^{(n)}$ . Robert J. Adler (Haifa)

Figure 3b: Sample review from *Mathematical Reviews* set in Autologic Times



**TEX Users Group**  
**Stanford University, August 11-14, 1985**  
**Terman Engineering Center Auditorium**

# Fine Typesetting with $\text{\TeX}$ Using Native Autologic Fonts

ARVIN C. CONRAD

Menil Foundation  
1519 Branard Street  
Houston, TX 77006  
`acc@rice.edu`

## ABSTRACT

*The Image of the Black in Western Art:* Volume 4, Parts 1 and 2, by Hugh Honour, plus its companion French translation *L'Image du noir dans l'art occidental*, t. IV (in two parts) were set with  $\text{\TeX}$ , and represent 1,200 final pages of high-quality typesetting and fine book design. These books are the latest in a series published by the Menil Foundation as part of an ongoing 30-year research and publishing project. Previously published volumes in the series dictated conformity to an existing design and the classic Monotype Baskerville fotography.

### 1. Introduction

For almost thirty years Menil Foundation has been conducting and publishing research on the representations of black Africans and Afro-Americans in Western art. The enclosed brochure gives a brief description of our publications and the scope of our efforts.

So reads one of the Menil Foundation Black Image Project introductory letters, referring to a monumental publication series: *The Image of the Black in Western Art*.

Economical and timely publishing of this research was the driving force behind the decision to adopt  $\text{\TeX}$  to typeset the most recent volume in this series.

Volume 4 was set at  $\text{\TeX}$ Source in Houston, Texas in running galleys to the specifications of a designer in Switzerland using  $\text{\TeX}$  and output in Autologic Baskerville II on an Autologic APS Micro5. Accuracy and economy were key factors in the decision to have typesetting for this project controlled by the editorial offices in Houston, where the two editors using `emacs` directly coded the manuscript in  $\text{\TeX}$  running on two Sun-3 workstations.

Economy was realized by producing a minimum number of silver film galleys during the course of the project. Production of inexpensive but conformant laser galleys became essential. Crucial to this economy and visual conformity was the ability to emulate the native phototypesetter font family with the laser fonts. During the design phases, laser proof-galleys acceptable to the designer were produced using special emulation features in a *dvi*-to-PostScript convertor designed by Stephan von Bechtolsheim. Together with its companion utilities, this *dvi2ps* convertor/driver allows emulation of any native typesetter font with any PostScript outline or pixel font.

This paper discusses this process, some of the problems encountered and their solutions, and some lessons learned: the results are available for all to see.

### 2. $\text{\TeX}$ 's Role

$\text{\TeX}$ , "a new typesetting system intended for the creation of beautiful books" (Knuth 1984:v) could provide the necessary sophistication for high-quality work that is expected in the Foundation's publications. Examples of  $\text{\TeX}$ 's finesse and facility at addressing complicated typesetting problems are well-known in the world of technical typesetting. However, until recently the only fonts available for use with  $\text{\TeX}$  were those derived from METAFONT descriptions. While for some purposes this presents little problem for a publisher, it often imposes restrictions unacceptable to the graphic designer. In

classical typesetting environments, there are literally thousands of fonts available. A general solution to fine typesetting using *TEX* must rationally address the requirements of all designers. Locally, except for documentation, it is unlikely that we would ever use Computer Modern for any publication.

### **3. Fine Fontography, *TEX*, and the Graphic Designer**

A fine book is more than type, graphics, and photos; it is a pleasing, coherent combination of these elements. The graphic designer is responsible for this coherency and beauty. The graphic designer has been around longer than the typesetting process and it sometimes seems that his main objective is to make a job less profitable for the typesetter by demanding what is often called quality typography. It is the designer's job to "direct" the typesetter to produce his typeset output in a way that will make the finished product, usually in print, fulfill its purpose. The ultimate purpose of the work may be visual attractiveness and readability, or it may have other, multiple goals.

## **4. Image of the Black in Western Art: A Case Study**

### **4.1 Production Environment**

The first books in *The Image of the Black* series were prepared and printed in Europe on second-generation (50 lines-per-minute) phototypesetting equipment (Monotype). The most recent volumes were prepared in *TEX* on Sun workstations and proofed on Apple LaserWriter Plus and Apple LaserWriter II NTX laser printers. The final galleys were output on an Autologic Micro5 CRT (1,000 lpm) output device located at *TEX*Source's plant a few miles from the Foundation's facilities. The *dvi* files were transferred to PC diskettes or tele-communicated to a PC running the TextSet (ArborText) DVIAPS driver.

### **4.2 Font Matching**

The overall design parameters were driven by the previous volumes in the series. The new typography needed to be consistent with the preexisting type. Volumes 1 and 2 had been set in Europe using a Monotype Baskerville face. To achieve the closest match many samples were required. Having an output device that is capable of producing type output in tenths of a point (10.1, 10.2, etc.) made it possible to find a "look," or appearance, that visually matched the weight and size of the previous volumes.

One of the first problems encountered was the classic difference between manufacturers: the uppercase alphabet and its dimensional relationship to the lowercase alphabet. If the paragraph was set to match capital height, then the text ran too long and if it was reset to match the lowercase alphabet, the "color" of the paragraph (the visual appearance of a page or paragraph) was undesirable. "Good color" means there is nothing about the preparation of the page that detracts from one's original objective: to read and retain the information on the page. "Bad color" results in a lower comprehension and retention. Words too close together or too far apart (especially a combination of both in the same paragraph), too much or too little leading, the presence of "rivers" (wordspaces that make vertical, white gullies through a page of print), all produce "bad color."

The ultimate solution was to define the range of point sizes for the copy to be set in (10.8 through 11.4), then set the paragraph in all possible point sizes (tenths of a point increments) and with different leading values. The results were then compared, on an individual basis, with the paragraph from the published books. The result was a choice that allowed the new volumes to match as closely as possible the previous volumes. The magnification function available in *TEX* lent itself well to solving a problem which arose late in the design phase. The original manuscript was in English, then the French translation was produced from it. French tends to run longer for the same thought. Since both language versions were to use the same mechanicals, the equivalent running length of the typeset sections became a critical factor. Through more laser type and photo galley tests it was determined that if a global magnification of 985 (i.e., a 1.5% reduction) was applied to the French chapter text, the overall effect would not be annoying and it would, on average, act to resynchronize the English and French mechanicals. After numerous tests and conversations the production staff agreed upon some basic font calls for the structural elements of the book.

#### **4.3 Laser Proofing: Font Emulation**

The cost of typesetting a book depends on the number of galleys to be produced and on the number of changes required before the final mechanical boards are pasted up. If preliminary silver galleys can be kept to a minimum and laser proof galleys utilized for the design and editorial stages, then significant cost savings are possible. Typically laser proofs cost a tenth to a twentieth of equivalent film galleys. If this approach was to be successful, the laser proofs would need to mimic the final galleys as closely as possible. These laser galleys would be all that the author, editors, and designer would have to work with until the final stages of paste-up. Acceptable representation of the Autologic font face at the laser stage would be critical.

#### **4.4 Laser Resolution Pixel Representation of Autologic Fonts**

One solution to laser proof galleys is to produce pixel representations of the typesetter proprietary font outlines in .pxl format for downloading to the LaserWriter. For many reasons, not all related to technical considerations, this was not possible within the cost and time constraints imposed by the production schedule.

#### **4.5 PostScript Laser Fonts**

A more interesting and general solution was to substitute PostScript outline fonts for the proprietary fonts during laser proofing. In brief, this process derives width information from the proprietary font to produce a set of standard .tfm files for the font. This facility is included in the ArborText dviaps software package. The resultant .tfm files are loaded into the local TeX font path environment. Next in the installation procedure, these .tfms are used to produce a corresponding .pdr file which includes the font mapping information and width values. At the time of dvi-to-PostScript conversion this .pdr file is referenced by the convertor to produce the font width vector which acts to impose the Autologic native widths on the PostScript outlines.<sup>1</sup> A major practical advantage of this method over that of pixel representation is the freedom offered by the scalable nature of PostScript outline fonts. During the "color" trial galley stages this greatly facilitates the easy changes of font sizes which would not be possible with pixel representation where each new trial size would need to be generated and stored.

### **5. Proof of the Pudding**

Examples of both laser proofs and tear sheets are included in the Appendix. Sample 1 and Sample 2 are equivalent chapter pages (in both laser proof and tear sheet form) from the English and French publications. The endnotes shown in sample 3 and sample 4 illustrate the extremely tight setting which had to be achieved.

### **6. Acknowledgements**

These are all books which are success stories partly because of the facilities developed over the last three years within the Menil Foundation publications arm. Much of this success is due to the talents and patience of highly skilled professionals in the fields of book design, fine typography, and computer science.

Specifically, the designer of record for *The Image of the Black* series, Hanspeter Schmidt, oversaw the production of a truly beautiful set of volumes. Steve Bencze, proprietor of TeXSource, provided much more than the output of the final galleys. His knowledge of fontography and classical typesetting, as well as his exemplary patience during the type-matching trials, were the technical basis for the success.

None of this would have been possible without the talents and generosity of Donald Knuth and the TeX community. Finally, the publications program owes much of its success to the talents and efforts of Stephan von Bechtolsheim for the development of the dvi-to-PostScript convertor software. Without the font emulation facilities incorporated into this convertor, utilization of arbitrary fonts would not have been possible.

---

<sup>1</sup> For full appreciation of the font emulation processes one needs to refer to Bechtolsheim's TeXPS manual and installation document, which is a very complete description of the system and currently runs to more than a hundred pages with examples.

Ultimately responsibility for quality publications rests with management decisions. In the case of the Menil Foundation and Black Image Project, many far-reaching decisions had to be formulated before actual results could be demonstrated. It is to their credit that this project took the direction that it did and that the final books represent high levels of quality in all aspects.

## Bibliography

### 1. Technical reference titles used in the process described in the paper:

Bechtolsheim, Stephan v. *Another Look at T<sub>E</sub>X*, West Lafayette, IN (self-published), 1986, 1987, 1988.

—. *T<sub>E</sub>X in Practice*. Heidelberg: Springer-Verlag. Forthcoming.

—. Documentation for the dvi-to-POSTSCRIPT Convertor, 1986, 1987, 1988. Final version included in the current T<sub>E</sub>XPS distribution, available from Stephan von Bechtolsheim.

Knuth, Donald E. *The T<sub>E</sub>Xbook*. Reading, MA: Addison-Wesley, 1984.

### 2. Titles produced using the process and techniques described in the paper [technical colophon information pertaining to volume in square brackets]:

Barnes, Susan J. *The Rothko Chapel, An Act of Faith*. Houston: Rothko Chapel, 1989, 128pp.

[Designer: Don Quaintance; font: Autologic Galliard.]

Camfield, William. *Marcel Duchamp Fountain*. Houston: The Menil Collection and Houston Fine Art Press, 1989, 184pp.

[Editor, T<sub>E</sub>X compositor: John Kaiser; Designer, hands-on T<sub>E</sub>Xer: Don Quaintance; font: Autologic Caslon.]

Davezac, Bertrand. *Spirituality in the Christian East, Greek, Slavic, and Russian Icons from The Menil Collection*. Houston: The Menil Collection, 1989, 134pp.

[Designer, editor, T<sub>E</sub>X compositor, John Kaiser; font, Adobe Palatino and Greek pixel fonts derived from Silvio Levy's Greek METAFONT descriptions.]

Guidieri, Remo, F. Pellizzi, and S.J. Tambiah. *Ethnicities and Nations: Processes of Interethnic Relations in Latin America, Southeast Asia, and the Pacific*. Houston: The Rothko Chapel, 1988, 408pp.

[Designer: Harris Rosenstein; T<sub>E</sub>X compositor, Geraldine Aramanda; font: Autologic Times New Roman.]

Honour, Hugh. *The Image of the Black in Western Art*, Volume 4, Parts 1 and 2. Cambridge, MA: Harvard University Press, 1989.

[Designer: Hanspeter Schmidt; font: Autologic Baskerville II.]

Honour, Hugh. *L'Image du noir dans l'art occidental*, t. IV [two parts]. Paris: Gallimard, 1989

[Designer: Hanspeter Schmidt; font: Autologic Baskerville II.]

Karageorghis, Vassos. *Blacks in Ancient Cypriot Art*. Houston: The Menil Collection, 1988, 63pp.

[Designer: Don Quaintance; font: Autologic Times New Roman.]

Printz, Neil and Remo Guidieri. *Andy Warhol: Death and Disasters*. Houston: The Menil Collection and Houston Fine Art Press, 1988, 136pp.

[Designer: Marilyn Muller; font: Autologic New Times Roman.]

Wood, Peter and Karen C.C. Dalton. *Winslow Homer's Images of Blacks: The Civil War and Reconstruction Years*. Austin: University of Texas Press, 1988, 144pp.

[Designer: Don Quaintance; T<sub>E</sub>X compositor: Geraldine Aramanda; font: Autologic Bembo.]

## Sample 1: English text: PostScript laser proof (left) and Autologic tear sheet (right)

## HONOUR TEXT

## PROOFS

## Text:II/ 3

and perhaps still more the way in which it was painted. And yet, from an iconographical point of view, this black woman could hardly be more traditionally conventional, cast in the "narrative" role of a servant and the pictorial role of a figure whose dark complexion sets off the pallor of a white woman. There is also a striking contrast between the way in which Manet depicted Olympia herself with chilly realism suggesting portraiture, and the black woman with generalized features carrying a bouquet of flowers painted in delicately fresh, one might almost say rococo, colors.

Manet painted *Olympia* for exhibition in the Salon hoping, no doubt, that it would help to establish his place in and also mark his development of the great tradition of figure painting. Hence its very obvious debts to artists he admired—to Courbet, Delacroix, Ingres, Goya, and especially Titian. It presented a contrast with and also a kind of critical comment on the innumerable images of odalisques provocatively flexing their ample thighs, displayed in practically every European art exhibition of the time. By depicting sincerely his own vision of the contemporary world, comparing and contrasting what he saw before him with reminiscences of artistic images of similar subjects, Manet was attempting to return to what he considered essential principles. The scene is set in France in the 1860s. Olympia is, Zola wrote,

a girl of sixteen, doubtless some model whom Edouard Manet has quietly copied just as she was. Everyone exclaimed that this nude body was indecent. That is as it should be since here in the flesh is a girl whom the artist has put on canvas in her youthful, slightly tarnished nakedness. When other artists correct nature by painting Venus, they lie. Manet asked himself why he should lie. Why not tell the truth? He has introduced us to Olympia, a girl of our own times, whom we have met in the streets pulling a thin shawl over her narrow shoulders.<sup>40</sup>

Representing Olympia as a common prostitute, he stripped away the subterfuges by which images of the naked female body, as an object of male desire and possible purchase, had been given respectability. It was well known that there were black women in Parisian brothels. At the same time the Goncourt brothers, gathering material for a realist novel, jotted in a notebook a reminder to "make the prostitute's friend a Negress, study the type, and incorporate it in the story."<sup>41</sup> But it was in Orientalist paintings that white women were most often accompanied by blacks. And Olympia's attendant might seem to intrude from this fantasy world to present a contrast between falsity and truth as well as skin color.

## FLESH FOR SALE

The Scottish painter David Roberts found the slave market in Alexandria "peculiarly disquieting" when he inspected it shortly after arriving in Egypt in 1838. "The slaves were mostly girls; some from Circassia were well dressed; others, negroes, squatted on the ground with scanty bits of matting thrown round them, and in a sun that would have killed a European" he told his daughter. "It was altogether a sickening sight, and I left it proud that I

and perhaps still more the way in which it was painted. And yet, from an iconographical point of view, this black woman could hardly be more traditionally conventional, cast in the "narrative" role of a servant and the pictorial role of a figure whose dark complexion sets off the pallor of a white woman. There is also a striking contrast between the way in which Manet depicted Olympia herself with chilly realism suggesting portraiture, and the black woman with generalized features carrying a bouquet of flowers painted in delicately fresh, one might almost say rococo, colors.

Manet painted *Olympia* for exhibition in the Salon hoping, no doubt, that it would help to establish his place in and also mark his development of the great tradition of figure painting. Hence its very obvious debts to artists he admired—to Courbet, Delacroix, Ingres, Goya, and especially Titian. It presented a contrast with and also a kind of critical comment on the innumerable images of odalisques provocatively flexing their ample thighs, displayed in practically every European art exhibition of the time. By depicting sincerely his own vision of the contemporary world, comparing and contrasting what he saw before him with reminiscences of artistic images of similar subjects, Manet was attempting to return to what he considered essential principles. The scene is set in France in the 1860s. Olympia is, Zola wrote,

a girl of sixteen, doubtless some model whom Edouard Manet has quietly copied just as she was. Everyone exclaimed that this nude body was indecent. That is as it should be since here in the flesh is a girl whom the artist has put on canvas in her youthful, slightly tarnished nakedness. When other artists correct nature by painting Venus, they lie. Manet asked himself why he should lie. Why not tell the truth? He has introduced us to Olympia, a girl of our own times, whom we have met in the streets pulling a thin shawl over her narrow shoulders.<sup>40</sup>

Representing Olympia as a common prostitute, he stripped away the subterfuges by which images of the naked female body, as an object of male desire and possible purchase, had been given respectability. It was well known that there were black women in Parisian brothels. At the same time the Goncourt brothers, gathering material for a realist novel, jotted in a notebook a reminder to "make the prostitute's friend a Negress, study the type, and incorporate it in the story."<sup>41</sup> But it was in Orientalist paintings that white women were most often accompanied by blacks. And Olympia's attendant might seem to intrude from this fantasy world to present a contrast between falsity and truth as well as skin color.

## FLESH FOR SALE

The Scottish painter David Roberts found the slave market in Alexandria "peculiarly disquieting" when he inspected it shortly after arriving in Egypt in 1838. "The slaves were mostly girls; some from Circassia were well dressed; others, negroes, squatted on the ground with scanty bits of matting thrown round them, and in a sun that would have killed a European" he told his daughter. "It was altogether a sickening sight, and I left it proud that I

HONOUR TEXT      PROOFS      French Part 2, Chapter3

français conservateur par son sujet — une prostituée — et plus encore peut-être par son exécution. Pourtant, au point de vue iconographique, elle est aussi conventionnelle que possible dans son rôle «narratif» de servante et en tant qu'élément pictural destiné par son physique à faire ressortir la pâleur de la femme blanche. Le contraste est frappant aussi, dans le traitement des deux figures, entre le froid réalisme d'*Olympia*, qui fait penser à un portrait, et la banalité des traits de la femme noire, portant un bouquet de fleurs aux couleurs délicates, presque rococo.

Manet avait peint *Olympia* pour l'exposer au Salon en espérant évidemment qu'elle lui permettrait de prendre place dans le grand art, tout en y imprimant sa marque. D'où une référence manifeste aux artistes qu'il admirait, Courbet, Delacroix, Ingres, Goya et plus encore Titien. *Olympia* se différencie des innombrables odalisques aux lourdes cuisses provocantes qu'on pouvait voir pratiquement dans toutes les expositions européennes de l'époque et dont elle constitue en même temps une sorte de commentaire critique. En transcrivant sincèrement sa propre vision du monde contemporain, en comparant et en opposant ce qu'il avait devant les yeux avec ses réminiscences d'œuvres sur des sujets semblables, Manet voulait revenir aux principes essentiels selon lui. La scène se passe en France, dans les années 1860. Pour Zola, *Olympia* est «une jeune fille de seize ans, sans doute un modèle qu'Edouard Manet a tranquillement copié tel qu'il était. Et tout le monde a crié : on a trouvé ce corps nu indécent ; cela devait être, puisque c'est là de la chair, une fille que l'artiste a jetée sur la toile dans sa nudité jeune et déjà fanée. Lorsque nos artistes nous donnent des Vénus, ils corrigent la nature, ils mentent. Edouard Manet s'est demandé pourquoi mentir, pourquoi ne pas dire la vérité ; il nous a fait connaître *Olympia*, cette fille de nos jours, que vous rencontrez sur les trottoirs et qui serre ses maigres épaules dans un mince châle de laine déteinte.»<sup>40</sup> En faisant d'*Olympia* une banale prostituée, il dépouillait la représentation du corps féminin, objet masculin de désir et d'amour vénal éventuellement, des subterfuges qui avaient permis de lui donner une certaine respectabilité. La présence de Noires dans les maisons closes parisiennes était notoire. A la même époque sensiblement, les frères Goncourt, rassemblant les éléments d'un roman réaliste, notèrent dans un carnet : «faire de l'amie de la prostituée une nègresse, étudier le type et l'intégrer à l'histoire»<sup>41</sup>. Mais c'est surtout dans la peinture orientaliste que l'on trouve réunies femmes blanches et noires ; et la servante d'*Olympia* semble surgie de ce monde factice pour apporter non seulement le contraste de sa couleur mais aussi celui d'une figure artificielle, contraire à la vérité de l'autre.

LA CHAIR À L'ENCAN

Le peintre écossais David Roberts trouvait le marché d'esclaves d'Alexandrie «particulièrement troubant» quand il le visita peu après son arrivée en Egypte, en 1838 : «Les esclaves étaient pour la plupart des jeunes filles; quelques-unes, des Circassiennes, étaient bien vêtues; les autres, des nègresses, se tenaient accroupies, quelques rares nattes jetées autour d'elles et sous un soleil qui aurait tué un Européen», écrit-il à sa fille. «C'était un spectacle vraiment révoltant que je quittais, fier d'appartenir à un pays qui avait aboli l'esclavage»<sup>42</sup>. Un peu plus tard, Roberts rencontra le propriétaire d'un bateau d'esclaves et regretta de «connaître trop peu de mots arabes

français conservateur par son sujet — une prostituée — et plus encore peut-être par son exécution. Pourtant, au point de vue iconographique, elle est aussi conventionnelle que possible dans son rôle «narratif» de servante et en tant qu'élément pictural destiné par son physique à faire ressortir la pâleur de la femme blanche. Le contraste est frappant aussi, dans le traitement des deux figures, entre le froid réalisme d'*Olympia*, qui fait penser à un portrait, et la banalité des traits de la femme noire, portant un bouquet de fleurs aux couleurs délicates, presque rococo.

Manet avait peint *Olympia* pour l'exposer au Salon en espérant évidemment qu'elle lui permettrait de prendre place dans le grand art, tout en y imprimant sa marque. D'où une référence manifeste aux artistes qu'il admirait, Courbet, Delacroix, Ingres, Goya et plus encore Titien. *Olympia* se différencie des innombrables odalisques aux lourdes cuisses provocantes qu'on pouvait voir pratiquement dans toutes les expositions européennes de l'époque et dont elle constitue en même temps une sorte de commentaire critique. En transcrivant sincèrement sa propre vision du monde contemporain, en comparant et en opposant ce qu'il avait devant les yeux avec ses réminiscences d'œuvres sur des sujets semblables, Manet voulait revenir aux principes essentiels selon lui. La scène se passe en France, dans les années 1860. Pour Zola, *Olympia* est «une jeune fille de seize ans, sans doute un modèle qu'Edouard Manet a tranquillement copié tel qu'il était. Et tout le monde a crié : on a trouvé ce corps nu indécent ; cela devait être, puisque c'est là de la chair, une fille que l'artiste a jetée sur la toile dans sa nudité jeune et déjà fanée. Lorsque nos artistes nous donnent des Vénus, ils corrigent la nature, ils mentent. Edouard Manet s'est demandé pourquoi mentir, pourquoi ne pas dire la vérité ; il nous a fait connaître *Olympia*, cette fille de nos jours, que vous rencontrez sur les trottoirs et qui serre ses maigres épaules dans un mince châle de laine déteinte.»<sup>40</sup> En faisant d'*Olympia* une banale prostituée, il dépouillait la représentation du corps féminin, objet masculin de désir et d'amour vénal éventuellement, des subterfuges qui avaient permis de lui donner une certaine respectabilité. La présence de Noires dans les maisons closes parisiennes était notoire. A la même époque sensiblement, les frères Goncourt, rassemblant les éléments d'un roman réaliste, notèrent dans un carnet : «faire de l'amie de la prostituée une nègresse, étudier le type et l'intégrer à l'histoire»<sup>41</sup>. Mais c'est surtout dans la peinture orientaliste que l'on trouve réunies femmes blanches et noires ; et la servante d'*Olympia* semble surgie de ce monde factice pour apporter non seulement le contraste de sa couleur mais aussi celui d'une figure artificielle, contraire à la vérité de l'autre.

LA CHAIR À L'ENCAN

Le peintre écossais David Roberts trouvait le marché d'esclaves d'Alexandrie «particulièrement troubant» quand il le visita peu après son arrivée en Egypte, en 1838 : «Les esclaves étaient pour la plupart des jeunes filles; quelques-unes, des Circassiennes, étaient bien vêtues; les autres, des nègresses, se tenaient accroupies, quelques rares nattes jetées autour d'elles et sous un soleil qui aurait tué un Européen», écrit-il à sa fille. «C'était un spectacle vraiment révoltant que je quittais, fier d'appartenir à un pays qui avait aboli l'esclavage»<sup>42</sup>. Un peu plus tard, Roberts rencontra le propriétaire d'un bateau d'esclaves et regretta de «connaître trop peu de mots arabes

## I

## LE TRIANGLE ATLANTIQUE

1 Paris, *Salon de 1781*, n° 196. Louis Petit de BACHAUMONT, *Mémoires secrets pour servir à l'histoire de la république des lettres en France, depuis MDCCXLII jusqu'à nos jours; ou Journal d'un observateur...*, t. XIX, Londres, 1783, p. 361, cité par Denis DIDEROT, *Salons*, Jean SEZNEC éd., vol. IV, 1769, 1771, 1775, 1781, Oxford, 1967, p. 301.

2 Ottawa, The National Gallery of Canada, 58. Robert Hamilton HUBBARD éd., *The National Gallery of Canada. Catalogue of Paintings and Sculpture*, vol. I, *Older Schools*, Ottawa, Toronto, 1957, p. 112, n° 58.

3 John R. WILLIS, «New Light on the Life of Ignatius Sancho: Some Unpublished Letters», *Slavery & Abolition*, 1, 3, December 1980, p. 345-358, particul. p. 349-351. Les lettres de Laurence Sterne à Sancho furent publiées dans *The Works of Laurence Sterne*, 10 vol., London, 1793, vol. IX, p. 198-200; vol. X, p. 62-63.

4 *Letters of the late Ignatius Sancho, an African, to Which Are Prefixed Memoirs of His Life*, Joseph JEVILLE éd., London, 1782.

5 Olaudah EQUIANO, *The Interesting Narrative of the Life of Olaudah Equiano, or Gustavus Vasa, the African*, London, 1789, vol. II, p. 244; le frontispice est signé W. Denton pinx. D. Orme sculp. Un tableau supposé être le portrait d'Equiano est conservé à Exeter, Royal Albert Memorial Museum, 14/1943, et reproduit comme tel dans Paul EDWARDS, James WALVIN, *Black Personalities in the Era of the Slave Trade*, Baton Rouge, 1983, pl. 1a. Mais les traits du modèle sont différents de ceux que nous présente le médaillon du frontispice.

6 Sur ces écrivains, voir Id., *ibid.*

7 *Reminiscences of Henry Angelo, with Memoirs of His Late Father and Friends...*, vol. I, London, 1830, p. 452. Aucun de ces portraits n'a été jusqu'à présent retrouvé. Il existe également une gravure anonyme qui représente Soubise croisant le fer avec la duchesse de Queensberry.

8 Wilhelm A. BAUER, *Angelo Soliman, der hochfürstliche Mohr. Ein exotisches Kapitel Alt-Wien*, Wien, 1922. Son portrait

par Nepomuk Steiner a été gravé en mezzotinte par Johann-Gottfried Haid.

9 Voir David DABYDEEN, *Hogarth's Blacks: Images of Blacks in Eighteenth Century English Art*, Mundelstrup (DK), Kingston-upon-Thames, 1985.

10 Par exemple, de Johan Zoffany, *La famille de Sir William Young*, Liverpool, Walker Art Gallery, 2395; cf. *Walker Art Gallery, Liverpool. Foreign Catalogue. Paintings, Drawings...*, Liverpool, 1977, vol. I, *Text*, p. 231-232; vol. II, *Plates*, p. 299. Young, qui était né aux Indes occidentales, devint gouverneur de la Dominique. En revanche, la fillette noire figurant sur une peinture de Joseph Wright of Derby demande sans doute à être interprétée différemment; cf. Benedict NICOLSON, *Joseph Wright of Derby: Painter of Light*, London, New York, 1968, vol. I, *Text and Catalogue*, p. 35, 99, 228; vol. II, *Plates*, p. 48, pl. 73 (localisation actuelle inconnue). L'auteur identifie ce tableau avec l'œuvre mentionnée dans le livre de raison (manuscrit) de Wright comme «deux fillettes avec leur servante noire», et avec le tableau exposé à Londres en 1770 sous le titre *A Conversation of Girls* (cf. Londres, Society of Artists, p. 286, s.v. «Wright, Joseph (of Derby)», 1770, n° 155). A ce moment, Wright travaillait à Liverpool, le premier port négrier anglais. Un bateau, à l'arrière-plan du tableau, évoque la traite. La fillette noire figurait sans doute parmi les nombreux enfants dont la traite pourvoyait le pays alors que les deux blanches devaient appartenir à l'une de ces familles qui lui devaient leur fortune et où le peintre trouvait sa clientèle. Mais on trouve, en même temps, parmi les autres modèles ou amis personnels de Wright, des personnalités d'un tout autre bord, notamment le naturaliste et poète Erasmus Darwin et Thomas Day. Tous deux allaient soutenir activement la campagne contre la traite. *The Dying Negro* que Day écrira en collaboration avec John BICKNELL, est le premier poème anglais à développer longuement les thèmes abolitionnistes. Sans avoir influencé directement Wright, on retrouve dans son tableau une idée des Africains très analogue à celle que résument les vers:

*What tho' no rosy tints adorn their face,  
No silken tresses shine with flowing grace?*

*Yet of ethereal temper are their souls...*

(«Las! Point de teint pareil à la rose diaprée/ Ni de tresses de soie d'un flot d'or inondées/ Mais leur âme est parée de la grâce éternelle...»). Cf. Thomas DAY, John BICKNELL, *The Dying Negro, a Poem* (1773), dans *The Poetical Register, and Repository of Fugitive Poetry, for 1810-1811*, London, 1814, p. 357. Day, en

outre, était disciple de Rousseau: il tenta de mettre en pratique avec deux jeunes orphelines ses principes d'éducation, et l'on est tenté d'associer à ces idées la simplicité des toilettes des enfants du tableau de Wright. Quoique Wright ait probablement reçu la commande d'un portrait de groupe, il l'exposa sous un titre qui pouvait convenir à une pièce de libre invention et le vendit à un collectionneur qui acheta trois de ses tableaux de genre.

11 James WALVIN, *Black and White: The Negro and English Society, 1555-1945*, London, 1973, p. 46.

12 Voir le tome 2 de ce volume, p. 12-18.

13 Les informations les plus complètes publiées au XVIII<sup>e</sup> siècle furent celles apportées par un Danois, missionnaire morave aux Antilles, Christian Georg Andreas OLDENDORF, dans *Geschichte der Mission der evangelischen Brüder auf den caraibischen Inseln S. Thomas, S. Croix und S. Jan*, Johann Jakob BOSSERT éd., Barby, 1777 (2 vol.).

14 Amsterdam, Rijksmuseum, A 4075; Copenhague, Statens Museum for Kunst, 376; cf. cat. exposition, Paris (Grand Palais), 1976-1977, *L'Amérique vue par l'Europe*, par Hugh HONOUR, Paris, 1976, n° 83-84, p. 86-87 (ill.).

15 Paris, Secrétariat d'Etat aux Départements et Territoires d'Outre-Mer; *ibid.*, n° 307, p. 293-294 (ill.).

16 Londres, Society of Artists, p. 277, s.v. «Wickstead, Philip», 1777, n° 159; p. 215, s.v. «Robertson, George», 1775, n° 22-213, 422; 1776, n° 77-80; 1777, n° 112-115; 1778, n° 171-175. Nous reproduisons ici une estampe gravée par James Mason d'après le tableau exposé en 1775 sous le n° 212 et publiée par John Boydell en 1778.

17 Londres, Royal Academy, vol. I, p. 321, s.v. «Brunais, Augustin», 1777, n° 35. La citation provient d'une coupure de presse (sans date), Londres, Victoria and Albert Museum, collection «Cuttings from English Newspapers on Matters of Artistic Interest, 1686-1835», vol. I, p. 145. L'étude la plus complète sur Brunias est celle de Hans HUTH, «Agostino Brunias, Romano: Robert Adam's 'Bred Painter', *The Connoisseur*, 151, 610, December 1962, p. 265-269.

18 Localisation actuelle inconnue.

19 Long, qui résidait à la Jamaïque, écrit que les Africains sont «abrutis, ignorants, paresseux, rusés, perfides, cruels, voleurs, soupçonneux et superstitieux»; non seulement leur peau est sombre, mais ils ont

## I

## LE TRIANGLE ATLANTIQUE

<sup>1</sup> Paris, *Salon de 1781*, n° 196. Louis Petit de BACHAUMONT, *Mémoires secrets pour servir à l'histoire de la république des lettres en France, depuis MDCCLXII jusqu'à nos jours; ou Journal d'un observateur...*, t. XIX, Londres, 1783, p. 361, cité par Denis DIDEROT, *Salons*, Jean SEZNEC éd., vol. IV, 1769, 1771, 1775, 1781, Oxford, 1967, p. 301.

<sup>2</sup> Ottawa, The National Gallery of Canada, 58. Robert Hamilton HUBBARD éd., *The National Gallery of Canada. Catalogue of Paintings and Sculpture*, vol. I, *Older Schools*, Ottawa, Toronto, 1957, p. 112, n° 58.

<sup>3</sup> John R. WILLIS, «New Light on the Life of Ignatius Sancho: Some Unpublished Letters», *Slavery & Abolition*, 1, 3, December 1980, p. 345-358, particul. p. 349-351. Les lettres de Laurence Sterne à Sancho furent publiées dans *The Works of Laurence Sterne*, 10 vol., London, 1793, vol. IX, p. 198-200; vol. X, p. 62-63.

<sup>4</sup> *Letters of the late ignatius Sancho, an African, to Which Are Prefixed Memoirs of His Life*, Joseph Jekyll éd., London, 1782.

<sup>5</sup> Olaudah EQUIANO, *The Interesting Narrative of the Life of Olaudah Equiano, or Gustavus Vasa, the African*, London, 1789, vol. II, p. 244; le frontispice est signé W. Denton pinx. D. Orme sculp. Un tableau supposé être le portrait d'Equiano est conservé à Exeter, Royal Albert Memorial Museum, 14/1943, et reproduit comme tel dans Paul EDWARDS, James WALVIN, *Black Personalities in the Era of the Slave Trade*, Baton Rouge, 1983, pl. 1a. Mais les traits du modèle sont différents de ceux que nous présente le médaillon du frontispice.

<sup>6</sup> Sur ces écrivains, voir Id., *ibid.*

<sup>7</sup> *Reminiscences of Henry Angelo, with Memoirs of His Late Father and Friends...*, vol. I, London, 1890, p. 452. Aucun de ces portraits n'a été jusqu'à présent retrouvé. Il existe également une gravure anonyme qui représente Soubise croisant le fer avec la duchesse de Queensberry.

<sup>8</sup> Wilhelm A. BAUER, *Angelo Soliman, der hochfürstliche Mohr. Ein exotisches Kapitel Alt-Wien*, Wien, 1922. Son portrait

par Nepomuk Steiner a été gravé en mezzotinte par Johann-Gottfried Haid.

<sup>9</sup> Voir David DABYDEEN, *Hogarth's Blacks: Images of Blacks in Eighteenth Century English Art*, Mundelstrup (DK), Kingston-upon-Thames, 1985.

<sup>10</sup> Par exemple, de Johan Zoffany, *La famille de Sir William Young*, Liverpool, Walker Art Gallery, 2395; cf. *Walker Art Gallery, Liverpool. Foreign Catalogue. Paintings, Drawings...* Liverpool, 1977, vol. I, *Text*, p. 231-232; vol. II, *Plates*, p. 299. Young, qui était né aux Indes occidentales, devint gouverneur de la Dominique. En revanche, la fillette noire figurant sur une peinture de Joseph Wright of Derby demande sans doute à être interprétée différemment; cf. Benedict NICOLSON, *Joseph Wright of Derby: Painter of Light*, London, New York, 1968, vol. I, *Text and Catalogue*, p. 35, 99, 228; vol. II, *Plates*, p. 48, pl. 73 (localisation actuelle inconnue). L'auteur identifie ce tableau avec l'œuvre mentionnée dans le livre de raison (manuscrit) de Wright comme «deux fillettes avec leur servante noire», et avec le tableau exposé à Londres en 1770 sous le titre *A Conversation of Girls* (cf. Londres, Society of Artists, p. 286, s.u. «Wright, Joseph (of Derby)», 1770, n° 155). A ce moment, Wright travaillait à Liverpool, le premier port négrier anglais. Un bateau, à l'arrière-plan du tableau, évoque la traite. La fillette noire figurait sans doute parmi les nombreux enfants dont la traite pourvoyait le pays alors que les deux blanches devaient appartenir à l'une de ces familles qui lui devaient leur fortune et où le peintre trouvait sa clientèle. Mais on trouve, en même temps, parmi les autres modèles ou amis personnels de Wright, des personnalités d'un tout autre bord, notamment le naturaliste et poète Erasmus Darwin et Thomas Day. Tous deux allaient soutenir activement la campagne contre la traite. *The Dying Negro* que Day écrira en collaboration avec John Bicknell, est le premier poème anglais à développer longuement les thèmes abolitionnistes. Sans avoir influencé directement Wright, on retrouve dans son tableau une idée des Africains très analogue à celle que résument les vers:

What tho' no rosy tints adorn their face,  
No silken tresses shine with flowing grace?

Yet of ethereal temper are their souls...  
«Las! Point de teint pareil à la rose diaprée/Ni de tresses de soi d'un flot d'or inondées/Mais leur âme est parée de la grâce éternelle...». Cf. Thomas DAY, John BICKNELL, *The Dying Negro, a Poem* (1773), dans *The Poetical Register, and Repository of Fugitive Poetry, for 1810-1811*, London, 1814, p. 357. Day, en

outre, était disciple de Rousseau: il tenta de mettre en pratique avec deux jeunes orphelines ses principes d'éducation, et l'on est tenté d'associer à ces idées la simplicité des toilettes des enfants du tableau de Wright. Quoique Wright ait probablement reçu la commande d'un portrait de groupe, il l'exposa sous un titre qui pouvait convenir à une pièce de libre invention et le vendit à un collectionneur qui acheta trois de ses tableaux de genre.

<sup>11</sup> James WALVIN, *Black and White: The Negro and English Society, 1555-1945*, London, 1973, p. 46.

<sup>12</sup> Voir le tome 2 de ce volume, p. 12-18.

<sup>13</sup> Les informations les plus complètes publiées au XVIII<sup>e</sup> siècle furent celles apportées par un Danois, missionnaire morave aux Antilles, Christian Georg Andreas OLDENDORP, dans *Geschichte der Mission der evangelischen Brüder auf den caraibischen Inseln S. Thomas, S. Croix und S. Jan*, Johann Jakob BOSSART éd., Barby, 1777 (2 vol.).

<sup>14</sup> Amsterdam, Rijksmuseum, A 4075; Copenhague, Statens Museum for Kunst, 376; cf. cat. exposition, Paris (Grand Palais), 1976-1977, *L'Amérique vue par l'Europe*, par Hugh HONOUR, Paris, 1976, n° 83-84, p. 86-87 (ill.).

<sup>15</sup> Paris, Secrétariat d'Etat aux Départements et Territoires d'Outre-Mer; *ibid.*, n° 307, p. 293-294 (ill.).

<sup>16</sup> Londres, Society of Artists, p. 277, s.u. «Wickstead, Philip», 1777, n° 159; p. 215, s.u. «Robertson, George», 1775, n° 212-213, 422; 1776, n° 77-80; 1777, n° 112-115; 1778, n° 171-175. Nous reproduisons ici une estampe gravée par James Mason d'après le tableau exposé en 1775 sous le n° 212 et publiée par John Boydell en 1778.

<sup>17</sup> Londres, Royal Academy, vol. I, p. 321, s.u. «Brunias, Augustin», 1777, n° 35. La citation provient d'une coupure de presse (sans date), Londres, Victoria and Albert Museum, collection «Cuttings from English Newspapers on Matters of Artistic Interest, 1686-1895», vol. I, p. 145. L'étude la plus complète sur Brunias est celle de Hans HUTH, «Agostino Brunias, Romano: Robert Adam's 'Bred Painter」, *The Connoisseur*, 151, 610, December 1962, p. 265-269.

<sup>18</sup> Localisation actuelle inconnue.

<sup>19</sup> Long, qui résidait à la Jamaïque, écrit que les Africains sont «abrutis, ignorants, paresseux, rusés, perfides, cruels, voleurs, soupçonneux et superstitionnels»; non seulement leur peau est sombre, mais ils ont

# Keynote Address

DONALD E. KNUTH

Donald E. Knuth  
Computer Science Department  
Stanford University  
Stanford, CA 94305

## Notes on the Errors of $\text{\TeX}$

At this 10th TUG meeting, my goal is to describe a longish paper that I recently had some fun writing.<sup>1</sup> That paper [6] contains a complete listing of the errors I noted down as I was developing and maintaining the  $\text{\TeX}$  system during a period of more than 10 years. I've always believed that one of the best ways to learn is by a process of trial and error; hence I decided that the presentation of a true-to-life list of errors might be the best way to help other people learn the lessons that my experiences with  $\text{\TeX}$  have taught me. And I suspected that the people at this conference might be as interested in this history as anybody is.

Of course no single project can be expected to illuminate all the aspects of software development. But the error log of  $\text{\TeX}$  seems to provide useful data for understanding the problems of crafting a medium-size piece of software. It's hard to teach students the concept of "scale" — the enormous difference between textbook examples and larger systems — but I think that a reasonable appreciation of the complexity of a medium-size project can be acquired by spending about two hours reading through a complete log such as the one in [6].

My error log begins with all the corrections made while debugging the first version of  $\text{\TeX}$ , which was a program consisting of approximately 4,600 statements in an Algol-like language. The log ends with all the changes I made as  $\text{\TeX}$  was becoming a stable system, as  $\text{\TeX}$  began to have more than a million users on more than a hundred varieties of computers. By studying the log you can see all the stages in the evolution of  $\text{\TeX}$  as new features replaced or extended old ones — except that I did not record the changes I made when I re-wrote the original program  $\text{\TeX}78$  and prepared the final one,  $\text{\TeX}82$ .<sup>2</sup>

Altogether the error log contains 865 entries so far. I've tried to analyze this data and to introduce some structure by assigning each of the errors to one of 15 categories:

- A Algorithmic Anomaly
- B Blunder, Botch
- C Cleanup for Consistency
- D Datastructure Debacle
- E Efficiency Enhancement
- F Forgotten Function
- G Generalization, Growth
- I Interactive Improvement
- L Language Liability
- M Mismatch between Modules
- P Promotion of Portability
- Q Quest for Quality

---

<sup>1</sup> The preparation of this paper was supported in part by National Science Foundation grant CCR-86-10181.

<sup>2</sup> Those changes were summarized briefly in another publication for early users [1].

- R Reinforced Robustness
- S Surprising Scenario
- T Trivial Typo

Categories A, B, D, F, L, M, R, S, T are *bugs*, which definitely needed to be removed from the code; categories C, E, G, I, P, Q are *enhancements*, which improved  $\text{\TeX}$  but were not obligatory. I consider both bugs and enhancements to be *errors*, for if I had designed a perfect system in the beginning I would not have made any of these changes and my error log would have been empty.

The most important lessons I learned can be summarized in the following theses, which my paper [6] defends and explains in detail:

1.  $\text{\TeX}$  would have been a complete failure if I had merely specified it and not participated fully in its initial implementation. The process of implementation constantly led me to unanticipated questions and to new insights about how the original specifications could be improved.
2.  $\text{\TeX}$  would have been much less successful if I had not used it extensively myself. In fact, when  $\text{\TeX}$  was new I thought of 100 ways to improve it as I was typesetting 700 pages over a period of several months, at a nearly constant rate of one enhancement per 7 pages typed.<sup>3</sup>
3.  $\text{\TeX}$  would have been much less successful if I had not put considerable effort into writing a user manual for it myself. The process of explaining the language gave me views of the system that I never would have perceived if I had merely designed it, implemented it, and used it.
4.  $\text{\TeX}$  would have been much less successful if I had not scrapped the first system and written another system from scratch, after having the benefit of several years' hindsight.
5.  $\text{\TeX}$  would have been much less successful if I had not had the voluntary assistance of dozens of people who regularly gave me feedback on how to improve everything. The network of volunteers eventually became worldwide, perhaps because I decided that  $\text{\TeX}$  should be in the public domain.
6. I recommend that everybody keep an error log such as the one I kept for  $\text{\TeX}$ . The amount of extra time required is negligible (less than 1%), and the resulting records help us to understand ourselves and our fallible natures.
7. The methodology of structured programming reduced my debugging time to about 20% of what it was under my habits of the 60s. Furthermore, structured programming gave me enough confidence in my code that I did not feel the need to test anything for six months, until the entire system was in place and ready for testing. Therefore I saved considerable time by not having to do any prototyping.
8. Although certain features of programming languages can justly be considered harmful, we should not expect that eliminating such features will eliminate our tendency to err. For example, 12 of my errors can be ascribed to misuse of **goto** statements [3]; but that accounts for only 1.4% of the total, and I also made mistakes when using **while**, **case**, **if-then-else**, etc.
9.  $\text{\TeX}$  proved to be highly reliable and portable because it was subjected to a "torture test," which is quite different from anything a sane user would write but which really tries hard to make the system fail. We should strive energetically to find faults in our own work, even though it is much easier to find assurances that things are OK.

My experiences agree well with Peter Naur's hypothesis [7] that programming is "theory building," and they strongly support his conclusion that programmers should be accorded high professional status. But I do not share Naur's unproved assertion that "reestablishing the theory of a program merely from the documentation is strictly impossible." On the contrary, I believe that improved methods of documentation (which I have called "literate programming" [4, 2]) are able to communicate everything necessary for the maintenance and modification of programs. I think it's fair to claim that more than 100 people, perhaps more than 1000, now understand the "theory" of the  $\text{\TeX}$  program after merely reading its documentation [5]. For I have seen numerous examples of electronic communications in which many people have demonstrated such knowledge by making excellent special-purpose extensions to the existing code and by giving highly appropriate advice to users.

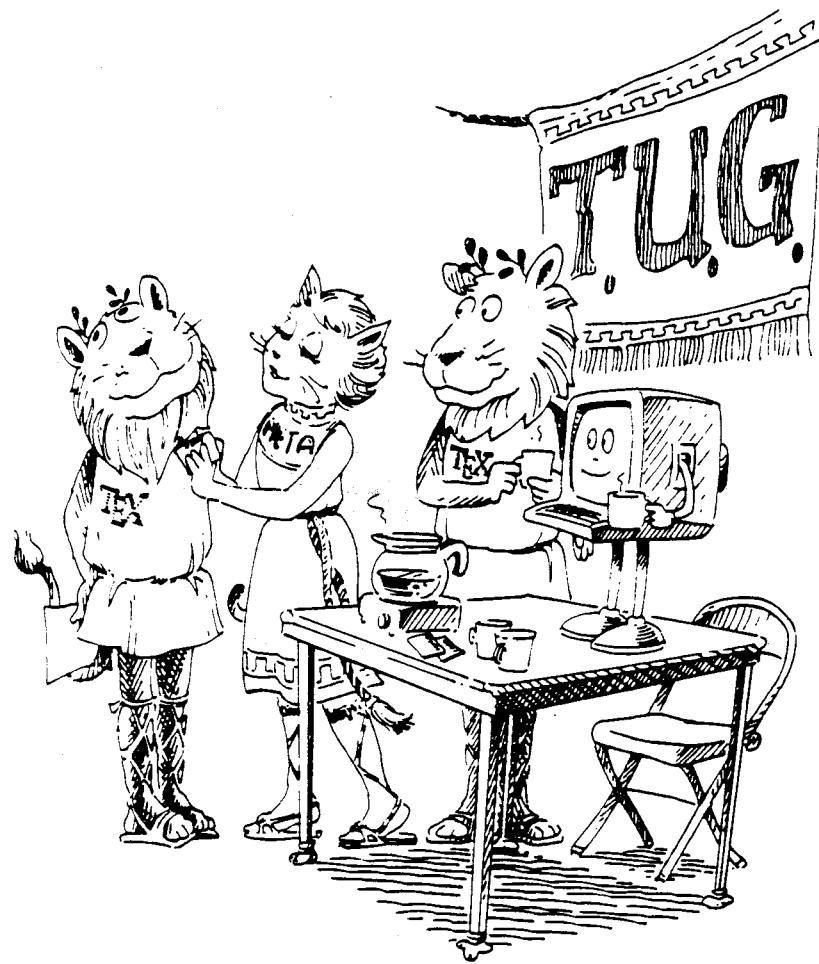
Therefore I now look forward to making further errors in my next project.

---

<sup>3</sup> On the other hand, the new ideas ceased when I went on to type hundreds of additional pages; 700 was enough! I got most of the later suggestions from other people, and I was able to appreciate them because of my own experiences.

## Bibliography

- [1] Beeton, Barbara, ed. “*T<sub>E</sub>X and METAFONT: Errata and Changes*,” (dated 09 September 11, 1983). Distributed with *TUGboat* 4, 1983.
- [2] Bentley, Don. “Programming Pearls.” *Communications of the ACM* 29:364–369, 471–483, 1986.
- [3] Knuth, Donald E. “Structured Programming with `go to` Statements.” *Computing Surveys* 6:261–301, December 1974. Reprinted with revisions in *Current Trends in Programming Methodology*, Raymond T. Yeh, ed., 1:140–194, (Englewood Cliffs, N.J.: Prentice-Hall, 1977); also in *Classics in Software Engineering*, Edward Nash Yourdon, ed., pp. 259–321, (New York: Yourdon Press, 1979).
- [4] Knuth, Donald E. “Literate Programming.” *The Computer Journal* 27:97–111, 1984.
- [5] Knuth, Donald E. *T<sub>E</sub>X: The Program*. Reading, Mass.: Addison-Wesley, 1986.
- [6] Knuth, Donald E. “The Errors of *T<sub>E</sub>X*.” *Software Practice & Experience* 19:607–785, 1989.
- [7] Naur, Peter. “Programming as Theory Building.” *Microprocessing and Microprogramming* 15:253–261, 1985.



T<sub>E</sub>X Users Group  
Tufts University, July 20-23, 1986  
P. T. Barnum Auditorium

# Of the Computer Scientist, by the Computer Scientist, for the Computer Scientist

MICHAEL DOOB

Department of Mathematics  
University of Manitoba  
Winnipeg, Manitoba  
R3T 2N2 Canada  
`mdoob@uofmcc.bitnet`  
`mdoob@ccu.umanitoba.ca`

## ABSTRACT

Having been in existence for almost ten years, is  $\text{\TeX}$  being used by the writer of mathematics who may have little or no interest in the beautiful programming properties of  $\text{\TeX}$  and only wants a tool to write papers? What about the use of  $\text{\TeX}$  in other disciplines? Is the structure of the (mark-up) language too complicated for the casual user? What can be done to help new users adjust to the extra demands of  $\text{\TeX}$ ?

This presentation is meant to motivate some discussion along these lines. In particular, the lack of appropriate learning aids is addressed. Some of the needs will be presented using the author's recently written introduction to plain  $\text{\TeX}$  (Doob 1989). In particular, the question of appropriate instructional levels and methods will be discussed.

### 1. What's the Problem?

We users of  $\text{\TeX}$  have no doubt of the beauty of our resulting documents. This feeling of beauty is allowed by the fine control available to the writer of the  $\text{\TeX}$  input file; this control enables different writers to produce documents to their own very exacting standards. The combination of the precision of  $\text{\TeX}$  viewed as a computer program and the aesthetic appeal of the resulting document is certainly a powerful attraction and part of the excitement of using  $\text{\TeX}$ . It may come as something of a shock when  $\text{\TeX}$  is met with less than total enthusiasm by others. The claim of this paper<sup>1</sup> is that a major cause of this lack of enthusiasm is the large amount of material that must be absorbed before  $\text{\TeX}$  may be used for even relatively simple jobs, and that there are insufficient tools available to ease the new user through this difficult period. If we look at the recent changes in the  $\text{\TeX}$  world, we can see why this has become increasingly significant.

One of the interesting  $\text{\TeX}$  developments during the past few years has been increasing variety of users. Only five years ago it was necessary for users of  $\text{\TeX}$  to install it themselves. This involved sending for a computer tape and having, to some extent, systems-level knowledge of a mainframe operating system. Users also had to write the device drivers necessary since they were readily available in prototype only. Until just a few years ago, the primary users of  $\text{\TeX}$  were principally computer scientists, their students, and a few other computer literate (or at least computer enthusiastic) users. *TUGboat*, the journal of the  $\text{\TeX}$  Users Group (TUG), consisted almost entirely of technical articles devoted to  $\text{\TeX}$  implementation. The use of  $\text{\TeX}$  by mathematicians was exceptional. Occasionally a mathematical preprint would be prepared using  $\text{\TeX}$ , but not much more; *AMS-TeX* was still very young. By contrast,  $\text{\TeX}$  is now commonly seen in the mathematical community. Several publications

---

<sup>1</sup> This research has been supported by NSERC grant OG0007457.

of the American Mathematical Society are published using T<sub>E</sub>X. *Mathematical Reviews*, also published using T<sub>E</sub>X, now accepts reviews from mathematicians via electronic mail in *AMS-T<sub>E</sub>X* format, and these reviews in turn are now available as a database on compact disc (see Latterner and Woolf (1989) in this issue).

We have also seen T<sub>E</sub>X used for non-mathematical purposes. The published *Proceedings* for the 1987 and 1988 Annual Meetings<sup>2</sup> contained articles showing T<sub>E</sub>X being used to typeset linguistics material (Thiele 1988), for in-house use by TV Guide (Barnhart and Ness 1988) and eventual typesetting of their weekly *TV Guide*, and even to keep a database of canine histories (Harris 1988).

Given that T<sub>E</sub>X is being used by newer and more varied groups, are there any significant problems? A further claim of this paper is that there is at least one important problem that needs to be solved; namely, the existing level of user documentation is so limited in scope that it is seriously impeding the migration of T<sub>E</sub>X from areas that are highly computer-oriented to even closely related areas. And what is the solution? Since TUG has been so active in promoting the distribution of T<sub>E</sub>X, at organizing both elementary and advanced classes at the annual meeting and elsewhere, and at communicating newer developments in T<sub>E</sub>X, and at announcing the existence of new products, it (or, rather, its members) is the natural source of the solution.

As a small step towards the solution, I have written a self-study manual called *A Gentle Introduction to T<sub>E</sub>X* (Doob 1989), which is freely available. I found the writing of this manual, the subsequent teaching from it, and reactions to it quite interesting and to some extent surprising. Some of these experiences will be interspersed within the following sections. Some of the suggestions may seem self-evident and superfluous; having looked at a number of manuals, however, I would say that they are still worth putting down on paper. There is a lot of work to be done if T<sub>E</sub>X is to be widely used; I hope that this will be a start.

## 2. From Where Are We Coming?

At the 1987 meeting of TUG in Seattle, a group of members interested in T<sub>E</sub>X training met for the first time. At that time it was (naïvely) thought that a generic introduction to T<sub>E</sub>X might be created by taking the main sections from existing manuals and glueing them together. I volunteered to gather things and to put together a first attempt at a general manual.

As the manuals arrived, it became clear that, while all had some good points, it was simply impossible to put them together. Many of them were highly site-dependent; others were obviously re-cycled lecture notes; some had clearly been put together in haste. In short, what existed was a series of learning aids that were designed for and could only be used in conjunction with a (human) tutor.

And then there is *The T<sub>E</sub>Xbook*. The amount of information in that book continues to amaze me; its organization is wonderful for the experienced user. But no book can be all things to all people. The very enthusiastic can learn all they need to know from *The T<sub>E</sub>Xbook*. For the less enthusiastic or less skilled, it is a rather unpleasant experience to try to do so. It is not a good starting point for most T<sub>E</sub>X users (as, I'm sure, it was never intended to be).

The problem has been further exacerbated by the methods TUG has used to teach T<sub>E</sub>X. By far the greatest technique has been what might be called the "university approach." A lecturer gives a course over one or two days, perhaps with a little lab time, and the student madly takes notes. In some undefined future spare time, the student is supposed to go over the notes and then know how to use T<sub>E</sub>X. Of course, this is not a good way to learn something like T<sub>E</sub>X where concomitant practice is such an important part of the learning process. And so I wrote the aforementioned self-study manual in which students could both practice and learn at their own pace.

At present, the great American T<sub>E</sub>X manual is still waiting to be written. Of course, there may be just such a manual sitting on a file server somewhere in the far reaches of the electronic net. If this is the case we need to give it more exposure. In any event, the creation of appropriate manuals that may be widely used is surely an important concern of TUG.

---

<sup>2</sup> The *Proceedings* for the TUG meetings of 1987 and 1988, although published separately, both carry a 1988 publication date. The *Proceedings* from the 1987 conference appear in the series *T<sub>E</sub>Xniques* 5, edited by Dean Guenther; the 1988 conference papers were published in *T<sub>E</sub>Xniques* 7, edited by myself -Ed.

### **3. Who Wants It?**

There would be little point in creating a new manual unless there were a need for it. It is, then, reasonable to ask if there is really any demand for such a manual, and if so, how great is it. Earlier this year, unknown to me, a one-sentence statement indicating that I had written a self-study guide appeared in the American Mathematical Society *Notices* (Barwise 1989); it also gave my electronic mail address. It was something of a surprise to me when, with no warning, dozens of requests for the manual cascaded into my mailbox<sup>3</sup>. It would seem that there are many mathematicians who would like to use *TeX* but for some reason cannot do so with the available materials. If there is this unmet demand among the mathematicians, it is safe to say that there is a similar need among other physical scientists and also among the social scientists.

Why is there such a sudden demand? In the last few years, the *TeX* world has completely changed. Up to 1985 the *TeX* user needed to be a somewhat sophisticated computer user. But once *TeX* migrated to the microcomputer and laser printer environment, it became trivial to run and print *TeX* documents. Our teaching methods in TUG have not really adapted to this change.

It should be self-evident that different audiences need different manuals. So our need is not for just one manual but, in fact, for a series of manuals, each one to target a different potential group of users. Some manuals have tried to be both an introduction for the new user and a reference manual for the experienced user; the results are not usually satisfactory. In particular, it is necessary to know the computer experience, the sophistication, and the level of knowledge of your audience before starting out.

With the large number of sophisticated TUG members, it should be possible to create such manuals. To make it a little easier, I have included in the next section a dozen suggestions that might make the writing a little easier. Needless to say, this is not exhaustive, but it might make the way a bit smoother.

### **4. A Dirty Dozen**

#### **4.1 The Right Definition (for you)**

As we mentioned before, the nature of any manual is influenced by the target group. When writing a *TeX* training manual, the target group must be well defined before starting. This group will then determine things like topics covered, examples, exercises, and even writing style.

##### **Suggestion 1: Know your user**

Many of the manuals that I received were far too broad in the attempted scope. The pace and background must be different for manuals aimed at a typical mathematician as opposed to a computer scientist. The user with other backgrounds needs to be handled still differently. The amount of space to be devoted to each topic needs to be adjusted in each case. For example, in one case it might be preferable to cover the different types of mathematical typesetting in great detail while in another, it might be better to see how to generate appropriate footnotes and endnotes. In any case, the topics covered must be controlled and appropriate. The object is not to display all that the writer knows.

A good list of different topics has been given by Childs *et al* (1988) in their discussion of possible course syllabi, and further suggestions were given by Childs (1989).

##### **Suggestion 2: Know all of the topics to be covered first**

A huge manual is truly intimidating. Although most manuals were of reasonable size, there were signs that they grew (like Topsy) as they went along. If you want to write a work that is encyclopedic and is several hundred pages long, that's fine — just don't consider it an introductory manual. If you want to write a reference card which fits on two sides of a sheet of paper, that's fine too. An introductory manual probably should be well under 100 pages. Don't be surprised if you find it

<sup>3</sup> Added in proof: the number of requests is now approaching 250.

difficult to stick to the exact number of pages; in fact you should probably plan for a 10% margin of error. But it is certainly necessary to have some kind of limit in mind.

#### **Suggestion 3: Decide on the document size first**

#### **4.2 Getting the User Going**

There is a huge hump in the beginning of the learning curve for TeX. Priority must be given to easing the new user into the intricacies of TeX. In particular, the TeX source of the your manual should be available easily, and it should be straightforward to produce hard copy at a typical TeX installation. This probably means that the source should be in one file with no auxiliary files being used. This also implies that the index and table of contents will be written directly into the document. It's possible, of course, to write the table of contents at the end of the document in one pass, but the index is more troublesome. For my introductory manual, I put in the index and table of contents directly into the TeX source file, but left the hooks in the introductory macros so that the knowledgeable user could generate new ones easily. And it's better to avoid documents that require two passes of the TeX program to generate and read auxiliary files, since this tends to have site-dependent restrictions.

#### **Suggestion 4: Make it easy for the user to get a hard copy**

It is inevitable that some site-specific information will be needed. For the new user, it is hoped that this is kept to a minimum. In any case, if at all possible it should be in one place, either as a separate section or as an appendix. This allows for greater portability of the document; it also lets the reader know what is part of TeX and what is part of the site implementation of TeX. It's easy for the new user of TeX to confuse them.

#### **Suggestion 5: Isolate site-specific instructions**

#### **4.3 How and What To Include?**

Now for the real nitty-gritty. In planning an introductory TeX manual, it is often helpful to use the analogy of the structure of a foreign language text. A first-year text of this sort has rather limited goals. There is just enough information given to start using the language. It is certainly not expected that the student will write sonnets after the first few weeks of study. An introductory TeX manual must also have limited goals. As with a foreign language text, sections need to cover a relatively small amount of material, with successive sections being cumulative. Similarly, when material from different sections is used for a new idea, this should be explicitly pointed out.

#### **Suggestion 6: Make evident the interdependency of different sections**

The first steps to learning TeX are rather difficult. Parks (1988) and McCaskill (1988) have both noted that it may take extended periods of time before a real facility is attained. No one would expect a person to learn a foreign language without reviewing each lesson carefully. Normally this is done with exercises, translations, and tables.

#### **Suggestion 7: Have lots of exercises and examples**

It is useful to have exercises which review material from the immediately previous section, as well as new concepts. Later exercises in one section can reinforce the earlier exercises in the same section. The point of the exercise should be clear to the student.

It is, of course, necessary to give lots of examples. When doing so, it is useful to have a `\verbatim` or `\literal` macro that will allow you to use exactly the same code for the example and the listing that generated the example. In a surprisingly large number of cases, examples were given where the supposed TeX input source was wrong. This is, of course, disastrous.

In a foreign language text there are often tables of the following sort:

Present tense of the verb *to be*:

	Singular	Plural
1st person	I am	We are
2nd person	You are	You are
3rd person	He/she/it is	They are

Such tables give an overview of the material covered and are often useful for TeX manuals, too. They also serve as a useful quick reference once the reader has mastered the material.

Here is an example of a similar table from the beginning of my manual. It follows a discussion of characters that have special purposes in the TeX input file.

#### Characters requiring special input

Character	Purpose	Input for literal output
\	Special symbols and instructions	\$\backslash\$
{	Open group	\$\{\$
}	Close group	\$\}\$
%	Comments	\%
&	Tabs and table alignments	\&
-	Unbreakable space	\-{}-
\$	Starting or ending math text	\\$
^	Math superscripts	\-{}^
-	Math subscripts	\-{}_
#	Defining replacement symbols	\#

#### Suggestion 8: Use tables to recapitulate sections

As mentioned before, it is important that a manual have definite limits to its scope. It's quite useful to let the reader to know where to get further information when desiring to go beyond that scope. Not surprisingly, *The TeXbook* is often the most useful reference. Here is a macro that I used to point the reader towards further information:

```
%%%%%%%
% macro to put TeX references in right margin %%%%%%
\newdimen\theheight
\def \TeXref#1{%
    \vadjust{\setbox0=\hbox{\sevencrm\quad\quad\TeX book: #1}}%
    \theheight=\ht0
    \advance\theheight by \dp0      \advance\theheight by \lineskip
    \kern -\theheight \vbox to \theheight{\rightline{\rlap{\box0}}}\%
    \vss}%
}%
%%%%%%
```

Then `\TeXref{1--9}` will cause a reference to appear in the right margin (now look to the right). The macro is obviously easy to adapt to different references.

#### Suggestion 9: Tell your reader where to find more information

Sometimes it is better to give an explanation that is not fully complete than to get lost in details that are not important. By mathematical analogy, consider two possible definitions of the real numbers: (1) real numbers are all possible decimals, and (2) real numbers are all possible greatest lower bounds

of sets of rational numbers. The first definition is (at best) rather loose and glosses over details such as non-uniqueness and repeating decimals; however, it does have the advantage of having a direct connection to familiar objects. The second definition is mathematically better than the first, but without the insight of the truncated decimals of the first definition, the second one has less intuitive understanding. When introducing a new topic, it is sometimes better to bend the truth a little (needless to say, it is hoped that the bending of the truth will be intentional). In particular, it is necessary to set a depth to which the material will be presented and then to consistently work to that level. As an example, in the manual I wrote, there is no mention of horizontal or vertical modes. Now it is obvious that a lack of knowledge about modes is *very* limiting. It is also clear that many error messages will be totally incomprehensible (so what's new?). Some topics, vrules and hrules for example, can only be discussed in a rough way, much as with the first definition of the real numbers. Nonetheless, in my (somewhat arbitrary) opinion, the neophyte TeX user can learn how to do lots of good documents without knowing about modes. Many decisions of this type were necessary for my manual; they require more time and thought than might be expected.

**Suggestion 10: Some judicious fibs are O.K.**

#### 4.4 A Few Final Comments

As has been mentioned earlier, it is important to have many exercises; as with learning a foreign language, it is necessary to *do* as well as *to read*. Give some thought as to whether or not the answers should appear directly after the exercises. It is evident to me that the typical reader wants them there; whether or not it is the best thing to do is a matter of debate. It really depends on whether the perspective of the manual is to be that of a teaching document or that of a reference document. There are at least two other possibilities: the answers can be put in an appendix, perhaps with a switch to determine whether or not they are actually printed, or they can be put into the TeX source code.

The TeX source code is also instructive. You may be assured that any macros that you write will be observed at some time by the readers of the manual. It is worth the trouble to comment your macros well.

**Suggestion 11: Make your TeX source code self-documenting**

The easiest way to distribute the TeX is via electronic mail. In principle, this should pose no problem; one of the strong points of TeX is that it uses only standard characters. My experience is that there are (EBCDIC-to-ASCII) problems. In one case, all the curly brackets disappeared (I leave it to the reader to imagine what this did to the output). It is prudent to give a sample set of characters as a comment at the beginning of your source file; sometimes it is possible to reverse the damage by simple global editing. I do something like the following:

```
%% Thanks for your interest in A Gentle Introduction to TeX. At the
%% moment it is in draft form, i.e., subject to correction, but it
%% is pretty close to the final copy. Any comments on this manual
%% would be appreciated: these may be typesetting, English, or
%% TeX criticisms.
%%
%%%                         Michael Doob
%%%                         Department of Mathematics
%%%                         The University of Manitoba
%%%                         Winnipeg, Manitoba R3T 2N2
%%%                         Canada
%%%                         mdoob@uofmcc          (bitnet)
%%%                         mdoob@ccu.umanitoba.ca    (internet)
%%
%% Here is a character listing to check to be sure that no
```

```
%% unwanted translations took place within the bowels of the net.  
%% Upper case letters: ABCDEFGHIJKLMNOPQRSTUVWXYZ  
%% Lower case letters: abcdefghijklmnopqrstuvwxyz  
%% parentheses, brackets, braces: () [] {}  
%% Exclaim, at, sharp, dollar, percent: ! @ # $ %  
%% Caret, ampersand, star, underscore, hyphen: ^ & * _ -  
%% vertical bar, backslash, tilde, backprime, plus: | \ ~ ' +  
%% plus, equal, prime, quote, colon: + = ' " :  
%% less than, greater than, slash, question, comma: < > / ? ,  
%% period, semicolon: . ;  
%%
```

#### Suggestion 12: Make your source code available via electronic mail

There are other manuals waiting to be written. We have already seen that there is a need for different elementary manuals. More advanced ones are non-existent. Most of the suggestions for elementary manuals also apply to advanced ones. Would someone, for example, like to write one on how to write output routines (that work)? How about introductory *TEX* wizardry? And maybe one on efficient *TEX* programming? How about *METAFONT*? I feel that I've done my part by writing an introductory manual — what about the rest of you TUG members?

### Bibliography

- Barnhart, Elizabeth, and David Ness. "Layout for *TEX*." *TEXniques* 7:97–115, 1988.
- Barwise, Jon. "Computers and Mathematics." *Notices of the American Mathematical Society* 36:241–243, 1989.
- Childs, S. Bart. "Teaching *TEX*" *TUGboat* 10(2):156–163, 1989.
- Childs, S. Bart *et al.* "Syllabi for *TEX* and *METAFONT* Courses." *TEXniques* 7:117–127, 1988.
- Doob, Michael. *A Gentle Introduction to *TEX**. Available from author, 1989.
- Harris, Robert L. "Using *TEX* to Produce Kennel Club Yearbooks." *TEXniques* 7:97–115, 1988.
- Knuth, Donald E. *The *TEXbook**. Reading, Mass.: Addison-Wesley, 1984.
- Latterner, Dan, and W.B. Boolf, "*TEX* at *Mathematical Reviews*." *TUGboat* 10(4):639–654.
- McCaskill, Mary K. "Producing NASA Technical Reports with *TEX*." *TEXniques* 7:1–10, 1988.
- Parks, Berkeley. "*TEX* Tips for Getting Started." *TEXniques* 7:129–138, 1988.
- Thiele, Christina. "*TEX*, Linguistics, and Journal Production." *TEXniques* 5:5–26, 1988.



**TEX Users Group**  
**University of Washington**  
**August 23 - 26, 1987**

# Mastering T<sub>E</sub>X with Templates

HOPE HAMILTON

National Center for Atmospheric Research  
P.O. Box 3000  
Boulder, Colorado 80307  
[hamilton@mmm.ucar.edu](mailto:hamilton@mmm.ucar.edu)

## ABSTRACT

Large scientific laboratories and government research sites, with from 1,000 to 5,000 employees, are the natural domain of T<sub>E</sub>X. However, the degree of secretarial (i.e., key-boarding) motivation for learning T<sub>E</sub>X in such an environment, is far below what is generally found in business, commerce, and industry. One such national laboratory, NCAR, has avoided near mutiny by the support staff and decreased learning reluctance by providing on-line templates for all frequently used types of correspondence. Statistics of the learning curve, comparisons of teaching techniques, and examples of templates illustrate a hard-earned victory for T<sub>E</sub>X, at long last the preferred scientific word processing software of NCAR scientists as well as support staff.

### 1. Introduction

The National Center for Atmospheric Research (NCAR)<sup>1</sup> is located in Boulder, Colorado. Its origins, some twenty-five years ago, involved a small group of scientists — physicists, mathematicians, engineers — whose common interest was the natural phenomena of the atmosphere and oceans. It was their hope that, by collaborating and combining efforts and expertise, they could work toward a better understanding of those aspects of weather that endangered lives and destroyed property. Inherent in this mutual goal was the critical need to improve weather prediction, for the benefit of mankind. This national atmospheric research center was, and is, sponsored by the National Science Foundation in Washington.

Today, NCAR has close to 1,000 employees. Its most outstanding product takes the form of scientific papers, carefully refereed by colleagues throughout the world, and published in some twenty monthly American and international journals. These papers and articles must be produced, frequently in camera-ready form, by secretaries and clerical personnel, most of whose word processing knowledge began with the IBM electric typewriter.

### 2. T<sub>E</sub>X Arrives

Four years ago, T<sub>E</sub>X came to the attention of several NCAR scientists. Admired for its typeset appearance and praised for its ability to produce any and every type of mathematical equation needed for technical research reports, T<sub>E</sub>X was acquired and installed on the VAXes of several scientific groups. A few scientists with programming background began to use it. None of the support staff, i.e., secretaries, had seen or heard of T<sub>E</sub>X.

One of the more creative scientists placed a copy of *The T<sub>E</sub>Xbook* on my desk with the comment "See how you like this word processing program?" I scanned through the book, made a mental note to try to make some sense of it if and when I had the time, and returned to our state-of-the-art stand-alone workstations, Micoms and NBIs.<sup>2</sup> Returning a week later, the scientist inquired if I had started using T<sub>E</sub>X in our office production. I stammered: "No, not yet." I had been too embarrassed to follow up

<sup>1</sup> The National Center for Atmospheric Research is sponsored by the National Science Foundation.

<sup>2</sup> Two years earlier, the IBM Selectric had reigned supreme; and many long-term secretaries still resisted the upgrade to the large and cumbersome stand-alones.

with "Why should I? I can't understand the book. There's no one to help me. I'm not a programmer or a computer scientist." But the episode remained in my mind as a challenge. I decided to see our VAX system manager. He gave me brief instructions, taught me a few VAX DCL commands, showed me the location of the new laser printer, and promised to bring a terminal to my office.

The struggle began! I managed at last to hammer out a letter that resembled, lamely, the crisp NBI format we had all become accustomed to. I had no experience with which to equate my knowledge of typewriters and NBIs to the commands and output of *T<sub>E</sub>X*. Margins? Double spacing? Tabs? Indented paragraphs? *Equations*? And I had no one to ask.

I saved that first letter and stapled it to the input page of *T<sub>E</sub>X* commands. Grimly, I twisted and tweaked *T<sub>E</sub>X* into giving me a two-page memo. An abstract containing several equations finally appeared, to my amazement. My collection of samples increased. But it was painfully slow. At the slightest hint of a one-hour deadline, I returned to my simple and predictable NBI.

A breakthrough occurred about six months later. I met a secretary, Eileen Boettner, from another NCAR site seven miles away. We discovered that we had reacted to the *T<sub>E</sub>X* challenge in the same way. My new friend had also saved her samples, and we began to discuss our *T<sub>E</sub>X* problems. Each of us, in her own location, had become a resource to other secretaries. Our samples were hopelessly chained to typewriter terms and NBI keyboard procedures. Glue? Boxes? Modes? Those concepts were obviously meant for programmers and hackers. We would have to manage without them.

Management soon began to make it clear that *T<sub>E</sub>X* was highly desirable, if not absolutely necessary, for all scientific output.

### 3. *T<sub>E</sub>X* Templates Arrive

My friend and I, remembering how hard it had been for us to produce the most primitive correspondence in *T<sub>E</sub>X*, combined our samples into an NCAR technical report. By sharing copies, we could make it easier to guide others through those early months of discouraging trial and error.<sup>3</sup> We organized our material along the lines of NCAR's needs and uses. We created a table of contents; we color-coded the examples and their hard-copy input pages. Every example or template was given a number. Each template appeared in its final output form, attached to which was the corresponding page of *T<sub>E</sub>X* commands that produced it. Our "Index," as we titled it, quickly became an in-house best seller! Lines formed outside of our offices; numerous phone calls and mail requests interfered with our regular work. To date, our 350-page effort — primitive through it is — has undergone three printings totalling about 5,000 copies. Many users have reported that they, in turn, have made countless photocopies for their friends, staff, and colleagues. Requests have been filled from users as far away as Denmark, Australia, China, Japan, Germany, and Brazil.

We asked our VAX system personnel to make the Index available electronically. For example, if Item 15, "References", was needed by a user, he or she "copied" the corresponding template into his directory from the system. This eliminated the much dreaded keyboarding of a series of confusing — and impossible to debug — *T<sub>E</sub>X* commands, with the obvious certainty of typos and mistakes. It was simple to delete the words on the example and enter the current words. Users were much too mystified by the ubiquitous commands to touch anything preceded by a backslash. Our VAX system manager began to make tapes of the Index and mail them to users in universities and institutions upon request. To date, at least 6,000 *T<sub>E</sub>X* users have been able to start using *T<sub>E</sub>X* without computing or programming knowledge but, unfortunately, without understanding some of *T<sub>E</sub>X*'s unique and basic concepts.

As Management gave *T<sub>E</sub>X* the executive nod of approval, rumors of discontent from the tenured, senior secretaries were rampant. At several sites of our organization, groups of secretaries met; they aired and drafted their complaints. The core of this discontent centered around the fact that there had been no mention of re-classifications or salary increases for the staff who felt "pushed" into learning this difficult new word processing software. *T<sub>E</sub>X* obviously required skill and capabilities that had not been requisites in the traditional secretary's job description. *T<sub>E</sub>X* was different, better, and only programmers were able to master it. In short, the gauntlet thrown down was "We'll learn *T<sub>E</sub>X* if Management makes it worth our while."

---

<sup>3</sup> A *T<sub>E</sub>X* preview device was not available at our research site during this time.

NCAR's budget at that time did not include mass salary raises for the support staff. Management's hope may have been that TeX's efficiency and speed would reduce the number of support staff needed. To complicate the issue, more scientists could now TeX their own lengthy, equation-filled papers. The secretaries' complaints and petitions were fended adroitly, and TeX began to appear on job descriptions as a Required Skill instead of an Optional Skill. Motivation to learn the new software, however, was visibly lagging.

Meanwhile, news of the Index began to spread by word of mouth: a typist's tutorial for TeX—no formal training nor programming experience necessary. The scientists had enthusiastic praise for anything produced in TeX by the staff. TeX's superior output resulted in an elitist competition between scientists and supervisors. "My research paper is being produced in TeX." "My office is now using TeX for all output." "Our secretary is leaving to get married. We need to replace her with *someone who knows TeX*."

At this point, many doubts and questions began to give me increasing concern. Was the Index a satisfactory vehicle for teaching TeX? Did the Index succeed in motivating nonscientific personnel to learn TeX? Could this Index be used as effective documentation for learning TeX? My answer, as would be the answer of any experienced teacher, was "If it works, use it."

The title of this paper is "Mastering TeX with Templates." I have three years of experience that back up my claim that anyone without computer or programming experience can learn TeX by using a system of on-line templates. An important provision states that the examples, or templates, be of high-frequency use within an organization. Either Preview software or a nearby printer, preferably both, are highly desirable. The Index has the advantage of not being unnecessarily verbose, such as the conversational style and running format we find in traditional software documentation. Folksy and lengthily explanations of commands and exceptions serve only to frustrate the beginner. The oft-seen documentation breach of avoiding any representation of what appears on the user's screen is side-stepped by the input/output pages of the Index. Another critical factor for the learner is his environment of stress or pressure; if an impatient supervisor is anxiously awaiting the output, no learner can be receptive to new TeX concepts and detailed explanations.

#### 4. TeX by Osmosis

Inevitably there are calmer, quieter hours in the secretary's day. Aware that **h** represents horizontal and **v** vertical, the learner's perception of the meaning of all commands beginning with **h** and **v** can result in cautious explorations:

"Change \vskip1truein to \vskip2truein if twice the space is needed. An \hfill can shift the location of the date in a line, depending on its placement."

A sense of control rewards the keyboarder who timidly experiments with variations in the baselineskip. Commands such as \hsize/\vsize, \hfill/\vfill, \eject, \centerline, \indent, \hskip/\vskip, \nopagenumbers, \raggedright, and \hoffset/\voffset are not difficult to understand. Greek letters and mathematical symbols are as close as Appendix F in *The TeXbook*. \obeylines and \obeyspaces are a boon to those who still wistfully hold out for WYSIWYG. \settabs are a welcome discovery when problems arise with the cc: after the closing of a letter. Font changes may require a visit (but only once) to the system manager, and aligned columns in tables may possibly be the last frontier in TeX mastery.

With the Index in hand, beginners can start TeXing without delay, and this early success is the most motivating factor of all. There is a maxim that all teachers heed: No one can learn until he or she is ready to learn. And as far as learning TeX is concerned, a learner is not ready until he wants to learn.

In my experience, the language itself used in TeX commands encourages the learner to experiment with altering the templates to serve his needs. This kind of learning is self-rewarding. An encouraged learner can evolve into a curious, audacious learner who may even decide to defer his doubts to the heretofore neglected *TeXbook*. The occasional familiarity of commands found within its pages continues the cycle of curiosity and reward. In a matter of weeks, the learner is ready to consider the more discrete concepts. It occurs to him that a formal TeX class might be helpful. He has unconsciously begun to shed the old typewriter habits; he realizes that the flexibility and power of TeX can elevate him to

levels and standards of production that will be recognized and praised by peers and supervisors.

## 5. Conclusion

If you have yet to succeed in motivating reluctant personnel to break away from out-moded word processing methods and begin experimenting with TeX, I suggest that you consider this approach. Tenured administrative staff, because of a lack either of self-confidence or of technical education, can surprise us by their enthusiastic support of TeX. Extend such a opportunity to them. Meet them half-way with a collection of templates based on familiar output. There may be costly consequences, however. It may be necessary later to budget for their advanced TeX classes. In time, there may also be a need to cover all of their expenses to annual TUG meetings.

## Bibliography

- Boettner, Eileen, and Hope Hamilton. *Definitive NCAR Index for TeX for NCAR Scientists*. Boulder: National Center for Atmospheric Research, NCAR Technical Note, NCAR/TN-266+1A, 1986.
- Knuth, Donald E. *The TeXbook*. Reading, Mass.: Addison-Wesley, 1984.

## Appendix A: Customized TeX Templates

The following table of contents reflects my organization's word processing activities:

Abstracts (See Manuscripts)	Top, centered
Contents, Table of (See Tables)	Begin with Introduction
1. Equations	Roman and arabic
Within text (see also Manuscripts)	Discretionary
Centered/displayed	Corner
2. Figure Captions	Book/chapter headings
Examples	
Fonts (see Typefaces)	
3. Footnotes	15. References
Examples	Examples
Forms (see Reviews, Spacing, Tables)	Résumés (see Vitae)
4. Format/Default	16. Reviews
TeX's built-in settings/no commands necessary	Manuscript/article
(See also Spacing, Letters)	NSF
Hyphens (see Spacing)	SCD
5. Indenting	17. Seminars
Paragraphs	Examples
Quotations/narrower margins	18. Spacing
Numbered items	Horizontal
Outline	Centering
Columns (see also Lists, Tables)	Flush left/right
6. Letterheads	Between words/letters
7. Letters	Word break/hyphen
Business (letterhead included)	Vertical
Business (letterhead, address, cc:,	Single/double
2nd page heading)	Between paragraphs
Business (multiple addresses)	Line skipping/page break
Business (use with letterhead)	Examples
Personal	19. Tables (see also Tabs)
cc: (see Tabs)	Budget
8. Lines/underlines/overlines	Manuscript
9. Lists (see also Tabs)	Ruled
10. Manuscripts	Typeset
Journal publication	Forms
Conference, camera-ready	Flow Charts
Journal, camera-ready	Routines
AGU abstract	Contents
Book, typeset,	20. Tabs (see also Tables)
11. Margins (see also Format)	Transparencies (see View-graphs)
Examples	21. Title Pages
12. Memos	Examples
Examples	
13. Minutes	23. View-graphs
Notices (see Seminars)	Examples
14. Page Numbers	24. Vitae, Curriculum
Bottom, centered	Examples
No page numbers	APPENDICES
Roman and arabic	A. Greek/symbols
Discretionary	B. Error Messages
Corner	C. Quick Command Shortcuts

## Appendix B: Samples of a Title Page and a Memo

The examples that follow illustrate early steps in the transfer from typewriting or stand-alone word processor usage to simple T<sub>E</sub>X output.

### Example 1: Title Page

Output:

# AIRBORNE LASER AND DOPPLER RADAR SYSTEMS

## PART II: DESIGN CRITERIA

by

K. L. Rolff

Arles National Laboratory, Denmark

and

D. L. Adrian

National Center for Atmospheric Research\*  
P. O. Box 3000, Boulder, CO 80307

Input:

```
\vskip.3truein
\centerline{\bf AIRBORNE LASER AND DOPPLER RADAR SYSTEMS}
\vskip .2truein
\centerline{\bf PART II: DESIGN CRITERIA}
\vskip .4truein
\centerline {by}
\vskip .26truein
\centerline {K. L. Rolff}
\vskip .17truein
\centerline {Arles National Laboratory, Denmark}
\vskip .3truein
\centerline {and}
\vskip .3truein
\centerline {D. L. Adrian}
\vskip .17truein
\centerline {National Center for Atmospheric Research\footnote{*{The National
Center for Atmospheric Research is sponsored by the National Science
Foundation.}}}
\centerline {P. O. Box 3000, Boulder, CO 80307}
```

---

\* The National Center for Atmospheric Research is sponsored by the National Science Foundation.

**Example 2:** Memo

Output:

1 October 1987

MEMO TO: Algernon P. Esworthy  
FROM: Mary R. Stuart  
SUBJECT: Definitive NCAR Index for TeX

This *Index* does not pretend to be an extension of Donald Knuth's comprehensive *TeXbook*. It contains, instead, a condensation of the text-formatting output most frequently used by the scientific community of NCAR. Also, as a special *lagniappe*, each example includes its own NCAR computer location; scientists can access and thus copy our "templates" to their directories where they can be used and altered to accommodate their needs.

The authors thank our many TeX-using friends who have shared their discoveries — serendipitous or otherwise. Their generosity has contributed greatly to the wide range of examples included in the Index.

End of Memo

cc: W.C. White  
University of Illinois

Input:

```
\parskip=4pt
\parindent=30pt
\nopagenumbers
\raggedright
\hsize=5truein
\vsize=6truein
\hfill {1 October 1987}
\vskip .2truein
\settabs\+\noindent&MEMO TO: \qquad&\cr
\+\&MEMO TO: &Algernon P. Esworthy \cr
\medskip
\+\&FROM: &Mary R. Stuart \cr
\medskip
\+\&SUBJECT: &Definitive NCAR Index for \TeX \cr
\bigskip
```

This {\it Index} does not pretend to be an extension of Donald Knuth's comprehensive {\it \TeX book}. It contains, instead, a condensation of the text-formatting output most frequently used by the scientific community of NCAR. Also, as a special {\it lagniappe}, each example includes its own NCAR computer location; scientists can access and thus copy our "templates" to their directories where they can be used and altered to accommodate their needs.

The authors thank our many \TeX -using friends who have shared their discoveries --- serendipitous or otherwise. Their generosity has contributed greatly to the wide range of examples included in the Index.

```
\bigskip
\centerline {End of Memo}
\vskip .3truein
\settabs +\noindent &cc: \quad & \cr
\+&cc: &W.C. White \cr
\+&& University of Illinois \cr
```

# Using WordPerfect 5.0 to Create $\text{\TeX}$ and $\text{\LaTeX}$ Documents

ANITA Z. HOOVER

University of Delaware  
002A Smith Hall  
Newark, DE 19716  
[anita@vax1.acs.udel.edu](mailto:anita@vax1.acs.udel.edu)  
ACS03174@UDACSV

## ABSTRACT

This paper addresses the issues associated with using WordPerfect 5.0 on PCs to create  $\text{\TeX}$  and  $\text{\LaTeX}$  documents. WordPerfect macros and keyboard layouts are used as an interface to aid in the input process of  $\text{\TeX}$  and  $\text{\LaTeX}$  control sequences. As a result, a major portion of this paper is devoted to discussing the basic concepts of WordPerfect macros, keyboard layouts, and the macros specifically developed to produce  $\text{\TeX}$  and  $\text{\LaTeX}$  control sequences.

### 1. Background

In 1984, the University of Delaware's administration decided that WordPerfect would become the faculty's standard word processing package, and so they initiated a grant program which allowed faculty to purchase IBM PCs with WordPerfect at a considerable savings. By 1987 a similar decision had been made for the secretarial staff; near the end of 1988 the actual implementation of this decision was completed, and WordPerfect became the standard word processing package for PCs throughout the University.

During the last year, however, many departments, such as Engineering, Computer Science, Math and Physics, realized that producing technical documents was beyond the scope of WordPerfect, especially those documents that contained complicated mathematics. Many faculty members in these departments had heard of the high-quality mathematical typesetting that  $\text{\TeX}$  and  $\text{\LaTeX}$  produced and thought that either of these packages would be an appropriate tool for producing their documents. It was this interest that brought  $\text{\TeX}$  and  $\text{\LaTeX}$  to the PC environment here at the University of Delaware.\* But the problem was to train the secretaries to use these packages. As usual, documents needed to be typed by yesterday, and the secretaries became overwhelmed and frustrated by having to learn both WordPerfect and the proper control sequences for  $\text{\TeX}$  and/or  $\text{\LaTeX}$  in a hurried manner. Many questions began to surface, for example, "Why do I have to type  $\backslash\alpha$  to get  $\alpha$  when I used to type only Alt-A on my EXXON system?"

In trying to answer such questions, I realized that WordPerfect macros could be used to help speed up the learning and typing process by automatically providing the basic control sequences for setting up a  $\text{\TeX}$  or  $\text{\LaTeX}$  document. The more common math control sequences, such as those that produce lowercase and uppercase Greek characters, are defined in a WordPerfect keyboard layout that contains a number of predefined macros. I should note that it is *not* my intention to tell a new user of  $\text{\TeX}$  or  $\text{\LaTeX}$  to go out and buy a PC with WordPerfect in order to use the macros I developed to produce  $\text{\TeX}$

\*In trying to reproduce this paper at other sites, it was brought to my attention that it would not  $\text{\TeX}$  successfully because the **save size** was exceeded. Our site requires a large version of  $\text{\TeX}$  for producing books, but I did not think this paper would cause any problems. However, upon closer examination I realized that the figures used to create the keyboard layouts (4 and 5) proved to be quite complex and thus required a **save size** of 800 or larger. I decided to use our time-sharing system Vax1 (VAX 8650 running version 4.3 Berkeley UNIX) to make communication easier for submitting this paper, so I have not actually run this on a small system and therefore you may experience difficulties due to memory and processor speed limitations.

and  $\text{\LaTeX}$  control sequences. However, these macros do provide new users of  $\text{\TeX}$  and  $\text{\LaTeX}$ , already familiar with WordPerfect, with a good starting point for creating high-quality technical documents in  $\text{\TeX}$  and/or  $\text{\LaTeX}$  format.

## 2. WordPerfect Macros

A WordPerfect macro provides a way of assigning a name to a series of keystrokes you often repeat. If you need to perform the same operation frequently, you can enter the keystrokes once and save them in the form of a macro. When you want to repeat the operation at a later time, you need only tell WordPerfect the name of the macro (the file where the keystrokes are stored), and WordPerfect will “replay” the keystrokes, automatically performing the task. This concept is common to most software packages that provide macros. For example, WordPerfect macros work much like  $\text{\TeX}$  and  $\text{\LaTeX}$  macros, which are shortened forms of your most commonly used control sequences or combined control sequences that you use repeatedly. So it seems that putting  $\text{\TeX}$  and  $\text{\LaTeX}$  control sequences into WordPerfect macros is not as unrelated as it may have appeared upon first inspection.

It is quite easy to create a WordPerfect macro if you follow three basic steps:

1. Plan the macro.
2. Save any files before you start writing the macro.
3. Write the macro.

### 2.1 Plan the Macro

You should have a clear idea of what you want the macro to accomplish before actually writing the macro. If you are creating a simple macro, such as one that produces  $\backslash\alpha$ , only a few keystrokes may be required. However, if you are creating a more complicated macro, such as prompting the user for an environment name in  $\text{\LaTeX}$  in order to produce the output shown in Figure 1, it becomes more important to write down the basic steps and actions necessary in order for the macro to work correctly in both new and already existing documents.

```
\begin{environment name entered}  
-  
\end{environment name entered}
```

Figure 1: Macro output

### 2.2 Save the Current File

As you create a macro, the functions you use are actually performed on the text on your screen. WordPerfect is a WYSIWYG word processor, so while you can create a macro on a blank screen, having a document on the screen enables you to see the effects of each function you include in the macro. Since your text will be altered by the keystrokes you enter, you should always save your file before creating a macro.

### 2.3 Write the Macro

1. Press the **Macro Def** key (Ctrl-F10)<sup>†</sup> The message Define Macro: appears in the bottom-left corner of the screen.

<sup>†</sup>Conventions used in this document: Text appearing in **typewriter** font refers to text you type or a key sequence you press, text appearing in **bold** is a descriptive name of a macro or key definition. Remember, what one writes is not always what one means, so I hope you will not find **typewriter** font where **bold** should be and vice versa. Just keep in mind that when you see the word hold, press, or type, then I want you to carry out the action that follows, otherwise I am just referring to that macro or key definition by name.

2. Name your macro; this can be done in two ways:

- (a) Type a name 1–8 characters long (conforming to the standard rules for naming DOS files) and then press the **Enter** key. This will create a file name with the extension **.WPM**. All named macros for this project were of the form **M-xxxxxx**. This was done in order to eliminate confusion with user-defined macros. For example, **m-be** is a named macro, stored in the file **M-BE.WPM**, that prompts the user for an environment name and produces the output in Figure 1; see Figure 2 for the description and WordPerfect codes of this macro.

Many macros can be created using this method. For this project, named macros were created for the following circumstances:

- If the control sequence is more than three characters long, including the backslash character (\).  
The reason for limiting what control sequences should be defined as named macros is based on the naming convention chosen for this project. If the control sequence name is too short, it is not advantageous to have to type a longer name for the macro.
- If the control sequence is made up of two separate control sequences, as in a **LATEX** environment; for example, **\begin{equation}** followed by **\end{equation}**.  
This would save time typing, plus help prevent errors which arise when an ending environment control sequence is forgotten.

See Table 1 for a listing of the named macros defined for this project and a short description of each.

- (b) Hold down the **Alt** key and press a single letter to create an **Alt** macro. This will create a file name with the extension **WPM**. For example, if you were to hold down the **Alt** key and press the letter **A**, then you would create the macro **Alt-A** and the file name would be **ALTA.WPM**. The basic lowercase Greek characters were assigned to the **Alt** macros in this project. For example, **Alt-A** is a macro that produces the **\alpha** control sequence. Such key assignments are very common on multi-function keyboards. See Figure 3 for the description and WordPerfect codes of this macro.

3. Enter a description of the macro (up to 39 characters). Most macro descriptions for this project stated whether the macro was designed for **TEX** and/or **LATEX** and the control sequence(s) it produces.

#### 2.4 Comments

Using the naming method described above, all lowercase Greek characters were defined as **Alt** key macros (**Alt-A**... **Alt-Z**). Therefore, uppercase Greek characters had to be given 1–8 character names. However, it became obvious that a WordPerfect keyboard layout could expand the current number of single-keystroke macros. By using a WordPerfect keyboard layout, you can assign macros to the **Ctrl** key as well as the **Alt** key. Therefore, the uppercase Greek characters could be assigned to the **Ctrl** keys, plus other frequently used **TEX** and **LATEX** control sequences could be assigned to the other available **Alt** and **Ctrl** keys (for example, **Alt-**; and **Ctrl-\**). In order to proceed with this task, we need to examine the following:

- What is a keyboard layout?
- How do you create a keyboard layout?
- How does a keyboard layout work?

Macro: Edit

File C:\MACROS\M-BE.WPM

1 - Description LaTeX \begin{env name}... \end{env name}

2 - Action

```
{DISPLAY OFF}
{ASSIGN}0~~
{TEXT}0~Enter environment name: ~
{End}{Enter}{Del}{Up}{End}{Enter}
\begin{{VAR} 0}
{Enter}{Enter}
\end{{VAR} 0}
{Home}{Left}{Up}
{DISPLAY ON}
```

Selection: 0

Figure 2: WordPerfect macro for any L<sup>A</sup>T<sub>E</sub>X environment control sequence

Macro: Edit

File C:\MACROS\ALTA.WPM

1 - Description TeX and L<sup>A</sup>T<sub>E</sub>X \alpha

2 - Action

```
\alpha
```

Selection: 0

Figure 3: WordPerfect macro for TeX and L<sup>A</sup>T<sub>E</sub>X \alpha control sequence

$\backslash\Gamma$	$\Delta$	$\leftarrow$	$\Theta$	$\rightarrow$	$\Upsilon$	$\Xi$	$\uparrow$	$\downarrow$	$\ell$
$\mathbf{Q}$	$\mathbf{W}$	$\mathbf{E}$	$\mathbf{R}$	$\mathbf{T}$	$\mathbf{Y}$	$\mathbf{U}$	$\mathbf{I}$	$\mathbf{O}$	$\mathbf{P}$
$\gamma$	$\delta$	$\epsilon$	$\theta$	$\tau$	$\upsilon$	$\xi$	$\iota$	$\circ$	$\rho$
$\nabla$	$\Sigma$	$\Phi$	$\approx$	$\Lambda$	$\hbar$	$\Pi$	$\leftrightarrow$	$\Omega$	
$\mathbf{A}$	$\mathbf{S}$	$\mathbf{D}$	$\mathbf{F}$	$\mathbf{G}$	$\mathbf{H}$	$\mathbf{J}$	$\mathbf{K}$	$\mathbf{L}$	
$\alpha$	$\sigma$	$\phi$	$\mid$	$\lambda$	$\eta$	$\pi$	$\kappa$	$\omega$	
$\simeq$	$\equiv$	$\Psi$	$\propto$	$\infty$	$\sim$	$\partial$			
$Z$		$C$	$V$	$B$	$N$	$M$			
$\zeta$	$\chi$	$\psi$	$\times$	$\beta$	$\nu$	$\mu$			

Figure 4: Key definitions for Alt-A ... Alt-Z and Ctrl-A ... Ctrl-Z

### 3. WordPerfect Keyboard Layout

A WordPerfect keyboard layout provides the ability to reassign, or remap, keys on the keyboard. This enables all of the **Alt** and **Ctrl** keys to be mapped to any function you want. In this project, all seventy-seven possible keys combinations have been defined to produce the most frequently used TeX and/or L<sup>A</sup>T<sub>E</sub>X control sequences in a single keystroke. See Figure 4 for the description of the **Alt-A...Alt-Z** and **Ctrl-A...Ctrl-Z** key definitions for this project. See Figure 5 for all other **Alt** and **Ctrl** keys defined for this project.

It is quite easy to set up a keyboard layout if you follow three basic steps:

1. Plan the key definitions.
  2. Create a keyboard layout.
  3. Define the keys.

### **3.1 Plan the Key Definitions**

Before you create a keyboard layout, you should plan what keys you will define and what control sequence(s) each key will produce. This is very similar to defining a macro, so you should have a clear idea of what you want the key to accomplish before actually defining it.

### 3.2 Create a Keyboard Layout

1. Press the **Setup** key (**Shift-F1**).
  2. Press 6 to select the **Keyboard Layout** option.
  3. At the **Setup: Keyboard Layout** screen, press 4 to select the **Create** option. The message **Keyboard filename:** appears in the bottom-left corner of the screen.

Type a name 1-8 characters long (conforming to the standard rules for naming DOS files) and then press the **Enter** key. The keyboard layout for this project has been named **TEXLATEX**

Figure 5: Key definitions for all other possible Alt and 'Ctrl keys

and is stored in the file **TEXLATEX.WPK**. See Figure 6 for an example of the **Setup: Keyboard Layout** screen.

You are now ready to enter your key definitions.

### 3.3 Define the Keys

**At the Keyboard:** Edit screen, there are two ways to define a key. See Figure 7 for an example of this screen.

1. Press 4 to select the **Key: Create** option.

This option is used for creating new key definitions.

2. Press 6 to select the **Macro: Retrieve** option.

This option is used for creating new key definitions by assigning a macro to the key.

The message **Key:** appears in the bottom-left corner of the screen. Press the key to be defined. In this project, only the **Alt** and **Ctrl** key combinations will be re-mapped. These key combinations can be entered by holding down the **Alt** key and pressing a single letter or by holding down the **Ctrl** key and pressing a single letter. There are some keys that do not provide a **Ctrl** key definition. If nothing happens after entering a key combination, then you have probably stumbled across one of the unavailable **Ctrl** key combinations. See Figure 5 above for the listing of these keys.

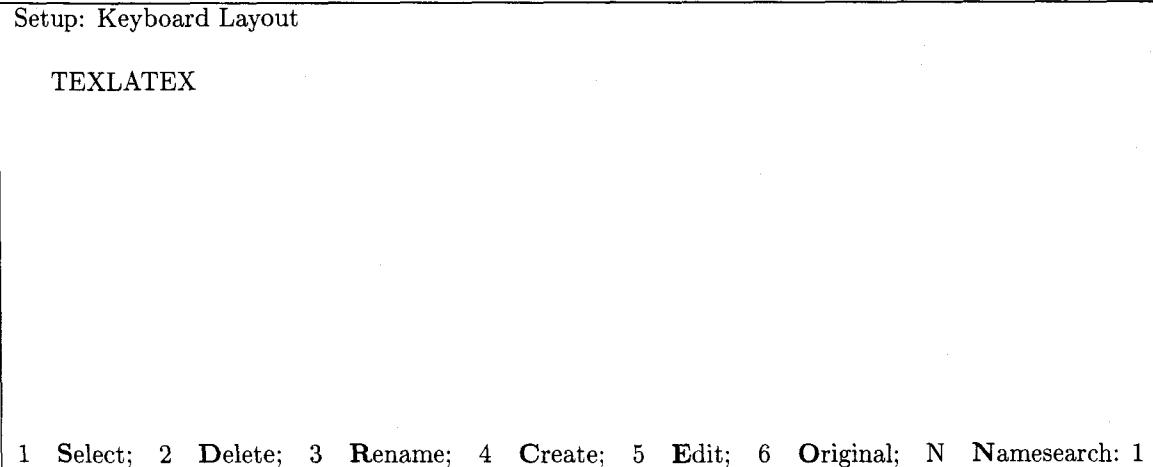


Figure 6: WordPerfect **Setup: Keyboard Layout** screen

Keyboard: Edit		
Name: TEXLATEX		
<u>Key</u>	<u>Description</u>	<u>Macro</u>
<b>Key:</b> 1 Edit; 2 Delete; 3 Move; 4 Create; <b>Macro:</b> 5 Save; 6 Retrieve: 1		

Figure 7: WordPerfect **Keyboard: Edit** screen

### 3.4 Comments

Now that the keyboard layout is in place, it is necessary to assign to the **Alt** keys the macros that are already stored in macro form as **ALTx.WPM** and to the **Ctrl** keys the macros that are already stored in macro form as **M-xxxxxx.WPM**. This is done quite easily by using the **Macro: Retrieve** option described on the previous page. For example, to assign **ALTA.WPM** as the **Alt-A** key definition, at the **Keyboard: Edit** screen do the following:

1. Press 6 to select the **Macro: Retrieve** option.
2. The message Key: appears in the bottom-left corner of the screen. Hold down the **Alt** key and press the letter **A**.
3. The message Macro: appears in the bottom-left corner of the screen. Type the name of the macro to be assigned to this key and press the **Enter** key; for this example, type **alta**.

You will now see an entry made on the **Keyboard: Edit** screen defining the **Alt-A** key. See Figure 8 for the updated screen.

Keyboard: Edit		
Name: TEXLATEX		
<u>Key</u> Alt-A	<u>Description</u> TeX and LaTeX \alpha	<u>Macro</u> 1
Key: 1 Edit; 2 Delete; 3 Move; 4 Create; Macro: 5 Save; 6 Retrieve: 1		

Figure 8: WordPerfect **Keyboard: Edit** screen with the **Alt-A** key defined

Once the macro has been integrated, you may or may not need to edit it. If you do, it is not difficult: press 1 to select the **Key: Edit** option from the **Keyboard: Edit** screen and notice that the key edit screen for key definition is exactly the same as the macro edit screen for macro definition. See Figure 9 for the description and WordPerfect codes of this key definition.

#### 4. Using WordPerfect Macros and the Keyboard Layout

It is important to keep in mind that the use of these macros and this keyboard layout is no different from the normal WordPerfect environment. However, two steps must be taken in order to use the macros and the keyboard layout:

1. Define where the keyboard layout and macro files will reside.
  - (a) Press the **Setup** key (**Shift-F1**).
  - (b) Press 7 to select the **Location of Auxiliary Files** option.
  - (c) Press 3 to select the **Keyboard/Macro Files** option.
  - (d) Type the directory name where the macros files and keyboard layout files are located (e.g., C:\MACROS) and press the **Enter** key.  
If you already have defined a directory to store macro and keyboard layout files, then you must install the macros and keyboard layout to create **TeX** and **LAT<sub>E</sub>X** documents in that directory.
  - (e) Press 0 ("zero") to return to the **Setup** screen.
2. Select the keyboard layout defined for creating **TeX** and **LAT<sub>E</sub>X** documents.  
Macros have been defined to select the **TEXLATEX** keyboard layout automatically and also to switch back to the original keyboard layout which is needed when creating normal WordPerfect documents. See **Changing Keyboard Layouts** in Table 1 for these macros.
  - (a) Press 6 to select the **Keyboard Layout** option.
  - (b) Use the up and down arrows to highlight the **TEXLATEX** keyboard layout. Press 1 to select this keyboard layout.
  - (c) Press 0 to return to your document.

You are now ready to use the keyboard layout and macros.

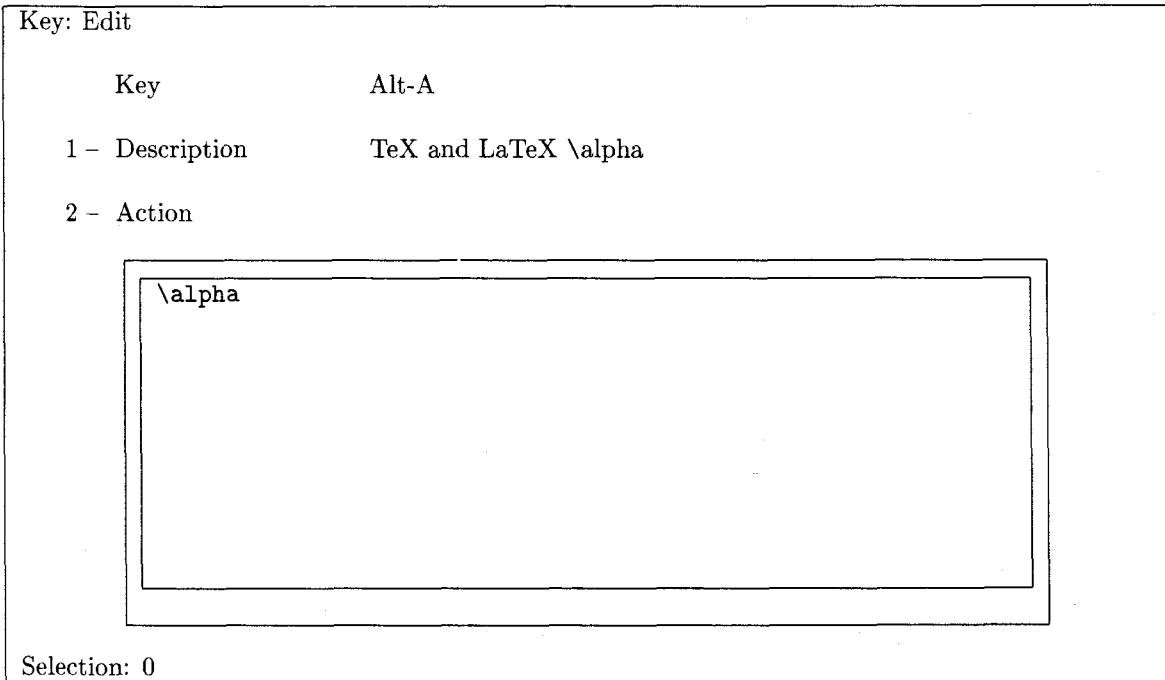


Figure 9: WordPerfect key definition for TeX and L<sup>A</sup>T<sub>E</sub>X \alpha control sequence

## 5. Conclusions

The intent of this project was to help speed up the learning and typing process for setting up a TeX and/or L<sup>A</sup>T<sub>E</sub>X document by automatically providing the basic control sequences. The project appears to have accomplished both of these with several interesting points:

1. The control sequences that were assigned to the **Alt** and **Ctrl** keys are common in most word processing systems, so there was very little time required to memorize which control sequences were produced by which key.
2. The amount of typing was drastically reduced as a result of being able to use a single keystroke in place of long control sequence names.
3. Using named macros also saved typing, but more importantly helped prevent errors when a multiple line control sequence such as those used to begin and end L<sup>A</sup>T<sub>E</sub>X environments were used in a document.
4. Repetitive use of the defined keys and named macros helped in the process of learning the actual control sequences for TeX and L<sup>A</sup>T<sub>E</sub>X. Creating a TeX and/or L<sup>A</sup>T<sub>E</sub>X document would be possible without the help of these aids; however, the overwhelming consensus of those who have used these WordPerfect macros and the keyboard layout is that the document would take more time to create as a result of having to type out all of the control sequences.
5. Don't get carried away! The control sequences assigned to keys or defined as named macros were carefully chosen in order to optimize typing time, reduce errors, and enhance learning. Make sure you take the time to evaluate whether or not a key definition or named macro for a particular control sequence will enhance the productivity of the user rather than make it worse. For example, why have the user type ALT-F10 and m-11 to produce << when they could type \ll instead.

In general, typing time was reduced, learning capability was enhanced, and the number of errors caused by misspelled control sequences or forgotten control sequences diminished.

### **5.1 Comments**

This project was a success for two reasons.

1. The macro and keyboard layout feature of WordPerfect provided an easy-to-use interface for  $\text{\TeX}$  and  $\text{\LaTeX}$ .

This concept can be applied to other word processing packages, providing technical secretaries with similar advantages who need to use  $\text{\TeX}$  and  $\text{\LaTeX}$  in their jobs.

2.  $\text{\TeX}$  and  $\text{\LaTeX}$  became less intimidating to the technical secretaries, which allowed them to put aside their frustrations and concentrate on their jobs.

This is a very important part of why non-technical users shy away from  $\text{\TeX}$  and  $\text{\LaTeX}$ . We need to reach out and identify such tools to give these users the power and beauty of  $\text{\TeX}$  and  $\text{\LaTeX}$  without the frustration. It will be the non-technical users who will shape the future of  $\text{\TeX}$  and  $\text{\LaTeX}$  and broaden its scope of use.

### **6. Acknowledgements**

I would like to thank Kenneth E. Gadowski of Academic Computing Support for his help in reviewing this paper, Karen M. Kral of Academic Computing Support for sharing her wealth of knowledge about WordPerfect macros and WordPerfect keyboard layouts, and Doris A. Wood and Denise J. Brzoska of the Physics and Astronomy Department for their help in choosing the  $\text{\TeX}$  and  $\text{\LaTeX}$  control sequences that were mapped on the keyboard layout. I would especially like to thank Denise J. Brzoska for the time she spent evaluating the macros and the keyboard layout.

### **7. Update**

It has been brought to my attention that several key definitions in the keyboard layout did not work. After investigating the problem, I discovered that the keyboard being used was not an IBM keyboard, but in fact a Keytronic Professional Series KB515 (supposedly an IBM compatible). The key definitions that did not work were  $\text{Ctrl-}\backslash$ ,  $\text{Alt-}[$ ,  $\text{Alt-}]$ ,  $\text{Alt-};$ ,  $\text{Alt-}'$ ,  $\text{Alt-`}$ ,  $\text{Alt-},$ ,  $\text{Alt-.,}$  and  $\text{Alt-/-}$ .

### **Bibliography**

Academic Computing Support. *Creating and Using WordPerfect 5.0 Macros*. University of Delaware, Newark, October 1988.

Academic Computing Support. *More About WordPerfect 5.0: Macros*. University of Delaware, Newark, May 1989.

Knuth, Donald E. *The \TeXbook. Computers and Typesetting* Vol. A. Reading, Mass.: Addison-Wesley, 1986.

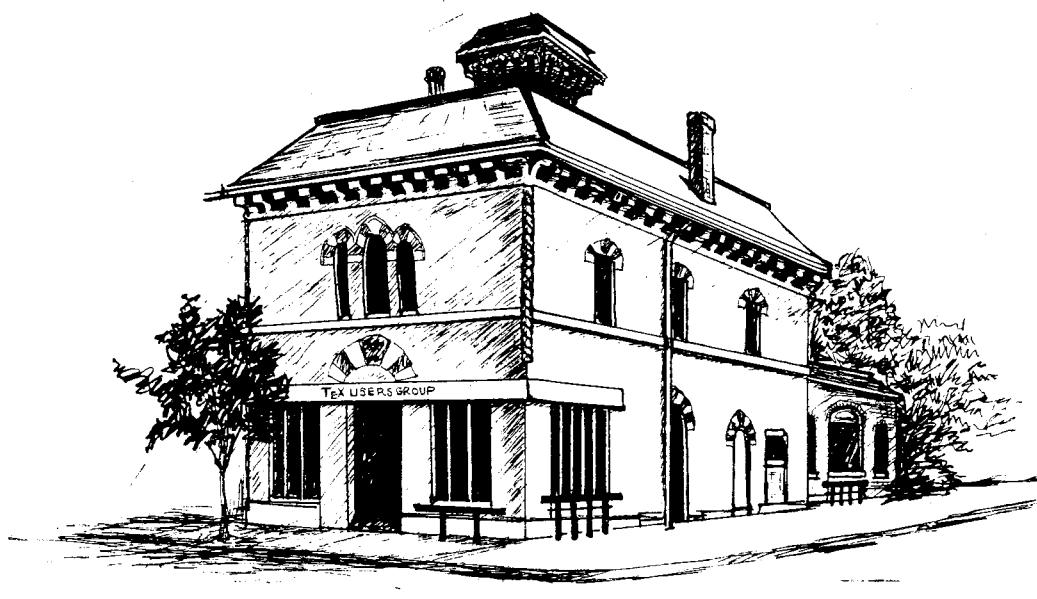
Lamport, Leslie. *\TeX: A Document Preparation System*. Reading, Mass.: Addison-Wesley, 1986.

McComb, Gordon. *WordPerfect 5.0 Macros and Templates*. New York, NY: Bantam Books, 1988.

WordPerfect Corporation. *WordPerfect for IBM Personal Computers*. Version 5.0, May 1988.

Math Macros	
m-eq m-\$\$ m-dm m-eqa m-sr m-ob m-ub m-ol m-ul	\begin{equation} ... \end{equation} \$\$ ... \$\$ \begin{displaymath} ... \end{displaymath} \begin{eqnarray} ... \end{eqnarray} \stackrel{ }{ } \overbrace{ } \underbrace{ } \overline{ } \underline{ }
Text Macros	
m-be m-ctr m-lst  m-part m-chap m-sec m-ssec m-sssec m-par m-spar m-ft m-fig m-tab m-vs m-hs m-doc  m-font m-size m-tex m-latex	\begin{env name you enter} ... \end{env name you enter} \begin{center} ... \end{center} \begin{enumerate} ... \end{enumerate} \begin{itemize} ... \end{itemize} \begin{description} ... \end{description} \part{ } \chapter{ } \section{ } \subsection{ } \subsubsection{ } \paragraph{ } \ subparagraph{ } \footnote{ } \begin{figure} ... \end{figure} \begin{table} ... \end{table} \vspace{ } \hspace{ } \documentstyle [ ] { } \begin{document} ... \end{document} font type changes font size changes TeX logo \TeX L <sup>A</sup> T <sub>E</sub> X logo \LaTeX
Changing Keyboard Layouts	
m-spec m-orig	Selects the TEXLATEX keyboard layout Selects the ORIGINAL keyboard layout

Table 1: WordPerfect macros by name



TEX USERS GROUP  
McGill University, Montréal  
August 21-24, 1988

# T<sub>E</sub>X for the Word Processing Operator

ROBIN L. KUBEK

ORINCON Corporation  
9363 Towne Centre Drive  
San Diego, CA 92121

## ABSTRACT

The purpose of this paper is to illustrate the need for a training program for word processing operators, secretaries, general management support staff, *et al*, in the use of plain.*tex* without intimidating them with T<sub>E</sub>X's programming language.

The paper will describe a technique developed by this author to train those familiar with word processing software, such as Microsoft Word, in the basic use of T<sub>E</sub>X by comparing T<sub>E</sub>X commands to word processing commands and functions. This type of training concentrates on T<sub>E</sub>X *operations* alone — the philosophy being that it is possible to learn to drive the car without knowing how to put the engine together.[1] By the use of parallel language, it will be shown that it is possible to have support personnel unfamiliar with T<sub>E</sub>X producing basic text manuscripts (letters, memos, etc.) and simple math within a few hours' worth of training.

### 1. Introduction

The subject matter of this paper was born after several years of frustration over the non-availability of T<sub>E</sub>X training at a local level, in an easily understandable form, *and* geared toward secretarial/management support personnel who don't have degrees in physics, engineering, mathematics, or computer science — *but* — are expected to type and format technical papers as if they did, *and* — by the way — should be fluent in Greek as well!<sup>1</sup>

Investigation in my local area has shown that T<sub>E</sub>X exists on the VAXes and PCs of many high tech R&D firms and at the universities, *but is not supported internally* by the Information Systems Support Division of any company or taught as a class at any school. T<sub>E</sub>X use seems to be largely confined to the technical professionals with degrees in the fields listed above, who have taught themselves with only the *T<sub>E</sub>Xbook* to guide them (and wouldn't have a clue as to how to teach anyone the program who doesn't speak *at least* three programming languages). For those of us who had to look up the word "algorithm" several times in the dictionary and still wouldn't recognize one even if it asked us to dance, a major dilemma arose when faced with the prospect of learning T<sub>E</sub>X "by the book". Not only is the *T<sub>E</sub>Xbook* oriented toward programming, but a lot of the commands don't even look like they're in English!<sup>2</sup>

Furthermore, T<sub>E</sub>X really has nothing to do with word processing, *per se*. It is not a word processing software package like WordPerfect or Microsoft Word; it is not a computer language such as FORTRAN or Pascal; and then there are funny words associated with T<sub>E</sub>X like \baselineskip, \parskip, \halign, \vbox, etc. (and let's not forget all those #!\$%\* curly braces). In other words, T<sub>E</sub>X doesn't look like anything a secretary or word processing operator has ever seen before in her/his life! Unfortunately, the very fact that T<sub>E</sub>X looks so foreign is the major reason many support personnel pass on the opportunity to learn it; and, coupled with the fact that there is no formal support network where the T<sub>E</sub>X trainee can turn when she runs into problems (which is often in the beginning), makes T<sub>E</sub>X a very intimidating program to tackle. It is true that "the best software in the world is useless if

<sup>1</sup> If you made it all the way through this sentence/paragraph, it is advisable to breathe now!

<sup>2</sup> Examine the spelling of \eqalign and \eqalignno. Where is the *u* that's supposed to follow the *q*?

you don't know *how* to use it,"[2] and probably explains why TeX resides inside so many computers but is sadly underutilized. Another misconception is that TeX is only used for math.

So how do we, the TeX community, go about introducing our ferocious looking friend (who is really a pussycat, once you get to know him) to the word processing community at large? If you follow along, I will explain my in-house "TeXnique" of introducing TeX the quick and semi-painless way to the Word Processing Operator.

## 2. Training Wheels

The first big hurdle in teaching TeX to anyone is convincing them that (a) no matter *what* they've heard, TeX cannot bite,<sup>3</sup> and (b) that while all those \backslashes do look a bit bizarre, TeX commands really do make sense. Once you've accomplished that, begin to familiarize your student with the basic TeX commands.

### 2.1 The Preamble

The preamble (the first several lines at the top of a file) is the best place to start comparing TeX commands to their equivalents in the word processing software. Below is an abbreviated list that I give to my TeX trainees of typical preamble commands and their meanings.

\magnification=\magstep1

Magnifies the font 1.2 times. If you are using a 10-point font, this magnification will make the type appear at approximately 12 points, which looks like 10 pitch (pica) on a typewriter. If you do not specify magnification, the default is \magstep0. 10 points with no magnification looks like 12 pitch (élite) on a typewriter. **Note:** On Macintosh, the TeX fonts do not magnify well past \magstep1. If a larger font is called for, use a PostScript font.

\font\cm=cmr12 at 12pt

Font definition for 12 point Computer Modern Roman (TeX font). **Note:** The normal TeX default font is cmr10. Any font that is not the default MUST be defined and called out prior to use.

\footline={\hss\folio\hss}

Your footer. \hss stands for Horizontal Stretch or Shrink and is "infinite glue". By having \hss on either side of \folio (which is the macro for the page number), your page number will be centered at the bottom of your page. **Note:** This \footline is the default setting in TeX and doesn't have to be placed in your preamble unless you are changing the font size, putting the number in either the right or left corner, or adding other information.

\pageno=1

Self-explanatory. If your page number is to begin with a lowercase roman numeral (for tables of contents, acknowledgement pages, etc.), you would give \pageno a value of -1, -2, etc. — whatever number you wish to start with (-1 = *i*, -2 = *ii*, -3 = *iii*, and so on). **Note:** The \folio macro in your \footline defines any \pageno less than 0 as a lowercase roman numeral.

\hsize=6.5in

This is the horizontal size of the page. Unlike ordinary word processing (where you set the *margin* space), in TeX you set the *text* space. \hsize=6.5in is the default horizontal setting in TeX and does not have to be placed in your preamble. *However*, until you become comfortable with the differences between TeX and ordinary word processing, it is a wise idea to list the differing commands up front.

\vsize=8.9in

This is the vertical size of the page. Again, text space is being set rather than margin space. This \vsize is the default setting and does not have to be set in the preamble, but, like \hsize, it is a wise idea to keep it up there in the preamble until you become comfortable. **Note:** With an \hsize of 6.5in and a \vsize of 8.9in, you will end up

---

<sup>3</sup> Well, maybe nibble a little, but not hard!

with a 1-inch margin on all four sides of your paper (and isn't that what you would have set your margins to anyway?).

\baselineskip=12pt

\baselineskip is the amount of space between lines, or, more precisely, from the bottom (or base) of a line to the bottom of the next line. In regular typesetting and some desktop publishing software, "leading" is the term used, and means the same as \baselineskip. In general, the leading or \baselineskip should be 2 points greater than the font for single-spaced text — or 12 points for a 10pt font. For space-and-a-half and double-spacing with a 10pt font, the math is simpler: a 15pt \baselineskip on a 10pt font for space-and-a-half, and a 20pt \baselineskip for double-spacing.

\parskip=6pt plus 3pt minus 2pt

Additional space between paragraphs — but not as much as "two returns". The plus and minus points give TeX additional shrink and stretch "glue" to use at its discretion.

\parindent=20pt

This is the paragraph indent. 20 points is approximately the same as 5-space indenting.

\overfullrule=1pt

The \overfullrule command is a handy little device that has no real equivalent in ordinary word processing. This command, when set for a value greater than 0, will put a black line beside any text that exceeds the \hsize on any line. **Note:** You will receive an Overfull \hbox message in your log when you send your document to typeset, but if your "overfullness" is less than 5 points, you probably will not be able to see it on the page, and thus not be able to fix it, without the line generated with the \overfullrule command.

\raggedbottom

No, this is *not* a description of how you will feel after a day of TeX training! This command tells TeX to permit a small amount of variability in the bottom margins on different pages in order to make the other spacing uniform (p. 111 of the *TeXbook*). If you choose not to use \raggedbottom, the default is \normalbottom, which will make all of your bottom margins the same on every page — even though it means stretching the text space on the page and making it look funny.

### *Word Processing — First Pass*

Once you've established the preamble, it's a good idea to have your student type the commands into a document. Not only does this get her into the feel of typing \backslash in front of just about *everything*, it's also a good opportunity to show her that she can use *any* editor/word processing application that suits her fancy. On a VAX, EDT is my editor of choice, but I've also input TeX files in Word-11, TEDI, and on Macintosh, in Microsoft Word.

One word of caution in regards to inputting TeX with a word processing application. You may use all of your "gold" key applications, editor keypad, and user-defined keys (UDKs) as you normally would to move around in your document, cut and paste, etc. However, you may not *format* in that application. No bolding, underlining, tabbing, super- or subscripting, etc. If you are using an application that puts in internal formatting (such as Version 4.0 and above in Word-11), you must strip the file of the formatting *before* sending it to TeX. Your best bet is to simply type in block paragraphs and let the text word wrap. While in a word processing application, you can still type your TeX commands, \backslash and all.

On Macintosh I've found that, while typing in Microsoft Word is fine, it takes a lot more steps to get it into *Textures* (the Mac version of plain.tex). First, the document must be stripped of formatting by saving as a text-only file, then copied into the clipboard. *Textures* must then be opened and the clipboard pasted in. If the document is too large to move with one cut and paste, *Textures* has to be closed, the text-only Word document re-opened, and the whole cut-and-paste procedure performed again. Consequently, when teaching TeX on a Mac, I wholeheartedly encourage the student to just type in *Textures* — which operates similarly to EDT.<sup>4</sup>

---

<sup>4</sup> Of course, if she's stubborn, I'll let her type in Word and go through the whole horrid procedure of moving the

## 2.2 Macros

Once you've gotten past the preamble (and the dreaded first document), macros are a real snap. Macro simply means *definition*. Macros work a lot like UDKs or cut and paste. Instead of storing a string of keystrokes, commands, etc. in memory under a key or in a buffer, you simply define your string (`\def`) and assign it a label (`\cc`). The definition of the keystrokes is then enclosed in curly braces (`{}`). For example:

```
\def\cc{\centerline{}} This macro defines a \phantom (invisible) \centerline the depth of  
1 \baselineskip (or, in plain English, a blank line). The empty set of curly braces next to  
\centerline means the set is empty.
```

The preamble commands are made up of macros (*i.e.*, `\magnification`, `\footline`, `\pageno`, etc.). The *TeXbook* lists all of the macros that come with *TeX* (hundreds of them). In addition, you can write *your own* macros (just like `\cc` above). Macros can be abbreviations for typing complex commands, or simply for words, phrases, or sentences that are used over and over again. They can contain font changes or formatting such as bolding, italicizing, underlining, and the like. By writing a macro, you can save yourself untold number of keystrokes (or “mouse clicks” or dragging down menus, etc.). You can even “nest” macros within macros. For instance:

```
\def\bb{\cc\cc\cc}
```

By using the macro `\cc` inside your definition, you've now got a macro for 3 blank lines. Some other macros that I have found helpful are:

```
\def\ie{{\it i.e.}\},} macro for italicized “i.e.” with italic correction5  
\def\ea{{\it et al}\},} macro for italicized “et al.” with italic correction  
\def\eg{{\it e.g.}\},} macro for italicized “e.g.” with italic correction  
\def\underscore#1{$\underline{\smash{\vphantom{y}}}\hbox{#1}}$} macro for underlining  
text so that the underline goes below a lowercase descender6 rather than through it.
```

Regarding `\underline`, above, I am unaware of *any* word processing application where this feat is possible using the underline key/command. This is also one macro where it is wiser to quote Joe Isuzu<sup>7</sup> rather than to try and explain to a *TeX* first-timer the meanings of all the commands in the definition.

### Word Processing — Second Pass

Once your student has grasped the concept of macros, have her go back to her document (the one with the preamble in it), and create some of her own macros for it. If you want to get her *really* crazy, show her how to write a *whole* document using *nothing but* macros! See Figure 1.

## 2.3 (The Dreaded) Math

Now that you have your student (a) thoroughly confused, and/or (b) totally in awe of your incredible genius and unique abilities, it is time to drop the “Big One”<sup>8</sup> on her — *Typing Mathematics Equations!!!!*

Contrary to every word processing operator's belief, math typing *does not* have to be one of those situations where, if given a choice between typing the math and hurling yourself in front of train — you'd choose the train!<sup>9</sup> The key to typing math in *TeX* is to “walk (and talk) yourself through the equation.” In other words, if you “say” the equation, you can type it. The hardest part of math typing, *TeX* style, is remembering all those #!\$%\* curly braces!

The answer to all of your student's questions, when it comes to math typing is “*TeX* will take care of that.” *TeX* will:

---

document. I can guarantee she'll abandon Word entirely for the second document, and I won't even have to nag.

<sup>5</sup> Italic correction squeezes in a liitle bit more space between the last italicized character and the next “normal” character. Otherwise, the slant of the italics could make the letters/characters overlap one another or make them appear too close together.

<sup>6</sup> Fancy terminology for the part of the letter which extends below the baseline: *g*, *j*, *p*, *q*, and *y*.

<sup>7</sup> “Trust me”.

<sup>8</sup> A little organ music would be nice right here, preferably with very deep bass chords.

<sup>9</sup> That is, if you can type the math in *TeX*. If I had to type math in ordinary word processing, I too would choose the train!

```

\def\mom{\par Dear Mom, I love you forever.}
\def\cookies{\par Please bake me some chocolate chip cookies.}
\def\butter{\par You're the best Mom in the Whole Wide World!}
\def\junk{\par Can we go to MacDonald's tonight for dinner?}
\def\ps{\par By the way, if my teacher calls, I really didn't throw
      the whole can of red paint on her!
\medskip
({\it It was only half a can\/})
}

\def\snow{\mom \cookies \butter \junk \ps}
\snow

Dear Mom, I love you forever.
Please bake me some chocolate chip cookies.
You're the best Mom in the Whole Wide World!
Can we go to MacDonald's tonight for dinner?
By the way, if my teacher calls, I really didn't throw the whole can of red paint on her!
(It was only half a can)

```

Figure 1: A macro to go crazy by ... and the end results!

- center
- proportionalize all the delimiters
- change font size for super- and subscripting
- do Greek and symbols (*all symbols!*)
- put equation numbers where they belong
- put equal space above and below a displayed equation and text
- *not change* your line spacing when typing math in a text sentence
- make your equal signs line up in a complex equation
- change the baby and get dinner started before you get home!<sup>10</sup>

Instruct your student that all she needs to do to invoke math mode is type a dollar sign or two; one \$, and she can type math inside of a text sentence. To leave math mode, she simply types another \$ (a lot easier than having to use code keys, alternate font codes, or leaving the document altogether, going into another program to type, and then having to port the equation back). If she wishes to type a displayed equation, she uses two dollar signs (\$\$) to get in and two \$\$ to get back out. Of course, with math, you can instruct until you turn blue, but the best way to teach the concept of math typing is...

#### *Word Processing — Third Pass*

The hands-on method is the only way to go when it comes to math typing. Have your student create a document (or use the same one she's been practicing in), and give her some *very simple* math equations to start out with. For example:

$$\frac{1}{2} = \frac{\alpha}{\beta}, \quad e = mc^2, \quad \text{and } x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Of course, you will have to give her the commands for  $\pm$  and  $\sqrt{\phantom{x}}$ . The next thing is to tell her that whenever she says a command (such as “over”) that she will use a \backslash in front of it. Tell her to think of superscripts as “up” and to use the caret symbol (^); subscripts are “sub” and the command is the underline symbol (\_). Curly braces are also something that need to be walked

<sup>10</sup> No, no — just checking to see if you're still with me on this one.

through. Learning their placement just comes with a little time and practice. It helps if, in talking the student through, you tell her to say "begin" and "end" with each equation and its parts, and to put a { with the "begin" and a } with the "end". Furthermore, tell her to type (in regular English words) everything that she says in the equation (numbers, of course, can be typed in regular English numbers). Thus, the equations above, would look like this:<sup>11</sup>

```
$\begin{aligned} & \{1\over 2\}\end{aligned} = \begin{aligned} & (\text{begin})\{\alpha\over\beta\},(\text{end})\\ & \$e = mc^2,\\ & \$x = \begin{aligned} & (\text{begin})\{-b\text{ (plus or minus)}\}\pm \\ & (\text{square root})\sqrt{b^2 - 4ac}\end{aligned}(\text{end})\\ & \over 2a\end{aligned}.(\text{end})\$
```

### 3. Conclusion

*...and at the end of the training session, T<sub>E</sub>X said ‘‘\relax’’, and it was good.*

By now, you've given your student a smattering of the things that T<sub>E</sub>X can do. This is by no means a complete smattering. It is, however, enough for an afternoon's worth of training — and enough to get your student's appetite whetted for more T<sub>E</sub>X. It is enough to get your student thinking about all the things she might be able to accomplish with T<sub>E</sub>X. And it is enough to make the first few chapters of the *T<sub>E</sub>Xbook* semi-understandable in one reading.

There are quite a number of basic T<sub>E</sub>X skills that have not been touched upon here. Those include tab fields for tables; itemizing for outlines; commands for underlining, bolding, and italicizing; and commands such as \leftline, \rightline, and \centerline (not to mention how to read error messages, or get back at T<sub>E</sub>X for some of the snotty comments he's prone to make). But stop and think about what *has* been accomplished here. The training, as outlined above, has taken a program that has generally been perceived as mysterious, forbidding, and unfamiliar, and made it more understandable and *accessible*. It has given your student enough to get started in T<sub>E</sub>X, and enough to produce a basic text document — and be comfortable doing it. It has shown her that math math typing can be pretty straightforward and much simpler than in any other software application that she may be familiar with. Your student will, of course, have many questions over time, and more than once will require your assistance. She will not know what makes T<sub>E</sub>X tick, or really *why* T<sub>E</sub>X works at all. *But*, she will be able to “drive the car.”

There will, of course, be other afternoons to show your student (and presumably, co-worker) how to set up tables. You can walk her through \item and \itemitem over the phone. The same holds true for \leftline, etc. But, hopefully, you can see where this technique of in-house training can lead, and you will be able to go about setting up your own training program to meet the needs of your company and its employees.

One word of warning, however; once you have taught a word processing operator the basics of T<sub>E</sub>X, she may not want to type in WordPerfect, Microsoft Word, Word-11, or any of the myriad assortment of ordinary word processing applications, ever again. She may develop an extreme disdain for ragged right margins and non-proportional spacing. She may even begin to talk funny (offering the opinion that an “overfull \hbox is by far a worse problem than an underfull \vbox!”). Furthermore, the more she learns about T<sub>E</sub>X, the more she's going to want to learn. She may no longer be content to merely “drive the car” — she's going to want to know how to put the engine together. You just may be creating a monster!<sup>12</sup>

### Bibliography

- [1] Knuth, Donald E. *The T<sub>E</sub>Xbook*. Reading, Mass.: Addison-Wesley. 1984.
- [2] Rees, Clair. “The Training Issue.” *WordPerfect, The Magazine* March 1989, p. 4.

<sup>11</sup> The “(begin)”, “(end)”, “(up)”, etc. (all the words enclosed by parentheses) that you see typed in the equation commands are *only* to show what you want your student to *say* while she is typing. In the actual file, these words would not be typed.

<sup>12</sup> Grrrr!

# T<sub>E</sub>X and Its Versatility in Office Production

JO ANN RATTEY-HICKS

Department of Mechanical and Materials Engineering  
Washington State University  
Pullman, WA 99164-2920  
Bitnet: jarhicks@wsuvml

## ABSTRACT

Probably the two most challenging situations an office administrator faces are: 1) strategies to maximize the efficiency, and 2) planning for increased output. Over the past ten years, tremendous changes have occurred: the PC has literally replaced the word processor which previously had replaced the typewriter; fax machines can now send documents faster than special overnight mailing services; photocopiers, built for speed and efficiency, are overwhelming as they devour tons of paper in mere seconds. In the face of all these advancements, secretaries have been expected to adapt to and accept all these changes.

But with the various types of PCs came the problems of selecting the right type of software, drivers, printers, and fonts. The costs alone for upgrading software and equipment are high — especially when an office needs three or four different software programs for word processing, graphics, spreadsheets, and databases. When the Department of Mechanical and Materials Engineering at Washington State University was in the process of upgrading their existing computer and word processing facilities, much thought was given to these problems. It was decided to utilize a program which was already installed on the mainframe, T<sub>E</sub>X, and build everything else around T<sub>E</sub>X. The department has been using T<sub>E</sub>X and T<sub>E</sub>XT<sub>1</sub> (a program developed by WSU based on T<sub>E</sub>X) for the past four years. The focus of this paper is on the integration of T<sub>E</sub>X and T<sub>E</sub>XT<sub>1</sub> into the office workplace.

During the early 1980s, the Department of Mechanical and Materials Engineering at Washington State University developed a comprehensive Strategic Planning Guide. This Guide reflected the goals and strategies necessary to meet the demands of a growing department. One of the fundamental issues was that of academic computing (i.e., research and instruction), and the need for more computers and modern computer facilities for faculty and students. As monies were appropriated for computer growth and expansion, the Department began to hire new faculty with strong backgrounds in computers. However, it was soon apparent that the existing word processing facilities were not adequate to handle the large typing loads. At that time the Department owned several IBM typewriters and an IBM Mag Card II which was shared by the department's two technical secretaries. The Department decided to upgrade to state-of-the-art word processing equipment and purchased two IBM Displaywriters.

The impact was felt immediately as secretarial workloads were handled more efficiently and the processing time for typing requests were cut nearly in half. The new equipment was relatively easy to use and the staff were very enthusiastic about working with the new word processing stations. However, the Displaywriters were cumbersome when it came to typing mathematical equations and other technical material, and there was very little written documentation available on this topic. Through trial and error, the secretarial staff devised a method for typing equations on the Displaywriter, but the technique was awkward and tedious.

In 1984, the Department conceded that its office equipment was antiquated. The personal computer industry had completely replaced the more conventional word processing systems. The only possible solution was to integrate personal computers into the office system. However, there were so many

different models of personal computers on the market, along with the various types of software, drivers, printers, and other hardware. The variety in software opened up new areas of office development: databases for compiling mailing lists and reports, spreadsheets for budgets and other accounting needs, and graphics for tables, charts and slides. The Department realized it needed to re-evaluate the role of its administrative office and address these new areas. The faculty also voiced their opinions about what types of software and hardware they wanted the office staff to use, and it soon became apparent that this was going to become very expensive. It was about this time that one of the secretaries, Jo Ann Rattey, discovered a solution to the personal computer problem.

She had agreed to type a 45-page appendix containing complex equations and tables for a doctoral candidate's dissertation. The doctoral student had been using a program on the mainframe called *T<sub>E</sub>X<sub>T</sub>1* to type his dissertation. He showed her samples of his work and she quickly realized that this program could easily solve many of her own problems regarding technical typing. She showed these samples to departmental chair and the office supervisor, and they agreed to give the program a try. The secretary spent the next several weeks learning the fundamentals of *T<sub>E</sub>X* and *T<sub>E</sub>X<sub>T</sub>1*, and using a computer terminal installed in her office, typed a few short proposals and papers. The faculty raved about the professional quality of their proposals and papers, and insisted that training sessions be conducted. Another feature that made *T<sub>E</sub>X* and *T<sub>E</sub>X<sub>T</sub>1* very desirable was the fact that all users had free access to the software. This alone meant considerable savings since the only equipment needed to use *T<sub>E</sub>X* and *T<sub>E</sub>X<sub>T</sub>1* was a computer terminal connected to the mainframe and the training/reference manuals. At this point in time, personal computers were still very expensive and the Department was trying to equip every faculty and staff office with a connection to the mainframe.

When the personal computers were finally installed in the main office, they were linked to the mainframe with a terminal emulator. This greatly added flexibility in that work could now be processed either using MicroSoft Word or the two typesetting programs on the mainframe. At first little, if any, work was done using MS Word on the PCs because Word was not equipped to handle technical typing. In fact, MS Word was behind the Displaywriter software when it came to technical typing. Of course, there were additional software packages available to interface with Word and make technical typing possible, but the main goal of the Department was to avoid spending a lot of money on different software. Therefore, with free access to *T<sub>E</sub>X* and *T<sub>E</sub>X<sub>T</sub>1* on the mainframe, it did not make sense to purchase additional software to interface with MS Word.

One particular problem did come up which made it necessary to decide whether all material should be typeset on the mainframe. The Computing Center decided to install new IBM laser printers in several locations on campus and to charge all users a flat rate per page printed. This resulted in the Department incurring an average of \$600.00 in monthly printing charges. It was decided to use Word for non-technical material, i.e., letters, and memos. This decision was also beneficial to the majority of the faculty who typed their own correspondence and submitted their disks to be formatted and printed on the small HP LaserJets in the main office. All other material (reports, papers, proposals, slides, and exams) would be typeset using *T<sub>E</sub>X* and *T<sub>E</sub>X<sub>T</sub>1*.

Interestingly, as the secretaries became more experienced using *T<sub>E</sub>X* and *T<sub>E</sub>X<sub>T</sub>1*, they began to experiment by typesetting brochures, programs, slides, bar charts, flow charts, and other things normally sent out to the Publications and Printing Office (see Appendix A for sample formats). The department quickly discovered that documents could be produced in-house approaching the quality available from Publications and Printing. However, special needs quickly surfaced: for example, how to integrate graphics into typeset documents. The department's Assistant for Computer Services, Mike Shook has modified Tektronics' PLOT10-TCS sub-routine package to generate files which can be used by *T<sub>E</sub>X<sub>T</sub>1* and printed on an IBM 3820 page printer. He has also been using PC image processing software in conjunction with a TV camera/frame grabber which creates PSEG 3820 files which *T<sub>E</sub>X<sub>T</sub>1* can use to produce full page images. Now text and graphics can be compiled into one *T<sub>E</sub>X* file and printed. This feature has resulted in less dependence on the College of Engineering's Design Office, and allows the user to control the design content.

The department felt that a database software package was a good investment, and dBASE III was purchased. However, it was soon discovered that *T<sub>E</sub>X* and *T<sub>E</sub>X<sub>T</sub>1* would not interface with database software. This became critical when a faculty member wanted to send material to an extended list of addresses stored on a 5.25" floppy. The addresses were uploaded to the mainframe and put into a

*T<sub>E</sub>X* file. The file was quickly formatted using *T<sub>E</sub>XT1* commands and printed out on plain bond paper in a three-column format (see Appendix B for sample file), and xeroxed onto address labels. dBASE was used to add delimiters to the original address file on the floppy disk and merged into a shell letter typed with Word. Although this process seems complicated, it actually took less than one hour to print out over 100 letters and addresses.

Recently, the department purchased new HP LaserJet printers and connected them to the mainframe via Gandalf boxes. These boxes make it possible to switch the printers from parallel to serial ports at the touch of a finger, thus allowing the user to print from either the PC or the mainframe. In order to print from the mainframe, the Computing Center had to establish each HP printer as a separate print destination and an access file had to be established on each user's account. The access file allows the user to print from the mainframe and controls who may use the printer.

The major advantage of printing on the LaserJets is that there are no monthly printing charges, and this has resulted in a further drop in departmental printing charges. There are three problems, however, in using the HP LaserJets to process *T<sub>E</sub>X* files: 1) Hewlett Packard does not support the IBM Sonoran fonts; 2) the printers are slower than the IBM 3820 and 3812 printers, and 3) the *texdvi* file does not allow selective printing of pages. As it stands now, the font that is supported by HP — the Computer Modern font is adequate for most printing needs — but it is hoped that these problems will be corrected within the next few years.

## Appendix A: Sample *TEXT1* Formats for Creating Flow Charts

```

.....
\nottitlepage
\drawboxmat{10pt}
\drawboxrulesize{1pt}
\line{\hfill{\drawbox{\vbox{\hsize=1in{Assume \lbr \hs{10pt} @P_{E_o} =
P_E@}}}}\hfill}
\vs{-2pt}
\cl{@\downarrowarrow@}
\line{\hs{2.1in}\drawbox{\vbox{\hsize=1.8in{Calculate @\Delta P_{E_o}@ \lbr \hs{15pt} from @(\Delta P_E/\Delta P_{E_o})@}}}\raise15pt
\hbox to .7in{ \leftarrowfill@\Delta P_E@}\hfil}
\vs{-2pt}
\cl{@\downarrowarrow@}
\line{\hfill{\drawbox{\vbox{\hsize=1.8in{Calculate gas flow\lbr\hs{15pt}
velocity using @\Delta P_{E_o}@}}}}\hfill}
\vs{-2pt}
\cl{@\downarrowarrow@}
\line{\hfill{\drawbox{\vbox{\hsize=1.8in{Calculate @\Delta P_{T_o}@ \lbr \hs{15pt} using gas velocity}}}}\hfill}
\vs{-2pt}
\cl{@\downarrowarrow@}
\line{\hs{2.1in}\drawbox{\vbox{\hsize=1.9in{Calculate @\Delta P_T/
\Delta P_{T_o}@ \lbr\hs{32pt} and evaluate Z}}}}\raise15pt
\hbox to .7in{ \leftarrowfill @\Delta P_T@}\hfil}
\vs{4pt}
\hs{1in} Z not converged
\vs{2pt}
\cl{@\bigcirc@}
\line{\hs{3.044in} @\Bigg\downarrow@ \raise1pt\hbox{Z converged}\hfil}
\line{\hfill{\drawbox{\vbox{\hsize=1in{Stop and \lbr\hs{17pt} print out}}}}\hfill}
\hfill
.....
% Default page dimensions and margins
\pageformat{\pagelength{11in} % 792pt = 11in
            \pagewidth{8.5in} % 612pt = 8.5in
            \topmargin{1in} % 72pt = 1in
            \bottommargin{1in}
            \leftmargin{.50in} % 86pt = 1.2in
            \rightmargin{.50in}
            \bindingadjust{0in}}
}% end pageformat
\normalbottom % text height will be the same for each page.
               % Bottom lines will be even.
% Specific Font for Computer Modern Sans Serif
\ssf{cmss}{9}{11pt}
% Markup for Document ==> \cmss9; (implies \isk{11})
%
% Specific Font for Computer Modern Sans Serif
\ssf{cmss}{10}{12pt}
% Markup for Document ==> \cmss10; (implies \isk{12})
%
\cmss9;
\def\boxit#1{\drawbox{\vbox to 40pt{\vfill#1\vfill}}}

```

```

\hs{144pt}\boxit{\hsize=.56in \nin Develop\lbr Target \lbr Specs}
\vs{.50in}
\line{\boxit{\hsize=.6in \nin Recognize \lbr a \lbr Need}
\hfill
\boxit{\hsize=.6in \nin Set \lbr Design\lbr Objectives}
\hfill
\boxit{\hsize=.6in \nin Create\lbr Alternative\lbr Designs}
\hfill
\boxit{\hsize=.6in \nin Screen to\lbr Satisfy\lbr Objectives}
\hfill
\boxit{\hsize=.6in \nin Select\lbr "Best"\lbr Design}
\hfill
\boxit{\hsize=.6in \nin Prepare \lbr a \lbr Model}
\hfill
\boxit{\hsize=.6in \nin Test and \lbr Evaluate \lbr Model}
\hfill
\boxit{\hsize=.6in \nin Communi- \lbr cate the \lbr Design}}
\vs{.50in}
\line{\hs{100pt}
\boxit{\hsize=.6in \nin Gather \lbr the \lbr Information}
\hs{100pt}
\boxit{\hsize=.6in \nin Improve\lbr the\lbr Design}
\hfill}
\vs{1in}
\cl{The Expanded Design Process - 11 Step.}
.....
\paraphindent{18pt}\paragraphskip{12pt plus 6pt minus 1pt}
\justify \singlespace
\normalbottom
\selectfontset{cm12}{defaultprop}{default} %
\defaultprop\rm
\selectfontset{cm10}{tablefont}{default} %
\selectmathset{cm97m}{mathdefault} %
\mathdefault
\nottitlepage
\sfs{cmsan11}{elevenptss}{13pt} \sfs{cmsan8}{eightptss}{10pt}
\def\leaderfill{\leaders\hrule height2pt \hfill}
{\offinterlineskip
\tablefont
\halign{\strut\lft{\#}\hfil& \vrule# &\hs{2pt}{\#}&\hs{2pt}{\#}&\hs{2pt}{\#}&
\hs{2pt}{\#}
&\hfil \vrule# &\hs{3pt}{\#}&\hs{2pt}{\#}&\hs{2pt}{\#}&\hs{2pt}{\#}&
\hfil\vrule#
&\hs{3pt}{\#}&\hs{2pt}{\#}&\hs{2pt}{\#}&\hs{2pt}{\#}& \hfil \vrule#\hfil\cr
&&\multispan{14} Months & \cr
&&&&&&&&&&\cr
Activity & J & F & M & A & M & JU & JL & A & S & O & N & D &\cr
&&&&&&&&&&\cr
\noalign{\hrule}
&&&&&&&&&&\cr
ISC sub-project: &&&&&&&&&&\cr
&&&&&&&&&&\cr
A. feature abstraction update && \multispan5\leaderfill &&& \multispan5
\leaderfill && \cr

```

```

B. engagement abstraction update && \multispan5\leaderfill &&& \multispan5
\leaderfill&&\cr
C. run time environment design update && \multispan3\leaderfill && && && && &
&\cr
D. run time environment hardware and software update && & \multispan6
\leaderfill && && && &\cr
E. control system design && \multispan3 \leaderfill && && && && &\cr
F. implement control algorithm && && && &\multispan4 \leaderfill && && && &\cr
G. ISC system testing && && && && \multispan5\leaderfill&&\cr
&& && && && && &\cr
Process models sub-project: && && && && && && &\cr
&& && && && && &\cr
H. process model development && && &\multispan5\leaderfill && && && &\cr
I. mechanistic model parameter investigation &&\multispan{10}\leaderfill && && &
\cr
J. control/process model system simulation&& &&\multispan8\leaderfill && && &
\cr
&& && && && && &\cr
Sensor sub-project: && && && && && &\cr
&& && && && && &\cr
K. design of AE fluid coupled sensor for CNC mill &&\multispan3 \leaderfill && &
&& && && &\cr
L. fabrication of AE sensor && &&\multispan4\leaderfill && && && &\cr
M. develop AE sensor monitoring environment for ISC && &&\multispan5 \leaderfill
&& && && &\cr
N. investigate AE signal for end mill surface generation && && && && &
\multispan6
\leaderfill && \cr
O. fabricate OM-2 force sensing system && && && &\multispan3 \leaderfill && &
&& &\cr
P. calibrate and test force sensing system && && && && && &\multispan4
\leaderfill &\cr
&& && && && && &\cr
OM-2 installation sub-project:&& && && && && && &\cr
&& && && && && &\cr
Q. complete mechanical and hydraulic installation && \multispan2\leaderfill && &
&& && && &\cr
R. install controllers && &\multispan5\leaderfill && && && &\cr
S. initialize controller logic and test system && && && &\multispan6
\leaderfill && \cr
T. prepare final report && && && && &\multispan2\leaderfill && \cr
&& && && && &\cr
\noalign{\hrule}}}
.....
% Default page dimensions and margins
\pageformat{\pagelength{11in} % 792pt = 11in
            \pagewidth{8.5in} % 612pt = 8.5in
            \topmargin{1in} % 72pt = 1in
            \bottommargin{1in}
            \leftmargin{.75in} % 86pt = 1.2in
            \rightmargin{.75in}
            \bindingadjust{0in}}
}% end pageformat
\normalbottom % text height will be the same for each page/

```

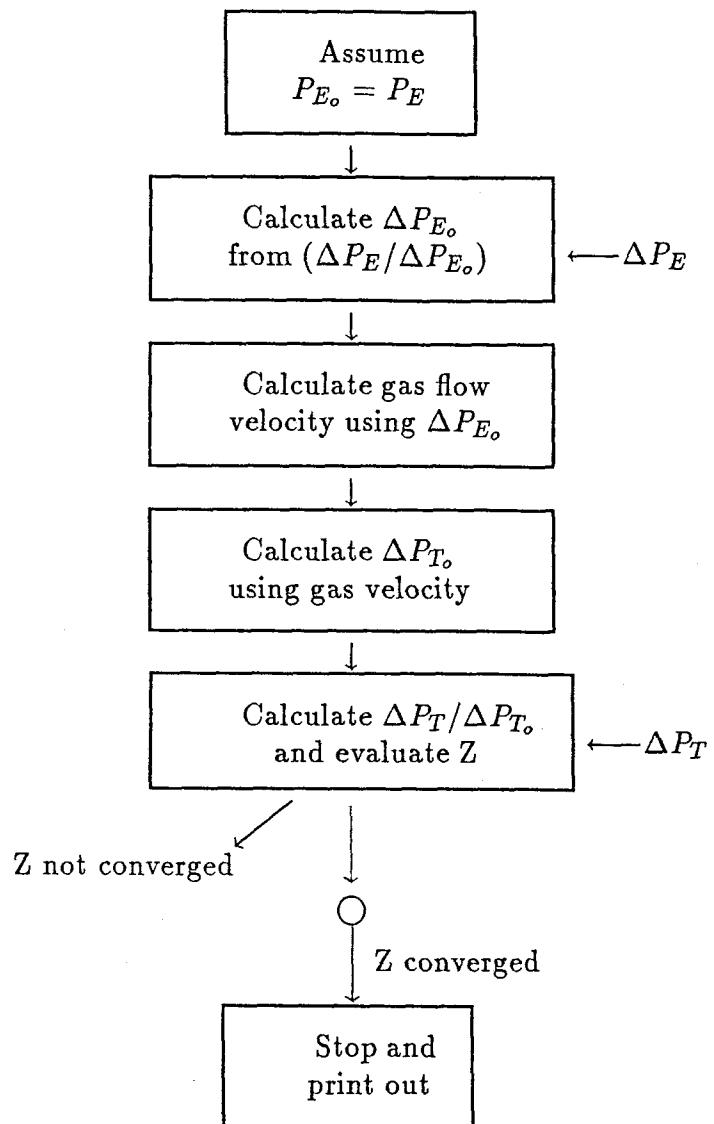
```

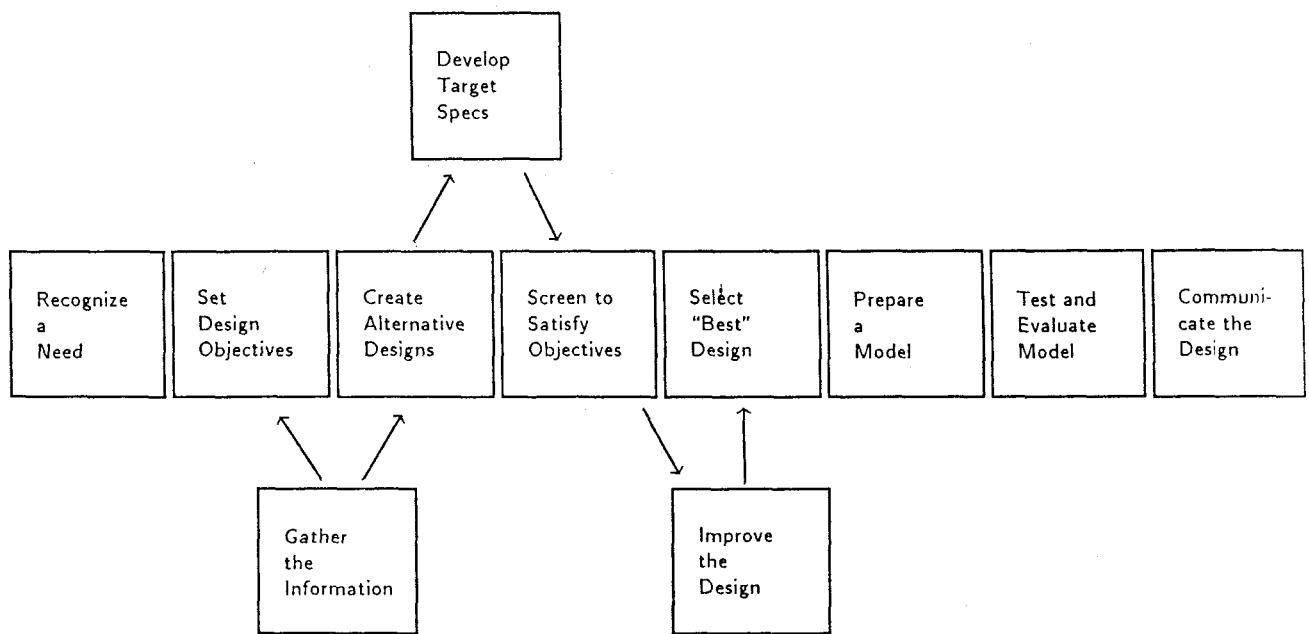
% Bottom lines will be even.

\notitlepage
\pin{Opt}
\cl{\bf{MECHANICAL \& MATERIALS ENGINEERING}}
\cl{\bf{ORGANIZATIONAL CHART}}
\vs{10pt}
\drawboxmat{10pt}
\drawboxrulesize{2pt}
\line{\hfill{\drawbox{\vbox{\hsize=2.1in {\nin\bf{DEPARTMENT CHAIR}}}}}\hfill}
\vs{-5pt}
\cl{@\Bigg\downarrow@}
\vs{2pt}
\line{\hfill{\drawbox{\vbox{\hsize=.8in{FACULTY}}}}}\hfill}
\vs{2pt}
\newbox\arrow \setbox\arrow=\hbox{@\downarrow@}
\def\daf\lower8pt\copy\arrow
\line{\hfill\da
    \hs{-1.2\wd\arrow}
    \vrule width 2in height .4pt depth Opt
    \vrule width .4pt height 10pt depth Opt
    \vrule width 2in height .4pt depth Opt
    \hs{-1.2\wd\arrow}
    \da\hfill}
\vs{2pt}
\line{\hs{18pt}{\drawbox{\vbox{\hsize=1.8in{GRADUATE STUDIES\vs{Opt}
    COMMITTEE (ME)}}}\hfill {\drawbox{\vbox{\hsize=1.5in{SECRETARY
    IV}}}} \hs{32pt}}
\vs{2pt}
\line{\hfill - Office Staff \hs{24pt}}
\vs{8pt}
\line{\hs{18pt}{\drawbox{\vbox{\hsize=1.8in{\nin GRADUATE STUDIES\vs{Opt}
    COMMITTEE (MSE)}}}\hfill {\drawbox{\vbox{\hsize=1.9in {ASSISTANT
    FOR\vs{Opt}COMPUTER SYSTEMS}}}} \hs{4pt}}
\vs{8pt}
\line{\hs{11pt}{\drawbox{\vbox{\hsize=1.9in{UNDERGRADUATE \vs{Opt} STUDIES
    COMMITTEE}}}\hfill {\drawbox{\vbox{\hsize=2in {ASSISTANT
    FOR\vs{Opt}RESEARCH FACILITIES}}}}}
\vs{8pt}
\line{{\drawbox{\vbox{\hsize=2.05in{LABORATORIES, EQUIP-\vs{Opt}MENT \&
    FACILITIES\vs{Opt} COMMITTEE}}}\hfill {\drawbox{\vbox{\hsize=1.75in
    {PROGRAM \vs{Opt}ASSISTANT III:\vs{Opt} STUDENT SERVICES}}}} \hs{13pt}}
\vs{8pt}
\line{\hs{60pt}{\drawbox{\vbox{\hsize=1.2in{SAFETY \vs{Opt}
    COMMITTEE}}}\hfill {\drawbox{\vbox{\hsize=2in {TECHNICAL
    SERVICES\vs{Opt}SUPERVISOR}}}}}

```

## Appendix B: $t_{EX1}$ Output of Appendix A

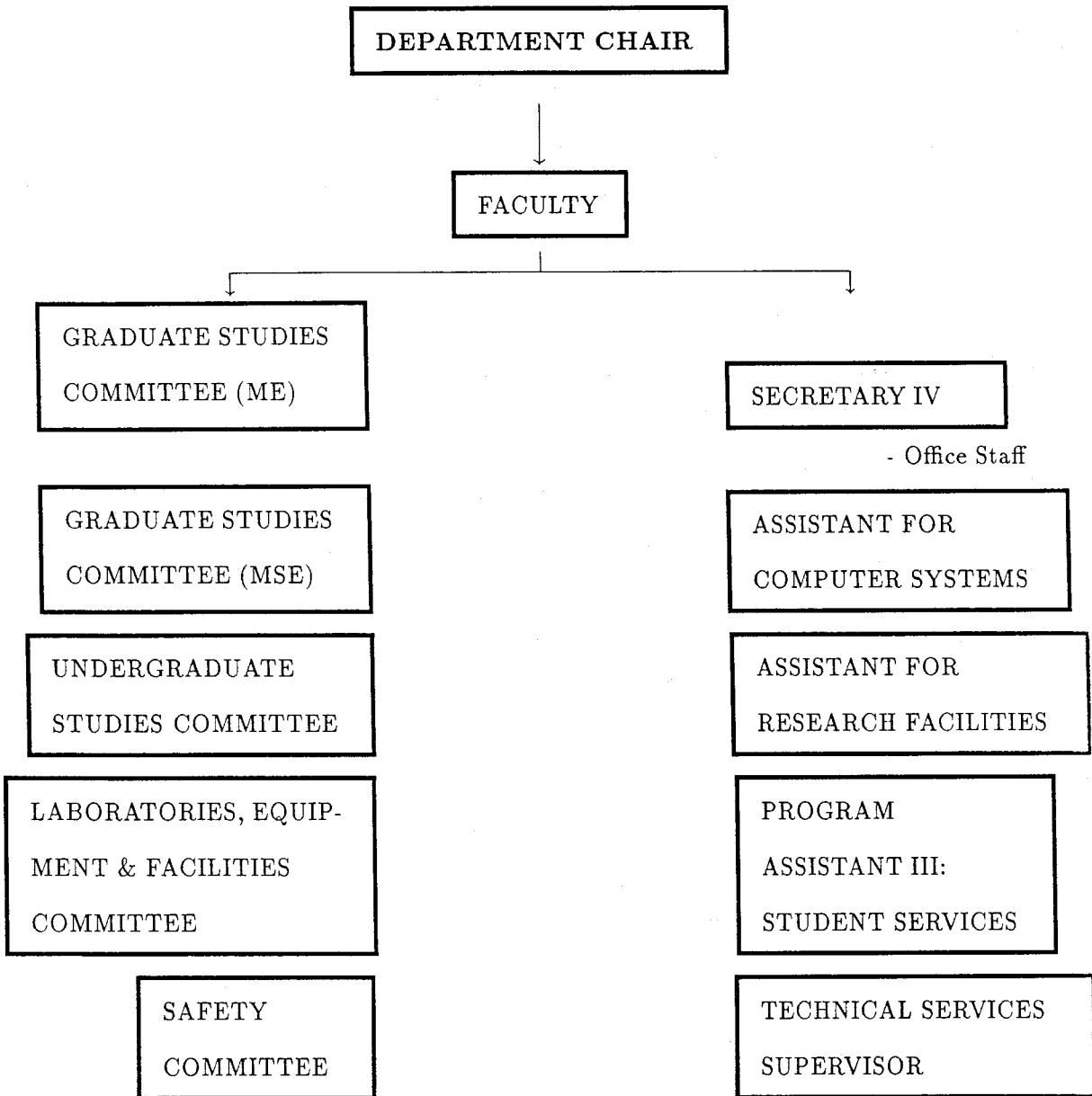




The Expanded Design Process - 11 Step.

Activity	Months												
	J	F	M	A	M	J	JU	JL	A	S	O	N	D
ISC sub-project:													
A. feature abstraction update													
B. engagement abstraction update													
C. run time environment design update													
D. run time environment hardware and software update													
E. control system design													
F. implement control algorithm													
G. ISC system testing													
Process models sub-project:													
H. process model development													
I. mechanistic model parameter investigation													
J. control/process model system simulation													
Sensor sub-project:													
K. design of AE fluid coupled sensor for CNC mill													
L. fabrication of AE sensor													
M. develop AE sensor monitoring environment for ISC													
N. investigate AE signal for end mill surface generation													
O. fabricate OM-2 force sensing system													
P. calibrate and test force sensing system													
OM-2 installation sub-project:													
Q. complete mechanical and hydraulic installation													
R. install controllers													
S. initialize controller logic and test system													
T. prepare final report													

MECHANICAL & MATERIALS ENGINEERING  
ORGANIZATIONAL CHART





TEX USERS GROUP  
STANFORD UNIVERSITY  
STANFORD, CALIFORNIA  
AUGUST 20-23, 1989

# T<sub>E</sub>X in México

MAX DÍAZ

Aurion Tecnología, SA de CV  
Arquímedes #219-1  
Col. Polanco  
11570 México, DF  
MEXICO  
(52-5)203-2662, FAX 203-5170

## ABSTRACT

Since 1985, Aurion has been a leading supplier of typesetting solutions in Mexico, with applications ranging from simple word processing, to magazines and books, encyclopedias, product catalogs, directories, and ending with the massive Presidential Inform. Most of these are T<sub>E</sub>X-based solutions, where Aurion develops the macros, and accessory programs, installs the system and trains the production personnel.

Examining some of these complex publications, we will show the problems encountered, and show how we go about making the whole cycle succeed: from convincing the production managers to use T<sub>E</sub>X, to finally achieving the output they wanted. We will also examine some cases where we do not recommend T<sub>E</sub>X, and mention the limitations of T<sub>E</sub>X we have run into.

### 1. Introduction

T<sub>E</sub>X has had considerable success in Mexico, perhaps helped by the fact that it is pronounced exactly like MÉXICO in Spanish: *may'-he-coh*.

In most cases, the success has been in understanding the users' needs, then providing them with a complete solution: selecting and installing the system, writing special macros with documentation, developing special utilities like word processor converters, training the production personnel, telephone support, and anything else that is needed. The applications are varied and the solutions are quite satisfactory.

Section 2 describes the more challenging and interesting T<sub>E</sub>X projects we have encountered; Section 3 talks about how we have managed to make them successful; and finally, Section 4 lists some of the major limitations that T<sub>E</sub>X has for commercial applications.

### 2. Case Studies of T<sub>E</sub>X Applications

The simplest T<sub>E</sub>X application in Mexico is as a word processor, with a set of macros that allow fifteen or twenty effects (center line, change font, etc.) and prevent the user from receiving any unpleasant error message; this is of course a very limited system, but the easiest one to learn (training took half an hour) for the very high typographic quality it gives. This is used by one of the secretaries of the President of Mexico to type speeches, memos, invitations, to label envelopes, and other office applications.

Many other applications are being pursued in Mexico, from magazines to advertisements to commercial typesetters. We have selected the most significative and describe them in the next four subsections.

#### 2.1 Electronic Publishing — Databases

More recently, this has been called "database publishing". It is a technique to obtain a camera-ready original as automatically as possible from a database, avoiding re-typing the text, eliminating manual

markup and pasteup, etc. The idea is very simple: given a format and macros, export the records from the database with the macro calls embedded; then compose and do only minor manual corrections. The important part is that the publishing part becomes more or less automatic and transparent, and the user from then on has to worry only to keep his data updated.

We have developed a dozen similar applications, of which the more interesting are:

#### *Bibliografía Mexicana, National Library of Mexico*

This was the first database publication we undertook. It is a bi-monthly publication made from a database in a minicomputer; originally, the text was printed on an impact printer, re-typed again on a composer machine and then pasted up manually. When we started the project, the publication was a couple of years behind schedule.

The solution was simple enough: connect a PC to the mini to download the data with the proper markup, and then compose and print camera-ready output to a laser printer. The only difficult part was to take care of the many rules of spacing, fonts and breaks to make the output resemble the Library of Congress format (see Figure 1), then automate the running heads, index and table of contents. As a mere experiment, we also produced catalog cards (see inset in Figure 1).

#### *Membership Directory, American Chamber of Commerce of Mexico*

The directory is similar to the previous one, but in the last phase the user had to add invocations to macros to leave space for advertisements (quarter, half, whole page) and to balance the last page of each section. There were many problems having to do with `Overfull hbox`, because uppercase hyphenation was forbidden due to the mixed appearance of many proper names in Spanish or other languages (see Figure 2).

The first year, the user complained sourly of the time it was taking to get the directory done, not seeing that it required macro and database development. The second year, one operator produced the whole 200-page directory in a few days.

#### *J.T. Baker Catalog*

This is a 400-page product catalog produced on an annual basis only, due to its complexity. Initially, the user attempted producing it with an interactive DTP<sup>1</sup> program, only to discover it took him more time than to do it manually, even though it required pen drawing and rub-on letters for some of the symbols. This is a very good example of the kind of publication made best with TeX automation, although it appears to require a lot of manual tune-up.

Besides the obvious complexity of the pages (see Figure 3), there were other major challenges:

- Each group is a chemical product; it can be broken from a column to the next, but not over pages.
- There are certain rules for vertical spaces, but not all groups have the same elements.
- Chemical formulas have links on top or below or both.
- The dot fill from *Producto* to *Pureza* must always appear; if either is too long, the line must be broken and the dot fill begin at the left margin.

Currently, production of the whole catalog is truly a matter of hours, to the delight of the user: not only for the incredible savings in typesetting, but also because it is the kind of publication that needs to be done frequently, due to changes in products and prices.

## **2.2 Electronic Publishing — Automation**

#### *Calendar, Sistemex*

This is a yearly calendar (see Figure 4) with space for appointments; it shows the current day and week number, the days left, the current month's calendar on even pages, and next month's on odd. At the top of each day, it shows events celebrated that day. Final printing takes place in two colors.

All of this is wholly automated, using programs that compute the calendar, pull out the events out of a database, and with a special output driver that produces two pages for the different color plates.

---

<sup>1</sup> "DTP" is the current acronym for "Desktop Publishing -Ed.

The publisher reported spending two weeks of production time for every month of the calendar, when it was done with traditional methods. He now spends two hours.

#### *Study of Operations, Mexican Association of Stock Brokers*

A monthly financial report with tables and histograms (Figure 5). Starting with a spreadsheet program to print the data to disk, a collection of special programs take on from there. One parses the columns and with a given scale (thousands, millions), computes the histogram, then the table shown at the lower left is typeset, selecting the first five that are in turn set into the small table on the upper right (overlapped onto the histogram). The programs produce a *T<sub>E</sub>X* file that is next \input into a master file, which is then processed and printed.

Current production currently takes literally a few minutes.

### **2.3 Standard Formats**

#### *Encyclopedia of Mexico*

Although it is a fourteen-volume work, with six hundred pages per volume, this is actually a fairly simple two-column publication, because for this edition relatively few photographs were inserted. It is a good example of what can be automated with a set of two dozen macros, some of them even invisible (such as boldface characters that were entered in the word processor); it is the most common kind of application and solution that we have installed.

The difficulties for the macros were in the placement of figures (a space was left and the photograph was pasted up latter), since it required, for instance, that a page with a two-column photograph would not simultaneously contain a one-column photograph, and similar placement rules. It also required that lines would match from one column to the other, forcing all vertical dimensions to be multiples of the leading and preventing *T<sub>E</sub>X* from expanding \parskip glue, thus in turn creating widow lines, which the user fixed by adding a word or two. It is worth mentioning that the user had a terrible time trying to get tables done right; when they were typesetting the Encyclopedia, the Ampersand utility mentioned in Section 3 was not available.

The whole work is now finished; it took the publisher one laser printer, three PCs, and about a month to complete each volume from typing to proof correction to camera-ready originals.

### **2.4 Presidential Inform**

This annual *Inform*<sup>2</sup> is the most ambitious project we have undertaken. It consists of two thousand legal size, landscape pages, with about 70% of them with intricate tables (see Figures 7 and 8), for which *T<sub>E</sub>X*'s \halign just would not work.

To complicate things further, the production of the whole report takes place in about two months. The data comes in all formats imaginable, from typewritten text to any number of word processor or spreadsheet formats, to mainframe databases. The markup had to be extremely simple, due to the technical level of the operators involved in the project. Finally, both the *T<sub>E</sub>X* processing and the printing had to be very efficient, which did not permit using one of the few available macro packages for tables.

There were a number of formidable challenges for *T<sub>E</sub>X*:

- Table contents are many pages long.
- Table heads have over 60 different designs on several levels, with variable heading text.
- There are side notes, with automatic numbering somewhere in the table or heading, with numbering reset to 1 on each page. Sometimes this note is repeated over several pages or the whole chapter.
- In some columns (e.g., *Unidad de Medida*), when the word is repeated, a quote is inserted, but at the beginning of page the repeated word should appear wholly.
- Table contents sometimes have a simple linear structure (Figure 7), but others have a tree-like structure (Figure 8), with main paragraphs (to the very left) having sub-items and these in turn sub-sub-items.

---

<sup>2</sup> Consisting of a speech read by the President of Mexico the first day of September, and nine annex volumes with information about the Secretariats, budgets and similar information.

The project was ready in time before September first last year after developing the following:

- A utility to design interactively the more than 60 table heads and be able to modify them quickly to the user's specifications. This produced *TeX* files to be used later.
- About 30% of the work utilized 58 of those formats, with linear structure and a markup simple enough to be made with a standard word processor.
- The other 70% was for 2 kinds of tables, that in Figure 7 (simple) and that in Figure 8 (complex); for one or other reason, neither could be marked up efficiently with a text editor and so we developed database and data entry programs to aid in the process. This program then automatically produced *TeX* files with all necessary macro invocations.

The necessary *TeX* macros are rather long and complicated to be even begun to be described here.

### 3. The Whole Cycle

At the dawn of Desktop Publishing, there was a joke about laser printouts. Production managers would say they were plain "trash", while finance managers, learning of costs, would say they were "not bad trash". Back in 1985, when we started showing *TeX* printouts to professional typographers, they were very pleased with the many capabilities of *TeX* on pagination, justification, math typesetting, and of course lower costs. But they were dismayed by other aspects: difficulty to learn, low quality laser printouts, non-standard fonts, etc.

Some things that are obvious to academic users are not to commercial users. For example, entering formulas like:

$$\$ \int_0^\infty f(x) dx \$$$

instead of entering symbols in a special keyboard, requires some knowledge of math, which many excellent typists lack. Or the accents: having to type `as\i\l` instead of `asi`, as it appears in a PC screen, and as is handled by many word processors is unacceptable for production work. Or the fonts: the Computer Modern fonts have their merits, but they are no substitute for the many classic and widely-used font designs. We realized this and similar things when we installed the first systems outside of universities.

The first task was to develop a hyphenation algorithm for Spanish, which turned out to be quite simple, since accents do not affect Spanish hyphenation, and since the exception dictionary is small; it works rather well, except for the fact that *TeX* will not hyphenate words in the vicinity of an accent. Next, for the accents, we managed to convince Personal *TeX* to produce a non-standard *TeX* that could input files with PC extended characters and translate them to control sequences (`\`a` to `\'a`), a true relief to typists and correctors. We did not attempt to translate the thousands of *TeX* control sequences nor the *TeXbook*, since people technical enough to understand it would be proficient enough in English to read it.

The second task was to develop a set of macros that had the basic things needed in any publication, as an addenda to *plain*: all kinds of paragraph shapes, boxed text, multi-column (two to thirteen) output with optional balancing, formats for letters, inclusion of images, optional but very precise crop marks, etc. All of this is written in a way that allows easy customization for different applications; for example, the `\footnote` macro is written with about ten parameters affecting the vertical spacing between them, the font used for text and numbering, leading, paragraph shape, etc. As well, many rather fancy `\output` routines can be written in 10 lines, with high precision in the positioning of headers, text box and other elements of the page. This set of macros is essentially a re-write of the old Fácil-*TeX* macro package (written for *TeX-78* in 1980-81) that circulated in the public domain.

Documentation for this and for the basic control sequences of *plain* was written in Spanish, and a three-day (12 hour) hands-on course was designed to teach it all. After this training, most users can do simple things and can use the *TeXbook* as a reference. Users ranged from university professors, to phototypesetter operators, to people who had to be told how to insert a floppy disk into a PC. In general, about 75% of these are still happily *TeXing*, and many have trained other users.

Having finished this training, we started work with users on their particular applications, sometimes doing the macro development in their presence for their additional understanding. We usually supervised closely the production of their first documents until both were satisfied. After this, we seldom received further requests for major help, only occasional questions.

As for output drivers, in the early stages we developed the screen previewer MAXview, which

is very simple to use, fast, handles high-resolution monitors, and has the enormous advantage from the very first versions, to be able to use the laser printer fonts and reduce them for any screen. This program was later adapted to drive high-performance laser printer video controllers (such as Tall Tree's JLaser and LaserMaster's CAPCard); these drivers have the same functionality as MAXview, and were designed to be very fast (up to 5 pages per minute) and precise in sizes and margin positioning. They also handle legal size, landscape printing (without special fonts), Paintbrush images, can reduce pages to show 4, 9, or 16 pages per sheet for fast proofing, etc. A special version can handle a resolution of  $600 \times 300$ , when used with specially generated `cmr` fonts, or with Bitstream outlines. We also have in the works a driver that takes advantage of the laser and screen controller boards ability to scale fonts on the fly. MAXview made it outside of Mexico, but not the laser printer drivers.

As an outgrowth of the Presidential Inform project, we developed an interactive utility called Ampersand that allows designing and filling very complex tables, to the extent that users will hardly ever need to use `\halign` again for tables. To do it, one designs interactively the table layout, the heading, rulers, sizes, fonts, leading, etc., producing a `TEX` file, compatible with `plain`. Then with another program, the table data is appropriately marked up for insertion into the table, which may be more than one page long. Finally, both files are `\input` and the table is ready. The markup is made simple enough that you can even easily edit those files to change something, without needing to start over. The utility is so flexible, it even allows making flow or organization diagrams, as shown in Figure 9. A first release will be available at the time of the TUG meeting.

#### 4. `TEX` Drawbacks and Limitations

`TEX` is an extraordinary tool, but awfully hard to use if special page layouts or complicated elements such as tables are needed. It is too much to ask a user in a commercial environment to undertake reading the *T<sub>E</sub>Xbook* to attack applications such as those described in Section 2, or to be happy with the layouts found in a macro package such as `LATEX`, whose styles are probably as hard to define as in `plain`. We doubt that any of those applications would have been successful without extensive joint work with the users and special training for the operators. For this same reason, the least successful of our customers are in sites such as publishing houses, which constantly require many special definitions and which cannot and will not depend forever on our help.

There are other inherent limitations of `TEX` that we don't see how to solve without considerable effort, and are important in commercial applications:

- The `\doublehyphendemerits` takes care of two consecutive hyphens, but does not allow to set a maximum of three or any given number of consecutive hyphens, for example, which becomes critical in narrow text. Once, trying to set very narrow text (7 or 8pc), we had to allow more than two consecutive hyphens, but then obtained a paragraph with 20 consecutive ones! A partial solution is to increase the `\hyphenpenalty` and allow greater `\hbadness` and similar parameters.
- Even when allowing only two hyphens, `TEX` allows any number of adjacent lines ending in punctuation signs, something that many typographers consider unacceptable.
- Many typographers in Mexico do not allow two lines to begin or end with the same (usually short) word, like "the", and `TEX` has no control over this (called alley streets or rivers). Fixing this kind of thing, as the last two items, requires much effort, as you have to insert a `\break`, compose again, preview, and so on.
- Although not considered a very good practice in fine typography, allowing variable interletter spacing in lines is sometimes the only way to get a paragraph to right justify without re-phrasing.
- `TEX` does not handle tracking (changing the character widths by a uniform predefined amount, to make fonts tighter or wider), without generating a whole new `tfm` and fonts, or without special extensions to the program itself and to the output drivers.
- If a publication is needed with, for example, side notes to the right on odd-numbered pages and to the left on even-numbered pages, because of `TEX`'s look-ahead algorithm, one is tempted to use macros like `\ifodd \pageno...` which naturally don't usually work near page breaks. Of course, there are ways around it (as in the *Presidential Inform* side note numbering), but usually in rather complicated and particular ways.

Other limitations of `TEX` have been solved by Ferguson's Multilingual version of `TEX`: for example, being able to specify the minimum size of a word and minimum letters at beginning and end, to allow

hyphenation, and the ability to hyphenate accented words.

Given a typesetting problem, in general if it is easier to describe *how* to do it than *to do it*, we would still recommend **T<sub>E</sub>X**; such is the case with a straightforward text book, but not in the case of a multi-column magazine whose layout changes from issue to issue. A recent example of the first instance is the Department of Publications of the University of Mexico. Producing several dozen books per month, these are part of about only 50 collections with the same page design; macros can then be developed with similar syntax over different collections so that typists can be trained to do the initial markup, with one or two moderately educated T<sub>E</sub>Xperts composing, previewing and fixing final details. Some of these collections, such as the art books, are too complicated to be handled by macros and are being produced with an interactive DTP system.

However this is rapidly changing, as many other DTP systems allow defining page layouts interactively, and markup of text with a text editor, but in the previewing phase, allow interactive modifications without going back to the original text (Ventura Publisher, for example). Nevertheless, for most applications and the solutions described in Section 2, **T<sub>E</sub>X** is still a good choice.

## Bibliography

Knuth, Donald E. *The T<sub>E</sub>Xbook*. Reading, Mass.: Addison-Wesley, 1984.

## Figure 1: Bibliografía Mexicana

<p><b>Ciencias sociales (300)</b></p> <p><b>Sociología (301)</b></p> <p>González Loyola, Pablo Introducción a la teoría sociológica / Pablo González Loyola. — Querétaro : Universidad Autónoma de Querétaro, 1982. 237 p. ; 21 cm. — (Universidad Autónoma de Querétaro. Ser. humanidades. Colec. historia y sociología) Bibliografía: p. 225-236. 1. Sociología. I. Universidad Autónoma de Querétaro. II. t. III. Ser.</p> <p>301</p>	<p><b>Educación (370)</b></p> <p><b>Generalidades (371)</b></p> <p>Universidad Nacional Autónoma de México. Dirección General de Servicios Médicos (7a Jornadas Internas de Trabajo : 1982 : México, D.F.) Memorias / VII Jornadas Internas de Trabajo. — México : UNAM, Dirección General de Servicios Médicos, 1982. viii, 367 p. ; il. ; 20 cm. "25 y 26 de noviembre de 1982. Auditorio de Centro Médico Universitario, Ciudad Universitaria, D.F."</p> <p>1. Universidad Nacional Autónoma de México — Asuntos sanitarios — Congresos. I. t.</p> <p>371.710672</p>
<p>Pelto, Pertti J.</p> <p>El estudio de antropología / Pertti J. Pelto, George D. Spindler. Con un capítulo final en que se sugieren métodos a los maestros para la enseñanza primaria y secundaria / por Raymond H. Muessig y Vincent R. Rogers ; tr. Antonio Garza y Garza. — México : UTHEA, 1980. xii, 199 p. ; 17 cm. — (Manuales UTHEA. Arte ; 356) Traducción de: The study of anthropology. Incluye bibliografías ISBN 968-438-356-8</p> <p>1. Antropología — Est. y ens. I. Spindler, George D. II. Muessig, Raymond Henry. III. Rogers, Vincent Robert, 1926—. IV. Garza y Garza, Antonio, tr. V. t. VI. Ser.</p> <p>301</p>	<p><b>Educación elemental (372)</b></p> <p>Beatriz : muñeca viva : con guardarropa para todo el año [Modelo]. — México : Fernández, [197—?] 1 muñeca de cartón, 20 vestidos, trajes y abrigos, 11 sombreros y zapatos, 1 tijeras : cartón y papel, col. ; en caja, 34 x 25 x 4 cm. — (FESA ; 4276) Modelitos.</p> <p>1. Actividades creativas y trabajos preescolares. I. Ser.</p> <p>372.55</p>
<p>Gobiernos centrales (351)</p> <p>Madrid Hurtado, Miguel de la, Pres. de México, 1984—</p> <p>Palabras del Secretario de Programación y Presupuesto Miguel de la Madrid H. en la Reunión Titoria del Programa de Capacitación de Empl la S.P.P. : ciudad de México, 22 de mayo — México : SPP, Dirección General de Relaciones Públicas, 1981. 6 p. ; 23 cm. — (Méjico. Secretaría de Programa Presupuesto. Dirección General de Difusión y Relaciones. Ser. intervenciones y entrevistas ; no. 75) ISBN 968-809-262-2</p> <p>1. Empleados — Capacitación — México — Alo conf. 2. México. Secretaría de Programación y Pres Personal — Dirección. I. México. Secretaría de Pro y Presupuesto. Dirección General de Difusión y Rela blicas. II. Reunión Evaluatoria del Programa de Ca de Empleados de la Secretaría de Programación y Pi (1981 : México, D.F.). III. t. IV. Ser.</p> <p>351.7220972</p>	<p>Eréndira y sus trajes regionales [Modelo] : muñeca viva. — México : Fernández, [197—?]. 1 muñeca de cartón, [16] trajes, 1 tijeras : cartón, papel y plástico, col. ; en caja, 34 x 25 x 4 cm. — (FESA ; 4276) "Los más vistosos trajes típicos de México". "Se para solita".</p> <p>1. Actividades creativas y trabajos preescolares. I. Ser.</p> <p>372.55</p>
<p>Wilcox, Francis Orlando, 1908— ed.</p> <p>The Atlantic community: progress and prospects. Edited by Francis O. Wilcox and H. Field Haviland, Jr. New York. Praeger [1963] viii, 294 p. 21 cm. "The essays ... originally appeared in a special issue of International organization ... vol. xvii, no. 3, summer, 1963." Bibliographical footnotes.</p> <p>1. North Atlantic region — Politics. I. Haviland, Henry Field, 1919— joint ed. II. International organization. III. Title.</p>	<p>D844.W52</p> <p>Biblioteca Nacional</p> <p>327</p> <p>83-20149</p> <p>86[4] TEX</p>

## Figure 2: Membership Directory

# A

<b>A SUS ORDENES, S.A.</b> Av. Chapultepec 57, Mezzanine Col. Centro Del. Cuauhtémoc 06040 México, D.F. Lic. Fernando Yllanes Ramos, President; Paul P. Payne, Vice-President; Lic. Gerardo Salinas Baigts, General Manager. Employment agency. Established 1968 • Personnel 25	578-9348	<b>A.E. VON HAUCKE, S.A. DE C.V.</b> Salamanca 34, 2o. Piso Col. Roma Del. Cuauhtémoc 06700 México, D.F. Ing. Juan M. del Castillo, General Director; Ing. José del Castillo, Comptroller; Lic. Pedro R. Pérez Orozco, Commercial Director. Manufacturers of modular office furniture. Established 1968 • Personnel 110	533-2762
<b>A. CALDERONI, S.A.</b> Ayuntamiento 161 Col. Centro Del. Cuauhtémoc 06040 México, D.F. Alfredo Calderoni García, General Director. Manufacturers of industrial carbon. Established 1962 • Personnel 7	510-3887	<b>A.F. ROMERO &amp; COMPANY, INC.</b> 477 Railroad Boulevard P.O. Box 989 Calexico, CA 92231 U.S.A. Alex Romero, Executive. Customs brokers.	(619)357-1183
<b>A. PALAZUELOS Y COMPAÑIA, S.C.</b> Colima 114, 1er. Piso Col. Roma Sur Del. Cuauhtémoc 06700 México, D.F. Alfredo Palazuelos, President; C.P. Félix Palazuelos, Director; Andrés Palazuelos, Manager. Customs brokers, air freight, shipping containers. Established 1920 • Personnel 200	533-1430	<b>A.H. ROBINS DE MEXICO, S.A. DE C.V.</b> Autopista México-Querétaro Km. 37 Apdo. Postal 71 54730 Cuautitlán, Mex. Ennio Sánchez Jasso, Vice-President and General Manager; Earle A. Jurgensen, Marketing and Sales Manager; José Luis Pacheco F., Purchasing Manager; C.P. Félix Ovando Venegas, Finance Manager. Manufacturers of pharmaceuticals. Established 1960 • Personnel 200 • Telex 171311	565-7300
<b>A.B. CHANCE DE MEXICO, S.A. DE C.V.</b> Camino Real de Santa Mónica 238 Col. Vista Hermosa 54080 Tlalnepantla, Méx. Ing. Julio T. Moreno, President and General Director; C.P. Helio Rodríguez, Finance Director; Ing. Pedro Ochoa, Commercial Director; C.P. Florentino Rodríguez, Administrative Manager; Ing. Mario Villavicencio, Materials Director. Manufacturers of electrical capital goods. Established 1952 • Personnel 289 • Telex 172634	397-1333	<b>A.S., S.A. DE C.V.</b> See Guadalajara Members	
<b>A.C. NIELSEN COMPANY</b> José Luis Lagrange 103, Desp. 11 Col. Chapultepec Morales Del. M. Hidalgo 11570 México, D.F. Apdo. Postal M-2638 06000 México, D.F. A. Traslosheros, General Manager; J. Oviedo, Administrative Manager; I. Chávez, Finance Manager. Retail audits, consumer panel, advertising expenditure measurement, ad-hoc studies, special analysis. Established 1967 • Personnel 509 • Telex 1773104	395-0399	<b>AADELE TURISMO, S.A.</b> Versalles 21-202 Col. Juárez Del. Cuauhtémoc 06600 México, D.F. Adele Sabalette, Director; Lic. Delfor I. Sabalette Graf, Public Relations Manager; C.P. Ignacio Colín, Comptroller. Travel agency. Established 1979 • Personnel 12 • Telex 1761144	546-6971 546-9214
<b>A.C. NIELSEN COMPANY DE MEXICO, S.A.</b> Av. Lerdo 251 Norte Col. Centro 32000 Ciudad Juárez, Chih. Lic. Ovidio H. Enríquez, General Manager; Eduardo Valle, Comptroller; William Nicolai, Finance Director. Marketing services. Established 1969 • Personnel 10,000 • Telex 33880	5-1022	<b>ABA, S.A. DE C.V.</b> Agustín Delgado 39-A Col. Tránsito Del. Cuauhtémoc 06820 México, D.F. Moisés Sidauy Cherem, President; Iván Chimal, Technical Director; Edmundo Vázquez, Plant Manager; Emilio Kichik Sidauy, Assistant Director. Manufacturers of polyethylene bags. Established 1960 • Personnel 250	542-4435 542-7057

**Figure 3: Product Catalog**

J. T. BAKER Reactivos y Productos para Laboratorio 880 AL

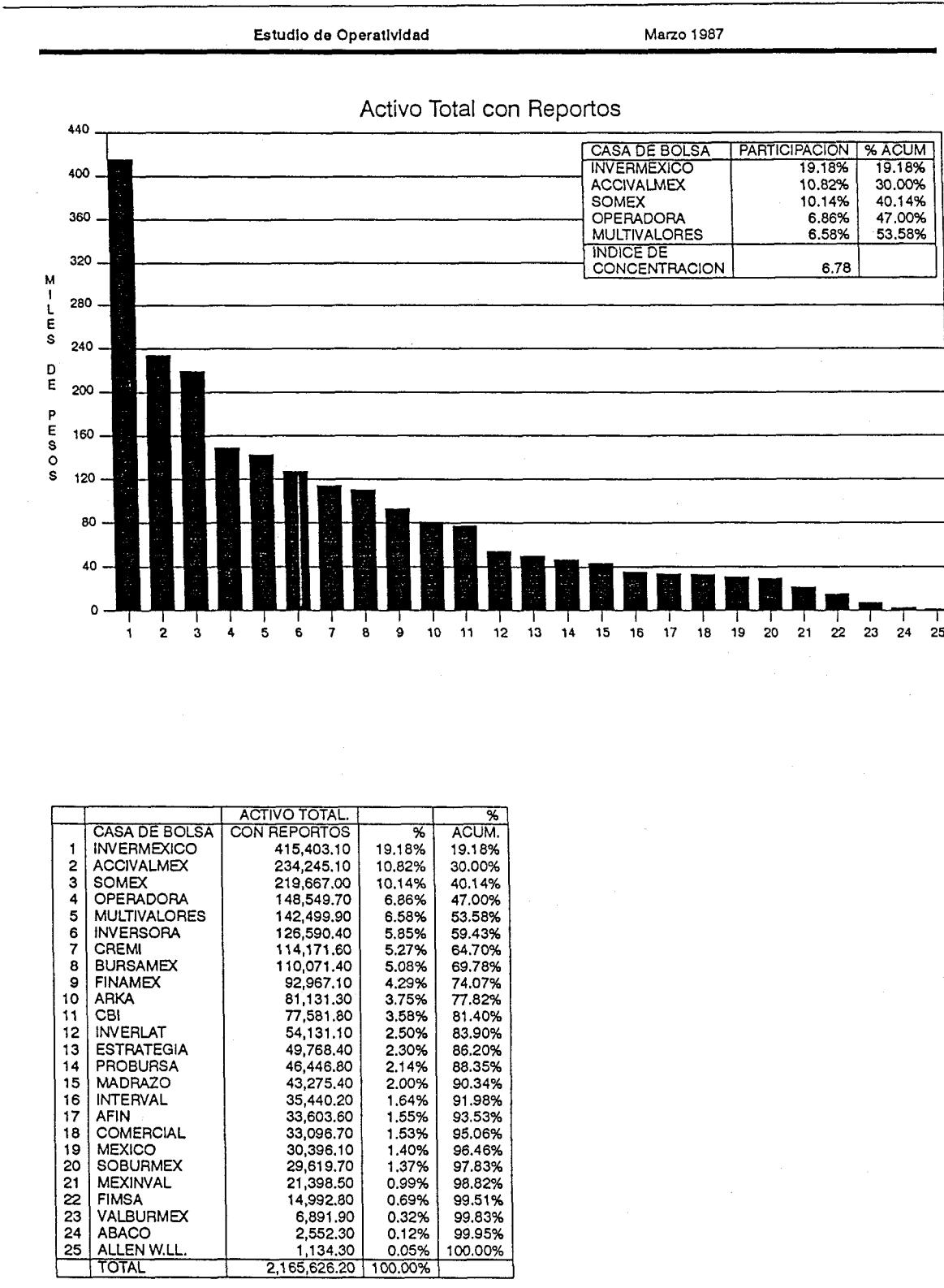
AQUAL

Código	Presentación	Código	Presentación
A-Aminoantiprina'BAKER'	$\text{CH}_3\text{N}(\text{C}_6\text{H}_5)\text{CO}(\text{NH}_2)\text{CCH}_3$ PM 203.25 P.F. 107-109°C esta es una nota impertinente esta es una nota impertinente esta es una nota impertinente Adecuado para Absorción de humedad ..... pasa prueba Adecuado para para para para para para Absorción de humedad etc etc ..... pasa prueba Adecuado para para para para para para para Absorción ..... pasa prueba	$\alpha\text{-Chloralose}$ (p.48) $\text{OCH}(\text{CCl}_3)\text{OCHCHCHOCH}(\text{CHOHCH}_2\text{OH})\text{O}$ PR 676 ALMACENAR EN AREA COLOR: Negro.	
A630-03 25 g A630-05 100 g 9094-07 20 L También viene en la siguiente presentación 9094-R 181 Kg CAS NO.: 83078-00-0 SPILL-KIT: <input checked="" type="checkbox"/> <input type="checkbox"/> EPA-HW: <input checked="" type="checkbox"/>	AA Estándares Ver Listado Alfabético	ABX BAKERBOND Ver La Sección de Productos de Investigación para Cromatografía	
ESCALA SAF-T-DATA: Salud Inflamable Reactividad Contacto <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/>	AC10 Celulosa Ver Celulosa AC10	ALCONOX® para uso manual y uso en lavadoras ultrasónicas A461-05 1.8 kg CAS NO.: - -	
ALMACENAR EN AREA COLOR: Naranja (Almacén general).		ESCALA SAF-T-DATA: Salud Inflamable Reactividad Contacto <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/>	
○ ABipyridine (p.30) $\text{N}:\text{CHCH:CHCH:CC:CHCH:CHCH:N}$ PA 5645 ALMACENAR EN AREA COLOR: Verde patriótico.	ALMACENAR EN AREA COLOR: Naranja (Almacén general).	ANHYDRONA,'BAKER ANALYZED' Reactivo (A.C.S.) Para secado (desecante, perclorato de magnesio) El análisis real del lote se reporta en la etiqueta $\text{Mg}(\text{ClO}_4)_2$ PM 223.21 Según especificaciones de A.C.S. Adecuado para Absorción de humedad ..... pasa prueba Límites Máximos de Impurezas: Ácido Libre Titulable ..... >0.005 meq/g Base Titulable ..... <0.025 meq/g Agua ( $\text{H}_2\text{O}$ ) ..... 8%	0828-01 500 g CAS NO.: 10034-81-8 EPA-HW: <input checked="" type="checkbox"/> IMO: 5.1-1475 NFPA: 1-0-0 OXY
□ ACamphor (p.44) $\text{CH}_3\text{COCH}_2\text{CH}[\text{C}(\text{CH}_3)_2]\text{CH}_2\text{CH}_2$ PG 24347 ALMACENAR EN AREA COLOR: Amarillo bilis.	ESCALA SAF-T-DATA: Salud Inflamable Reactividad Contacto <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/>	ESCALA SAF-T-DATA: Salud Inflamable Reactividad Contacto <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/>	
● ARaya arriba y abajo 1 $\text{CH}_3\text{NN}(\text{C}_6\text{H}_5)::\text{CH}_7\text{OCCH}_3$ PM 56757 ALMACENAR EN AREA COLOR: Ninguno.	ALMACENAR EN AREA COLOR: Amarillo (Reactiv).	ALMACENAR EN AREA COLOR: Amarillo (Reactiv).	
ACaffeine (p.39) $\text{CH}_3\text{NCON}(\text{CH}_3)\text{CO}\text{C:CN:CHNCH}_3$ PW 9890 ALMACENAR EN AREA COLOR: Café.			
ADichlorofluorescein (p.66) $\text{C}_6\text{H}_4\text{COOC}_6\text{H}_2\text{-2-Cl-3-OHO}_6\text{H}_2\text{-6-OH-7-Cl}$ PF 67 ALMACENAR EN AREA COLOR: Azul fosforescente		AQUALYTE PLUS, Coctel LSC para muestras acuosas y biológicas en ampolletas MILLI Ver Coctel LSC y productos accesorios	

Figure 4: Agenda Sistemex

AGOSTO																																													
LUNES 7		<i>Los Santos Reyes (Epifanía). Ntra. Sra. de Alta Gracia.</i> <i>Día del Geólogo. Día de la Enfermera. Fundación de Mérida, Yuc. (1542). Fiesta en los Reyes, Mich.</i>																																											
SEMANA 32	DIA 219	FALTAN 146	1989																																										
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="width: 10px;"></td><td style="text-align: right; padding-right: 5px;">8 Hrs.</td><td></td></tr> <tr><td></td><td style="text-align: right; padding-right: 5px;">9</td><td></td></tr> <tr><td></td><td style="text-align: right; padding-right: 5px;">10</td><td></td></tr> <tr><td></td><td style="text-align: right; padding-right: 5px;">11</td><td></td></tr> <tr><td></td><td style="text-align: right; padding-right: 5px;">12</td><td></td></tr> <tr><td></td><td style="text-align: right; padding-right: 5px;">13</td><td></td></tr> <tr><td></td><td style="text-align: right; padding-right: 5px;">14</td><td></td></tr> <tr><td></td><td style="text-align: right; padding-right: 5px;">15</td><td></td></tr> <tr><td></td><td style="text-align: right; padding-right: 5px;">16</td><td></td></tr> <tr><td></td><td style="text-align: right; padding-right: 5px;">17</td><td></td></tr> <tr><td></td><td style="text-align: right; padding-right: 5px;">18</td><td></td></tr> <tr><td></td><td style="text-align: right; padding-right: 5px;">19</td><td></td></tr> <tr><td></td><td style="text-align: right; padding-right: 5px;">20</td><td></td></tr> <tr><td></td><td style="text-align: right; padding-right: 5px;">21</td><td></td></tr> </table>					8 Hrs.			9			10			11			12			13			14			15			16			17			18			19			20			21	
	8 Hrs.																																												
	9																																												
	10																																												
	11																																												
	12																																												
	13																																												
	14																																												
	15																																												
	16																																												
	17																																												
	18																																												
	19																																												
	20																																												
	21																																												
MARTES 8		<i>S. Raymundo de Peñafort, S. Luciano, S. Crispín ob.</i> <i>Fiesta en Acámbaro, Gto. Estalla la trágica Huelga Obrera en Río Blanco, Ver. (1907).</i>																																											
SEMANA 32	DIA 220	FALTAN 145																																											
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="width: 10px;"></td><td style="text-align: right; padding-right: 5px;">8 Hrs.</td><td></td></tr> <tr><td></td><td style="text-align: right; padding-right: 5px;">9</td><td></td></tr> <tr><td></td><td style="text-align: right; padding-right: 5px;">10</td><td></td></tr> <tr><td></td><td style="text-align: right; padding-right: 5px;">11</td><td></td></tr> <tr><td></td><td style="text-align: right; padding-right: 5px;">12</td><td></td></tr> <tr><td></td><td style="text-align: right; padding-right: 5px;">13</td><td></td></tr> <tr><td></td><td style="text-align: right; padding-right: 5px;">14</td><td></td></tr> <tr><td></td><td style="text-align: right; padding-right: 5px;">15</td><td></td></tr> <tr><td></td><td style="text-align: right; padding-right: 5px;">16</td><td></td></tr> <tr><td></td><td style="text-align: right; padding-right: 5px;">17</td><td></td></tr> <tr><td></td><td style="text-align: right; padding-right: 5px;">18</td><td></td></tr> <tr><td></td><td style="text-align: right; padding-right: 5px;">19</td><td></td></tr> <tr><td></td><td style="text-align: right; padding-right: 5px;">20</td><td></td></tr> <tr><td></td><td style="text-align: right; padding-right: 5px;">21</td><td></td></tr> </table>					8 Hrs.			9			10			11			12			13			14			15			16			17			18			19			20			21	
	8 Hrs.																																												
	9																																												
	10																																												
	11																																												
	12																																												
	13																																												
	14																																												
	15																																												
	16																																												
	17																																												
	18																																												
	19																																												
	20																																												
	21																																												
<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr><td style="text-align: center; padding: 2px;">D</td><td style="text-align: center; padding: 2px;">L</td><td style="text-align: center; padding: 2px;">M</td><td style="text-align: center; padding: 2px;">M</td><td style="text-align: center; padding: 2px;">J</td><td style="text-align: center; padding: 2px;">V</td><td style="text-align: center; padding: 2px;">S</td></tr> <tr><td style="text-align: center; padding: 2px;">20</td><td style="text-align: center; padding: 2px;">21</td><td style="text-align: center; padding: 2px;">22</td><td style="text-align: center; padding: 2px;">23</td><td style="text-align: center; padding: 2px;">24</td><td style="text-align: center; padding: 2px;">25</td><td style="text-align: center; padding: 2px;">26</td></tr> </table>		D	L	M	M	J	V	S	20	21	22	23	24	25	26	<table border="1" style="display: inline-table; border-collapse: collapse;"> <tr><td style="text-align: center; padding: 2px;">6</td><td style="text-align: center; padding: 2px;">7</td><td style="text-align: center; padding: 2px;">8</td><td style="text-align: center; padding: 2px;">9</td><td style="text-align: center; padding: 2px;">10</td><td style="text-align: center; padding: 2px;">11</td><td style="text-align: center; padding: 2px;">12</td></tr> <tr><td style="text-align: center; padding: 2px;">27</td><td style="text-align: center; padding: 2px;">28</td><td style="text-align: center; padding: 2px;">29</td><td style="text-align: center; padding: 2px;">30</td><td style="text-align: center; padding: 2px;">31</td><td style="text-align: center; padding: 2px;">13</td><td style="text-align: center; padding: 2px;">14</td></tr> <tr><td style="text-align: center; padding: 2px;">D</td><td style="text-align: center; padding: 2px;">L</td><td style="text-align: center; padding: 2px;">M</td><td style="text-align: center; padding: 2px;">M</td><td style="text-align: center; padding: 2px;">J</td><td style="text-align: center; padding: 2px;">V</td><td style="text-align: center; padding: 2px;">S</td></tr> </table>		6	7	8	9	10	11	12	27	28	29	30	31	13	14	D	L	M	M	J	V	S							
D	L	M	M	J	V	S																																							
20	21	22	23	24	25	26																																							
6	7	8	9	10	11	12																																							
27	28	29	30	31	13	14																																							
D	L	M	M	J	V	S																																							
AGOSTO																																													

**Figure 5: Study of Operations**



## Figure 6: Encyclopedia of Mexico

### A-ABADEJO

**A.** Prefijo negativo en náhuatl, con mismo valor que la *alpha* privativa de los griegos. *Cualli* es bueno y *acualli*, malo; *neconi*, lícito, y *aneconi*, ilícito; *yel*, diligente, y *ayel* perezoso; *patiuh*, barato, y *apatiuh*, caro; *tlácatl*, persona, y *atlácatl*, inhumano. De *huehueti*, envejecer, y la *a* privativa, se forma *ahuéhuetyl*, el ahuehuete, árbol longevo "que nunca envejece". Duplicada, la *a* connota las ideas de perfección y abundancia: *aatzi* es alcanzar completamente una cosa; *aahuía*, llenarse de alegría; *aai*, hacer algo repetidas veces; *aamapoa*, leer mucho; *aami*, cazar en diversas partes. Como raíz de *atl*, agua, *a* se encuentra en innumerables vocablos, según se ve en etimologías de sustantivos y topónimos. *Acázul* es alberca; *ahuézotl*, sauz; *amózoc*, donde el río se ensancha; *analco*, en el otro lado del río; *anepantla*, en medio del agua; *atía*, licuarse algo; *acalpapano*, recrearse navegando.

**AATZIN.** (Del náhuatl *atl*, agua y *tzin*, reverencial: "venerable agua".) Caudillo de una de las tribus nahuas en su peregrinación desde Aztlan hasta el valle de México; se dice que fue el primero en llegar a Chapultepec. Está considerado como uno de los fundadores de Tenochtitlan; con Tenoch, llevó a cabo la unión con Tlatelolco. Son variedades del mismo vocablo *aátlatl*, *áhatl*, *áatl*, *ahuatzin*, *atzin*. Algunos autores suponen que no fue un personaje histórico, sino una personificación mitológica del agua.

**ABABÁBITE.** (*Poulsenia armata* Stand.; igual que *Coussapoa rekoi* Stand.) Árbol de la familia de las moráceas, de 25 m de altura, con jugo lechoso y provisto de pequeñas espinas en las ramas, peciolos, estípulas y nervaduras de las hojas. Éstas son grandes, ovales o elípticas, enteras, redondeadas en la base y con el ápice corto y acuminado; miden 50 cm de largo por 25 de ancho; y el peciolo, de 1.5 a 2.5 cm. Tiene flores unisexuales: las masculinas se dan en inflorescencias globosas, pedunculadas, de 1 a 1.5 cm; las femeninas, en receptáculos sésiles con tres a siete ejemplares. El fruto, carnoso y comestible, parecido a una chirimoya, mide de 2 a 3 cm de diámetro. La corteza tiene una fibra consistente, de aplicación industrial. Es común en las selvas muy húmedas de Guerrero, Oaxaca, Veracruz, Tabasco y Chiapas. Se le conoce también como *abababi*, *chirimoya*, *carnero*, *carne de pescado* y *masamorro*.

**ABACÁ.** (*Musa textilis* Need.) Planta filamentosa de la familia de las musáceas, de 7 a 8 m de altura. Las hojas, elípticas, de color verde oscuro y de terminales angulosas, miden 2 m de longitud. En el espádice, de posición inclinada, se contiene un

jugo lechoso y amarillento. Las brácteas, ovales, se enrollan hacia afuera y abrigan un promedio de 16 flores. Éstas, de estambres comprimidos y con cinco lóbulos, tienen los pétalos exteriores dos veces más largos que los interiores. El fruto es verde, duro, lento para madurar y no comestible. La planta fructifica al tercer año, pero es antes de la floración cuando debe extraerse la fibra denominada *cáñamo de Manila*; para ello, se remojan los tallos, se prensan, lavan y blanquean en agua acidulada con limón, ácido clorhídrico o una solución de soda. En 1958 fue traída de Filipinas, de donde es originaria. Se inició su explotación sistemática en Teapa, Tab., primero para utilizar las tierras que habían quedado ociosas al ser devastados los platanares por el "mal de Panamá", y luego con el fin de obtener la fibra que, desde 1964, se emplea en la producción de cables trenzados para calabrotes. El jugo que se extrae de la base de las hojas tiene propiedades curativas contra la tuberculosis, y el extracto de la raíz se usa para curar inflamaciones tiroideas.

**ABADEJO.** Nombre de varias aves pequeñas pertenecientes al orden Paseriformes, familia Muscicapidae, subfamilia Sylviinae, género *Regulus*.

(*Regulus satrapa*.) Ave que presenta plumaje verde olivo en las partes superiores y verde grisáceo en las inferiores. El macho tiene la corona color naranja con bordes amarillos y negros; la hembra, en cambio, la muestra amarilla con bordes negros. Ambos presentan una línea superciliar blanquecina y dos barras alares. Esta especie se encuentra en México en invierno, en las montañas altas. Se le conoce también como *reyezuelo de oro*.

2. (*Regulus calendula*.) Ave que tiene una coloración muy similar a la de la anterior, aunque la corona en el macho lleva un parche bermellón. Al igual que *R. satrapa*, se alimenta de insectos que recoge revoloteando por las ramas. Durante el invierno se le encuentra en todo el país. Recibe igualmente el nombre de *reyezuelo rojo*.

**ABADEJO.** Nombre de varios peces del orden Perciformes, familia Serranidae, género *Mycteroperca*.

(*Mycteroperca bonaci*.) Pez de cuerpo mediano, robusto y largo, cabeza grande, boca oblicua, opérculo con tres espinas planas, y aletas dorsales y anal cubiertas por escamas en la base. Es de color ocre claro, con hileras de manchas rectangulares más oscuras. Las mejillas y los lados ventrales presentan máculas hexagonales color bronce. El margen de las aletas verticales es negro y el de las pectorales, naranja. Habita sobre fondos rocosos y arrecifes coralinos. Los individuos pequeños

Figure 7: Presidential Inform I

VIII. Comercio, Fomento Industrial y Turismo  
Sector Turismo

CONCEPTO	UNIDAD DE MEDIDA	1978	1979	1980	1981	1982	1983	1984	1985	1986	1987 <sup>f</sup>	1988 <sup>e</sup>	OBSERVACIONES
1 Recursos Totales, Estructura por Sectores <sup>1</sup>	Mil M\$	1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.72	1 Incluye no sectorizados e intermediarios financieros.	
20.10 RECURSOS TOTALES, ESTRUCTURA POR SECTORES <sup>3</sup>	-	1 076.3	77 121.327	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.74	2 Esta es una nota.	
Disponibilidad <sup>a</sup>		1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.7	3 Incluye no sectorizados e intermediarios financieros.	
En sectores no financieros		1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.7	4 Esta es una nota.	
En sectores internacionales <sup>5</sup>		1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.7	W Esta nota aparece sin llamada en el texto (después se puede pegar la W en cualquier parte)	
Billetes		1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.7	5 Otra nota importante	
Cuenta de Cheques		1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.7	6 Incluye no sectorizados e intermediarios financieros.	
Liquidos		1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.7	7 Esta es una nota.	
A plazo menor de un año		1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.7	8 Incluye no sectorizados e intermediarios financieros.	
A plazo de un año		1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.7	9 Esta es una nota.	
1 Recursos Totales, Estructura por Sectores <sup>6</sup>	-	1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.7	10 Incluye no sectorizados e intermediarios financieros.	
Disponibilidad <sup>a</sup>		1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.7	11 Esta es una nota.	
En sectores no financieros		1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.7	12 Incluye no sectorizados e intermediarios financieros.	
En activos internacionales		1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.7	13 Esta es una nota.	
Billetes		1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.7	P Citras Preliminares	
Cuenta de Cheques		1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.7	E Citras Estimadas	
Liquidos		1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.7	n.d. No disponible	
A plazo menor de un año		1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.7		
A plazo de un año		1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.7		
1 Recursos Totales, Estructura por Sectores <sup>6</sup>	-	1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.711		
Disponibilidad <sup>a</sup>		1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.7		
En sectores no financieros		1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.7		
En activos internacionales		1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.7		
Billetes		1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.7		
Cuenta de Cheques		1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.7		
Liquidos		1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.7		
A plazo menor de un año		1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.7		
A plazo de un año		1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.7		
1 Recursos Totales, Estructura por Sectores <sup>12</sup>	-	1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.7	Fuente: Banco de México y Msc. La Lucha	
Disponibilidad <sup>a</sup>		1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.7		
En sectores no financieros		1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.7		
En activos internacionales		1 076.3	1 327.4	1 737.9	2 330.9	3 519.0	8 061.0	13 142.0	20 265.3	34 848.7	76 792.7		

1  
Secretaría de Hacienda Y Crédito Público  
Información Programática Presupuestal 1987

(dígitos presupuestales en millones de pesos)

Figure 8: Presidential Inform II

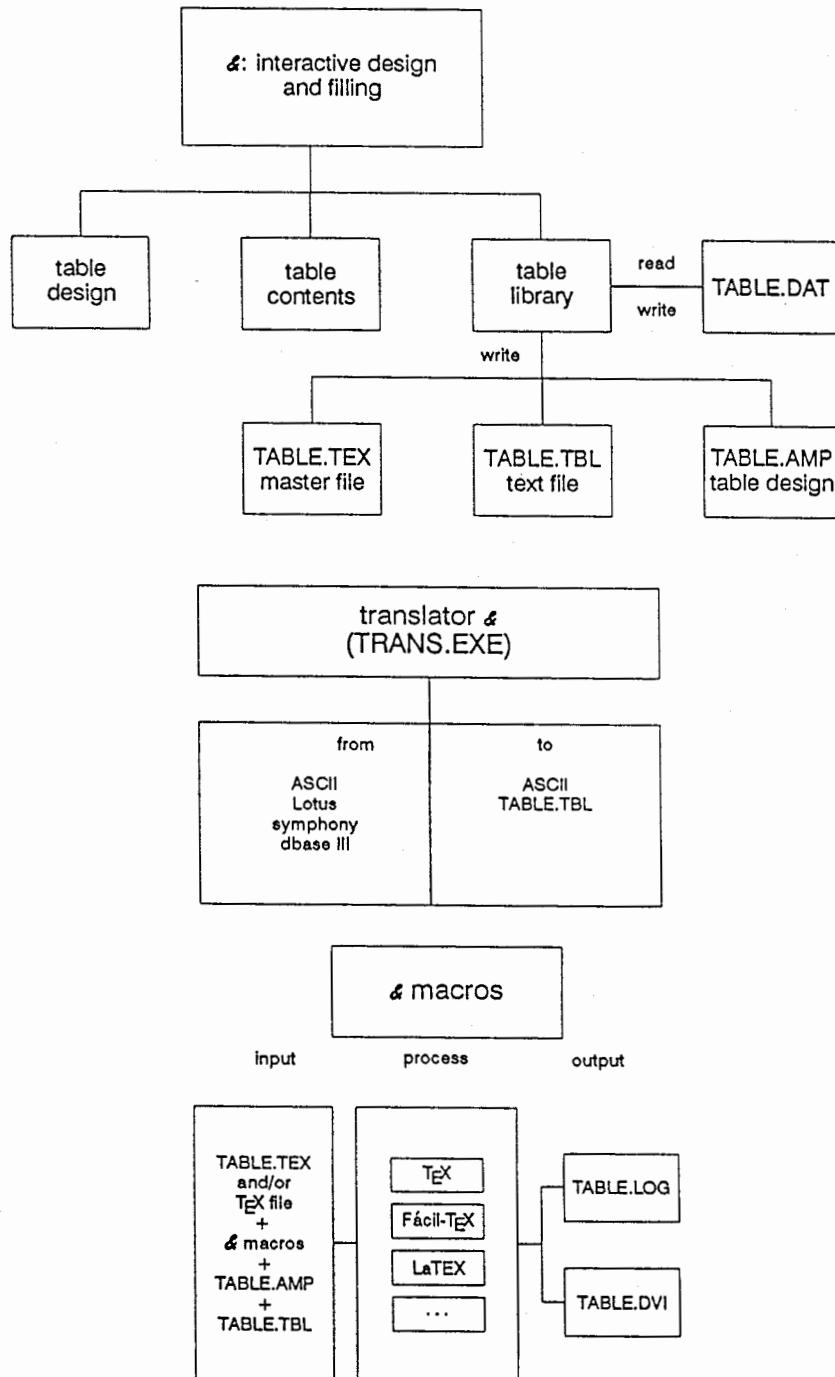
OBJETIVO	PROGRAMA SUBPROGRAMA	PRINCIPAL RESPONSABLE	PREBUPUESTO AUTORIZADO	PROGRAMA ANUAL			REALIZADO EN EL PERÍODO			OBSERVACIONES	
				GASTO CORRIENTE	GASTO DE INVERSIÓN	DESCRIPCIÓN	METAS ORIGINALES	UNIDAD DE MEDIDA	GASTO CORRIENTE		
Conocer y resolver controversias entre particulares y la Federación, fundamentalmente en asuntos fiscales.	1. IMPARTICIÓN DE JUSTICIA FISCAL (6)	Tribunal Fiscal de la Federación	2 937	2 937		Conocer y resolver los recursos interpuestos por el Estado a los particulares en contra de las Salas Regionales Visitar las Salas Regionales; preparar informe para su publicación de síntesis de tesis de independencia, y resolver demandas y notificaciones	Sentencia	2 100	6 719	\$3 6 772	1 El Presupuesto de Egresos de la Federación consigna para esta dependencia un presupuestal para cada uno de los subprogramas que ejecutan, por lo tanto en este caso no se aprecian los criterios de selección del principal responsable establecidos en la introducción general del presente anexo.
Contribuir al cumplimiento de las metas definidas en el Plan Nacional de Desarrollo y en el Programa Nacional de Financiamiento del Desarrollo, estableciendo los lineamientos y directrices de la política de precios y tarifas, así como determinar los ingresos del Sector Parastatal y efectuar la política de precios y tarifas a efecto de lograr el cumplimiento del presupuesto de recursos propios, principio al establecimiento de una estructura adecuada de precios relativos, evitar el desgobierno de precios y proteger el consumo básico.	2. FORMULACIÓN Y COORDINACIÓN DE LA POLÍTICA DE DETERMINACIÓN EN MATERIA DE PRECIOS Y TARIFAS DE LA ADMINISTRACIÓN PÚBLICA (E)	Unidad Técnica de Precios y Tarifas de la Administración Pública	541	541		Formular estudios sobre las políticas de costo y mediar plazo, relativos a la modificación de precios y tarifas a Programa Estudio por la Administración Pública Federal e investigar los efectos económicos por sectores y empresas que provocan las posibles modificaciones	"	6	796	796	7
Dirigir y coordinar la formulación, instrumentación, ejecución, seguimiento y evaluación del Programa Nacional de Financiamiento del Desarrollo y su operativo anual correspondiente; así como fortalecer y consolidar el ámbito regional dictando políticas congruentes a la descentralización y/o descentratización.	3. DIRECCIÓN Y COORDINACIÓN DE LA POLÍTICA DE FINANCIAMIENTO NACIONAL DEL DESARROLLO (E)	Oficina del C. Secretario de Hacienda y Crédito Público	9 780	9 780		Subprograma: 3.1 CONDUCCIÓN DE LA POLÍTICA	Subprograma: 3.2 CONTRALORÍA INTERNA	12	937	937	12
3.3 COMUNICACIÓN SOCIAL	Dirección General de Comunicación	Contraloría Interna	1 527			3.2 CONTRALORIA INTERNA	1 527	1	2 407	10	2 417
			2 120			3.3 COMUNICACIÓN SOCIAL	2 120	2 120	5 215	5 215	1
									105	105	28

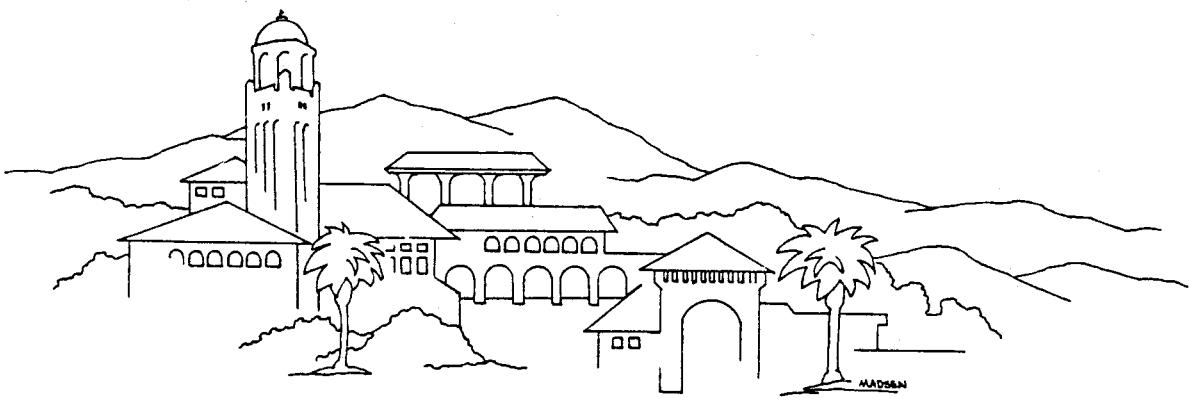
PP-88

Figure 9: Ampersand Utility

# ampersand

Utility for making tables in  $\text{\TeX}$





**T<sub>E</sub>X Users Group Meeting and Short Course  
Stanford University, July 25–30, 1982**

# TEX for 30,000

JAMES HASKELL, WALLY DESCHENE, AND ALAN STOLLEIS

James Haskell  
Director of Information Systems  
USDA—Forest Service  
Intermountain Region  
324 25th Street  
Ogden, Utah 84405

Wally Deschene  
Computer Specialist  
USDA—Forest Service  
Intermountain Fire Sciences laboratory  
P.O. Box 8089  
Missoula, MT 59807

Alan Stolleis  
Research Assistant  
Department of Computer Science  
Texas A&M University  
College Station, TX 77843-3112  
[alan@cssun.tamu.edu](mailto:alan@cssun.tamu.edu)

## ABSTRACT

Quality documents are an integral part of the Forest Service mission. These documents include reports to the public, the Congress and other oversight agencies. The integration of *TEX* into our existing word processing system (Data General's Comprehensive Electronic Office — CEO) meets our goal of providing a consistent user interface and does not require all employees to learn a new system.

### 1. Forest Service Organization and Mission

The Forest Service, which is the largest agency in the U.S. Department of Agriculture, is charged with the responsibility of managing 191 million acres of National Forest System lands. Included on these lands are more than 128,000 miles of streams and two million acres of lakes and reservoirs. Each year, National Forests host more than 40 percent of all outdoor recreation in the country. Forest Service programs include areas such as fire prevention, fire suppression, range, water, air, timber, mineral leasing, oil and gas leasing, wildlife, recreation, engineering, forest pest management, and an extensive research program.

To carry out these programs, the Forest Service employs nearly 30,000 people in 9 Regional Offices, 123 National Forest Supervisor Offices, 653 Ranger Districts, approximately 50 research locations along with the National Office in Washington, D.C. The entire workforce is distributed over 45 states.

### 2. Forest Service Technology Environment

Beginning in 1983, the Forest Service embarked upon the installation of a comprehensive nation-wide distributed processing system. The objectives of this system were to meet increasing demands placed on the agency for timely and accurate information, and to provide a consistent office automation environment throughout the Forest Service.

The computer hardware acquired includes more than 900 Data General MV/family super minicomputers. These systems communicate with each other and with mainframe systems at USDA National

Computer Centers at New Orleans and Fort Collins over an X.25 public packet switched network. Included in the network are over 200 remote satellite locations, data channels on Forest Service-owned microwave channels and other local area networks.

The office automation software being used is Data General's Comprehensive Electronic Office (CEO). The systems are used for electronic mail, document preparation, electronic filing, spreadsheets, database creation and maintenance, graphics, most business transactions, and a variety of other uses that even include business management activities at forest fire camps. The CEO word processing system is the backbone and is a typical word processing system, including features such as format rulers, centering, bold, underline, tabs, indents, footnotes, super/subscripts, and more.

Today, nearly every one of the 30,000 employees has access to a terminal somewhere in the system to conduct their every day duties. Employees are able to travel or relocate to any other Forest Service location and never need to learn a new system. The system enables the Forest Service to be more productive and share information faster, all at reduced costs to the public. Independent consulting firms have estimated a benefit/cost ratio of five to one.

### **3. Typesetting Needs**

Quality documents are an integral part of the Forest Service mission. These documents include reports to the public, the Congress and other oversight agencies. The Forest Service needs to produce the highest quality printed documents. The research community has needed a system that is available to everyone that can produce manuscripts with Greek letters and mathematical equations. Our desire is to use our existing system and not implement another technology strictly to fulfill this need. *T<sub>E</sub>X* on the Data General provides this capability. Integration into the CEO system has greatly simplified the use of *T<sub>E</sub>X*. Forest Service employees can use *T<sub>E</sub>X* to produce high quality printed documents knowing little or nothing about *T<sub>E</sub>X* itself. CEO commands are translated into *T<sub>E</sub>X* while *T<sub>E</sub>X* commands are simply passed through as the document is exported to *T<sub>E</sub>X*. The conversion and export processes are transparent to the user.

The integration of *T<sub>E</sub>X* into our existing word processing system meets our goal of providing a consistent user interface and does not require all employees to learn a new system.

### **4. *T<sub>E</sub>X* Interface**

*T<sub>E</sub>X* is used by inserting commands into a document. These commands instruct the *T<sub>E</sub>X* system how to produce the desired printed document. The results of the commands are not seen until the document is printed. The CEO word processing system, on the other hand, is almost a WYSIWYG (What You See Is What You Get) system: it performs functions by displaying the text in the selected format at the time the function is entered. For example, a line centered in CEO is displayed with the center symbol in column one and the correct number of spaces to make the text centered. In short, these systems use two entirely different methods for processing textual information.

*CEO\_to\_TEX* is a program developed to convert a CEO document into a file to be processed by *T<sub>E</sub>X*. This program will convert many of the CEO functions (e.g., center, text attributes, indents, etc.) as closely as possible to equivalent or comparable *T<sub>E</sub>X* commands. It is important to note that the *T<sub>E</sub>X* system, unlike the CEO system, uses proportional spacing to produce high-quality documents. The conversion program gives the user as much control as possible via CEO functions without degrading the quality of the *T<sub>E</sub>X* output. By doing this, the user can produce high-quality typeset documents from CEO without having to know any of the *T<sub>E</sub>X* commands. It can also be used by knowledgeable *T<sub>E</sub>X* users to convert existing CEO documents into *T<sub>E</sub>X* format without having to re-type the entire document.

For some users, the conversion program may be all that is needed to produce a final product. For others, it may be just the first step in the process.

#### **4.1 Features of the *CEO\_to\_TEX* Conversion Program**

For many of the CEO features, the conversion to *T<sub>E</sub>X* is a straightforward process. However, there are certain conversions that require special consideration. The following sections describe these special considerations.

### *Special T<sub>E</sub>X symbols*

The T<sub>E</sub>X system uses the following infrequently used typewriter symbols in its command syntax:

\ { } \$ & # ^ - % ~

The \, { and } symbols are the most frequently used of these for basic T<sub>E</sub>X commands and infrequently used in most word processing documents. Thus, we made the decision to sacrifice these symbols so that the CEO user could more easily embed T<sub>E</sub>X commands in a document. The remaining symbols are converted to the equivalent T<sub>E</sub>X symbol. The system was designed with the flexibility to allow the user to convert or not convert any of these symbols. By default, the \, { and } are not converted; all others are converted. The fact that the \, { and } are not converted allows the CEO user to embed T<sub>E</sub>X commands directly into a document. For example, the following will result in the text between the brackets printing in a monospaced typewriter text font:

The use of {\tt monospaced fonts} can be quite useful for software documentation. This example illustrates the fact that the three symbols are passed through to T<sub>E</sub>X with no conversion to represent the appropriate T<sub>E</sub>X commands. Any of the special symbols can be converted or not converted independently of one other. To do this, the user simply includes symbol/ON or symbol/OFF in a CEO user note prior to using the symbol. For example, the following sequence illustrates how to use T<sub>E</sub>X math mode in a CEO document:

```
!$/OFF!  
$$ Y = \alpha + \beta X $$  
!$/ON!
```

The exclamation points (!) represent a CEO user note. The user can turn any symbol conversion off and on anywhere in the document. If a user wants to use the \, { or } symbols in a document, the symbol/ON user note can be used to allow the symbol to be converted and not represent a T<sub>E</sub>X command.

### *CEO tabs and indents*

Tabs should only be used to produce tables, and as a paragraph indent. When tab stops are used, the most recent format ruler (current) must specify the tab stops required and only those stops. They must be used correctly in conjunction with the format ruler. Every tab stop in a line must be used when typing text even if the column is blank. A common error made when inserting tabs or indents is to type over existing tabs in one line and then tab past them in another line. This is a definite mistake when using the conversion program because it uses the current format ruler to construct T<sub>E</sub>X-compatible tab stops.

Tabs and indents were designed for specific functions and should not be used interchangeably. Tabs are used to position information in columns of a table and to start the beginning of a paragraph. Indents should never be used in a table. Indents are used to offset information from the left margin. All information following an indent will be offset to the current tab stop until the end-of-line character is encountered. Tabs and indents should never occur in the same line and/or paragraph of text.

### *Blank lines in tables*

Whenever the conversion routine encounters a tab character in any column other than the paragraph indent column (column 5), it switches to table mode. If a table is to contain blank lines, they should be entered with all tab stops specified. If the blank lines between lines of text in a table do not have tab stops in them, T<sub>E</sub>X will switch in and out of table mode for every blank line of the table. It is more efficient to specify null tabs in blank lines and maintain table mode than to switch T<sub>E</sub>X in and out of table mode.

### *Spaces in tables*

The T<sub>E</sub>X system normally compresses all spaces to one space and removes leading spaces in a table. To allow some flexibility, the conversion routine converts all spaces in tables into hard spaces. Thus, unlike T<sub>E</sub>X, CEO spaces can be used to align information within columns of a table.

#### *CEO hard space*

The CEO **hard space** feature can be used to make sure words with spaces in them always print on the same line. For example, the words "Dr. Childs" should always occur on the same line of print. To ensure that this always happens, insert the CEO hard space character between the two words. The CEO hard space will be translated to **T<sub>E</sub>X** and will perform the same function as in CEO.

#### *CEO block protect*

In CEO a block of text can be marked with the **block protect** feature. If the block of marked text cannot be printed on the current page, it will be started on the beginning of a new page. This feature is useful to ensure that a complete table or other block of text is printed on the same page (provided the entire table fits on one printed page). **block protect** can also be used to eliminate orphaned headings (paragraph or section heading located at the bottom of a printed page) by marking the heading and one or more lines of the section body. You should be careful not to mark large blocks of text with **block protect** because it may cause unnecessary white space to occur on the previous page.

#### *CEO end page*

CEO has two end-of-page types; soft and hard. A soft end-of-page is represented by a dotted line across the screen and automatically occurs every time the number of lines specified by the current format ruler is encountered. A hard end-of-page is represented by a solid line across the screen and is manually inserted by the **end page** function key. Soft end-of-pages are not converted to **T<sub>E</sub>X**, as **T<sub>E</sub>X** will automatically page text based on the current page size specification. Hard end-of-pages will be converted to **T<sub>E</sub>X** page ejects.

#### *CEO include*

The CEO **include** function is used to attach another document to the one currently being processed. The document to be included does not get inserted into the current document at the time the **include** function is invoked. Instead, the path name of the document is displayed inside the **include** symbols. At print time, the contents of the **include** document are inserted into the current document at the point where the **include** occurs. This feature is processed in the same manner by the **T<sub>E</sub>X** conversion routine.

**include** documents can contain other **include** documents; this is called nested **includes**. For example, a paper from the proceedings in the above example might contain five chapters and each chapter might be a separate document specified by an **include**. So the main document contains **include** documents of which one or more contains **include** documents. The conversion routine was tested to 200 levels of nested **includes** before it was terminated.

#### *CEO footnotes*

The CEO system provides an option for specifying footnotes in a document. The conversion routine will extract the footnote information from CEO and convert it to the **T<sub>E</sub>X** system.

#### *CEO headers and footers*

The CEO page headers and footers are used to put additional information, such as titles and page numbers, at the top and/or bottom of a printed page. CEO allows you to put up to six lines of text in both the headers and footers; however, it does not allow the use of text attributes, such as bold, underline, etc., but does allow text to be tabbed and centered. Also, the user can specify any valid **T<sub>E</sub>X** command in a CEO header and footer, which will be transferred directly to the **T<sub>E</sub>X** headline and footline.

#### *CEO bold and underline*

CEO provides a feature for bold, underline, bold underline, double underline and bold double underline. Underlining, however, is generally not used in typesetting because it does not look good:

For example, underlining dosen't always look very good.

The underline either smashes descenders or is spaced too far from words without descenders to compensate for words with descenders. The convention we have used is to convert underlined text to

italicized text and double underlined text to slanted text. The conversion program uses this convention for underlined and double underlined text. In addition, bold underlined text is converted to bold italicized text and bold double underlined text is converted to bold slanted text. Bold text in CEO is converted to bold text in  $\text{\TeX}$ .

#### *CEO center*

The CEO center function is converted to a  $\text{\TeX} \backslash\text{centerline}$ , which produces the same results.

#### *CEO quote marks*

In CEO, the double quotation is used for both the left and right double quote marks, whereas in typesetting, there is a difference between the left and right quote marks: the left double quote marks look like two backwards (and inverted) apostrophes ("") and right double quote marks look like two apostrophes (''). The conversion routine will keep track of CEO double quotes and convert them into the appropriate left or right quotes. The conversion routine assumes a double quotation immediately following a digit is an indicator for inches and does not convert it to either a left or right quote. If in fact, the double quote following a digit should be either left or right quote marks, it can be specified by turning  $\text{\TeX}$  mode on and entering either '' for right double quotes or '' for left double quotes and then turning  $\text{\TeX}$  mode off again.

#### *Dashes and hyphens*

In typesetting, there is a difference between a hyphen and a dash. In fact there are three different symbols. One is a hyphen, which is used for compound words like X-ray. The second is an en-dash, used for number ranges such as "lines 3-20". The third is an em-dash (normal dash), which is used for punctuation in sentences — like this. The difference is in the length of the line representing the symbol. The conversion program will pass the hyphen characters directly to  $\text{\TeX}$  and  $\text{\TeX}$  will treat one typed hyphen as a hyphen, two hyphens as an en-dash and three hyphens as an em-dash.

#### *CEO mandatory end-of-line*

In CEO, the mandatory end-of-line is produced by either the NEWLINE or CR key. The CEO mandatory end-of-line is used to end a sentence, end a line in a table, end a paragraph, exit indent mode or produce vertical white space in the printed document. The conversion for a mandatory end-of-line will depend on where it is encountered.

- If the mandatory end-of-line is encountered at the end of paragraph, the conversion routine will instruct  $\text{\TeX}$  to terminate the current paragraph, produce a specified amount of white space and begin a new paragraph. The new paragraph will be indented if the next line contains a tab skip to column five; otherwise, the paragraph will not be indented.
- If a mandatory end-of-line is encountered while in table mode, the current row of the table will be terminated.
- If a mandatory end-of-line is encountered while in indent mode, the conversion routine will instruct  $\text{\TeX}$  to exit indent mode.
- When two mandatory end-of-lines are encountered in sequence, the conversion routine will instruct  $\text{\TeX}$  to produce white space equivalent in vertical dimension to a line of text.

#### *Superscripts and subscripts*

CEO superscripts and subscripts will be converted to  $\text{\TeX}$  equivalents.

#### *Overstrike mode*

CEO supports an overstrike mode that allow characters to be printed on top of other characters. The conversion routine attempts to convert this feature to  $\text{\TeX}$ ; however, it is almost impossible to do because  $\text{\TeX}$  uses proportional spacing. This feature should not be used.  $\text{\TeX}$  should be able to produce any character needed.

#### *CEO discretionary hyphen*

CEO supports discretionary hyphens, which tell the software where a potential line break in a word

can be made. The conversion routine translates CEO discretionary hyphens to  $\text{\TeX}$  discretionary hyphens, which perform the same function as in CEO. If a line break does not occur in the word, the discretionary hyphen is ignored.

#### 4.2 Additional Conversion Features

The conversion program contains additional features which allow CEO users greater flexibility in producing typeset documents.

##### *Special symbols*

The following table contains a list of special symbols that are not part of the CEO character set but can be produced in  $\text{\TeX}$  documents. These symbols are specified with the CEO superscript text attribute. For example, the bullet symbol can be produced with a superscripted period, i.e.,  $\text{.^{.^{}}}$ . Note that the  $\text{^}$  symbols enclosing the period (.) are produced by the superscript text attribute and not by the keyboard character.

Superscript text	Description	CEO Examples
$\text{.^{.^{}}}$	Bullet	• This line starts with a bullet.
$\text{^R^}$	Registration symbol	This is a CEO® example.
$\text{^C^}$	Copyright	Copyright © 1988 by Data General Corp.
$\text{^T^}$	Trademark	The PRESENT <sup>TM</sup> utility is a versatile tool.
$\text{^o^}$	Degree Symbol	The temperature was -25°F today. (This is a lowercase o, not a zero)
$\text{^1/2^}$	Fraction	It rained 1 $\frac{1}{2}$ inches today.

Fractions can be produced using the superscript text attribute and any digits separated by a slash, for example,  $\text{1/3}$ ,  $\text{5/16}$ ,  $\text{19/64}$ ,  $\text{3/323}$ , etc.

##### *Special spacing features*

When a period is encountered by  $\text{\TeX}$ , it is assumed to be the end of a sentence and a space representing end-of-sentence spacing is put after the period. However, if the period is not the end of a sentence, as in "Dr. Childs", then the amount of space inserted is out of proportion with the rest of the spacing in the sentence. The conversion program scans the CEO text for the character strings Dr., Drs., Ph.D., Mr., Mrs., and Ms., and inserts a single space instead of end-of-sentence spacing.

### 5. ASCII Previewer

We have also provided an ASCII previewer to get a crude representation of what the typeset document will look like on our existing ASCII terminals. This screen representation allows the user to see page breaks, line endings and orphaned text. The previewer software was designed following the CEO user interface with consistent screens and function key mappings. Figures 1 and 2 demonstrate the user interface developed for implementation of  $\text{\TeX}$  in the Forest Service.

### 6. Data Tables

CEO decision support tools include a data table product that allows a user to enter data in a table format. In keeping with our desire to use existing products, a conversion of CEO Data Tables to  $\text{\TeX}$ -formatted tables is provided. This conversion takes data from the data table and builds the appropriate  $\text{\TeX}$  commands to produce a table. The user can select options to include table captions (above or below), column headings, and horizontal or vertical rules.

Msg: New: 0	Mar 28,89 5:26 PM	username:hostname
	FOREST SERVICE TEX	
----> 1. Typeset      (Typeset and print CEO document) 2. Preview     (Display typeset document) 3. View        (Display CEO document) 4. Edit        (Revise CEO document) 5. File        (Create AOS/VS file)		
Enter Choice:1		
To return to the previous menu, press the CANCEL/EXIT key. For assistance here (or on any other menu or question), press the HELP key.		

Figure 1: TeX environment main menu

Msg: New: 0	Mar 28,89 5:26 PM	username:hostname
	FOREST SERVICE TEX	
Document: TeX test	in	
Drawer: TeX drawer	Folder: TeX folder	
Pick one: (1. TeX, 2. LaTeX, 3. SliTeX, 4. BibTeX) 1		
Convert to TeX format (Y/N)? Y		
Printer name:		
How many copies? 1		
Run in background (Y/N)? Y		
Execute (Y/N)? Y		

Figure 2: TeX environment typeset menu

## 7. Using Graphics

Graphics produced by either CEO Drawing Board, Charting Tool, Present or application programs using the Graphics Kernel System (GKS), in either GKM or CGM format, can be inserted into a TeX document. The procedure for including a graphic consists of two parts; defining the graphic to TeX and preparing the graphic for inclusion.

## 8. Documentation

Documentation describing the Forest Service implementation of TeX is available from the authors. It includes a description of TeX, the Forest Service environment, CEO to TeX conversion features, previewing text on ASCII terminals, building tables with CEO Data Table, using graphics in a TeX document, and a description of other TeX products. Also included are appendices that provide the user with a glossary of technical terms, common TeX terms, a TeX font chart, Forest Service TeX macro library, a detailed description of using TeX in the Forest Service environment, TeX defaults for CEO conversion, TeX software release notice, and TeX software abstract for the Software Reference Center.

## **9. Summary**

The TeX and CEO integration software was just released within the Forest Service this past May. Fewer than 50 sites have the software installed, and two national training classes have been held, with more to follow soon. We also expect to release the CEO Data Table and the Graphics Inclusion software by the end of October.

The integration of a sophisticated typesetting system such as TeX into an existing word processing system produced an extremely powerful document processing tool for the Forest Service. Any employee having access to a Data General system anywhere in the organization can now produce the highest quality typeset documents.

Training consisted of 3 courses similar to the elementary TeX course offered by TUG. Others will be offered as our user community grows. Distribution will be in the form of magnetic tape, and also electronically through the Forest Service network. Distribution will include the TeX executables, the CEO\_to\_TEX conversion program, several Forest Service-specific macros, and the Computer Modern and other fonts. Additional macro packages will be distributed as the need is identified.

Integration of these two systems has allowed the Forest Service to maintain the integrity of its information processing standards developed over many years. This is viewed as a major accomplishment in an organization the size and complexity of the USDA—Forest Service. We hope to make a future report on the use of TeX in the Forest Service.

## **10. Acknowledgements**

The authors would like to express their sincere appreciation to Dr. Bart Childs, Texas A&M University, for making TeX available on the Data General systems and for his continued support throughout this project; and to Dr. Donald Knuth, Stanford University, for developing TeX and putting it in the public domain so projects like this could become a reality.

# **T<sub>E</sub>X Enslaved**

ALAN E. WITTBECKER

Digital Equipment Corporation  
110 Spit Brook Road (ZKO 1-2/C14)  
Nashua, NH 03062-2698

## **ABSTRACT**

Several documentation systems incorporate T<sub>E</sub>X as a text formatter, but use a generic markup language as a front-end. This means T<sub>E</sub>X is hidden from the users, and usually has been modified or extended. This paper discusses the advantages and disadvantages of such systems, as regards international standards, macro packages, and T<sub>E</sub>X itself.

### **1. Introduction**

T<sub>E</sub>X, the mathematical typesetting program, has a number of good qualities and unique features. It is extremely precise in its measurements, generous in its use of fonts, sophisticated in its hyphenation, ultra-sophisticated in its display of mathematical characters, and generally device-independent in its output. And, it is now fixed and completed; it is the same for everyone.

But, T<sub>E</sub>X has limitations. T<sub>E</sub>X processes text in a straight line. It cannot rotate characters or text, and it has difficulty drawing "freehand" lines. Although it can insert graphics at arbitrary locations, it cannot produce them.

T<sub>E</sub>X is a general-purpose composition program. It has a relatively large number of commands that make it hard to learn. Many of its concepts, such as zero-width boxes, are not intuitive. T<sub>E</sub>X is often inconsistent in the way it presents its primitives and arguments.

Programmatically, most of its formatting macros are low-end, aimed at producing basic structures, such as paragraphs and boxes. For higher-level objects, custom macros are needed.

### **2. A Front-End**

Many, if not most, of the perceived shortcomings of T<sub>E</sub>X could be solved by placing T<sub>E</sub>X behind a front-end system. The front-end would have its own simple, consistent, standardized commands for formatting documents. These commands would then be translated into T<sub>E</sub>X.

#### **2.1 Advantages**

Numerous advantages would accrue. Specifically:

1. The front-end could provide consistency and conformance with external standards, such as the Standard Generalized Markup Language (SGML).<sup>1</sup>

The syntax of an SGML-encoded document is independent of processing. With SGML, a document is ordered as a hierarchy of structural elements. The standard provides conventions for beginning and ending structural elements; every element is delimited by a pair of start and end tags. The standard also provides ways of defining document types and limiting the structural elements permitted within each.

T<sub>E</sub>X has been compared unfavorably with SGML standards; it has been labelled as a procedural, but not descriptive, text processor. Yet, some proposed SGML markup, 'lq' for instance (left

---

<sup>1</sup>As defined in International Standard (ISO) 8879 (10/86). "Markup" is used here to mean any non-text added to increase understanding of the text.

quad? long quote? loose quality?), is as cryptic as anything in `TEX`.<sup>2</sup> It is possible for SGML or `TEX` (in the form of macros) to be completely descriptive, depending on the interpretation.

2. The front-end could permit an object-oriented approach to documentation, which would seem more natural for users.
  - (a) A front-end language would be considered object-oriented if it supported four specific object properties (Cox 1987):
    - i. abstraction — a concise representation for a more complex idea, where the details are not essential to understanding the function. Each object is an instance of a class, its behavior limited by the properties of that class.
    - ii. encapsulation — an object is the unit of encapsulation of an abstraction — is a process by which an individual component is defined. Encapsulation ensures that an individual object can have private properties, not shared with other objects in the class.
    - iii. inheritance — classes that are sub-classes of others can inherit the properties of the super-classes. For example, in a document, a paragraph within a table would inherit the left margin of the table.
    - iv. polymorphism — inherited messages can be re-defined by sub-class; the same message in different classes could be polymorphically defined.
  - (b) The front-end could provide a set of object-oriented constructs which would then be used to form high-level, hierarchical document structures.
3. The front-end could, in fact, provide extensions, or perhaps “pretensions”, to `TEX`. This would simplify the operation of the macros. For instance, when an argument in `TEX` is used for several purposes, such as a chapter title, running head, contents entry, and index entry, each case is handled differently; the text is a box, mark, or write, depending on the macro. Expansion occurs at different times. The front-end could copy the text, so it does not need to be done in the macros.
4. The front-end could be simpler to use and easier to learn. Complex parts of the code would be located in lower levels, in `TEX`, so that higher levels, the front-end, would be less complex. The markup would be descriptive and consistent.

It could offer easier error correction. An SGML parser could check the context of elements (for example, emphasis, double columns, or indents) and issue error messages. Otherwise, this would have to be built into the `TEX` macros. Thus, `TEX` errors would be minimized. The front-end would detect errors and relate them to the elements used by the author, not to the arcane operations in `TEX`'s gut.
5. The front-end could be extraordinarily flexible. It could provide another level of optimization. It could, in fact, use other text processors under certain circumstances. It could offer multiple styles of document preparation; the same dataset could be available as a manual, print-file, or on-line book. The layout would be determined by a style definition.

## 2.2 Disadvantages

Putting a front-end on `TEX` does have disadvantages, however:

1. A comprehensive front-end may be almost as difficult as `TEX` to learn, especially if it has to serve a variety of applications.
2. The interface between the programs becomes complex.
3. Maintenance and debugging also become more complex and lengthy.

---

<sup>2</sup>See Coombs *et al* 1987.

4. Running time may be increased.
5. The front-end may not be as portable as  $\text{\TeX}$ .

### 3. An Example in Progress: Digital's VAX DOCUMENT

Front-ends have been advertised for a number of years. The Tyxset system (1985, Tyx Corporation, Reston, VA) was designed to hide  $\text{\TeX}$  from users with no typesetting experience, while letting them typeset. Page One (1987, McCutcheon Graphics, Toronto, Canada) offers templates for automatic book production on the Macintosh, where the skill requirement was listed as the ability to use a mouse. MARKUP (1987, Hewlett-Packard, Palo Alto, CA) is an SGML parser and application generator that uses  $\text{\TeX}$  to print internal documentation (Price 1987).

Digital Equipment Corporation's VAX DOCUMENT provides a front-end for a formatting engine that is currently "based on  $\text{\TeX}$ ". The engine has been modified and does not pass the trip test. Changes include making  $\text{\TeX}$  into a callable function, instead of a standalone program, providing 8-bit support (to permit fonts with 256 characters, instead of 128), raising the memory limit, replacing error messages with more normal VMS messages, and increasing the number of marks, dimens, and fonts.

This means that a VAX DOCUMENT-produced file cannot run through public domain  $\text{\TeX}$ , or similarly, an arbitrary  $\text{\TeX}$  file won't run through VAX DOCUMENT's text formatter. In fact, end users are not allowed to load their own format files. VAX DOCUMENT consists of the following components:

1. A proprietary generic markup language, based on preliminary international standards.
2. A tag translator, which reads the source file and produces an intermediate file (with  $\text{\TeX}$  syntax) that can be read by text formatting software.
3. The text formatter, which uses the input file from the translator and font definition files.
4. Device converters that produce output for different devices: monospaced, LN03 printers, Post-Script printers.

The text formatter controls page layout, typography, and output dependencies. Text formatting macros are coded using a proprietary  $\text{\TeX}$  macro package, dependent on  $\text{\TeX}$  extensions (and inclusive of future bug fixes). Most of the macros (and language dependencies, font loads, device-dependent characters) are considered internal to the product and are shipped to customers in machine-readable form only.

Invoking VAX DOCUMENT begins a chain of processing. One command calls the components of the system and their associated files. Except for the top level *verb* component, which parses the command line and some files and creates an item list to facilitate communication, all the components are data processors that manipulate an input file to produce a readable file for the next processor. The tag translator reads an ASCII input file with embedded generic tags and replaces them with definitions from a saved tag table file, which supplies a mapping from the tags to strings of  $\text{\TeX}$  macros. The tag translator output becomes input to the text formatter, which looks up the macros from a format file; the macros are parameterized by a design file.<sup>3</sup> At the end of the chain, a formatted file is displayed or printed.

The major components of VAX DOCUMENT are the tag translator and text formatter. Both components are macro processors, i.e., text substitution programs. Each processor defines a set of primitive macros that do "work". The primitive macros include a kind of programming language that allows an algorithm to be written. The tag translator macros are called *tags*, to conform with generic tagging and to avoid confusion with  $\text{\TeX}$  macros.

VAX DOCUMENT is still evolving. Currently, it supports government standards (DOD 2167, 2167a). Other features are being studied: automatic language strings; support for languages that read from right-to-left and left-to-right, as well as the mixture of the two; a modular form for the text formatter macros; an on-line bookreader.

---

<sup>3</sup>Users are not told how to modify the saved tag table file and format files, only design files.

#### 4. Summary

T<sub>E</sub>X is a sophisticated system for setting type. But T<sub>E</sub>X has a high learning curve, and T<sub>E</sub>X does not meet international standards.

T<sub>E</sub>X is a public domain program with all the attendant problems of public domain programs, such as maintenance and improvement — who will support and disburse T<sub>E</sub>X and for how long? By extending T<sub>E</sub>X, making it proprietary, companies such as Digital guarantee interest and investment in T<sub>E</sub>X. By using a front-end, they can complement the strengths of T<sub>E</sub>X with those of the front-end. Also, they are flexible in their choice of a formatter.

An object-oriented front-end that conformed to SGML standards could handle the logical structures of a variety of documents. That front-end could then harness T<sub>E</sub>X's impressive text-formatting capabilities.

#### Bibliography

- Coombs, J.H., A.H. Renear, and J.S. DeRose. "Markup Systems and the Future of Scholarly Text Processing." *Communications of the ACM* 30:933-947, 1987.
- Cox, Brad. *Object Oriented Programming: An Evolutionary Approach*. Reading, MA: Addison Wesley, 1987.
- Price, Lynne. "SGML and T<sub>E</sub>X." *TUGboat* 8(2):221-225.

# Methodologies for Preparing and Integrating PostScript Graphics

J.T. RENFROW

Jet Propulsion Laboratory  
4800 Oak Grove Drive  
Pasadena, CA 91109  
`TRENFROW@jpl-pds.jpl.nasa.edu`

## ABSTRACT

Graphical material represented in the PostScript language can be incorporated into TeX-based documents and ultimately printed on a printer having a PostScript interpreter. Successfully completing this incorporation process requires an understanding of PostScript and its use in programs to generate graphics and in programs associated with document preparation. It also requires an understanding of the particular DVI-to-PostScript translator program being used. PostScript concepts such as a stack of dictionaries, commands such as "translate", "rotate", "scale", and the idea of saving the state of virtual memory must be mastered. Different graphics generation programs use somewhat different mechanisms to produce the PostScript description of a graphic object. The graphics programs Adobe Illustrator, Cricket Draw, and MacDraft illustrate the diversity of possible PostScript output files. Text manipulation programs, such as AWK, can be used to examine these PostScript output files, establish the correct origin for these files when incorporated into documents, and prepare them for inclusion in TeX-based documents. A number of special cases need to be considered depending on the use of landscape or portrait modes in both the TeX-based document and the graphic object.

### 1. Introduction

Incorporating PostScript code into a document being processed by TeX is challenging initially but once the techniques and approach are worked out you will be able to incorporate PostScript code produced by graphics programs very rapidly and easily. To set up the process initially may require someone who is willing to "hack" or experiment a little. After this, anyone who can use TeX should be able to incorporate this type of PostScript code into a document with relative ease.

In this paper I will illustrate how to incorporate PostScript code representing graphical figures — lines, circles, shaded figures, text, etc. — into documents. I will not illustrate how to incorporate images but the techniques used here should apply to that graphical form also. I will illustrate the techniques using three common software packages: MacDraft, Adobe Illustrator, and Cricket Draw. I have chosen these three packages because they require somewhat different approaches as one incorporates their graphical material into the TeXed document.<sup>1</sup>

It seems that the success of incorporating PostScript graphics material into TeX documents can be very dependent on the hardware/software configuration. For that reason I am carefully documenting the configuration I use:

- a. Hardware — IBM PC/AT, Macintosh Plus, QMS 810 Laser Printer
- b. Software — DOS 3.3, MacLink Plus, Mac OS 4.2, Laser Prep 5.0, Cricket Draw 1.1, MacDraft 1.2a, Adobe Illustrator 1.1, MicroTeX Version 1.5A1, DVIPS 4.0.4, AWK (MS-DOS MKS Toolkit Version 2.2c from Mortice Kern Systems, Inc.)

If you are using other software (for example, a different DVI-to-PostScript processor), you will probably have to do things differently. Hopefully you will have enough courage or insight after reading this paper

---

<sup>1</sup> The phrase "TeXed document" means a document that was produced using the TeX typesetting system.

that you will try to incorporate graphics using your configuration.

I employ a uniform approach to the generation of PostScript and *T<sub>E</sub>X* material. The graphical material is generated on a Mac Plus using one of the three packages previously identified. Using techniques to be described later I generate a file containing the PostScript code associated with the graphical material. This file is transferred over to the IBM AT using MacLink Plus. The new file is processed by several AWK programs which insert, change, or remove PostScript commands as appropriate. Via these AWK programs I am also able to preview the diagram itself so that I can determine information needed to properly center the diagram on the printed page. Commands using the *\special* command in *T<sub>E</sub>X* are placed in my text file. These commands are then passed through untouched by *T<sub>E</sub>X* through to the *dvi* file. The *dvi* processor (in my case DVIPS from ArborText), then interprets the *\special* commands, includes the appropriate PostScript files, and prepares a PostScript file for the printer. This file is then sent via a parallel communication cable to the printer. The resulting document contains the integrated text and the graphical material.

These techniques of incorporating graphical material into *T<sub>E</sub>Xed* documents may not be useful in every case. If you are producing only one version of a document that contains only one or two diagrams then it may be simpler to paste in a copy of the graphical material. These techniques become useful when you have many figures to include in a document or when multiple versions or multiple drafts of the document will be prepared.

The remaining sections of this paper explain the techniques used to incorporate graphical material. First an explanation of relevant PostScript concepts is presented, including the concept of a PostScript header file. After this, the techniques for producing the PostScript file representing graphical material are presented. The techniques for manipulating and converting this PostScript file are then explained. Finally some techniques useful in printing the document are discussed. Appendix A contains some AWK code that is used in processing the PostScript files.<sup>2</sup>

## 2. Understanding PostScript

PostScript is a language used to describe where to put certain objects on a page. Of course this is a very simplified statement of the capability of PostScript, but it is sufficient for the mental model that I am trying to help you build for PostScript. I am including this material to help you *read* PostScript and write a few simple commands in it. To learn more about it you should consult either the second or the fourth reference in the Bibliography.

The initial coordinate system for PostScript has 72 units to the inch, with the origin at the lower left corner of the page. Thus the position (144,72) would correspond to a point 2 inches to the right of the lower left corner and 1 inch above the lower left corner.

PostScript uses post-fix notation; that is, the parameter or arguments are given first and then the command follows. For example, 100 100 moveto tells PostScript to move its current point to position (100,100). PostScript also assumes a stack architecture — parameters are placed on a stack and then removed or manipulated by the operators.

It is important to understand how PostScript knows what to do when it sees an operator. The simple answer is that the PostScript interpreter looks up the operator in a dictionary to see what it means. That is a simple model to follow. In fact, a user can create a stack of dictionaries for the PostScript interpreter to use. The interpreter looks up the command in the top dictionary first. If it finds the entry there then it does what the dictionary entry says. If it does not find the entry in the first directory it looks in the second directory. If it finds the entry there then it does what the dictionary entry says. The interpreter continues this process through all the directories on the dictionary stack until it finds some definition for the operator. If the interpreter has searched all the directories on the dictionary stack and it has not found the operator, it signals that a bad operator has been given. If the same operator name appears in two different dictionaries then the definition in the dictionary that is above the other in the stack will be used. This provides a very easy way to make sure that PostScript does exactly what you want with the commands you use. You create a new dictionary on top of all the other dictionaries and put your special commands in it. This is the way that most drivers

---

<sup>2</sup> A full set of documentation for this work, including complete examples, can be obtained by sending a message to either the electronic or physical address of the author.

for PostScript work. We will talk about these special dictionaries in the next section.

The PostScript commands that you need to learn for this work fall into a few categories:

1. Translating, scaling, or rotating the material to be placed on a page.
2. Creating a dictionary and putting it on the dictionary stack and then removing it.
3. Defining a new operator.
4. Saving and restoring the state of the entire PostScript interpreter or just the graphics state.

## 2.1 PostScript Movement Operators

These operators will be used to adjust your graphical material to appear on the printed page at the right location. *The methodology advocated in this paper is to place the bottom center of the diagram at the coordinates, (0,0).* The commands used are **currentpoint**, **moveto**, **translate**, **scale**, and **rotate**.

- a. **currentpoint** – This puts on the stack the coordinates of the “PostScript pointer” at the current time. This is useful when PostScript knows where it is but you don’t.
- b. **moveto** – This command moves the current location of the PostScript writing pen to the coordinates you designate. For example **72 72 moveto** moves the graphics pen to the point one inch up and one inch in from the bottom left corner of the paper.
- c. **translate** – This command translates the origin to a new point. For example, **144 144 translate** moves the origin up two inches and over two inches. A common use for this command in the context of placing graphical material into documents is **currentpoint translate**. You may have positioned your diagram so that the bottom center of your diagram is at (0,0). When the DVI-to-PostScript program tries to put your diagram on the page the current point may be (212,345.33). By putting **currentpoint translate** at the beginning of your graphics PostScript file, you have removed any need to know exactly where your diagram will go on the page.
- d. **scale** – This command scales the coordinate system. It can be used to shrink the coordinate system or expand it. The shrinking or expanding may be non-uniform (i.e., the shrink may be bigger in the *x* direction than in the *y* direction). For example, **2 2 scale** expands all the coordinates by 2. Now the point (72,72) would be two inches over and two inches up. We have to use the **scale** operator because DVIPS does some necessary scaling that we, nonetheless, must undo. Another useful but initially confusing example is **1 -1 scale**. This produces a flip of the graphics along the *x*-axis. We have to use this when dealing with CricketDraw output.
- e. **rotate** – This command rotates the coordinate system. In this paper it will be used to convert diagrams from landscape to portrait and vice versa.

The diagrams presented in Figure 1 illustrate how these operators can be used to place the bottom center of the figure at the origin. For each diagram, additional code has been created to put an axis and rectangles indicating four  $8\frac{1}{2} \times 11$  inch pages so that the reader can better see how the transformations are taking place. Two new operators (**makedarkrectangle** and **makedarktriangle**) have been defined to produce the geometric figures.

Determining which transformation should be applied first and which second, etc. has always been counter-intuitive to me. As you can see from the examples, you should first take the code that draws the figures, then precede it by the first transformation. Then precede all this by the second transformation, and then precede all that by the third transformation, and so on.

Figure (1a) represents the initial diagram which consists of one rectangle and one triangle. Figure (1b) represents the diagram with the bottom center of the diagram shifted to the origin. Figures (1c) and (1d) represent a translation of the original code to put the left side center at the origin and then a rotation to put the diagram “bottom centered” at the origin when the diagram is in a landscape position. Figures (1e) and (1f) represent the same initial translation to the origin as in Figure (1b) but then a scaling by .5 in each direction to shrink the diagram so that it is centered at the origin but in a smaller size. This last technique is useful for adjusting a diagram when it is initially too large to fit in the space allocated for it in the document.

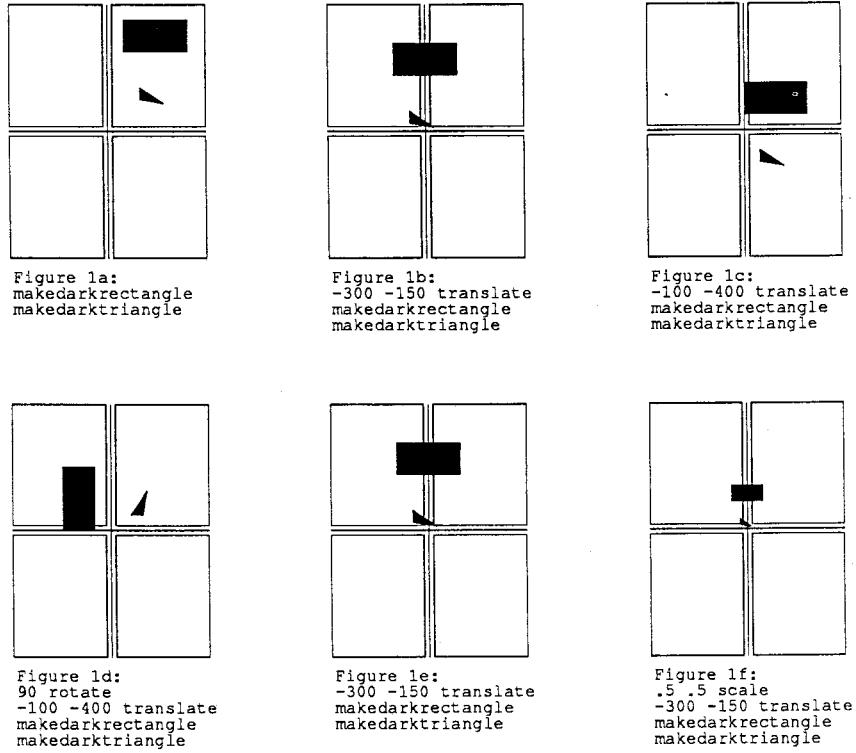


Figure 1: Illustrations of PostScript movement operators

## 2.2 PostScript Dictionary Operators

This paper covers only a few of the many PostScript operators used for manipulating dictionaries. The operators discussed here are used to create a new dictionary, and to put a dictionary on the stack of dictionaries being used by PostScript and to remove a dictionary from this stack. You need to realize that PostScript can store dictionaries “over in the corner” when it does not have them on the active stack of dictionaries. Via the appropriate command, a dictionary “over in the corner” can be brought out of storage and put back on the stack. All its entries are still intact.

The way to create a brand new dictionary is via a command string such as `/mydict 200 dict def`. This defines a new dictionary, which I have arbitrarily named `mydict`, which has room for 200 entries and which is currently stored “over in the corner”.

The way to bring a dictionary (for example, one named `myoldone`) that is “over in the corner” to the active stack is to give the command string `myoldone begin`. Notice that we did not have a “/” at the beginning of `myoldone` because the name is already known to PostScript.

The way to remove a dictionary from the active stack is to give the command `end`. This removes the top dictionary from the stack and places it “over in the corner”.<sup>3</sup>

## 2.3 PostScript Definition Operators

Just as macros are used in `TEX` to define useful combinations of basic commands, so procedures are used in PostScript to combine useful combinations of PostScript operators. A simple form of this definition is `/myownprocedure { some PostScript code } def`. This process causes the name `myownprocedure` to be added to the topmost dictionary in the stack of dictionaries and the corresponding PostScript code to be entered as the meaning of `myownprocedure`.

With this form of defining a procedure, when PostScript encounters `myownprocedure` it will find the corresponding PostScript code in the dictionary and proceed to look up the *current* definition of

<sup>3</sup> I have continued to use the phrase “over in the corner” to impress upon you that dictionaries don’t disappear just because they are removed from the stack.

each of the PostScript operators and then execute them. This in fact is the way that *T<sub>E</sub>X* behaves when macros are expanded normally.

There is another way to define procedures in PostScript which can be a bit confusing at first but is very helpful once it is understood. This involves the use of the **bind** operator. If a PostScript expression such as `/moveit { 100 100 translate } bind def` is used, then the **bind** operator looks up all the operators — in this case **translate** — in all the dictionaries. If it finds that **translate** has been re-defined from its original meaning (by having been defined again in some new dictionary) then **bind** does nothing. If it finds that **translate** has not been re-defined then it associates **translate** with its original definition and in fact causes this name to point directly to that code. This has two beneficial results. The first benefit is that when the **moveit** procedure is called, the PostScript interpreter will not have to go through all its dictionaries for the definition of **translate**. It can immediately execute the code associated with it. The second benefit is that even if between the time that the **moveit** procedure is defined and the time that it is executed another dictionary is added to the stack of dictionaries which re-defines **translate**, the **moveit** procedure is unaffected.

It is important to understand the necessity for using the **bind** operator at times. In some cases header files can interfere with one another and the person producing the document must load header files in such a way that the interference is avoided. This actually occurs when the Adobe Illustrator header file is loaded on top of the DVIPS header file. This will be discussed more later. This problem was solved by loading the Adobe Illustrator header file permanently, while the DVIPS header file dictionary was not on the dictionary stack, and the **bind** operator was used on the affected Adobe Illustrator code. The new release of DVIPS (Version 4.6) from ArborText has fixed this problem.

#### 2.4 PostScript State-Preserving Operators

When you insert PostScript code for diagrams into PostScript code that will cause the document to be typeset, the inserted code should make sure that when it finishes execution, the PostScript interpreter is in the same state as it was when the inserted code began execution. In *T<sub>E</sub>X* we could accomplish this by grouping, using { and }. In PostScript this is accomplished by using the **save** and **restore** operators. The way that these are used is as follows:

```
/vmsave save def  
many lines of PostScript code  
vmsave restore
```

In the remainder of this paper this convention is referred to as a “save–restore” encapsulation. Other distinct combination of letters could have been used instead of **vmsave**. The first line of code defines a “save” object, which reflects the state of the PostScript interpreter’s virtual memory at that time. The last line of code restores the state of virtual memory to this same state. All new definitions are forgotten, all new graphics operations are forgotten, etc. **Warning:** the operand stack is not fixed so you must make sure that your code leaves the operand stack just as it found it.

Usually any decent program producing PostScript code leaves the stack clean. These two lines of code are usually placed at the very beginning and end of the PostScript code for the diagrams that you are inserting. Sometimes the DVI-to-PostScript program will automatically insert these.

Another “save and restore” sequence used for saving the graphics state is **gsave** and **grestore**. These operators are easier to use than **save** and **restore** but are less powerful. The way that they are used is as follows:

```
gsave  
many lines of PostScript code  
grestore
```

The graphics state (i.e., the line width, the fill value, the current path, the current point, etc.) after the **grestore** will be the same as it was immediately before **gsave**. Any new definitions created will not be removed. Only the graphics state is restored.

### 3. Understanding and Using Header Files

Understanding header files can be easier if one is familiar with the concept of a file of macros for TeX. For example, the `plain.tex` file discussed in Appendix B of the *TeXbook* is such a set of macros. Both the header files and the files of macros are created to contain abbreviations for very commonly used sequences of commands. In both cases these files are loaded into the “interpreter” before the other code is sent to the interpreter. This can simplify the complexity of the other code that is sent since the code can use these abbreviated forms of commands rather than laboriously repeat the full sequence each time they are used.

Just about any program that produces PostScript output has a header file associated with it. DVIPS, Cricket Draw, Adobe Illustrator and MacDraft each have a header file associated with them. The header file is independent of the actual diagram and document that is produced. Thus, once a header file is analyzed and conquered (you feel like using that language after you have dealt with some header files) you never have to worry with it again. You don’t have to reconsider it each time a new diagram or document is produced. However, you need to make sure that the version of the header file that you have created and stored (perhaps some time ago) is the same one that the application is currently using. This is particularly important with the standard Laser Prep file used by MacDraft and most of the other Macintosh applications. Compare version numbers, which should be contained in each file.

Two nice examples of header files are those associated with Adobe Illustrator and Cricket Draw. In the following paragraphs I will tell you how to produce these header files and how to load them into the printer.

#### 3.1 Adobe Illustrator Header File

The header file for Adobe Illustrator can be produced simply. Create a diagram using Adobe Illustrator. For example, a diagram consisting of a single line is fine. Save this diagram. The Illustrator file will appear as a simple text file of PostScript commands. Using a text editor, open and edit this file. Extract the header file as follows. There will be some material at the beginning of the Illustrator file that is specific to the particular file and this can be removed. The header file then begins with the line

```
%%BeginProcSet:Adobe_Illustrator_1.1 0 0
```

and ends with the line:

```
%%EndProlog
```

Thus the whole header file is:

```
%%BeginProcSet:Adobe_Illustrator_1.1 0 0
% Copyright (C) 1987 Adobe Systems Incorporated.
% All Rights Reserved.
% Adobe Illustrator is a trademark of Adobe Systems Incorporated.
/Adobe_Illustrator_1.1 dup 100 dict def load begin
/Version 0 def
/Revision 0 def
% definition operators
/bdef {bind def} bind def
/ldef {load def} bdef
/xdef {exch def} bdef
% graphic state operators
/_K {3 index add neg dup 0 lt {pop 0} if 3 1 roll} bdef
(several more pages of definitions)

% font construction operators
/Z {findfont begin currentdict dup length dict begin
{1 index /FID ne {def} {pop pop} ifelse} forall /FontName exch def dup
```

```

length 0 ne
{/Encoding Encoding 256 array copy def 0 exch {dup type /nametype
eq
{Encoding 2 index 2 index put pop 1 add} {exch pop} ifelse} forall} if
pop
currentdict dup end end /FontName get exch definefont pop} bdef
end
%%EndProcSet
%%EndProlog

```

### 3.2 Cricket Draw Header File

To create a header file for Cricket Draw also is easy. Create a diagram in Cricket Draw consisting of one line or some other trivial diagram. While this file is open, choose "New" from the "File" menu and check the "PostScript" button in the dialog box, indicating you wish to create a PostScript file (as opposed to a "Draw" file). A new window will be created. Under the "Goodies" menu that has appeared with the activation of a PostScript window, select the entry "Create PostScript". This will display the PostScript code for the diagram, but not the header code. Now choose "Save As ..." from the "File" menu. Give this file a name and check "Complete" when indicating how to save the file. Exit Cricket Draw and use a text editor to examine the PostScript file. Like Illustrator, there will be code at the beginning and at the end of the file that is not a part of the header. This should be removed. The remaining material will be the header. The header file should look like:

```

/$cricket 210 dict def
$cricket begin
2 setlinecap
/d /def load def
/b {bind d}bind d
/l {load d}b
/e /exch l
/x {e d}b
/C /closepath l
/CP /currentpoint l
(several more pages of definitions)

/shadow {so}d
/charshadow {sc}d
/fountain {df}d
/offsetcalc {oc}d
/MakeOutlineFont {of}d
/MakeUnderlineFont {uf}d
/leftshow {ls}d
/rightshow {rs}d
/centershow {cs}d
/fullshow {ss}d
/coordinatefont {cf}d

%-----

```

One last line should be added to the header file in order to make it complete. That is the line  
`end`

and it is needed to close off and remove the dictionary that is being created for Cricket Draw.

### 3.3 MacDraft Header File

To create the header file for MacDraft may require Herculean strength. This header file is the standard Laser Prep file used by most applications that run on the Macintosh. I have included a list of suggestions on how to construct this file in Appendix B. It is definitely a non-trivial process.

We can now assume that we have three header files created. The next step is to load them into the PostScript interpreter's memory. There are four different techniques that can be used.

1. The header file can be downloaded directly into the permanent memory of the PostScript interpreter. The advantage to this is that the header file is accessible whenever you or anyone else needs it, until the power to the printer is turned off or the printer is reset. Another advantage is that you don't have to worry with how your DVI-to-PostScript processor would handle these header files. Still another advantage is that you will not be loading the graphics software's header (a.k.a. dictionary) on top of the DVI-to-PostScript package's header (a.k.a. dictionary). This last feature has already been referred to in Section 2.3. The disadvantage is that the header file takes up space in the printer's memory. On printers with a reasonable amount of memory (e.g., 2MB), this does not appear to be a problem.

To use this process requires placing the following additional line at the beginning of the header file:

```
serverdict begin 0 exitserver
```

The assumption here is that "0" is the password for your printer (it will be unless some printer guru has gone in and changed it). It may appear that you have placed an additional dictionary ("serverdict") on the stack and therefore you should place an additional "end" at the end of your header file. *You do not need to do this.* However, you should send a <ctrl-d> character to the printer at the end of this file to indicate that the end of this special file has been reached. I have done this on my system by creating a special file with only one character in it, namely <ctrl-d> (the HEX notation for this is 04). I send the header file and then this very small file to the printer.

2. The header file can be downloaded via a Macintosh computer attached to the printer. This technique is really useful only in downloading the header file associated with MacDraft and is the standard Laser Prep file for the Macintosh. This is by far the cleanest and simplest way to download this very complex file. Simply attach the Macintosh computer to the printer and print a Word document or a MacWrite document or a MacDraw diagram or a MacDraft diagram to the PostScript printer. This causes the header file to be downloaded permanently to the printer.
3. The header file can be downloaded as a part of printing the document itself. This process is dependent on the particular DVI-to-PostScript processor being used. If you are printing diagrams on multiple pages of the document, then you want to make sure that the header code survives from one page to the next one. If my header file is in `header.ps`, then I can place the following \special command at the beginning of the text for my document:

```
\special{ps: plotfile header.ps global}
```

and the header file will be correctly loaded. This particular command sequence is used by DVIPS and will most likely not work with other DVI-to-PostScript processors. The user should consult the manual for the particular DVI to PostScript processor used.<sup>4</sup>

4. If you are going to print only one diagram then you may want to include the header with the actual file itself. This technique is discussed later when the actual process for including a graphics file is considered.

We can now assume that the header file is loaded into the printer or else will be included with the diagram itself.

---

<sup>4</sup> There is a problem in using this technique (no. 3) with the version of DVIPS and Adobe Illustrator referenced in this paper. DVIPS's header re-defines a basic PostScript operator `concat`. When Adobe Illustrator's header is installed with the DVIPS dictionary already on the stack (as technique 3 would do), this operator, which is used by Adobe Illustrator, is not bound even though `bind` is used. The wrong definition of `concat` will be used. If the Adobe Illustrator header is loaded while the DVIPS dictionary is not on the stack, the `bind` operator will make sure that `concat` is defined properly and permanently. When the Adobe Illustrator dictionary is finally called during operations and placed above the DVIPS dictionary, the correct definition of `concat` will still be used.

#### 4. Producing PostScript Graphics

This section concerns the production of the PostScript code for the diagram itself. This production process depends on the application used.

- a. Adobe Illustrator – Using Adobe Illustrator, produce the diagram. Save the diagram. This entire file can be sent from the Macintosh computer to the IBM PC for processing.
- b. Cricket Draw – Using Cricket Draw, produce the diagram. Follow the procedures discussed with respect to Cricket Draw in the section on producing headers (Section 3.2). Instead of saving the diagram using the “Complete” option the file should be saved using the “brief” option. This will produce only the code for the diagram and not the header file.
- c. MacDraft – Using MacDraft, produce the diagram. Select the “Print” command. Position the cursor over the “OK” button but don’t click it. At the same time, press down the “Apple” or the “Four-Leaf” (different ways to describe the same key) key and the “F” key — then click on the “OK” button. The computer may sound some warning signals because these keys are pressed but this can be ignored. The message to look for after this sequence of key strokes and mouse clicks are performed is a message box which says “Creating PostScript file”. A file will be created called “PostScriptn” where  $n$  is an integer between 0 and 9. This file should be re-named to some meaningful name associated with the diagram.

#### 5. Transferring the Graphics

Any of the files produced on the Macintosh must be sent to the IBM PC. I use MacLink Plus which makes the job very easy. If you decide to use Kermit or some other file transfer protocol, then you will have to cope with the fact that on the Mac the EOL (*end-of-line*) is designated by just CR (*carriage return*) (0D in HEX) while in the IBM PC world it is designated by CRLF, (*carriage return-line feed*, 0D0A in HEX). There are fixes around this but a file transfer package such as MacLink Plus simplifies the work.

#### 6. Analyzing and Converting the Graphics Files

The graphics files that were produced on the Macintosh need to be altered in some standard ways to prepare them for incorporation into documents that will be processed by *T<sub>E</sub>X*. Each of the three graphics packages requires different changes. Also complicating the conversion is the fact that a diagram may have been produced in either portrait or landscape orientation and the document that will contain it may either be in portrait or landscape orientation. To handle all this repetitious file conversion work, I use the AWK programming language. It is ideally suited for this work. It is powerful, it runs on an IBM PC, it does text manipulation, and it can handle files in a combined interactive/batch mode.<sup>5</sup>

The following generic steps must be taken for all PostScript files sent over from the Macintosh:

1. Appropriate scaling information must be placed in the file to compensate for the uniform magnification provided by the DVI-to-PostScript processor.
2. The “bottom center” of the diagram must be determined and the graphics file must be adjusted to contain this information. It may be desirable to produce a test print for the diagram to make sure that it is correctly centered.
3. All unneeded PostScript code in the file must be removed. Any needed additional PostScript code must be added.

The matter of scaling depends on the DVI to PostScript processor. DVIPS magnifies all the PostScript code based on the magnification level indicated in the original document (i.e., what `\magstep` was used). For example, if `\magnification = \magstep1` is used, then a scaling factor of 1/1.2 must be included in the graphics file. This is done by placing the sequence `.8333 .8333 scale` in the appropriate (to be shown later) place in the graphics file.

The diagram that has been constructed on the Macintosh will need to be translated so that the bottom center of the diagram is at the coordinates (0,0). This makes the placement of the diagram in the document very easy. Since the diagram will not normally be constructed so that the bottom center of the diagram has these PostScript coordinates, a simple statement must be added to the file

---

<sup>5</sup> Complete AWK programs to handle all the various permutations of graphics packages and document and display orientations can be obtained via electronic mail from the author.

to adjust the origin. The following approach is used to determine the correct coordinates to use in making this code addition:

1. The original PostScript file is manipulated by an AWK program that adds code to produce a pair of very long lines that form a right angle on the page. The vertex of the right angle is printed at approximately one inch up and in from the lower left corner of the page. This modified code is sent directly to the printer. The resulting output contains two things — the diagram and a pair of reference lines. By visual inspection and by using a ruler with a scale in points, the person preparing the material can measure the relative displacement from each of these reference lines to the bottom center of the diagram.

It is unwise to use a corner of the paper as the reference point. The printer may have a systematic offset that it applies so that a corner of the paper does not correspond to (0,0). The orientation of the paper as it goes through the printer may be a little askew and this can also distort the measurement process.

2. A second AWK program is used to interactively ask the user for these two displacement coordinates and then to insert the correct PostScript **translate** command into the file to accomplish this needed translation.
3. A third AWK program can be used to print out this modified PostScript file to verify that the translation is correct.

Each graphics production package requires different changes to the contents of the file that it has produced:

- a. Adobe Illustrator – The header file information must be removed. The correct translation of bottom center must be added. The proper scaling must be added to compensate for the magnification found in the document. A “save–restore” encapsulation of the PostScript file must be put in place.
- b. Cricket Draw – A correction (**1 -1 scale 30 -762 translate**) must be inserted to correct for the fact that Cricket Draw does a flip of the diagram about the *x*-axis. A line of PostScript code must be added for each font used in the Cricket Draw diagram. The fonts used in the diagram will be listed in the header of the document. The AWK program can read these lines, remember the fonts, and then insert the needed lines at the appropriate place in the file. For example, if the two fonts used are Times-Roman and Helvetica, then the two lines to be added are:

```
(Times-Roman) coordinatefont  
(Helvetica) coordinatefont
```

The correct translation of bottom center must be added. The word **showpage** must be removed from the file so that printing will occur only after the entire page is done. The proper scaling must be added to compensate for the magnification found in the document. A “save–restore” encapsulation of the PostScript file must be put in place.

- c. MacDraft – The correct translation of bottom center must be added. The proper scaling must be added to compensate for the magnification found in the document. The line **F T cp**, which occurs near the end of the file, must be removed or “%ed out”. This line causes the page to print prematurely. A “save–restore” encapsulation of the PostScript file must be put in place.  
If the header code is to be included with the code for the diagram itself then the following alterations of the above additions must be used. The header material (minus the code **serverdict begin 0 exitserver**) should be inserted immediately after the line **/vmsave save def**. This will allow the header to be loaded before the code that calls on it is executed.

Appendix A to this paper contains the “before and after” codes for a simple diagram (the word “Hi”) for each of the three graphics packages.

## 7. Incorporation of Graphics Files into **TeX** Files

The files containing PostScript code for the graphical material must now be placed in the output stream that will be sent to the printer. This is done in a two-stage process and is dependent on the particular DVI-to-PostScript processor used. The following information applies to DVIPS.

The DVIPS processor prepares an environment in which you can place your diagram. When the `\special` command is given, DVIPS restores most of the pure PostScript environment (except the magnification level). You make the current point the center of your drawing environment by giving the command `currentpoint translate`. You will notice that this sequence has been added to all the sample files shown in Appendix A.

My placement methods for diagrams uses the "bottom center" approach. I determine the bottom center of the space on the page in which the diagram should fit. I find this approach much easier than if I have to specify the lower left corner of the diagram. Supposedly all the PostScript files should tell you where the bottom left corner of the diagram's "bounding box" is, but most of the programs don't do this correctly. Generally this space for the figures or diagrams is created through the use of a `\midinsert`, `\topinsert` or `\pageinsert`. If the diagram is 3 inches high then I can use the following code:

... and this is shown in Figure 1.

```
\midinsert
\vskip 3truein
\centerline{\hbox to 0pt{\special{ps: plotfile fig1.ps}}}
\medskip
\centerline{Figure 1 -- My First Figure}
\endinsert
```

A new paragraph . . .

Sometimes the diagrams are larger than will fit on one whole page using `\pageinsert`. As was mentioned earlier, these diagrams require additional scaling factors to reduce the size of the diagram. These scaling factors can be combined with the scaling factors used to correct for the document magnification. For example, if `magstep1` is used and we want to reduce the diagram to 85% of its original size anyway then, when using DVIPS, the correct scaling would be `.7083 .7083 scale` since  $.7083 = .85 \times (1/1.2)$ .

## 8. Additional Comments and Guidelines

The incorporation of graphics into documents definitely proceeds in two phases if it is to be successful. The first phase involves constructing the header files and building the AWK programs that will manipulate the PostScript files. This phase is tedious, subject to numerous errors, and will need to be completed only once. The second phase involves the production and conversion of the actual diagrams. This part is relatively easy. After approximately 25 diagrams were prepared in MacDraft, I converted and centered all of them in approximately one hour.

Sometimes it may help to start learning how to insert PostScript code by incorporating some pure PostScript code. You have no header to worry about. For example, the following code will cause one diagonal line to be constructed:

```
/vmsave save def
currentpoint translate
newpath
0 0 moveto
72 72 rlineto
stroke
vmsave restore
```

Once you have gotten this to work, you will have more confidence to tackle the bigger job of PostScript output from applications. Since PostScript code is simply ASCII text, you can always examine a (small) output file and get some idea of how the PostScript code for graphics has been inserted into the other PostScript code.

I feel that it is important to use AWK programs in order to perform this work. This allows for

the almost complete automation of this process. Dealing with all the cases of landscape diagrams being placed in portrait documents or landscape diagrams being placed in landscape slides requires either that you remember all the subtle but necessary corrections to make or that you depend upon a program to make them.

This paper has not addressed all the desirable features one would like in incorporating PostScript files. There are numerous other features that can be considered:

1. The PostScript code can be instructed to read the magnification setting that is current in the printer and scale the diagrams accordingly. Currently I need to know beforehand the magnification level that will be used in the document that will receive the graphics files.
2. Another form of auto-scaling can be built into the process. The *TEX* macros would be used to pass the size of the space available to the graphics file. Specially written PostScript procedures would be invoked to combine this information with information describing the size of the diagram and these procedures would then set the scaling factors correctly.
3. The positioning of text generated by *TEX* and to be placed in a landscape orientation when the body of the text is in a portrait position could be implemented. This would be useful in placing captions for landscape diagrams correctly in portrait-oriented documents.

I have discussed how you do this work but now you need to actually do it. What tools are available to help you? Your greatest new need will be to try to understand what is going on inside the PostScript printer, that inscrutable "black box". There are several possibilities:

1. There is a package called "LaserTalk", produced by Emerald City Software. It is excellent for sending code to a PostScript printer and getting error messages back. It is available for the Macintosh and IBM environments.
2. There are various error handling routines that can be downloaded to your printer. Then, when you send PostScript code to your printer and there is an error, *supposedly* a sheet will be printed showing the offending command and the state of the stack at the time the command was issued. Unfortunately there are times when you commit an error that will not allow this error handler to be invoked.
3. Communicate between the computer and the printer via a serial port that is set up to handle communication *from* the printer as well as *to* the printer. ArborText includes such a communications utility with the DVIPS program, called SPR. The value of this is that the PostScript interpreter will most always be able to use this method to send you the first offending command. It will not be able to use this method to send you any more information but at least this is a start.

A final comment: getting started with the process of incorporating PostScript code can be a trying or challenging experience. But once the process is established, incorporating graphical material can be easy, beneficial and very time-saving.

## 9. Acknowledgment

The work described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

## Bibliography

Adobe Systems Incorporated. *PostScript Language Reference Manual*. Reading, Mass.: Addison-Wesley, 1985.

Adobe Systems Incorporated. *PostScript Language Tutorial and Cookbook*. Reading, Mass.: Addison-Wesley, 1985.

Aho, Alfred V., Brian W. Kernighan, and Peter J. Weinberger. *The AWK Programming Language*. Reading, Mass.: Addison-Wesley, 1988.

Holzgang, David A. *Understanding PostScript Programming*. San Francisco, Calif.: SYBEX, 1987.

## Appendix A: PostScript Code and AWK Programs

The following material is the PostScript code for a simple diagram (the word "Hi") before and after modification for incorporation into a document. The code is included for Adobe Illustrator, Cricket Draw and MacDraft. In some cases large amounts of text has been removed and replaced by three lines consisting of a single period. This is done simply for economy of space in this paper. Normally no code would be removed.

### 1a: Adobe Illustrator Code Before Modification

```
%!PS-Adobe-2.0 EPSF-1.2

.

%%DocumentFonts:Courier
%%+Helvetica
%%BoundingBox:111 -395 252 -320
%%TemplateBox:288 -360 288 -360
%%EndComments
%%BeginProcSet:Adobe_Illustrator_1.1 0 0
% Copyright (C) 1987 Adobe Systems Incorporated.
% All Rights Reserved.
% Adobe Illustrator is a trademark of Adobe Systems Incorporated.
/Adobe_Illustrator_1.1 dup 100 dict def load begin
/Version 0 def
/Revision 0 def
% definition operators
/bdef {bind def} bind def

.

.

% font construction operators
/Z {findfont begin currentdict dup length dict begin
{1 index /FID ne {def} {pop pop} ifelse} forall /FontName exch def dup
length 0 ne
{/Encoding Encoding 256 array copy def 0 exch {dup type /nametype
eq
{Encoding 2 index 2 index put pop 1 add} {exch pop} ifelse} forall} if
pop
currentdict dup end end /FontName get exch definefont pop} bdef
end
%%EndProcSet
%%EndProlog
%%BeginSetup
Adobe_Illustrator_1.1 begin
n
%%BeginEncoding:_Helvetica Helvetica
[
39/quotesingle 96/grave
128/Adieresis/Aring/Ccedilla/Eacute/Ntilde/Odieresis

.

.

%%EndEncoding
%%EndSetup
```

```
0 g
.
.
.
%%Note:
/_Helvetica 72 12 0 0 z
[1 0 0 1 148 -380]e
(Hi)t
T
%%Trailer
_E end
```

#### 1b: Adobe Illustrator Code After Modification

```
/vmsave save def
%!PS-Adobe-2.0 EPSF-1.2
.
.
.
%%DocumentFonts:Courier
%%+Helvetica
%%BoundingBox:111 -395 252 -320
%%TemplateBox:288 -360 288 -360
%%EndComments
%%BeginProcSet:Adobe_Illustrator_1.1 0 0
% Copyright (C) 1987 Adobe Systems Incorporated.
% All Rights Reserved.
% Adobe Illustrator is a trademark of Adobe Systems Incorporated.
%%EndProcSet
%%EndProlog
%%BeginSetup
Adobe_Illustrator_1.1 begin
n
%%BeginEncoding:_Helvetica Helvetica
[
39/quotesingle 96/grave .
.
.
.
%%EndEncoding
%%EndSetup
currentpoint translate
.8333 .8333 scale
-320 288 translate
0 g
.
.
.
%%Note:
/_Helvetica 72 12 0 0 z
[1 0 0 1 148 -380]e
(Hi)t
T
%%Trailer
```

```
_E end  
vmsave restore
```

## 2a: Cricket Draw Code Before Modification

```
%!PS-Adobe-2.0  
%%Creator:Cricket Draw 1.1  
%%Title:Untitled #1  
%%CreationDate: 5/15/89 4:16 PM  
%%DocumentFonts: Helvetica  
%%BoundingBox: 0 0 612 792  
%%Pages:(atend)  
%%EndComments  
/vmstate save def  
/$cricket 210 dict def  
$cricket begin  
2 setlinecap  
%%LoadPrep:Cricket Procedures  
  
%----- Begin Main Program -----%  
gsave % Text Block  
0.000 1 -1 0.000 126.500 114.000 fixcoordinates  
/myshow /show load def  
0 setgray  
-63 -11 moveto  
/|_____Helvetica findfont 72 scalefont setfont  
{  
(Hi) show  
} leftshow  
grestore  
%----- End Main Program -----%  
showpage end  
vmstate restore  
%%Trailer  
%%Pages: 1
```

## 2b: Cricket Draw Code After Modification

```
%!PS-Adobe-2.0  
%%Creator:Cricket Draw 1.1  
%%Title:Untitled #1  
%%CreationDate: 5/15/89 4:16 PM  
%%DocumentFonts: Helvetica  
%%BoundingBox: 0 0 612 792  
%%Pages:(atend)  
%%EndComments  
/vmstate save def  
currentpoint translate  
.8333 .8333 scale  
1 -1 scale -100 -130 translate  
$cricket begin  
2 setlinecap  
%%LoadPrep:Cricket Procedures
```

```

(Helvetica) coordinatefont
%----- Begin Main Program -----%
gsave % Text Block
0.000 1 -1 0.000 126.500 114.000 fixcoordinates
/myshow /show load def
0 setgray
-63 -11 moveto
/|_____Helvetica findfont 72 scalefont setfont
{
(Hi) show
} leftshow
grestore
%----- End Main Program -----%
end
vmstate restore
%%Trailer
%%Pages:1

```

### 3a: MacDraft Code Before Modification

```

%!PS-Adobe-2.0
%%Title: Untitled-1
%%Creator: MacDraft
%%CreationDate: Monday, May 15, 1989
%%Pages: (atend)
%%BoundingBox: ? ? ? ?
%%PageBoundingBox: 30 31 582 761
%%For: Tom Renfrow
%%IncludeProcSet: "(AppleDict md)" 66
%%EndComments
%%EndProlog
%%BeginDocumentSetup
md begin

T T -31 -30 761 582 100 72 72 1 F F F F T T psu
(Tom Renfrow; document: Untitled-1)jn
0 mf
od
%%EndDocumentSetup
%%Page: ? 1
op
.
.
.
2 F /|_____Helvetica fnt
bn
-1.95671 0.(Hi)ashow
0 0 pen
254 147 gm
(nc 0 0 730 552 6 rc)kp
0 gr
104 146 194 227 0 rc
F T cp
%%Trailer

```

```
cd  
end  
%%Pages: 1 0  
%%EOF
```

### 3b: MacDraft Code After Modification

```
%!PS-Adobe-2.0  
%%Title: Untitled-1  
%%Creator: MacDraft  
%%CreationDate: Monday, May 15, 1989  
%%Pages: (atend)  
%%BoundingBox: ? ? ? ?  
%%PageBoundingBox: 30 31 582 761  
%%For: Tom Renfrow  
%%IncludeProcSet: "(AppleDict md)" 66  
%%EndComments  
%%EndProlog  
%%BeginDocumentSetup  
/vmsave save def  
currentpoint translate  
.8333 .8333 scale  
-250 -600 translate  
md begin  
  
T T -30 761 582 100 72 72 1 F F F F T T psu  
(Tom Renfrow; document: Untitled-1)jn  
0 mf  
od  
%%EndDocumentSetup  
%%Page: ? 1  
op  
. . .  
2 F /|_____Helvetica fnt  
bn  
-1.95671 0.(Hi)ashow  
0 0 pen  
254 147 gm  
(nc 0 0 730 552 6 rc)kp  
0 gr  
104 146 194 227 0 rc  
%%F T cp  
%%Trailer  
cd  
end  
vmsave restore  
%%Pages: 1 0  
%%EOF
```

Two AWK programs are included as examples to show the utility that the AWK language can provide for manipulating PostScript files. The first program reads in a PostScript file produced by MacDraft and puts in the code necessary to generate two reference lines. This altered file can be sent to a PostScript printer and the resulting diagram (with reference lines included) can be used to determine the "bottom center" of the diagram. It is assumed that the MacDraft header has already been loaded into the printer.

```
#Add the vmsave header
#Add the code to generate the reference lines
/^%%BeginDocumentSetup/ {print $0
    print "/vmsave save def"
    print "gsave"
    print "newpath"
    print "72 720 moveto"
    print "360 0 rlineto"
    print "stroke"
    print "newpath"
    print "72 720 moveto"
    print "0 -360 rlineto"
    print "stroke"
    print "grestore"
    next}
#Add the restore command at the end
/^end$/ {print $0
    print "vmsave restore"
    next}
#print the lines which don't contain an end of file marker
$0 !~ "\032" {print $0}
```

The second AWK program converts a Cricket Draw file from the original form transferred from the Macintosh to the form that can be included in the PostScript code for the document. Before the user runs this AWK program, it is assumed he has run another AWK program like the one above (but designed for Cricket Draw) which helps the user determine the "bottom center" of the diagram. The AWK program interactively queries the user for the two needed offset measurements. The results of the transformation are written to a file specified by the variable `outfile`. This variable is specified by the user in the command line that invokes the program. An interesting feature of this program is that it saves font name information from one part of the text file and then writes it out later.

```
#Ask user for offset information
BEGIN {print "How many points from the left line"
        print "is the center of the figure?"
        if (getline > 0) xoffset = $1
        print "How many points up from the bottom line is"
        print "the bottom of the figure?"
        if (getline > 0) yoffset = $1}
#Build the list of fonts needed in Cricket Draw
#Note: This command will be executed only after the
#/%%DocumentFonts:/ command has been used.
/^%%LoadPrep:/ {print "%-----" > outfile
    print "%Encode PS Fonts to match Mac Fonts" > outfile
    for (name in fonts)
        print "(" name ")" coordinatefont > outfile
    print "%-----" > outfile
    next
    }
```

```
#Put in the needed offset information
/^\\\$cricket/ {
    print "currentpoint translate" > outfile
    print "/xoffset " xoffset " def" > outfile
    print "/yoffset " yoffset " def" > outfile
    print "/specialmag " specialmag " def" > outfile
    print "specialmag dup neg scale" > outfile
    print "xoffset 100 add neg 500 yoffset neg add neg translate" > outfile
    next}

#Turn off the routine that gathers font information
/^%%BoundingBox:/ {action = 0; print > outfile ; next}

#Turn on the routine that collects font information
/^%%DocumentFonts:/ {action=1; fonts[$2]=0; print > outfile ; next}

#Another line that will contain font information
action == 1 {fonts[$2] = 0; print > outfile ; next}

#Remove the showpage command so that the page is not
#printed prematurely
/^showpage/ {print "end" > outfile ; next}

#Print the other lines which don't contain an end of file marker
{if ($0 !~ "\032") print > outfile }
```

## Appendix B: Suggestions on Creating MacDraft Header File

Producing a working version of the MacDraft Header File is a challenging task. Basically this involves producing a working version of the Laser Prep file on the Macintosh.

The first step is to capture the header file. This can be done by creating a file in Word or MacWrite that consists of some small amount of text — for example, the letter "a". Select the "Print" command. Just before you click on the "OK" button, depress the "Apple" or "Four leaf" key and simultaneously the "K" key and then click on the "OK" button. Ignore the strange sounds produced by the machine. A new file called "PostScriptn" will be produced on the system and you can find it with the Macintosh Desk Accessory "Find File". Open this file with some text editor.

Now you have to pare this header file down some. There are several large code segments at the end of the header. These look like:

```
currentfile ok userdict/stretch known
not and{eexec}{flushfile}ifelse
373A767D4B7FD94FE5903B7014B1B8D3BED0
2632C855D56F458B118ACF3AF73FC4EF5E81F57490.

.
.
.

0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
cleartomark
currentfile ok userdict/smooth4 known
not and{eexec}{flushfile}ifelse

.

.

F94E00EE41A71C59E5CAEED1EDBCF23D1DBA1
EE99B9BB356492923BD8B1BA83A87CEB0E07377A3
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
cleartomark
%%EndProcSet
```

These segments can be removed. The rest of the file can remain. If you plan to download the file permanently to the printer, then you need to add the line **serverdict begin 0 exitserver** at the beginning of the file. When you transfer this file from the Macintosh to the IBM PC and edit it with a text editor, you must make very sure that some of the long lines do not get broken up incorrectly. One particularly bad section is the part that lists the names of special characters (e.g., /agrave /acircumflex /adieresis /atilde /aring /ccedilla /eacute /egrave). Text editors may break these lines in the very middle of a word and this causes the PostScript interpreter to think that there is a new name and also a command which it does not understand. Text editors may also break -1 into - and 1 which will cause an error. Once you get all these edits made and no lines broken improperly, then you can send the header to the printer to see if you have any errors. At this time it is nice to use a utility like LaserTalk to analyze the code as it is being sent.

The suggestions don't work magic but they may help.

# *texpic* – Design and Implementation of a Picture Graphics Language in $\text{\TeX}$ à la *pic*

ROLF OLEJNICZAK-BURKERT

Eiselauer Weg 12  
D-7901 Beimerstetten  
West Germany

## ABSTRACT

*texpic* is a  $\text{\TeX}$  implementation of a graphics language similar to Kernighan's *troff* pre-processor *pic*.

Many features of the original *pic* are supported, including a variety of graphical objects (boxes, circles, ellipses, lines, arrows and others), directions of motion, controlling sizes of objects with variable and appropriate defaults, relative and absolute positioning of single objects or whole pictures (labels and corners are allowed), and much more.

There are two significant enhancements. Objects adapt to the size of their contents; that is, a circle may contain a table with mathematical equations, a box may contain the circle, etc. *texpic* objects and  $\text{\TeX}$  or  $\text{L}\text{\TeX}$  commands may be combined at will.

The implementation consists of two parts, a set of elaborate  $\text{\TeX}$  macros and a post-processor for drawing (in the *dvi* file), written in C. It should be emphasized that *texpic* is fully portable, i.e., every  $\text{\TeX}$  version, every preview and even every (correctly written) printer driver will work together with *texpic*.

## 1. Preface

Some years ago I attended a lecture on text processing. At that time I had just discovered  $\text{\TeX}$  and was filled with enthusiasm, but unfortunately the lecture dealt mainly with another system: the *troff* typesetting software, widely used under UNIX.

There ensued a friendly competition between the lecturer and me — with the goal being to typeset things the other one couldn't do. One time he won, another time I made a point, so the race was rather even.

With introducing *pic* one day, a powerful, but easy to use language for drawing pictures, implemented as a pre-processor to *troff*, the tables turned. Because  $\text{\TeX}$  has little to retort, I began to lose very often. To catch up, I decided to implement something similar in  $\text{\TeX}$ , not knowing what frustration (and fun) this would be!

## 2. Boxes — The Cornerstone of $\text{\TeX}$

Boxes are probably the only objects which are easy to implement in  $\text{\TeX}$ . This is because  $\text{\TeX}$  also uses a box concept which offers two possibilities. If we have specified width and height explicitly, we obtain just a box with these dimensions. Otherwise the smallest box is chosen which fits around its contents. For the frame of the box we need only horizontal and vertical lines — suitable commands already exist. Consequently we require the following:

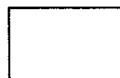
- Boxes have a minimum size.
- Between contents and frame there is a certain amount of free space.
- Boxes adapt to the size of their contents.
- Boxes are centered perpendicular to the current direction of movement.
- Minimum size, free space and the thickness of the lines are locally or globally changeable.

The resulting *TeX* macros are relatively straightforward. Producing a box with *texpic*, the complete syntax of the corresponding command is:<sup>1</sup>

```
\tpbox [attributes] [parameters] [contents];
```

An *attribute* such as *invis* describes a quality and is typically one word, whereas a *parameter* such as *width 3cm* influences the size of an object and consists of several words. Finally, the *contents* begin behind the last *parameter* or *attribute*, stop at the next semicolon and are often ordinary text. Subsequent sections will illustrate this.

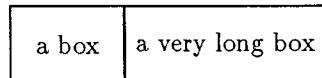
## 2.1 “Growing” Boxes with Minimum Size



a box

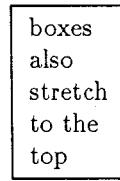
```
\tpbox; a box
```

As we can see, the box is centered on an imaginary horizontal line.



a box a very long box

```
\tpbox a box;  
\tpbox a very long box;
```



```
\tpbox \vbox{  
  \hbox{boxes}  
  \hbox{also}  
  \hbox{stretch}  
  \hbox{to the}  
  \hbox{top}  
};
```

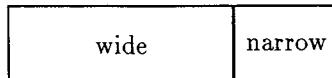
In the next example the current direction of movement is vertical which changes the centering of the box:



```
\par  
\tpbox;
```

## 2.2 Local and Global Changes

With *parameters* we can change various sizes of *one* object:

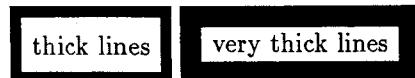


```
\tpbox width 3cm wide;  
\tpbox width 1cm narrow;
```



```
\tpbox width 0cm height 0cm narrower;  
\tpbox height 0cm width 0cm fill 0cm  
  very narrow;
```

Parameters which control the size of an object, control only the minimum size, i.e. if the contents don't fit, the object will still grow. The space between frame and contents is changed through *fill*. The *thickness* of the lines is also adjustable:



```
\tpbox thickness 4pt thick lines; \  
\tpbox thickness 8pt very thick lines;
```

<sup>1</sup> “tp” as a prefix for all names relating to *texpic* and should avoid name conflicts.

To achieve global changes we can simply change corresponding variables. As usual, braces control the scope:

all	boxes	have	equal	size
-----	-------	------	-------	------

```
{
  \tpboxwd=35pt \tpboxht=20pt
  \tpbox all; \tpbox boxes; \tpbox have;
  \tpbox equal; \tpbox size;
}
```

normal
--------

```
\tpbox normal;
```

~~all boxes have minimal size~~

```
{
  \tpboxwd=0pt \tpboxht=0pt \tpboxfill=0pt
  \tpbox all; \tpbox boxes; \tpbox have;
  \tpbox minimal; \tpbox size;
}
```

### 2.3 Sharing Attributes

With the attribute **same** we can make an object have the same size of the last one, provided that the contents fit:

all boxes have	the same	size
----------------	----------	------

```
\tpbox all boxes have;
\tpbox same the same;
\tpbox same size;
```

The first **same** is the attribute, the second is ordinary text! **\tppphantom** can be used, if the biggest box is not the first one:

1	12	123	1234
---	----	-----	------

```
\tpboxwd=0in \tpboxht=0in
\tppphantom{\tpbox 1234;};
%
\tpbox same 1; \tpbox same 12;
\tpbox same 123; \tpbox same 1234;
```

With **invis** we can make an object invisible, i.e. we suppress the frame. This attribute will prove useful later, when we want to position objects at different places:

an	invisible	box
----	-----------	-----

```
\tpbox an;
\tpbox invis invisible;
\tpbox box;
```

### 2.4 Boxes Around Other Objects

More complicated examples are possible — boxes are *bona fide* members of the TeX world:

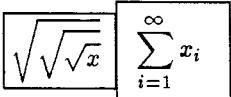
a. first item
b. second item

```
\tpbox
\hbox to 3cm{\vbox{
  \item{a.} first item
  \item{b.} second item
}}\hss
;
```

centered
----------

```
\centerline{
  \tpbox centered;
}
```

Notice, that \item would use the entire line, therefore \hbox is used to limit line length. Similarly, \centerline pushes the \tpbox to the middle.



The last example shows a particularly valuable feature: nesting. Most L<sup>A</sup>T<sub>E</sub>X macros also work with *texpic* boxes. So a box around a **tabular** or a box inside a **tabular** can be used. This is very useful for positioning.

### 3. Circles — Do they have to be so special?

Now on to the circles which should provide exactly the same features as the boxes above. As we will see, however, circles are much more complicated than boxes.

### 3.1 Two Dead Ends

To draw circles there are two approaches which will not work, at least not very well or with considerable restrictions:

1. Drawing in  $\text{\TeX}$  is possible, but this is very slow and there are also limitations regarding the number of circles, i.e., points, on the same page. See also the preface from the  $\text{P}\ddot{\text{T}}\text{C}\text{\TeX}$  Manual.
  2. Use of a printer language for drawing is possible with the `\special` command, though this means a commitment to one printer and therefore a loss of portability.

The second solution would be sufficient at the moment, but as in the original *pic*, references to objects should eventually be implemented. Because there is no way to get the current coordinates on the page in *TEX*, we could have transferred this problem to the printer language as well. However, this would certainly not improve any portability aspects.

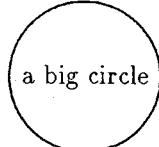
### 3.2 A Post-Processor for Drawing

Looking for other ways to obtain the coordinates of an object we discover the **dvi** file which is absolutely device independent. Reading this file (and some **tfm** files to get the widths of single characters) we are able to track the current position.

The main point, however, is that we can draw in the dvi files. This is a bit subtle, since we must pay attention to some pointers. With the use of the `\special` command and a post-processor written in C, the same features as for boxes are possible:

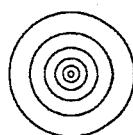


\tpcircle;



a big circle

```
\tpcircle a big circle;
```



```
\tpcercrad=Opt \tpcercfill=ipt  
\tpcircle{\tpcircle{\tpcircle{  
\tpcircle{\tpcircle{\tpcircle{  
};};};};};};};
```

#### 4. Directions and Movements — Not quite the same

Changing the current direction of movement in *pic* is possible at any time — “north”, “south”, “west” and “east” are allowed. Besides that, we can change the current point by a movement. Both features can be implemented in *TeX*, however, with some restrictions.

##### 4.1 Directions

Because we want to allow arbitrary objects, we require the following points:

- All four points of the compass are allowed.
- Macros `\tp...begin` and `\tp...end` enclose the objects of one “row”.
- Inside a “row” all arbitrary objects are possible.
- Every single object must be surrounded by braces.

In the following examples the default sizes have been decreased a little:

3	2	1
---	---	---

```
\topleftbegin
  {\tpbox 1;} {\tpbox 2;} {\tpbox 3;}
\topleftend
```

3
2
1

```
\tpupbegin
  {\tpbox 1;} {\tpbox 2;} {\tpbox 3;}
\tpupend
```

Directions can be combined with other objects as usual:

3
2
1
1

```
\topleftbegin
  {\tpbox 1;}
  {\tpupbegin
    {\tpbox 1;} {\tpbox 2;} {\tpbox 3;}
  \tpupend}
  {\tpbox 3;}
\topleftend
```

From a  to a .

```
\tprightbegin
{From a } {\tpbox box;}
{ to a } {\tpcircle circle.;}
\tprightend
```

Especially for positioning objects these features are very useful.

##### 4.2 Movements

An arbitrary change of the current position is not possible in *TeX*, therefore the design of movements is poor and rather restricted:

- A single `\tpmove` changes the reference point for a default value in the current direction.
- Specifying an optional direction moves only the next object.
- All default values are changeable locally and globally.

The first example shows the “normal” use of `\tpmove`, the second moves only one object:

two	horizontal	movements
-----	------------	-----------

```
\tpbox two;
\tpmove width 5pt;
\tpbox horizontal;
\tpmove same;
\tpbox movements;
```

	second
first	third

```
\tpbox first;
\tpmove up {\tpbox second;};
\tpbox third;
```

The `\tpmove` command in the last example does not change the reference point!

## 5. Arrows and Corners — Tying objects together

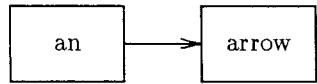
As said before, *pic* supports a “link” mechanism:

... to 3rd last circle ...

Since the actual position on a page is not available, this feature cannot be implemented in *T<sub>E</sub>X*. Because we are already using a post-processor for drawing circles, it is not very difficult to extend the C program to store the positions of the objects. The communication is done again with the `\special` command of *T<sub>E</sub>X*.

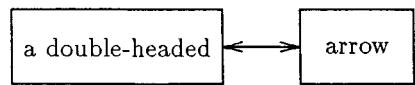
### 5.1 Arrows

To work not only with lines we implement arrows:

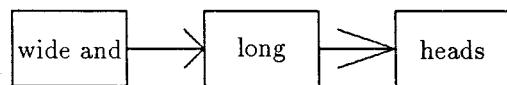


```
\tpbox an;
\tparrow;
\tpbox arrow;
```

*T<sub>E</sub>X* does not support slanted lines and *L<sub>A</sub>T<sub>E</sub>X* does not permit arbitrary slopes. Therefore, the arrow-head is drawn by the post-processor. There are two new *parameters* and one new *attribute* relating to the arrowheads:



```
\tpbox a double-headed;
\tparrow double;
\tpbox arrow;
```



```
\tpbox wide and;
\tparrow headwidth 0.2in; \tpbox long;
\tparrow same headheight 0.3in; \tpbox heads;
```

### 5.2 Links

Links to objects are much better than coordinates for connecting objects with lines or arrows. Because the original syntax of *pic* is not ideal for scanning, I changed the syntax slightly from `line from 2nd box to 3rd circle` to `\tpline from 2.box to 3.circle;`:



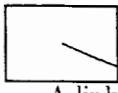
```
\tpcircle; \par \hskip 3cm \tpcircle;
\tpline from 1.circle to 2.circle;
```

Counting up results in absolute links. Relative links are constructed by counting backwards:



```
\tpcircle; \par \hskip 3cm \tpcircle;
\tparrow from 2.circle to 3.circle;
\tparrow from -2.circle to -1.circle;
```

If one link is missing the current position is used:

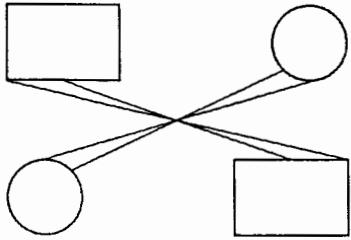


A link from a text.

```
\tpbox; \par
A link from a
\tpline from 1.box;
text.
```

### 5.3 Corners

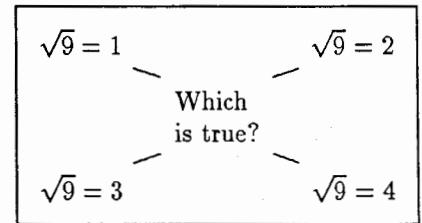
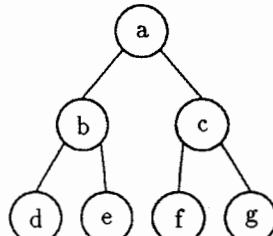
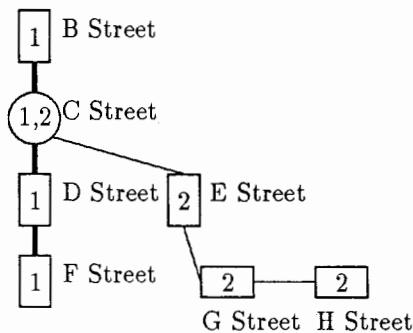
Links can even refer to eight compass points on the perimeter of an object:



```
\tpbox; \hskip 2cm \tpcircle;
\par \vskip 1cm
\noindent \tpcircle; \hskip 2cm \tpbox;

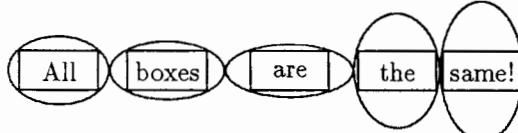
\tpline from -1.box.n to -2.box.s;
\tpline from -1.box.ne to -2.box.sw;
\tpline from -1.circle.n to -2.circle.s;
\tpline from -1.circle.ne to -2.circle.sw;
```

As you can see, circles also have "corners". With these features fancy pictures become possible. However, they require too much code to be shown here:



### 6. Ellipses — Circles with a catch

Unfortunately, ellipses differ considerably from circles since there is not just *one* smallest suitable ellipse around an object:

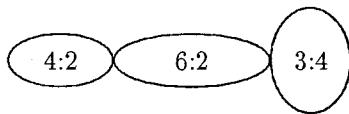


Because an ellipse has *two* major axes, it seems reasonable to require a fixed ratio for them:



```
\tpboxwd=30pt \tpboxht=15pt
%
\tpellipse {\tpbox invis;};
\tpellipse {\tpbox invis width 40pt;};
\tpellipse {\tpbox invis height 30pt};
```

If width or height are specified, the shape of the ellipse will change:

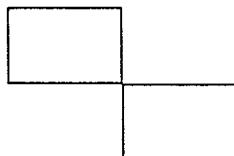


```
\tpelliwd=30pt \tpelliht=20pt \tpellifill=0pt
%
\tpellipse width 40pt 4:2;
\tpellipse width 60pt 6:2;
\tpellipse height 40pt 3:4;
```

Within each ellipse the ratio of its axes is displayed.

## 7. Shifted Objects — With and without size

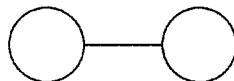
Sometimes it is useful to move whole objects. To do this, there are two new attributes: `with` and `at`. Unlike `at`, `from`, and `to`, `with` does not permit a link:



```
\tpbox;
\tpbox with .nw at -1.box.se;
```

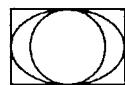
Again with the C post-processor, the implementation is simple: only one change in position has to be made. But there is a problem: objects which are to be moved must be set without any size; otherwise, they will need some place on the page and after being moved this place would be empty! So the user has to worry about the space.

The “corners” of lines and arrows are abbreviated with ‘s’ for ‘start’ and ‘e’ for ‘end’:



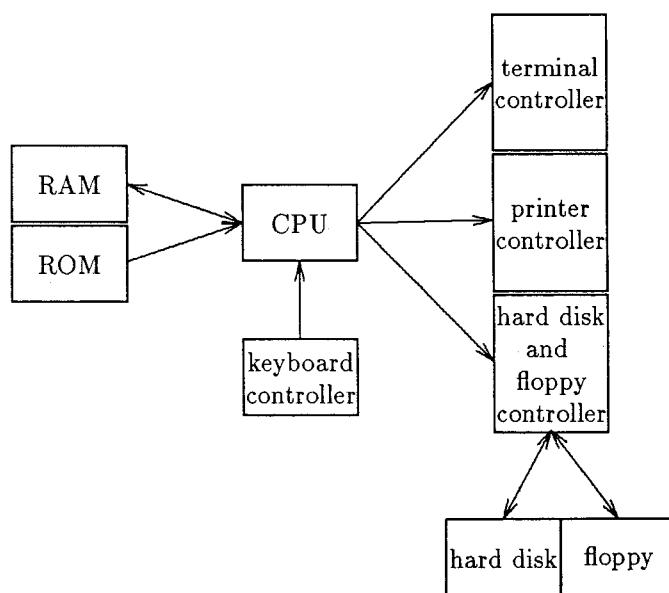
```
\tpbox invis;
\tpline;
\tpcircle with .e at -1.line.s;
\tpcircle with .w at -1.line.e;
```

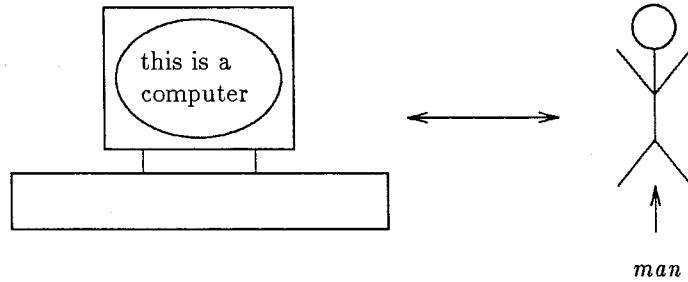
Without the `with` attribute, the center of the desired object is used:



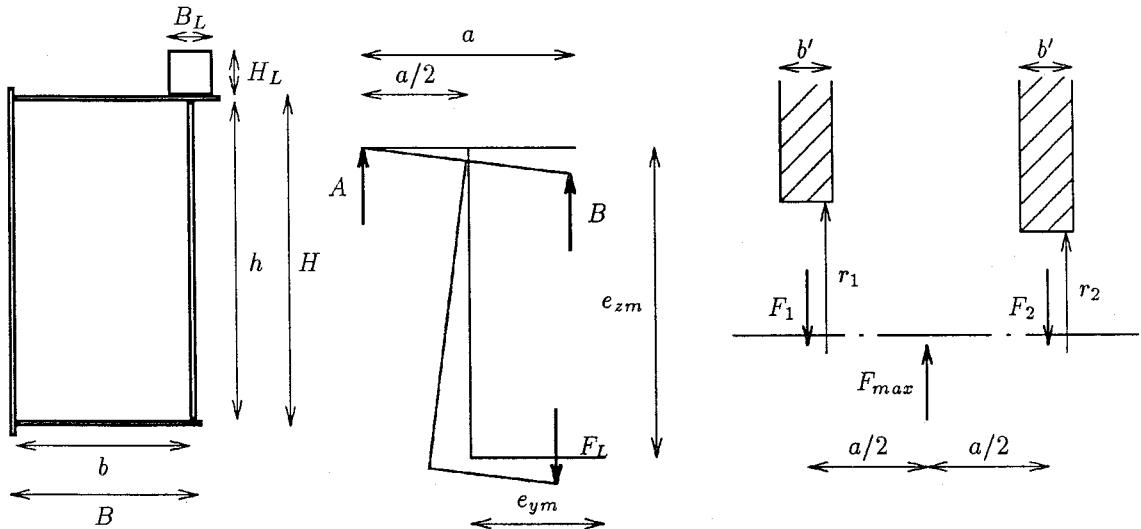
```
\tpbox;
\tpcircle at -1.box;
\tpellipse at -1.box;
```

Once again, more elaborated pictures are possible:





A friend of mine, a mechanical engineer, supplied the following pictures, which I would never have thought possible:



## 8. Conclusion

### 8.1 What has been done

Using *texpic* simple pictures in the style of *pic* can be drawn within a *TEX* document. Graphical objects have been implemented which may be used with several attributes and positioned in different ways. *pic* syntax was modified slightly to accomodate *TEX* conventions. Furthermore, there are two significant enhancements:

- Objects adapt to the size of their contents.
- The contents may be almost anything.

This results in a very smooth integration of text and line drawings. Through a C program as a post-processor operating on the *dvi* file, we achieved a very portable and absolutely device independent solution. Some points of the original *pic* were not implemented:

- *pic* itself serves as a target language for other pre-processors (*grap*, *chem*, etc.) New features in *texpic*, however, will most likely have to be constructed within *TEX* as well.
- In *pic* a picture can be scaled to near arbitrary dimensions. I see no way to do this in *TEX*.

### 8.2 What can be done

A few more features are probably practical:

- Generalization of the corners, for example *nnw* or *sssee*. This requires only a little bit of mathematics.
- Arcs of a circle and splines. This is possible with some mathematics and the C program.

- References to the dimensions of an object, e.g.:
 

```
... -1.box.ht ...
```
- Local scopes for objects. This requires an extension to the management of object stacks.
- Labels for objects or whole pictures, as in:
 

```
\tpbox name ellipsoid;
\tpline from ellipsoid.w ...
```
- Coordinates with addition and subtraction. This is very simple, because we already have the coordinates in the C program. The only thing to do is to build an interface to TeX, e.g.:
 

```
\tpbox with .n at -1.box.s minus (12,15);
```
- Interpolating a point, e.g.:
 

```
\tparrow from <1/3,-1.box.n,-2.box.s> ...
```

There is a syntactical problem: a link must consist of one word.
- Projecting object coordinates, e.g.:
 

```
\tparrow from (1.box.s,-2.ellipse.n) ...
```
- Printing and positioning text. The ideal would be along the lines of “printf”, because this is simple to implement in C.

### 8.3 What might be done

I am afraid the following features would be rather difficult to implement:

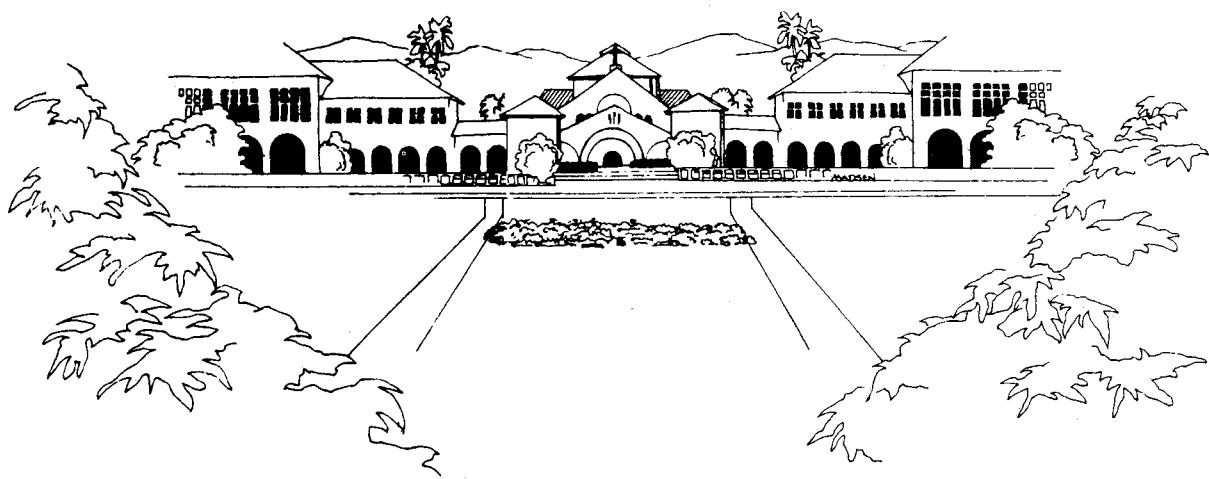
- Grids with automatic scaling. There is a question: what should a good grid look like?
- Drawing arbitrary functions. This requires all sorts of mathematical and syntactical support.
- Simple graphics in the style of *grap*, a pre-processor of *pic*.
- Rotation of objects. This would result in substantial changes since then every object must be drawn by the post-processor.

It is interesting to note that further refinement of features appears to shift more and more responsibility out of TeX and on to the post-processor. Is the ideal solution a graphical co-processor to TeX?

## Bibliography

- Adobe Systems Incorporated. *PostScript Language — Tutorial and Cookbook*. Reading, Mass.: Addison-Wesley, 1985.
- Adobe Systems Incorporated. *PostScript Language — Reference Manual*. Reading, Mass.: Addison-Wesley, 1985.
- Aho, Alfred V., Brian W. Kernighan, and Peter J. Weinberger. *The AWK Programming Language*. Reading, Mass.: Addison-Wesley, 1988.
- Appelt, Wolfgang. *TeX für Fortgeschrittene*. Bonn: Addison-Wesley, 1988.
- Bentley, J.L. and Brian W. Kernighan. “grap — A Language for Typesetting Graphs.” *CACM*. August 1986.
- Elan Computer Group. “pic — Reference Manual”.
- Foley, J.D. and A. Van Dam. *Fundamentals of Interactive Computer Graphics*. Reading, Mass.: Addison-Wesley, 1982.
- Hearn, D. and M.P. Baker. *Computer Graphics*. Reading, Mass.: Addison-Wesley, 1986.
- Jordan, B.W., W.J. Lennon and B.D. Holm. “An Improved Algorithm for the Generation of Nonparametric Curves.” *IEEE Transactions on Computers*. December 1973.
- Kernighan, Brian W. “pic — A Language for Typesetting Graphics.” *Software — Practice and Experience*. January 1982.
- Knuth, Donald E. *The TeXbook*. Reading, Mass.: Addison-Wesley, 1986.
- Knuth, Donald E. *TeX: The Program*. Reading, Mass.: Addison-Wesley, 1986.

- Kopka, Helmut.  *$\text{\LaTeX}$  — Eine Einführung*. Bonn: Addison-Wesley, 1988.
- Lamport, Leslie.  *$\text{\LaTeX}$  — User's Guide & Reference Manual*. Reading, Mass.: Addison-Wesley, 1986.
- Schreiner, Axel T. "Lecture on Text Processing." Given at the University of Ulm, Dept. of Computer Science, 1987/88.
- Schwarz, Norbert. *Einführung in  $\text{\TeX}$* . Bonn: Addison-Wesley, 1988.
- Wichura, Michael J. *The  $\text{\PCTEX}$  Manual*. Providence, Rhode Island:  $\text{\TeX}$  Users Group, 1986.
- Wonneberger, Reinhard. *Kompaaktführer  $\text{\LaTeX}$* . Bonn: Addison-Wesley, 1987.



**AMS-TEX82 Users Course and TEX Users Group Meeting**  
**Stanford University, July 11-15, 1983**  
**Terman Engineering Center Auditorium**

# T<sub>E</sub>X at *Mathematical Reviews*

D.C. LATTERNER AND W.B. WOOLF

*Mathematical Reviews*

P.O. Box 8604  
Ann Arbor, MI 48107-8604  
dcl@math.ams.com  
wbw@math.ams.com

## ABSTRACT

A brief description of the history of *Mathematical Reviews* (MR) is followed by a detailed discussion of the use of T<sub>E</sub>X in the current MR production system. The flow of a sample item through the MR "pipeline" is traced, showing how T<sub>E</sub>X is used to produce in-house tracking and proofreading documents as well as the MR journal and its related publications. The design and use of generically tagged data files extracted from the MR database are described, with an explanation of how this approach takes advantage of the powerful formatting capabilities of T<sub>E</sub>X. Finally, the use of T<sub>E</sub>X in the on-line version of MR is demonstrated.

### 1. Introduction

*Mathematical Reviews* (MR) is the world's pre-eminent secondary journal covering research in pure and applied mathematics.<sup>1</sup> It was founded in 1940; it has grown from 2,120 items in 1940 to about 50,000 per year currently.

In the early years, MR was produced from bibliographic information maintained on file cards and from review manuscripts prepared in pen and ink or on typewriters by reviewers around the world. Over the last fifty years, there has been some change in the form or manner of reviews submitted by reviewers: the handwriting is still quite varied and occasionally hard to read, the typing is still frequently error-prone and smudgy, but the typewriter has become a word processor. (The future is also beginning to show in the few manuscripts each month which are submitted electronically.) But more dramatic changes have taken place in the way the bibliographic information is maintained. From file cards in the 1940s (from which information was repeatedly re-typed in the process of preparing review forms and typesetting MR issues, indexes, annual indexes and cumulative indexes), and "ditto" masters in the fifties, sixties and seventies (which allowed one keyboarding for file cards and forms, but still required a separate keyboarding for each issue, index, annual index and cumulative index), MR moved to a database management system (SPIRES) and a computer typesetting system (STI, from Science Typographers, Inc.). This proprietary typesetting system allowed the same keyboarded input to be used for cards and forms as was used for all issues and indexes, but only the outputs created by the STI company from its typesetters carried the typeset appearance; cards and forms carried the raw coded input from which the STI typesetting program would create camera-ready copy. In 1983, MR changed to a CODASYL database management system (SEED), and yet another migration is under way at present, to a relational database management system (INGRES).

While MR was still using STI, it began experimenting with T<sub>E</sub>X. A program was written to convert the STI code into T<sub>E</sub>X, and a few macro files were written to format in-house forms. The advantages of T<sub>E</sub>X over STI soon became apparent. Outputs could be had in typeset rather than raw coded form

---

<sup>1</sup> A secondary journal carries abstracts and/or reviews (with bibliographic information) of articles published in primary journals and collections, as well as of monographs.

and, most importantly, MR was no longer dependent on an outside source for its typesetting needs. By 1985, MR had fully adopted  $\text{\TeX}$  as its typesetting language.

The following sections detail the ways in which  $\text{\TeX}$  is used to provide multiple outputs from the MR database, and the efficiencies and economies that this implies.

## 2. The MR Production System

From the time an article is received at MR to the time a review is actually published (and even beyond, to its appearance on-line in MathSci), the bibliographic data and the review undergo much examination and correction. Along the way, a variety of programs extract information from the MR database for proofreading purposes and report generation. These database extraction programs, along with the  $\text{\TeX}$  macro files ("headers") that format the extracted data, form the backbone of the MR  $\text{\TeX}$  production system.

### 2.1 The MR Database and Extraction Programs

The MR database is currently managed by a CODASYL-type database management system running on one of the DEC-20 machines at the headquarters of the American Mathematical Society (AMS) in Providence, RI; it contains bibliographic information on about 300,000 items reviewed in MR from 1985 to the present or awaiting publication. These several years of information are kept on-line to support referencing and author identification functions. Older material is available on archived copies of the database. The database is bibliographic only, i.e., it contains only author, reviewer, title, journal and publisher information, and a variety of flags for tracking purposes. Due to space considerations, the reviews themselves are not stored in the database, but rather as text files linked to the bibliographic information in the database by means of unique accession numbers, on-line before publication and on tape thereafter.

While the database resides on the DEC mainframe, much of the production work, particularly the typesetting, is off-loaded to a network of Apollo workstations located at the MR offices in Ann Arbor, MI. Data files from the mainframe are transferred to the Apollos via FTP or KERMIT. Most proof output is printed on an Imagen 5320, with an Imagen 8/300 serving as a secondary printer. Camera-ready copy is obtained from an Autologic APS- $\mu$ 5 in Providence.

Although the idea of generically tagged files is certainly not a new one, particularly in the  $\text{\TeX}$  arena, the use by MR of the principle provides a good illustration of the advantages of this approach in a  $\text{\TeX}$ -based production system. Since the format of the extracted data files is central to the way in which  $\text{\TeX}$  is used in production at MR, a brief discussion of the design principles behind the extraction programs follows.

The database extraction programs retrieve information from the database and format it in a manner easily manipulable by an appropriate set of  $\text{\TeX}$  macros. Figure 1<sup>2</sup> shows the output of the extraction program MAKMRB (MRB refers to MR bibliographic file) and an example of one type of  $\text{\TeX}$  document produced from this mrb file. This example illustrates the characteristics of an MR extracted data file:

1. Each data entity is individually tagged. Volume, year and issue, for example, are each tagged separately rather than combined into one field.
2. Data elements are tagged with unique, descriptive control sequence names and terminated with the control sequence  $\backslash endx$ .
3. All data associated with a particular item are pulled, even though certain elements may not actually be typeset in a given document. Figure 1 shows the element  $\backslash etype$  in the tagged file, although that element is not used in the output shown. Other documents, however, may typeset this field.
4. The tagged data do not contain formatting instructions.

Since most  $\text{\TeX}$  documents generated from an extracted file include only a subset of the data pulled, MR uses a system whereby the first macro file  $\backslash input$  is a "tags" file. The tags file contains macro definitions instructing  $\text{\TeX}$  to ignore all the possible tagged data fields associated with a particular

<sup>2</sup> All figures can be found at the end of the article -Ed.

extraction program. It is followed by a document-specific header which re-activates and formats the desired fields. For example, the line

```
\def \etyp #1\endx{}
```

is found in the file `makmr.tags`, in effect “nullifying” the `\etyp` field. In the header for a “daily box proof”, this field is re-activated with the definition

```
\def \etyp #1\endx{\qquad \hbox{\tt ET =} #1}}
```

while in the MR issue header, there is no re-activation and the definition supplied by `makmr.tags` remains in effect. Thus, the `tags` file is an efficient way to provide `TEX` with a definition for every tagged field, so that undefined control sequences can be avoided.

This approach to the design of the data files has allowed MR to take advantage of `TEX`’s tremendous formatting capabilities. The same generically tagged file can be formatted into literally any type of document simply by writing or modifying a set of `TEX` macros.

## 2.2 The Spine

Perhaps the best way to illustrate the production system at MR is to track the flow of a reviewed item through the MR “pipeline”. The process begins with the receipt of a “spine” — a book or an issue of a journal — into the MR library. An identification number, called a spine number, is assigned to the spine and certain basic bibliographic elements such as the book title or issue volume and number are keyed into the database.

To take a real example, suppose that Vol. 91, no. 8 of *The American Mathematical Monthly* arrives at the MR office. The next unused spine number, S117 897, is assigned and the issue volume and number are input into the database, along with flags indicating how the bibliographic information should be displayed in print.

After a day’s worth of spines has been entered in the database, several `TEX`ed documents are generated using an extraction program called SPINES. The first, a “spine form”, displays all the current information held in the database for a particular spine. The spine form is physically attached to the spine; any corrections or comments about the spine are recorded on the form and the database is later updated to reflect these changes. Figure 2 shows the output of the SPINES program and a sample spine form.

An interesting aspect of this particular form is the barcode at the bottom. Generated by `TEX` macros, it represents the spine number (`Issue cno:` on the form). Barcodes are used on a variety of MR documents for tracking purposes. Every four months, an inventory of items at various stages in the pipeline is conducted in order to identify old or lost items. Barcodes enable this function to be performed quickly and efficiently.

Mail logs are produced along with the spine sheets using the same extracted input files. These are lists of all spines added to the database on a particular day, separated into books and journal issues. The issue of *The American Mathematical Monthly* would appear on the log along with all the other journal spines that were added to the database that day, as in the following excerpt:

S117 893	Acta Math. Hungar. <b>53</b> (1984), no. 1–2
S117 896	Comm. Math. Phys. <b>120</b> (1984), no. 4
S117 897	Amer. Math. Monthly. <b>91</b> (1984), no. 8
S117 901	Bull. Calcutta Math. Soc. <b>80</b> (1984), no. 6

Next, spines are arranged into an “editors’ box” and routed to the MR editorial staff for decisions on whether they are within the scope of the areas of mathematics reviewed in MR, and for classification according to the Mathematics Subject Classification scheme. Travelling with the editors’ box is another `TEX`ed document based on the output of the SPINES program — an editors’ box log that lists the spines contained in the box. To continue the example, the editors scan *The American Mathematical Monthly*,

Vol. 91, no. 8, assigning preliminary 2- or 3-digit classifications to those articles they consider to be of interest to the mathematical community.

### 2.3 The Bibliographic Item

After the editors finish with the spines, they return them to the library where the staff provide a bibliographic "set-up" for the selected articles. The spines containing the selected articles are collected together into a "daily box" and a daily box log is generated using the SPINES program. New paper-specific accession numbers — called control numbers — are assigned to the chosen articles, and the bibliographic information for those items is keyed into temporary files using a program called MRPADD (MR paper add).<sup>3</sup> An extraction program called PADGAL is then used to pull the information from the MRPADD files, and a daily box proof is produced, using the output of PADGAL in combination with a  $\text{\TeX}$  header file (see Figure 3). The daily box proof is proofread and corrected and the information is loaded into the database.

Now that the complete bibliographic data for the items in a daily box are held in the database, the real workhorse of the extraction programs — MAKMRB — is used. Its first task is to create the input file for generating "editor assignment forms", forms that are used by the editors to record their assignments of reviewers to the items in the daily box that will eventually be reviewed (see Figure 4). The editor adds the first three letters of the reviewer's last name and his reviewer code (a unique number assigned to the reviewer, used internally at MR) to the form. There are also numeric codes on the form for those items that the editor suggests be reviewed from the author's summary, from the preface, etc. The forms are routed back to data entry staff and the assignments are keyed into the database. The example shows that M.S. Cheema, with a reviewer code of 03162, was assigned to review "The toilet paper problem".

### 2.4 The CMP Issue

After three weeks' worth of daily boxes have been added to the database (this amounts to 3000+ items), an issue of *Current Mathematical Publications* (CMP) is ready to be produced. The process begins with the creation of a giant `mrb` file containing the bibliographic information for all the items entered over the three-week period. Much of the CMP is generated in one way or another from this `mrb` file or parts of it, although some sections require further "massaging" by other programs in order to combine entries, add cross-references and section heading information, and provide some formatting instructions to  $\text{\TeX}$ . The source file for the CMP "Books Listed in This Issue" section, for example, is actually an unembellished `mrb` file, while that of the "Complete Bibliographic Listing by Subject Classification" section requires further `mrb` file processing. Figure 5 shows the familiar example as it appears in the "Bibliographic Listing" section along with its  $\text{\TeX}$  source. For illustration purposes, a fictional co-author has been introduced to show how cross-references are added to the source file.

The various sections of the CMP issue are actually  $\text{\TeX}$ ed twice. The first run generates output that is designed to make the job of proofreading easier: it is set in single-column format (or in some cases double column, if the section is ultimately printed in more than two columns) with plenty of space in the margins for marking. Also, the output is magnified, since the type size used in CMP is quite small. After proofreading and correction, the issue is re- $\text{\TeX}$ ed (this time in the format of the published CMP), camera-ready copy is ordered off the APS- $\mu$ 5, and the issue is delivered to the printer.

Because MR tends to use fairly small type in its publications, the scenario just described — a first run that generates enlarged output for proofreading and a final run that produces copy as it will appear in print — is an often repeated one at MR. This ability to switch from one output mode to another is a feature of many of the  $\text{\TeX}$  macro files used in MR production. Horizontal spacing (i.e., line breaking) remains the same from the first run to the final run; it is only the placement of the material on the page and the page length that changes. All of this is accomplished with switches built into the macros: a `\FirstRun` switch generates proofreading copy, a `\FinalRun` switch produces final

---

<sup>3</sup> The MRPADD interface was designed to facilitate input and to make the correction process easier. Inputting the bibliographic information directly into the database would be cumbersome.

copy. For the most part, the output routines used to produce final copy are fine-tuned enough so that re-runs for bad page or column breaks are rare.

## 2.5 Review Forms

Once the database has been updated with the reviewer assignment, a “review form” is printed and mailed to the reviewer along with a copy of the article to be reviewed. A detailed look at this form demonstrates the great versatility of  $\text{\TeX}$  in the MR production environment. The form consists of several parts (see Figure 6):

1. An overlay that partitions the form into various sections. (The macros for this overlay consist of a variety of rule boxes and some text which are put into a box the size of the page and \copy'd for each new form produced.)
2. The bibliographic information in the top left-hand corner derived from the `mrb` file.
3. The reviewer's address, also taken from the `mrb` file, at the bottom left-hand corner, positioned so that it appears in the window of the envelope when the form is folded.
4. The barcode at the very bottom of the form.

An interesting sidelight to the production of review forms has to do with so-called “pre-publication” items. These are articles from selected journals that have been accepted for publication by the editors of those journals but have not yet appeared in print. In order to accommodate this type of item efficiently, a new tag — `\isprepubl` — was added to `MAKMRB`. The presence of this tag in the `mrb` file for a review form will (1) add the phrase “*to appear*” to the bibliographic heading on the review form and (2) generate a separate attachment describing the treatment of pre-publication articles at MR. This example points out the ease with which new tagged fields can be introduced into the MR production stream. In this particular case, all that was needed after `MAKMRB` was programmed to output the tagged field was the addition of the line to ignore the field in `makmrbl.tags`, and a re-definition of `\isprepubl` in the header that produces the review form.<sup>4</sup>

MR strives to be as timely a publication as possible through procedures such as the pre-publication process just described and periodic inventories of items in the MR office. Another method used to keep MR up-to-date is the mailing of reminder letters to reviewers who have not returned their written reviews for a significant period of time. Every three weeks a list of such reviewers is compiled along with a  $\text{\TeX}$  source file containing their addresses and a list of the items they have not returned. The letters are then  $\text{\TeX}$ ed and mailed.

## 2.6 The Review

There are two ways in which a review is received back at the MR office. Since January of 1988, MR has been accepting manuscripts through the electronic mail system; currently, several dozen reviews a month are received electronically. Most reviewers, however, still type, typeset or write their reviews on the review forms provided and return them to MR by mail. Once received back at MR, the reviews are routed to the copy editors for copy editing and verification of references, and to the editors for substantive editing. They are then passed on to keyboarders in Providence and Ann Arbor. Reviews are input using the macros of  $\text{\AMS-TeX}$  as a standard, although there are a few cases where conflicts between  $\text{\AMS-TeX}$  and `plain.tex` are resolved in favor of `plain.tex` (for instance, MR uses \. for the dot accent as defined in `plain.tex` rather than \D of  $\text{\AMS-TeX}$ ). However, since MR is a journal of abstracts rather than complete papers, the mathematics generally is not difficult to code and there are relatively few instances where such conflicts occur. As noted earlier, the review text is not entered into the database but rather is written to a text file and tied to its bibliographic information by means of its control number. Thus, the name of the example review text file is `761401.msr` (`msr` refers to manuscript review).

After the review text has been keyed, the file is merged with its bibliographic information, re-named to be an `mrg` file (merge), and  $\text{\TeX}$ ed to produce an “item proof” (see Figure 7). At this point, the

<sup>4</sup> In the CMP “Complete Bibliographic Listing by Subject Classification” section, the `\isprepubl` tag also produces the phrase “*to appear*” for the entry. Later, after the article is actually published, the item appears a second time in CMP followed by the symbol  $\circ$ , triggered by the presence of the tag `\wasprepubl` in the `mrb` file.

item actually begins to look like an entry in an MR issue. In fact, the same *T<sub>E</sub>X* headers used to process the MR issue are used here to generate item proofs. Like the CMP headers, they contain a set of switch options for producing several different output types. Thus, there is an *\ItemProofSwitch*, a *\FinalRunSwitch* that calls up the macros for typesetting the MR issue as it appears in print, a *\ClippingsSwitch* for preparing "clippings" of reviews be sent to publishers of books, and a number of others.

Item proofs are routed to the editorial staffs for proofreading and editing, and then back to the keyboarding staff for correction. On the average, an item is run through *T<sub>E</sub>X* twice before it is declared "clean" and ready for the issue "pool".

### 2.7 The MR Issue

Once a month, reports are run on the approximately 4000 items ready for publication to give the editors a final opportunity to remove from, add to, or substitute for, items in the issue pool. After any necessary adjustments have been made, the MR issue is created: MR numbers are assigned (recall that up to this point there have been only control numbers), cross-reference information is created, and bibliographic information is extracted (using *MAKMRB*) and merged with the review text files in the issue pool. This process results in 61 *mrg* files, one for each subject area.

Next, *T<sub>E</sub>X* is run on each of the 61 section files to produce a first set of page proofs that are then scanned by editorial staff. An author index, a "key" index (a list of "unauthored" items such as collections or conference proceedings) and a list of new serials being reviewed in MR are also produced and proofread. After all corrections have been made, there is a final page proof run, this time in three or four large chunks, rather than section by section. Once again the proofs are scanned and, if there are no further corrections to be made (generally there are none at this stage), camera-ready copy is ordered off the *APS-μ5*.

### 2.8 MR Sections

MR provides mathematicians with the option of subscribing to a section or sections of MR in which their main interests lie. For instance, a combinatorialist who subscribes to MR Sections for section 05 is provided every month with a "mini" version of MR consisting of reviews from section 05 along with an author and key index for that section. Although MR has looked at the possibility of re-*T<sub>E</sub>X*ing the individual sections of MR and using laser printer output for MR Sections, it has so far been more cost-effective simply to re-use the camera copy from the MR issue. The author and key indexes for MR Sections, however, must be generated from scratch, since they represent only those authors and unauthored items within that section (whereas the indexes in the MR issue are alphabetized lists of names and keys taken from the issue as a whole).

### 2.9 The Annual Index

After the items in the December issue of MR have been assigned MR numbers (even before that issue is *T<sub>E</sub>Xed*), production of the Annual Index of MR begins. The Annual Index consists of a number of parts:

1. a bibliographic index by author name
2. a bibliographic index by "key"
3. a list of all serials appearing in MR
4. a list of journals that are translations of other journals
5. the complete list of author institution codes used in MR
6. some information on the transliteration of Cyrillic in MR
7. a bibliographic index by subject classification
8. the MR subject classification scheme

*Mrb* files containing all items that have been reviewed over the past year, as well as items that will not be reviewed but have appeared in CMP, are extracted. The format of the items in the bibliographic indexes is much like that of the "Complete Bibliographic Listing by Subject Classification" section of CMP. In fact, the same programs used to generate that section of CMP are employed here with only a few minor changes.

Thus, the Knuth paper mentioned earlier would be found in the author index under the letter

"K" along with any other papers Knuth may have written that were reviewed in the same year, and under section "05" of the subject index. Information on the journal in which the paper appeared could be looked up under "*Amer. Math. Monthly*" in the serials list; Knuth's institutional address could be obtained from the author institution code list under "1-STF-C"; and a description of the article's classifications, 05A10 and 05A15, could be found in the subject classification scheme (see Figure 8a-e).

The process of producing an Annual Index is spread over a period of about three months. Annual section indexes, including subject, author and key indexes, are also produced for MR Sections subscribers.

### **2.10 MathSci and MathSci Disc**

Since 1982, the AMS has provided the mathematical community with the information contained in MR and CMP in an on-line database now called MathSci. Users of MathSci can find information on any item published in MR since 1959 (the review text is available only for the years 1980 to the present), as well as those items from CMP not yet in MR. With the adoption of  $\text{\TeX}$  as its typesetting language in 1985, MR was able to add a new dimension to the MathSci database: users equipped with a  $\text{\TeX}$  software package (MathSci $\text{\TeX}$ , obtainable from the AMS) can print or preview the results (for items from 1985 or later) of their MathSci sessions in typeset form rather than as encoded on-line records (see Figure 9a-b).

MathSci Disc makes available a subset of the MathSci database (MR 1985–1988, plus 68,000 CMP entries) on CD-ROM. Search records can be downloaded to hard disk for typesetting with  $\text{\TeX}$  software.

### **3. Conclusion**

The advantages to a bibliographic database operation of a typesetting language that provides input for all manner of outputs — from office forms to issues and indexes — cannot be overstated. That the language is in the public domain and has been widely adopted by the mathematical community, and that it can be used to drive many varied output devices and fonts is of substantial additional value. While the current mode of operation at MR seems standard and routine to those who work there, it is useful to reflect on how modern, state-of-the-art, and efficient this mode is, compared to what was available only a decade ago.

```
\paper 761 401\endx
\mrnum 86a:05006\endx
\cno 761 401\endx
\hdyr 84\endx
\etyp J\endx
\status PUBL\endx
\cv 17\endx
\ci 2\endx
\editor AG\endx
\psubj 05A15\endx
\ssubj 05A10\endx
\authno \endx
\namepub Knuth, Donald E.\endx
\inst 1-STF-C\endx
\instname Department of Computer Science, Stanford University\endx
\instaddr Stanford, California, 94305\endx
\mttitle The toilet paper problem.\endx
\lang English\endx
\mrabr Amer. Math. Monthly\endx
\jname The American Mathematical Monthly\endx
\vol 91\endx
\yr (1984),\endx
\iss no.~8,\endx
\pp 465--470.\endx
\jorissn 0002-9890\endx
\jorcoden AMMYAE\endx
\tyrevtext Signed review\endx
\revf Cheema, M. S.\endx
\revr M. S. Cheema\endx
\revl (1-AZ)\endx
\revcode 03162\endx
\endpaper
```

**86a:05006 05A15**  
**Knuth, Donald E. (1-STF-C)**  
**The toilet paper problem.**  
*Amer. Math. Monthly* **91** (1984), no. 8, 465–470.

Figure 1: MAKMRB output and sample TeX document

```

\spines \endx
\isscno S117 897\endx
\maillog AA\endx
\issindate 841012\endx
\isszero \blanks \endx
\counttag \blanks\endx
\jorcoltype \blanks\endx
\jorkey AMEMM\endx
\jorabbrev Amer. Math. Monthly\endx
\pubkey 0-88385B\endx
\pubname Mathematical Association of America\endx
\pubtitle Math. Assoc. America\endx
\publoc Washington, DC\endx
\pgprf N\endx
\issvol 91\endx
\vol N\endx
\issyr 84\endx
\yhd N\endx
\issno 8\endx
\ord N\endx
\issname \blanks\endx
\jctitle \blanks\endx

```

Issue cno:	S117 897	Count:	_____
Entry date:	84/10/12	Journal collection type:	_____
Issue zero:		Related cno:	_____
*****			
AMEMM/Amer. Math. Monthly			
0-88385B/Mathematical Association of America, Washington, DC			
Pageproof flag:	N	*****	
Volume:	91	Paren. number:	
Volume slash flag:	N	Part number:	
Issue year:	84	Supplement:	
Year hanging date:	N	Hanging date:	
Issue number:	8	Order flag:	N
Multi issue number:		Duplication:	
Issue name:	_____		
JC title:	_____		



Figure 2: SPINES output and spine form

```
\dbox 841015\endx
\isscno S117 897\endx
\jorkey AMEMM\endx
\cno 761 401\endx
\ettype J\endx
\pp 465--470\endx
\ioflag N\endx
\psubj 05A\endx
\mttitle The toilet paper problem.\endx
\namepub Knuth, Donald E.\endx
\inst 1-STF-C\endx
\instname Department of Computer Science, Stanford University\endx
```

S117 897 761 401 ET = J  
PP = 465-470  
IO = N  
PCLASS = 05A  
T = The toilet paper problem  
A = Knuth, Donald E.  
INST = 1-STF-C Department of Computer Science, Stanford University

Figure 3: Daily box proof input and output

Check here if this item should be given  
high priority (gold slip) treatment.

761 401

et:J

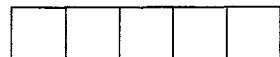
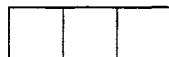
CMP 17:2

05A

Knuth, Donald E. (1-STF-C)

The toilet paper problem.

*Amer. Math. Monthly* 91 (1984), no. 8, 465–470.



(reviewer last name - first 3 letters)(reviewer code)

DATE :

EDITOR :

Summary ..... (2)

From text ..... (9)

From sum ..... (4)

From preface ..... (10)

EDS ..... (1)

Introduction ..... (5)

Title ..... (12)

From intro ..... (7)

..... (11)

Date Produced : 1/26/85



761 401

Figure 4: Editor assignment form

---

```
\newaut Knuth, Donald E.\endx
\inst 1-STF-C\endx
\others ({\it with\}/} Drofnats, R. J. \instc{1-ABC-D})\endx
\mttitle The toilet paper problem.\endx
\mrabr Amer. Math. Monthly\endx
\vol 91\endx
\yr (1984),\endx
\iss no.~8,\endx
\pp 465--470.\endx
\cmpclass 05A\endx
\endpaper
\newaut Drofnats, R. J.\endx
\see Knuth, Donald E.\endx
```

**Knuth, Donald E.**(1-STF-C) (*with* Drofnats, R. J. (1-ABC-D)) The toilet paper problem. *Amer. Math. Monthly* 91 (1984), no. 8, 465–470. 05A  
**Drofnats, R. J.** See **Knuth, Donald E.**

Figure 5: CMP “Complete Bibliographic Listing by Subject Classification” TeX input and output

MATHEMATICAL REVIEWS											
761 401 et : J	CMP 17:2	AG 05A	TOG:	RP							
Knuth, Donald E. (1-STF-C) The toilet paper problem. <i>Amer. Math. Monthly</i> <b>91</b> (1984), no. 8, 465–470.		REVIEWER: Please give 5-character classification(s) according to 1980 Subject Classification (1985 Revision). (See the most recent MR Annual Subject Index.)  Conventions (use colored pencil): Greek : underline in red German Fraktur : print or type roman letter, underline in green Script : print or type roman letter, encircle in blue Boldface : underline with a wavy blue line  Do not underline for italics. (Letters used as mathematical symbols are automatically italicized by our printer.)									
<b>PLEASE TYPE WITH EXTRA SPACE BETWEEN LINES</b>											
<table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>T</td></tr> <tr><td>C1</td></tr> <tr><td>C2</td></tr> <tr><td>L</td></tr> <tr><td>E1</td></tr> <tr><td>E2</td></tr> <tr><td>Q</td></tr> </table>					T	C1	C2	L	E1	E2	Q
T											
C1											
C2											
L											
E1											
E2											
Q											
Professor M. S. Cheema Department of Mathematics University of Arizona Tucson, AZ 85721		CONTINUE ON SEPARATE SHEET IF NECESSARY. If the paper is in Russian or Chinese, and you consider it to have exceptional merit warranting translation, check here. <input type="checkbox"/> This work has been specially commissioned for inclusion in MATHEMATICAL REVIEWS, or subsequent compilations of reviews, in accordance with the terms of Section 101 of the Copyright Act of 1976. All rights to this review, including copyright, belong to the American Mathematical Society.									
											

Figure 6: Reviewer form

**Mathematical Reviews Item Proof:** Run 1985/10/17/14:10  
Batch (Page: 1) Item:1 Complete

CE	Ed		Co		RP	Fin
----	----	--	----	--	----	-----

Keyed by: AMP

761 401

**99a:99999 05A15**

**Knuth, Donald E.** (1-STF-C)

**The toilet paper problem.**

*Amer. Math. Monthly* 91 (1984), no. 8, 465–470.

The toilet paper dispensers are designed to hold two rolls of tissues, and a person can use either roll. There are two kinds of users. A big chooser always takes a piece from the roll that is currently larger, while a little chooser does the opposite. When the two rolls are the same size, or when only one is nonempty, everybody chooses the nearest nonempty roll. Assume that people enter the toilet stalls independently at random, with probability  $p$  that they are big choosers and probability  $q = 1 - p$  that they are little choosers. If two fresh rolls of toilet paper, both of length  $n$  are installed, let  $M_n(p)$  be the average number of portions left on one roll when the other one first empties. The purpose of this paper is to study the asymptotic value of  $M_n(p)$  for fixed  $p$  as  $n \rightarrow \infty$ . Let  $M(z) = \sum_{n \geq 1} M_n(p)z^n$ , and  $C(z) = \sum_{n \geq 1} c_n z^n$ ,  $c_n = \binom{2n-2}{n-1}/n$  (Catalan numbers) be the generating functions. It is proved that  $M(z) = (z/(1-z)^2)((q - C(pqz))/q)$ . Let  $r$  be any value greater than  $4pq$ ; then  $M_n(p) = p/(p-q) + O(r^n)$  if  $q < p$ ,  $M_n(p) = ((q-p)/q)n + p/(q-p) + O(r^n)$  if  $q > p$ , and  $M_n(\frac{1}{2}) = 2\sqrt{n/\pi} - \frac{1}{4}\sqrt{1/\pi n} + O(n^{-3/2})$ .

*M. S. Cheema* (Tucson, Ariz.)



761 401

Figure 7: Item proof

**Knuth, Donald E.** The toilet paper problem. *Amer. Math. Monthly* 91  
(1984), no. 8, 465–470. **86a:05006** 05A15 (05A10)

Figure 8a: Excerpt from annual author index

**Amer. Math. Monthly** The American Mathematical Monthly. Math.  
Assoc. America, Washington, DC.

Figure 8b: Excerpt from annual serials list

**1-STF-C**  
Department of Computer Science  
Stanford University  
Stanford, CA 94305

Figure 8c: Excerpt from annual institution code list

**Knuth, Donald E.** The toilet paper problem. **86a:05006**

Figure 8d: Excerpt from annual subject index

05A10 Factorials, binomial coefficients, combinatorial functions  
[See also 11B65.]  
05A15 Combinatorial enumeration problems, generating functions

Figure 8e: Excerpt from subject classification scheme

AN- 1541094|  
 AN- <MR Number> 86a#05006|  
 AN- <Paper Number> CMP 761 401|  
 TI- The toilet paper problem.|  
 AU- Knuth, Donald E.  
     (Department of Computer Science, Stanford University,  
     Stanford, 94305, California)|  
 CS- 1-STF-C|  
 JN- Amer. Math. Monthly, 91, no. 8, 465--470.|  
 PY- 1984|  
 SN- 0002-9890|  
 CO- AMMYAE|  
 LA- English|  
 DT- Journal|  
 SF- MR (Mathematical Reviews) AMS|  
 RL- MEDIUM (20 lines)|  
 AB- The toilet paper dispensers are designed to hold two rolls of tissues,  
     and a person can use either roll. There are two kinds of users. A big  
     chooser always takes a piece from the roll that is currently  
     larger, while a little chooser does the opposite. When the two rolls  
     are the same size, or when only one is nonempty, everybody chooses  
     the nearest nonempty roll. Assume that people enter the toilet stalls  
     independently at random, with probability  $p$  that they are big  
     choosers and probability  $q=1-p$  that they are little choosers. If  
     two fresh rolls of toilet paper, both of length  $n$  are installed,  
     let  $M_n(p)$  be the average number of portions left on one roll  
     when the other one first empties. The purpose of this paper is to  
     study the asymptotic value of  $M_n(p)$  for fixed  $p$  as  
 $n \rightarrow \infty$ . Let  $M(z)=\sum_n M_n(p)z^n$ , and  
 $C(z)=\sum_n c_n z^n$ ,  $c_n = \frac{1}{n+1}$  (Catalan numbers) be the generating functions. It is proved that  
 $M(z)=(z/(1-z)^2)((q-C(pz))/q)$ . Let  $r$  be any value greater than  
 $4pq$ ; then  $M_n(p)=p/(p-q)+O(r^n)$  if  $q < p$ ,  $M_n(p)=((q-p)/q)n+p/(q-p)+O(r^n)$  if  $q > p$ , and  $M_n(p)=2\sqrt{\pi}-\frac{1}{\sqrt{\pi}}+\frac{1}{2}n^{-3/2}$ .|  
 RE- <Name> Cheema, M. S.|  
 RE- <Location> (Tucson, Ariz.)|  
 RT- Signed review|  
 DE- \*COMBINATORICS -Classical combinatorial problems --Combinatorial  
     enumeration problems, generating functions (05A15); COMBINATORICS  
     -Classical combinatorial problems --Factorials, binomial coefficients,  
     combinatorial functions (05A10)|

Figure 9a: MathSciTeX input file

1541094	- Dialog Number
86a#05006	- MR Number
CMP 761 401	- Paper Number

*Title:* The toilet paper problem.

*Author:* Knuth, Donald E. (Department of Computer Science, Stanford University, Stanford, 94305, California)

*Corporate Source:* 1-STF-C

*Journal:* Amer. Math. Monthly, 91, no. 8, 465–470. *Year:* 1984 *ISSN* 0002-9890

*CODEN:* AMMYAE

*Language:* English *Document Type:* Journal

*Subfile:* MR (Mathematical Reviews) AMS *Length:* MEDIUM (20 lines)

*Review:*

The toilet paper dispensers are designed to hold two rolls of tissues, and a person can use either roll. There are two kinds of users. A big chooser always takes a piece from the roll that is currently larger, while a little chooser does the opposite. When the two rolls are the same size, or when only one is nonempty, everybody chooses the nearest nonempty roll. Assume that people enter the toilet stalls independently at random, with probability  $p$  that they are big choosers and probability  $q = 1 - p$  that they are little choosers. If two fresh rolls of toilet paper, both of length  $n$  are installed, let  $M_n(p)$  be the average number of portions left on one roll when the other one first empties. The purpose of this paper is to study the asymptotic value of  $M_n(p)$  for fixed  $p$  as  $n \rightarrow \infty$ . Let  $M(z) = \sum_{n \geq 1} M_n(p)z^n$ , and  $C(z) = \sum_{n \geq 1} c_n z^n$ ,  $c_n = \binom{2n-2}{n-1}/n$  (Catalan numbers) be the generating functions. It is proved that  $M(z) = (z/(1-z)^2)((q - C(pqz))/q)$ . Let  $r$  be any value greater than  $4pq$ ; then  $M_n(p) = p/(p-q) + O(r^n)$  if  $q < p$ ,  $M_n(p) = ((q-p)/q)n + p/(q-p) + O(r^n)$  if  $q > p$ , and  $M_n(\frac{1}{2}) = 2\sqrt{n/\pi} - \frac{1}{4}\sqrt{1/\pi n} + O(n^{-3/2})$ .

*Reviewer:* Cheema, M. S. (Tucson, Ariz.)

*Review Type:* Signed review

*Descriptors:*

\*COMBINATORICS -Classical combinatorial problems -Combinatorial enumeration problems, generating functions (05A15); COMBINATORICS -Classical combinatorial problems -Factorials, binomial coefficients, combinatorial functions (05A10)

Figure 9b: MathSciTeX output

# Lexicography with TEX

JÖRGEN L. PIND

Institute of Lexicography  
University of Iceland  
Reykjavík  
Iceland  
[jorgen@lexis.hi.is](mailto:jorgen@lexis.hi.is)

## ABSTRACT

At the Institute of Lexicography at the University of Iceland, TEX is used for the typesetting of dictionaries. Currently we are in the process of bringing out a large etymological dictionary which is typeset in TEX with PostScript fonts. Details of this project are presented. The value of generic or logical coding over typographical coding is emphasized.

### 1. Background

In this paper I will discuss the use of TEX in the work carried out at the Institute of Lexicography of the University of Iceland. The Institute was founded in 1948 and has as its major aim the production of an historical dictionary of Icelandic from 1540 (when the first printed book appeared in Icelandic) up to the present, a dictionary somewhat along the lines of the Oxford English Dictionary.

During the past forty years, a lot of material has been gathered for the dictionary. The main collection of the Institute comprises some 2.5 million dictionary slips; others include, for instance a collection of words from the spoken language. These other collections contain perhaps 300,000 slips in all. Near the end of 1982, it was decided to begin evaluating the collection with the aim of publishing an historical dictionary of the language. At the same time it was decided to embark on computerizing the Institute itself.

The first computational project involved registering the main collection so as to open more paths into the collection itself. A database of all the words contained in the collection was set up. The word class, date of oldest and newest citation, the oldest source, number of citations kept in the collection and the word type (whether the word is a compound, an affixed word or a 'simple' word) were registered for each word. This database contains a total of just over 600,000 words. This is a surprisingly high figure but is explained in part by word-compounding, which is an active process in the Germanic languages, not the least in Icelandic.

In some respects, this database file can be viewed as a first approximation to a dictionary although a very primitive one, since it does not have any grammatical analysis to speak of. Yet, because the material is stored in a database (as opposed to a linear alphabetized order), it does enable us to escape from the "tyranny of the alphabet" and gives us multiple access paths to the collections of the Institute.

The editing of historical dictionaries has usually proceeded in alphabetical order, the work being brought out in installments over a period of decades. This is an approach which is in many respects less than ideal since the editor is forced to deal with words which do not form a coherent set under any reasonable linguistic criterion. We would therefore like to proceed in a different manner, dealing with individual word classes at a time. The availability of the computer makes this relatively easy to accomplish. It has now been decided by the governing board of the Institute that the editing work will concentrate on the verbs with the aim of producing an historical dictionary of verbs as the first volumes of what will hopefully later become a comprehensive historical dictionary of Icelandic.

This work was begun in 1985. The editorial strategy involves some novelties compared with traditional methods (e.g., Kuhn 1982), in that each citation is furnished with a set of editorial descriptors detailing the grammatical and semantic features of the citation itself. This is done on-line with the

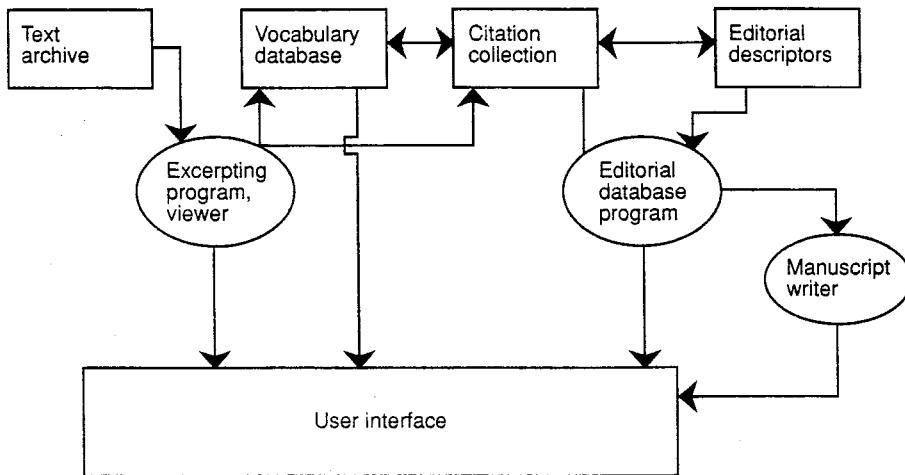


Figure 1: A model for a “lexicographer’s workbench”

description being stored in a database system. The database system is then used to make  $\text{\TeX}$ -encoded scripts which  $\text{\TeX}$  then changes into beautifully typeset pages.

In 1986, in a talk presented at the NordData Conference in Stockholm, I outlined the approach we were using and illustrated it with a figure of a “lexicographer’s workbench” (Figure 1), commenting that a number of features had not been implemented. “This holds especially for the ‘manuscript writer’. Our work has not yet reached the stage where this is in great demand, but we envisage, for example, the possibility of using the database to turn out manuscripts for a typesetting program like  $\text{\TeX}$ ” (Pind 1986:87).

Well, this was written before we even had a version of  $\text{\TeX}$  running at the Institute! As a matter of fact, though we expected that typesetting would be something that we would deal with much later, a lot of work over the past couple of years has been devoted to the typesetting side of lexicography. There are two reasons for this. The first is that the lexicographer very much wants to be able to print proofs from the lexicographic database that show some resemblance to a traditional dictionary. The second is the fact that we have been engaged in producing an Icelandic etymological dictionary working from the author’s manuscript. This project will be described in detail below.

## 2. Icelandic $\text{\TeX}$

In October 1986 I first acquired  $\text{\TeX}$ . Unfortunately it was not possible for me at that time to work with Icelandic in  $\text{\TeX}$  since a number of characters were missing from the Computer Modern fonts which are needed in Icelandic, such as *thorn* and *eth* (`\char'034` and `\char'037` in Figure 2). Additionally, Icelandic has a number of accented characters and, as is well known,  $\text{\TeX}$  will not hyphenate words which contain floating accents. In January 1987, however, I acquired Doug Henderson’s METAFONT for MS-DOS and this enabled me to get started on making Icelandic versions of the Computer Modern fonts. The first version was limited to 128 characters, due to limitations in the drivers then available. The special Icelandic characters are accessed as ligatures, as recommended by Knuth (1984:46). An Icelandic hyphenation table was made using Frank Liang’s PATGEN-program. This table has turned out to perform excellently. A number of changes have also been made to the plain and  $\text{L}\text{\TeX}$  macros to accommodate Icelandic. The development of Icelandic  $\text{\TeX}$  was originally carried out on an IBM PC/AT. In early 1988 we switched over to AIX on an IBM PC/RT and got Rick Simpson’s excellent port of  $\text{\TeX}$  and METAFONT to that machine. This has since been the platform on which we have operated.

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	Γ	Δ	Θ	Λ	Ξ	Π	Σ	Υ	
'01x	Φ	Ψ	Ω	á	é	í	ó	ú	"0x
'02x	ı	J	~	'	-	~	-	°	
'03x	~	ý	æ	ö	þ	Æ	P	ð	"1x
'04x	Ð	!	„	#	\$	%	&	,	"2x
'05x	(	)	*	+	,	-	.	/	
'06x	0	1	2	3	4	5	6	7	"3x
'07x	8	9	:	;	ø	=	Ø	?	
'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	
'13x	X	Y	Z	[	“	】	^	·	"5x
'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	
'17x	x	y	z	-	—	”	~	..	"7x
	"8	"9	"A	"B	"C	"D	"E	"F	

Figure 2: The Icelandic Font imr10

### 3. The Making of an Etymological Dictionary

I turn now to a discussion of the making of one particular dictionary, a 1250-page etymological dictionary of Icelandic which will appear later this year. The etymological dictionary is the work of one man, the late Ásgeir Blöndal Magnússon, who worked at the Institute for over forty years. When keyboarding of the text began in 1985, it was expected that the dictionary would eventually be typeset by a professional printer. Indeed, at that time we did not even have *TeX* at the Institute as mentioned above. However, I acquired the *TeXbook* early in 1985 and *TeX* did influence the keyboarding of the manuscript.

We were immediately confronted with the diverse floating accents which any etymological dictionary contains and I decided that we would adopt the *TeX* coding scheme as our model for the keyboarding. I now have some doubts about the suitability of this scheme as I will elaborate on later. However, it must be emphasized that I never expected that in fact the dictionary would end up being typeset with *TeX*.

The dictionary was keyboarded directly from the author's handwritten manuscript (having been collected on slips of the traditional kind so loved by lexicographers before the advent of computing). The PC-Write editor was used to input the manuscript, since it uses near ASCII-files and is easily configured. It was limited to 60K files but this did not cause any trouble. When the whole manuscript had been input, it amounted to 151 files containing just over 7Mb of text.

Proofreading and checking the manuscript turned out to be a major task, even more so since the author died in 1987. In December 1988, it became clear that it would be possible to publish the dictionary this year and arrangements were made with the largest printing house in Iceland to take care of the typesetting and printing. The typesetting was to be done on a Linotronic 300. At that time it turned out, however, that the typesetting process would be difficult since a lot of accents were missing from the fonts which were available. Some of these could be ordered from Linotype, others had to be made specifically at what we felt was an exorbitant price. I think I can fairly say that the printers were none too happy with the prospect of typesetting this massive book.

At this point, I decided that I would have a go at typesetting the dictionary myself with *TeX*. This fitted also very well into our overall strategy since we had decided that we would use *TeX* in the

future as our typesetting engine and had, as a matter of fact, already made some experiments with the dictionary of verbs. Using the etymological dictionary as the first major test-case was of course in some respects ideal since if we could accomplish *that*, we felt we could cope with any dictionary. I should point out that this is by no means the first Icelandic book typeset with  $\text{\TeX}$ . The first one was actually a book about the Macintosh personal computer written by the present author (Pind 1987) — my apologies for not having chosen a weightier subject for the occasion! A number of other books have appeared in Icelandic typeset with  $\text{\TeX}$ , with more on their way. The dictionary is, however, by far the most ambitious and also the first (Icelandic) book of its kind typeset with  $\text{\TeX}$ .

## 4. Some Details

Figure 3 shows a page from the dictionary printed on the Linotronic 300.

### 4.1 The Dictionary Entry

A typical article from this page is the one for *áðan*, meaning ‘just now’. This is coded as follows:

```
\hword{áðan} ao. ‘fyrir skömmu’; \shword{áður},
\dag\shword{áðr} ao. ‘fyrr’. Sbr. fær. \wform{áðan(i)},
\wform{áður}, nno. \wform{\aa{}dan}, \wform{\aa{}der}, fd.
\wform{adens}, fsæ. \wform{\apans}, nsæ. \wform{ij\aa{}ns\}/};
sk. fe. \wform{\={æ}dre}, fsax. \wform{\=adro\}/} ‘undir
eins’, fhp. \wform{\=atar\}/} ‘fljótur, skilningsskarpur’; likl.
einnig í ætt við lettn. \wform{\~atrs} ‘bráður, fljótur til’
og lith. \wform{otr\‘us\}/} ‘ákafur’.
```

This, admittedly, doesn’t look particularly nice, but the output from the Linotronic sure does and that is what counts. Each article starts with a headword which is given by the  $\hword$  macro. Other categories shown in the extract are  $\shword$  which identifies a ‘subsidiary headword’ and  $\wform$  which identifies a word, either one from a different language or one cross-referenced in the dictionary.

As can be seen in Figure 3, it is often the case that there are multiple meanings for one word, each one entered as a separate headword. These are distinguished by a decimal number in front of the word itself. These numbers are given as an optional parameter (enclosed in square brackets) for the  $\hword$  macro which is defined as follows:

```
\def\hword{\futurelet\PossBracket\hwordbranch}
\def\hwordbranch{\ifx\PossBracket [%
  \let\next=\hwordwithno
  \else
  \let\next=\hwordwithoutno
  \fi
  \next
]
\def\hwordwithno[#1]#2{%
  {\leavevmode\hbox to 10pt{}\bf#1 #2\mark{#2}}}
\def\hwordwithoutno#1{%
  {\leavevmode\hbox to 10pt{}\bf#1\mark{#1}}}
```

We use  $\futurelet$  to check for the presence of a bracket. If it is present, the macro  $\hwordwithno$  is executed, otherwise the macro  $\hwordwithoutno$  is used. Note that the indent is specified with an  $\hbox$ . Since this occurs at the very beginning of a paragraph, it is necessary to leave the vertical mode explicitly, using  $\leavevmode$ . The  $\text{\TeX}$  primitive  $\mark$  enables us to automate the typesetting of the headwords at the top of each page, which shows the range of entries on a particular page. For this dictionary, which is set in two-column format, a version of Knuth’s double-column output routine from Appendix E of *The TeXbook* has been used. These output macros make use of  $\vsplit$  to divide the page into two columns. The  $\headline$  macro is as follows:

```
\def\headline{\hbox to \pagewidth{%
  \tenbf\strut\hbox to 14pc{\firstmark\hfil}\%
  \hfill\folio\hfill\hbox to 14pc{\hfil\splitbotmark}}}
```

klausturs'; sbr. fær. *abbatī*, fsæ. *ab(b)ot(e)*, d. *abbed*. To. < lat. *abbas* (pf. *abbatem*) < gr. *abbas* < sýrl. *abbā* ‘faðir, munkur’. Sjá *abbadís*, en bæði þessi to. hafa borist inn í íslenskt mál með kristninni.

**Abraham** k. karlmannsnafn, komið úr hebr., eiginl. merking ‘faðir fjöldans’.

**ábrúðig(u)r** l. † ‘afbrýðisamur’; **þábrýði** h., kv. ‘afbrýði’, sbr. *afbrúðig(u)r*, *afbrýði*, *þafbrýgði* (v.l.). Sbr. nno. *ábruig* l., *ábry* h. (s.m.). Ekki er öruggt hvort hér er upphaflegra á- eða af-forskeyti; ef á- er eldra mætti vitna til fe. *onbregðan* ‘bregða hart við, hrökkva upp’ (en on- í fe. getur líka svarað til *and*-). Sé af- eldra ættu orðin að vera mynduð af so. \**abbregðan* eða *bregða af* ‘breyta um’ e.p.h., sbr. *afbragð*, *afbrugðinn*. Upphaflegt g í síðara lið orðsins hefur fallið niður og valdið uppþótarlengingu, -*brugðigr* > -*brúðig(u)r*, -*brygði* > -*brýði*. Sjá *bregða*.

**ábrystir, árestir, ábristir, ábre(i)stur, ábrystur** kv.ft. (17. öld) ‘sérstakur réttur gerður úr (hit-aðri) broddmjólk’. Orðið er líkl. dregið af að *bresta* eða *brysta* (*brista*) um það er mjólkurhlaupið tók að bunga upp í miðju og springa; sbr. nno. *bresta*, d. *briste* ‘skiljast (um mjólk)’. Óvist er hvort rita skal orðið með e eða i, en frábrigðilegar myndir þess gætu bent til gamallar u-st. beygingar: -*brestr*: *bristir*, *brestu* (nf. og pf. ft.). Aðrir telja að forliður orðsins á- sé s.o. og ær kv. ‘sauðkind’, en síðari liðurinn -*brystir* eigi skylt við ísl. *broddur* ‘broddmjólk’ og þ. máll. *briestermilch* (s.m.) (sem er raunar ummyndun úr *biest(milch)*). Vafasamt. Sjá *bresta* og *brystingur*.

**Absalon** k. karlmannsnafn, bíblíunafn ættæd úr hebr. *Abshalom*, eiginl. merking ‘guð faðir er friður og velsæld’.

**ábyrgur** l. ‘sem ber að gæta e-s, svara fyrir e-ð’; sbr. fær. *ábyrgur* (s.m.). Af sama toga eru so. **ábyrgjast** ‘tryggja, svara fyrir’, sbr. fær. *ábyrgjast* og nno. *ábyrgjast* (s.m.), og no. **ábyrgð** kv. ‘trygging, ...’, sbr. fær. *ábyrgd* kv. (s.m.). Sk. ísl. *bjarga* og *borga*, fe. *borgian* ‘ljá, fá að láni’, *borg* ‘trygging, ...’, fhþ. *borgēn*, *por(a)kēn* ‘tryggja gegn, fela til varðveislu’, sbr. nhþ. *borgen* ‘ljá’, ne. *borrow* ‘fá að láni’. Lo. *ábyrgur* er e.t.v. leitt af týndri forskeytri so. \**anburgian*, sbr. fe. *onbyrgan* ‘tryggja, ábyrgjast’.

**1 að, þat** fs. (ao.); sbr. fær. *at*, no. *ad*, nno. *át*, sæ. *át*, d. *ad*, fe. *æt*, fhþ. *az*, gotn. *at*, lat. *ad*; aðalmerkingin virðist vera ‘í áttina til’ e.p.u.l. Stundum talið sk. fír. *ad* ‘lög, siðvenja’ < \**ado*- ‘markmið’, af ie. rót \**ad*- ‘ákveða, tiltaka (sem markmið)’. E.t.v. eru ísl. *til* (fs.) og -*tili* k. af sama toga (s.p.).

**2 að, þat** forskeyti; sbr. fe. *æt*, fhþ. *az*, gotn. *at*-, fír. *ad*-, lat. *ad*- og ísl. *að* fs. Oftast í eiginlegri merkingu ‘til, hjá’, sbr. *aðfall*, *aðförf*, *aðsókn*; í sumum tilvikum helst hin forna mynd forskeytisins *at* í nýmálinu, sbr. *athvarf*, *athöfn*, *atlot*, *atriði* o.s.frv.

Oft er erfitt að skera úr hvort um gamalt forskeyti eða síðar forskeytta fs. er að ræða. Sjá *að* (1).

**3 að, þat** nhm.; sbr. d. *at*, sæ. *át*, sama orð og fs. *að* (*at*), sbr. að forsetningar sömu merkingar í e. og þ., to og zu, eru einnig notaðar sem nhm.

**4 að, þat** st.; sbr. fær. *at*, nno., d. *at*, sæ. *att*; upphaflega sama orð og fn. *það*, *þat*, upphafs-þ-ð hefur fallið burt; sbr. e. *that* og þ. *das(s)* (fn. og st.); (\**ek veit þat*, *hann kómr* > *ek veit (þ)at hann kómr*).

**-að** viðsk. í no. *unað* h. (s.p.) < germ. \*-aiða- eða \*-éða-, þ.e. afleiðsla með viðsk. \*-ða- < ie. \*-to- af aí/é-sögn. Óvist er hvort um sama viðsk. er að ræða í *volað* og *forað* (s.p.). Örugglega annar uppruni í *hérað* (s.p.).

**aða, öðuskel** kv. (17. öld) ‘skel af kræklingsætt (modiola modiolus)’; sk. nísl. *öðlingur* ‘sælindýr af kræklingsætt (modiola phaseolina)’ og fær. *þða*, nno. *odskel* ‘öðuskel’ og fær. *þðulingur* ‘kræklingur (modiolaria nigra)’; *aða* er e.t.v. í ætt við *eðja* og nafngiftin af því dregin að skeldýr þessi fundust helst á sandleirum, sbr. *að sitja eins og aða í leiru*, sbr. og so. *aðast*. Önnur skyld orð væru þá d. *ajle* ‘for’, mlþ. *ad(d)e*, fe. *adul* ‘óhreinindi’. Ættartengsl þessarar orðsiftar eru að öðru leyti óljós. Sjá *aðast*, *eðja* og *öðlingur* (2); (*aða* e.t.v. stytting úr *öðuskel*).

**1 aðal** h. ‘eðlisfar, höfuðauðkenni’, sbr. forliðinn *aðal-* (2), *eðli*, *öðlast*. Sjá *aðall*.

**2 aðal** forl. ‘höfuð-, megin-’, sbr. *aðalstarf*, *að-alatriði*. Sjá *aðal* (1) og *aðall*. *Aðal-* kemur og fyrir sem forliður mannanafna, sbr. *Aðalbjörn*, *Aðalgeir*, *Aðalbjörg* o.fl. og á þá líkl. við ættgöfgi. Sjá *aðall* og *eðal-* og *Al-* (2) í pn.

**aðall** k. ‘yfirstétt (einkum í lénsþjóðfélagi); eðli eða höfuðeinkenni; meginhluti e-s’; sbr. d. *adel*, fe. *ædel* ‘tigínborinn’, *ædele* ‘göfugt ætterni’, fhþ. *adal* ‘ætt, ættgöfug fjölskylda’, gotn. *apala-* í pn. *Athalaricus*. Orðsiftin sýnist leidd af ie. \**ato*- ‘foreldri, faðir’, sbr. gotn. *atta*, fsl. *otici* ‘faðir’, og merkja í öndverðu arfleifð frá ættfeðrum eða -mæðrum. Önnur orð af sama stofni eru *aðili*, *eðli*, *óða*, *óðal*, *öðlast* og *öðlingur* (1). Merkingin ‘yfirstétt eða lénsaðall’ í norr. málum er fengin að láni úr þ.

**Adam, Ádam** k. karlmannsnafn úr hebr. *Ādām* ‘maður’. Aðrir telja að nafnið merki ‘hinn rauðleiti’.

**ádamant, adamas, adímas, átímas** k. (um 1500) ‘gimsteinn’. Sjá *demandur*.

**áðan** ao. ‘fyrir skómmu’; **áður, þáðr** ao. ‘fyrr’. Sbr. fær. *áðan(i)*, *áður*, nno. *áðan*, *áðder*, fd. *adens*, fsæ. *apans*, nsæ. *ijáns*; sk. fe. *ædre*, fsax. *ádro* ‘undir eins’, fhþ. *átar* ‘fljótur, skilningsskarpur’; líkl. einnig í ætt við lettn. *átrs* ‘bráður, fljótur til’ og lith. *otrùs* ‘ákafur’.

**aðast** s. ‘hreyfast hægt, mjakast áfram’; e.t.v. sk. *aða* og *eðja* og þá tekið mið af hægfara straumi eða

Figure 3: A sample page from the etymological dictionary

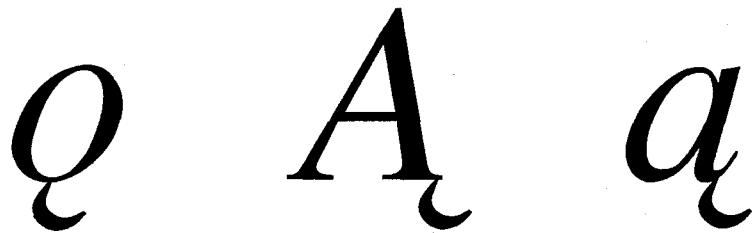


Figure 4: The placement of the ogonek accent under different letters

The `\firstmark` token marks the first headword on each page and the `\splitbotmark` marks the last headword on the page.

#### 4.2 Accents

In an etymological dictionary, there are  $n$  different accents which need to be taken care of.  $\text{\TeX}$  has an `\accent` primitive which positions accents over letters. Unfortunately the `\accent` primitive is limited to putting one accent over a letter. Often there is a need to put two accents over a single letter. For this it is necessary to write special macros.

One subtlety which  $\text{\TeX}$  does not address directly is the fact that it is not always possible to position accents without knowing what letter it is put over. This holds, for example, for the acute accent over a k. Ordinarily, the accent primitive works fine for positioning an acute accent over letters (see e.g., á, š), but when it comes to the k, the accent should not be positioned as in k̄, but rather as in K̄. A similar issue arises in regard to the positioning of the ogonek accent under letters. Thus it is usually placed under the middle of an o but at the right serif of the A and a (Figure 4).

These facts bring up the question of coding. The character set of the Computer Modern fonts is really quite limited since it only uses 128 character positions out of the 256 possible. Adobe PostScript fonts have many more characters (around 300) and the Bitstream fonts even more. In particular, many accented characters are part of the standard Adobe and Bitstream fonts and it would seem natural to use those. This makes it necessary to define the characters in a manner similar to that adopted by Knuth for the mathematical symbols. Thus we would have control sequences such as `\aacute`, `\oogonek`, `\ubreve`, etc. In fact, in many ways this is probably a better choice than using the accent coding. In our case, for instance, where we want to be able to use both the Adobe fonts and Knuth's Computer Modern — changing between them with a simple switch — a fragment of our font coding macros runs on along the following lines:

```
\newif\ifcmfonts
...
\ifcmfonts
  \message{*** Computer Modern fonts used ***}
  \font\tenrm=imr10 % im... Icelandic versions of Computer Modern
  \font\ninerm=imr9
  \font\eightrm=imr8
...
  \def\aa{\accent23a}
  \def\L{\leavevmode\setbox0\hbox{L}\hbox to\wd0{\hss\char32L}}
...
\else
  \message{*** PostScript fonts used ***}
  \font\tenrm=Timesro at 10pt
  \font\ninerm=Timesro at 9pt
```

```

\font\eightrm=Timesro at 8pt
...
\def\aa{\char7} % Yes, our encoding is a bit peculiar!
\def\L{\char3}
...
\fi
```

This makes it possible for us to define the characters without regard for the particular fonts we are using. Here a generic or logical approach to the coding of characters is adopted rather than the typographically-oriented coding of the *TeXbook*. The TeX coding scheme is natural for use with the Computer Modern fonts. When extending TeX to use other fonts it becomes less than ideal and thus it is appropriate to adopt another coding scheme which is not font-based.<sup>1</sup>

At one point I attempted to use the Bitstream fonts, but they turned out to be useless for our purposes since the implementation of the Bitstream-to-Tex conversion program from Personal TeX makes it impossible to freely arrange characters in the fonts. Thus it is not possible to use the Icelandic hyphenation table if using the Bitstream fonts.

Sometimes the plain macros are not up to the typesetting of floating accents. There is no TeX primitive which puts accents underneath letters. Knuth has defined some macros in plain which are used for this. To put a dot underneath a letter the macro \d is used and the macro \b is used to put a bar underneath a letter. The former macro is defined in plain.tex as:

```
\def\d#1{\oalign{\#1\crcr\hidewidth.\hidewidth}}
```

These macros work nicely for putting a dot and a bar underneath straight letters but are not adequate for *italic letters*. Re-definition of these macros along the following lines makes it possible to use these macros both for straight letters and *italic letters*.

```

\def\d#1{\ifnum\fam=\itfam
  \oalign{\#1\crcr\hidewidth\kern-0.1em.\hidewidth}%
\else
  \oalign{\#1\crcr\hidewidth.\hidewidth}\fi}
\def\b#1{\ifnum\fam=\itfam\oalign{\#1\crcr\hidewidth\hidewidth}%
  \kern-.3em\vbox to .2ex{\hbox{\char22}\vss}\hidewidth}%
\else\oalign{\#1\crcr\hidewidth\hidewidth}%
  \vbox to .2ex{\hbox{\char22}\vss}\hidewidth\fi}
```

These macros only work for the italic family. If they are to be extended to slanted letters, it will be necessary to introduce yet another conditional testing for membership in the \slfam.

TeX is not able to put two accents over a single letter. These are quite frequent in the etymological dictionary and, unfortunately, always occur in the italic font. After having tried some fiddling around with kerns and such things, which did not produce what I felt were adequate results, I looked at the definition of the \accent primitive in the listing for the TeX program (Knuth 1986:462–463). The positioning of accents is “straightforward but tedious” according to Knuth. And further:

Given an accent of width  $a$ , designed for characters of height  $x$  and slant  $s$ ; and given a character of width  $w$ , height  $h$ , and slant  $t$ : We will shift the accent down by  $x - h$ , and we will insert kern nodes that have the effect of centering the accent over the character and shifting the accent to the right by  $\delta = \frac{1}{2}(w - a) + h \cdot t - x \cdot s$ .

Well, I thought, this is what I need for the positioning of double accents. And so I decided to implement this in a TeX macro called \dblacc. The idea is to first put one accent over a letter and then use Knuth’s formula as if it were a single character needing one accent. The macro \dblacc needs to play with a number of variables. The names of these variables are similar to the ones Knuth uses in the equation above:

```

\newdimen\xheight % the accents are designed for the x-height
{\it\xheight=\fontdimen5\the\font} % x-height for the italic font
\newdimen\Shift \newdimen\A
```

---

<sup>1</sup> I should note that I have not implemented this scheme completely and letters which can only be set with a floating accent (e.g., doubly accented letters) are still coded with an accent-based coding.

```

\newdimen\X \newdimen\W \newdimen\H \newdimen\HT
\newdimen\XS \newcount\slant \newdimen\lk \newdimen\rk

The macro itself is defined as follows:

\def\dblacc#1#2#3{\leavevmode\setbox1=\hbox{#2#3}%
%   #1 topmost accent, #2 first acc, #3 character
\H=\ht1\W=\wd1\setbox0=\hbox{#1}\A=\wd0
\ifnum\fam=\itfam\slant=4
  \HT=\H\divide\HT by \slant
  \XS=\xheight\divide \XS by \slant
\else\slant=0
  \HT=\H\multiply\HT by \slant
  \XS=\xheight\multiply\XS by \slant
\fi
\lk=\W\advance\lk by -\A\divide\lk by 2\advance \lk by \HT
\advance \lk by -\XS\rk=\A\advance\rk by \lk
\Shift=\xheight\advance\Shift by -\H
\kern\lk\lower\Shift\hbox{#1}\kern-\rk\unhbox1\relax}

```

*T<sub>E</sub>X* is only able to handle integer arithmetic. In the *T<sub>E</sub>X* program, the slant parameter of the font is used to position the accent. Here a brute force approach is used and the *\slant* is set to 4, which is then used in division to equal multiplication by 0.25, which is the value of the slant parameter both in *cmti9* and *Times-Italic*.

Now, I must admit that this does look rather complicated and I felt that a simpler method could be found. I was aware of Peter Olivier's macros for setting double accents (Olivier 1988) but these do not work for the italic fonts. After having made the *\dblacc* macro, I saw Christina Thiele's macro *\diatop* (Thiele 1987). This macro does a pretty good job but does not assign correct width to the overall construction for doubly-accented letters and so is not suitable for running text (this could probably be easily fixed). However, I did run a small experiment timing the *\dblacc* and *\diatop* macros. After ascertaining that the former runs faster I sort of lost interest in redefining the macro! Anyway, the *\dblacc* macro has performed pretty well in this project and so there has not been a pressing need to change to something which perhaps is somewhat simpler.

### 4.3 PostScript

The book is typeset with *T<sub>E</sub>X* using PostScript to drive the typesetter. This approach was taken so that it would be possible to get high-resolution typesetting on the Linotronic 300. Using PostScript also demanded the use of Adobe typefaces since we did not have the Computer Modern faces for use with PostScript at the high resolution offered by the Linotype machine. Even so I doubt that we would have used the Computer Modern faces since they are not particularly well suited to typesetting in narrow columns. The lowercase alphabet length of *cmr9* is 118.0124 pt while the corresponding figure for *Times-Roman* at 9 points is 107.48698 pt (the dictionary is set using fonts at 9 pt). For the bold fonts there is an even greater difference in that *cmbx9* has a lowercase alphabet length of 136.1019 pt while *Times-Bold* is 114.50696 pt.

This difference of length shows up clearly in the number of *Overfull \hbox* messages when typesetting with these fonts. *T<sub>E</sub>X* has a parameter called *\tolerance* which enables the user to specify how much glue is allowed to stretch and shrink between words. Normally plain *T<sub>E</sub>X* sets *\tolerance* equal to 200 but this, as pointed out by Knuth (1984:28–29, 96), is much too strict for narrow column setting.

The following table shows some experimental runs with *\tolerance* equal to 500, 1000 and 5000 using either Adobe fonts (*Times-Roman*, *Times-Italic*, and *Times-Bold*) at 9 points or Computer Modern fonts (*cmr9*, *cmbx9* and *cmti9*). These experimental runs were done on the articles comprising the letter *s* totaling 4,318 paragraphs and running to 240 pages. It is evident that the number of over- and underfull boxes is dependent on both the fonts used and the setting of the *\tolerance* parameter. From these experiments it was decided to set the *\tolerance* to 1000 during the processing of the book.

Fonts	Box type	\tolerance		
		500	1000	5000
Adobe Times	overfull	397	309	22
	underfull	1	2	177
Computer Modern	overfull	1037	463	71
	underfull	3	3	421

Another approach, which I feel would be worth trying, would be to make a special version of Computer Modern designed for narrow settings. Knuth (1989) has recently given a fascinating example of the way he changed the parameters of the Computer Modern fonts for the Concrete fonts. Something similar can no doubt be done to make the fonts suitable for narrow columns.

The typesetting process used standard  $\text{\TeX}$  (with one exception) running with an Icelandic hyphenation table. The change from standard  $\text{\TeX}$  relates to the hyphenation where one change was made to the  $\text{\TeX}$  code. In section 902 of the program listing (1986:380), Knuth declares that “ $\text{\TeX}$  will never insert a hyphen that has fewer than two letters before it or fewer than three after it”. This is less than ideal for Icelandic where it is very common to hyphenate before the second last letter of a word. So slight changes were made to the code to accomplish this. Otherwise, the  $\text{\TeX}$  program is standard. In particular, I don’t think using Multilingual  $\text{\TeX}$  would have been to any advantage in this case since dozens of languages and dialects are referenced in the dictionary, many of them extinct and no doubt getting hold of hyphenation patterns for these would have been pretty difficult!<sup>2</sup> Correcting the overfull boxes was therefore done by hand. This, however, did not turn out to be a particularly onerous task.

Using PostScript with  $\text{\TeX}$  is really quite straightforward. It is of course necessary to supply the requisite  $\text{tfm}$  files. These can be rather easily generated from the AFM files provided with the Adobe fonts. For this I used the  $\text{aftopl}$  program on the UNIX  $\text{\TeX}$  distribution and changed it so that it would recognize the font encoding I had adopted. The  $\text{aftopl}$  program makes  $\text{pl}$  files which can then be changed to  $\text{tfm}$  files with the  $\text{pltotf}$  program. I have used ArborText’s  $\text{dvips}$  driver to generate the PostScript code from the  $\text{dvi}$  files. This has all worked quite satisfactorily.

The only thing which leaves something to be desired is the possibility for previewing the typeset pages. The IBM RT has a screen with a resolution of 118 points to the inch. It has an excellent and very fast previewer, enabling the user to jump to any page in a 100-page section of the dictionary almost instantaneously. No standard Adobe files exist for this resolution. So I tried making some up using the Bitstream fonts, e.g., Dutch for Times-Roman. It turned out of course that one foundry’s Times-Roman is not another’s. The widths of the characters are not comparable so what should be a nicely justified text comes out quite ragged on the screen. This should come as no surprise and was presumably one of the main motives behind Knuth’s development of the Computer Modern family, namely the need to generate a consistent set of fonts for use on devices with very different resolutions.

However, though I see a need for bringing up a set of correct PostScript screen fonts, the need is not pressing since the preview is only used to check for widow lines, overfull boxes and such things.

Though PostScript has a reasonable character set, some characters are missing which are needed in this project. These I have made up using the Fontographer font editor, a PostScript font editor running on a Macintosh. Fontographer is quite different from METAFONT. A character is made by drawing curves and lines on the Macintosh screen. Fontographer uses Bézier curves like METAFONT but it has no understanding of “meta-ness”, so each character has to be drawn on its own with the user attending to the overall aspects of the design. The output of Fontographer is a PostScript file which can either be downloaded to the printer or prepended to the PostScript file containing the text of the dictionary itself. Actually, only the latter approach seems to work on the Linotronic.

All things considered, I would like to stress that using PostScript fonts with  $\text{\TeX}$  has turned out to be much easier than I had at first imagined. It is of course true that PostScript has some limitations in that it always operates from a single design size. This leads for instance to small caps letters which are obviously of a lower quality than a specially designed small caps font. Also, the quality of the

---

<sup>2</sup> With a purely bilingual dictionary the advantages of using Multilingual  $\text{\TeX}$  are obvious.

letters at small point sizes leaves something to be desired, but this is not a problem for the setting of a dictionary which almost exclusively uses 9pt fonts. Those setting mathematics are of course aware of the limitations of the PostScript fonts for mathematics.

## 5. Other Projects

I have already mentioned the dictionary of verbs which will be the major work undertaken over the next few years. Considerable time has been spent on the database side of this project (which is now being ported from MS-DOS to UNIX), and also on the typesetting aspects. We have also embarked on a study of older Icelandic dictionaries. Some of these will be republished by the Institute, freshly typeset using *T<sub>E</sub>X*. The first three volumes are now underway: an Icelandic-Latin dictionary from 1683, an Icelandic-Danish-Latin dictionary from 1814, and a Danish-Icelandic dictionary from 1819.

The latter dictionary shows off some of *T<sub>E</sub>X*'s capabilities quite nicely. The dictionary is Danish-Icelandic, although we are primarily interested in the Icelandic vocabulary. A list of all the Icelandic words (with reference to the appropriate headword) will be included with the book. *T<sub>E</sub>X* automatically writes these words (which have been specifically marked in the dictionary) to a file and a special program then takes care of sorting and merging these entries which are then input to *T<sub>E</sub>X* again for typesetting.

## 6. Some Lessons

I don't think it will be necessary to explain to this audience why we have found *T<sub>E</sub>X* to be eminently suitable for lexicographic work. The typesetting is unquestionably of the highest order. Our experiences with *T<sub>E</sub>X* over the last couple of years have taught us many lessons. The most important of these is perhaps the following: When coding a manuscript, always code it at the most abstract level possible. This was not our approach when we embarked on the etymological dictionary. This was partly due to the fact that we were preparing a file for a typesetter. The virtues of logical or generic coding are many, as pointed out by Lampert (1988) for example, and Knuth (1989:31-32) has an interesting example of this relating to the different use of text numerals and mathematical numerals. The value of logical coding is apparent in the making of dictionaries where we are dealing with text which is relatively highly structured. By using logical coding it is relatively straightforward to use the same manuscript for typesetting as for input to a database system. This of course, is one of the ideas behind the SGML standard. *T<sub>E</sub>X* by itself does not force any particular style of coding on the user, but it does enable the use of generic coding, and I feel that this should be used to the fullest extent possible, especially perhaps in lexicographic work, where it would considerably ease the process of putting printed dictionaries on-line (Alshawi, Boguraev and Carter 1989). Of course, when it comes to the actual, final typesetting, it is not possible to completely bypass typographical coding. Thus, in order to get rid of widow lines and other such typographical blemishes, it is necessary to use purely typographic command such as *\looseness*. These commands are, however, few and can be easily isolated.

One advantage of this approach is that by relatively simple re-definitions of macros, it is possible to print completely different proofs of the same text, with cross-references or grammatical information highlighted in special ways. This has been tried and found to be highly useful.

This approach has now been consistently adopted for other works now being coded in *T<sub>E</sub>X* at the Institute. This holds for the historical dictionary of verbs mentioned earlier, as well as the series of reprints of older dictionaries. My aim, through these diverse types of dictionaries, is to produce a reasonably comprehensive macro package for the typesetting of dictionaries, a package which will enable the user to describe the logical structure of the text while *T<sub>E</sub>X* takes care of the formatting.

## Bibliography

- Alshawi, Hiyan, Bran Boguraev, and David Carter. "Placing the Dictionary On-Line." Pp. 41-63 in *Computational Lexicography for Natural Language Processing*. Bran Boguraev and Ted Briscoe, eds. London: Longman, 1989.
- Knuth, Donald E. *The T<sub>E</sub>Xbook*. Reading, Mass.: Addison-Wesley, 1984.
- Knuth, Donald E. *T<sub>E</sub>X: The Program*. *Computers and Typesetting*, Vol. B. Reading, Mass.: Addison-Wesley, 1986.

- Knuth, Donald E. "Typesetting Concrete Mathematics." *TUGboat* 10:31–36, 1988.
- Kuhn, Sherman. "On the Making of the Middle English Dictionary." *Dictionaries: Journal of the Dictionary Society of North America* 4:14–41, 1982.
- Lamport, Leslie. "Document Production: Visual or Logical." *TUGboat* 9:8–10, 1988.
- Olivier, Peter J. "Publishing 'Exotic' Documents with EXOTEX, A New Macro Package." Paper presented at *TeX88*, Exeter University, July 1988.
- Pind, Jörgen, "The Computer Meets the Historical Dictionary." *Nordisk DATAnytt* 16(10):41–43, 1986.
- Pind, Jörgen. *Bókin um Macintosh*. Reykjavík: Mál og menning, 1987.
- Thiele, Christina. "TeX, Linguistics, and Journal Production." Pp. 5–26 in *Conference Proceedings, Eighth Annual Meeting of the TeX Users Group*. Dean Guenther, ed. *TeXniques* 5. Providence: TeX Users Group, 1988.



## TeX Users Group

Stanford University, August 13–24, 1984

Terman Engineering Center Auditorium and The Graduate School of Business

# Olde Worlde T<sub>E</sub>X

MALCOLM CLARK

Imperial College Computer Centre  
Exhibition Road  
London SW7 2BP  
`texline@vaxa.cc.imperial.ac.uk`  
`mwc@doc.imperial.ac.uk`  
and  
T<sub>E</sub>Xpert systems  
36 Baker Street  
London W1M 1DG

## ABSTRACT

T<sub>E</sub>X and L<sub>A</sub>T<sub>E</sub>X are used world wide. In recent years user groups have started to spring up spontaneously, outside (or alongside) the umbrella of TUG. A brief outline of the established European groups is presented here, together with a selective account of some historical background. There are some concerns, which although not specifically European, are certainly non-US, and indicate areas which tend to be overlooked when viewing the T<sub>E</sub>X world from the New World.

### 1. Introduction

T<sub>E</sub>X is alive and well on both sides of the Atlantic. Readers of *TUGboat* will be familiar with contributions from outside North America. But it is not the purpose of this short paper to chart the migration of T<sub>E</sub>X from the New World to the Old Worlds. Besides anything else, I am bound to omit some of the early movement, simply because it is no longer evident. The early history of T<sub>E</sub>X is only poorly known, which is a great pity, since it might provide some useful information about the way that software evolves and becomes incorporated into the wider world. Many institutions and systems take T<sub>E</sub>X/L<sub>A</sub>T<sub>E</sub>X for granted — how did this happen? But fascinating as this might be (and worthy of ‘Trivial Pursuit — the T<sub>E</sub>X edition’), it is not the topic here.

The objective is merely to report on the present status of T<sub>E</sub>X user groups in Europe. As such, we are ignoring many aspects of T<sub>E</sub>X and T<sub>E</sub>X support.

One aspect of the T<sub>E</sub>X community which must be obvious to all who read the electronic communications facilities like T<sub>E</sub>Xhax is the readiness of T<sub>E</sub>X/L<sub>A</sub>T<sub>E</sub>X users to contact other T<sub>E</sub>X/L<sub>A</sub>T<sub>E</sub>X users by *email*, and their apparent reluctance to join TUG. Search the *TUGboat* membership listings for the various T<sub>E</sub>Xhax contributors — where are they? This aside covers people with *email* access. The growth of “personal” versions of T<sub>E</sub>X, where the person at the keyboard is less likely to have *email* or even bulletin board access, may point to an even larger community of effectively anonymous users.

### 2. European Organisations

Within Europe there are five groups with national or supra-national interests. The order in which they are presented here is completely arbitrary, and reflects neither size, age nor alphabet.

#### 2.1 DANTE

The German T<sub>E</sub>X users group (*Deutschsprachige Anwendervereinigung T<sub>E</sub>X*) is, as its name says, a *German-speaking* user group, and as such, includes members from Austria and Switzerland, as well as the Federal Republic. DANTE was founded in 1989. But there have been significant T<sub>E</sub>X/L<sub>A</sub>T<sub>E</sub>X activities in the German-speaking countries for several years. This year’s meeting in Eichstätt will

be the eighth annual meeting of German TeX users, and closely follows TeX89, the fourth European TeX Conference, to be held in Karlsruhe (quite independently, in September). We might also note that Frank Mittelbach was awarded the Knuth Scholarship for his work with LATEX, as well as Hubert Partl's work in customising LATEX to the various regional requirements within the DANTE catchment. DANTE promises to be very active and to promote the use of TeX in a very wide sense. One of their proposals is to provide TeX/LATEX courses in German (not before time!) and even to distribute a copy of Klaus Thull's public domain TeX (for the IBM/clone-pc) to every school in the area — principally for use by teachers. There is also a proposal to produce a 'newspaper', which it is intended will appear at the Eichstätt meeting in October.

The contributions of the German-speaking TeX users includes Norbert Schwartz' excellent introductory text to TeX (1988), and Reinhard Wonneberger's extremely useful 'compact' LATEX book (1988). Several other books are beginning to appear.

A common feature of most of the groups is the use of list servers. The list server which is operated from Heidelberg is a useful feature of DANTE's activity, enabling the electronic transmission of a large amount of public-domain TeXware.

## 2.2 GUTenberg

*Groupe (francophone) des Utilisateurs de TeX* is a little older than DANTE. Again, activity in France is nothing new. The second European TeX conference was held in Strasbourg in 1986, and Jacques Désarménien of Strasbourg is well known for his work in incorporating French hyphenation and national characters into TeX, since before 1984. GUTenberg was constituted on September 23rd 1988, but before that time an unofficial group did exist, and organised a number of meetings, including one entitled "TeX et les Sciences Humaines" (Paris, 1987). In fact GUTenberg traces its origins back to 1984, following a suggestion from Jacques André. The focus of GUTenberg appears to be Paris rather than Strasbourg. Both GUTenberg meetings (in 1988 and 1989) have been in Paris, but next year's meeting will be in Toulouse. Both Paris meetings were well attended and included many participants from the academic, research and commercial world. One of the encouraging features of GUTenberg membership is the support from the commercial sector (indicating that pragmatic concerns can be accommodated from within TeX/LATEX).

Although this apparently emphasises an exclusively French component to GUTenberg, it does include the other parts of the French-speaking world, like Switzerland, Belgium and Quebec. Michael Ferguson's ML-Tex (Multilingual TeX), which greatly eases the use of national characters and hyphenation, is a Canadian development which was wide applicability both outside and within the French-speaking world. GUTenberg is empowered to distribute VMS, UNIX and MVS implementations of ML-Tex to its membership (for free, but with copying restrictions). They also distribute a version of SBTeX (one of the public domain MS DOS compatible TeXs), together with various TeXtools.

Education plays an important role: this year's annual meeting (May 16th-17th; for a report, see *TUGboat* 10:150-153) included two day-long presentations on METAFONT and LATEX (naturally, in French).

One of GUTenberg's major contributions is the *Cahiers GUTenberg*. So far three editions of the *Cahiers* have appeared. Adopting a fairly common French convention, the very first edition was 'numéro zéro'. Production is of a high quality, and so too is the content. The *Cahiers* deserve far higher circulation outside the French-speaking world. Although not a GUTenberg "production", we must also note Raymond Séroul's *Le petit livre du TeX* (1989), an excellent introduction to TeX.

There is a list server which is used by the French group. Refreshingly, they are well-aware of the problems of the user who is not connected to electronic networks and are hoping to develop access to the list server through Minitel.

## 2.3 Nordic TeX Group

While the previous two groups are, in a sense, language specific, the Nordic group is a geographical supra-national group, including participants from Denmark, Norway, Sweden and Finland. The language problems are sufficiently diverse, even *within* some of the individual countries that no one set of language solutions (hyphenation/national character) is possible. A result of the heterogeneity of language is that the meetings tend to be conducted in English, as the *lingua franca* of the area. The

reluctance of the French and German groups to use English (because it is not spoken by the majority of the membership) is not found here. This does carry with it the possible implication that Nordic  $\text{\TeX}$  users may be drawn from the sector which has gone through an educational process which includes English as a foreign language. This may represent an ultimate restriction on the distribution of  $\text{\TeX}$  to the academic and research areas, and a slowing down of its penetration to more general technical and non-technical document preparation. (Equally, it seems that almost everyone speaks English, to the extent that Swedes may reply in English to a question posed in Swedish by a Dane — perhaps the scope for confusion is lessened!)

The Nordic group has had a number of meetings, and seems to be able to maintain a pattern of about one meeting per year — this year's was on June 12th (a report appeared in  *$\text{\TeX}line$  9:25–27*). The structure is quite informal, although there is an organising committee elected each year. Being drawn from a number of constituent countries, this has to be a well-balanced affair, with representatives from each country. Unlike the other European groups, membership in this group is free.

$\text{\TeX}$  has been used in the Nordic countries for some considerable time. The first version of  $\text{\TeX}$  which I ever used ( $\text{\TeX}78$  on a CDC NOS machine, driving an APS  $\mu$ -5 phototypesetter) was obtained from the University of Århus in the early 1980s.

There is a short  $\text{\LaTeX}$  book by Steen Larsen,  *$\text{\LaTeX}$  på Dansk* (1989), produced at UNI-C, in Danish, and a wealth of other introductory documentation in Norwegian, Finnish, and English. It is unfortunate that this material is not more widely known. Although few outside Finland may read Finnish, we can all appreciate the examples and the structure of such documents. This is probably a general point: there is undoubtedly support material available in France and Germany which does not surface outside those countries. It is inconceivable that they have no relevance to the rest of us. Sadly, the canonical  $\text{\TeX}$  and  $\text{\LaTeX}$  books are not really suitable for many beginning  $\text{\TeX}/\text{\LaTeX}$  users, excellent though they may be for the  $\text{\TeX}x$ pert and  $\text{\TeX}n$ ician.

Although there is no list server as such, the Nordic group is setting up an electronic mailing service over the Scandinavian Universities Network which will channel  $\text{\TeX}$  news. They also plan to handle electronic enquiries over the net.

Many of the suggestions presented by the Nordic representatives for modifications to  $\text{\TeX}$  (in order to handle foreign languages more easily) were suggested in a paper by Romberger and Sundblad in 1985 at the first European  $\text{\TeX}$  Conference. That paper also makes quite explicit the problems generated by keyboards which replace some crucial  $\text{\TeX}$  characters by national letters. Losing the \, {, }, [ and ] presents some interesting problems, and certainly delayed the adoption of  $\text{\LaTeX}$ .

## 2.4 Dutch $\text{\TeX}$ Group

The NTG (*Nederlandse  $\text{\TeX}$  Gebruikergroep*), is about one year old. Their recent two-day meeting in Utrecht (June 29th–30th; for a report, see  *$\text{\TeX}line$  9:29–30*) combined introductory courses on  $\text{\LaTeX}$  with a conventional conference. As a somewhat smaller grouping (Dutch is spoken by fewer people than either German or French), the Dutch are very aware of the need for international cooperation. Many of their concerns parallel those of other countries: the customisation of  $\text{\LaTeX}$  style files is a wheel which has been re-invented several times already, but the identification of the parts of *lplain* and the *.sty* files which require harmonisation have not been adequately documented at an international level, despite the applicability of a general solution to this particular problem. Both  $\text{\TeX}$  and  $\text{\LaTeX}$  are sometimes seen as unnecessarily “Anglo-Saxon” (whatever that means!), with embedded assumptions which are not truly warranted.

There are other  $\text{\LaTeX}$  concerns: the adjustment, or creation, of genuine “European” styles is well overdue. We are all aware of the need to use international sized paper (usually A4), and the implicit assumptions that  $\text{\TeX}$  and  $\text{\LaTeX}$  make of US paper sizes: equally, European typography is different from the rather narrow Addison-Wesley style typography exemplified by  $\text{\LaTeX}$ . This is no criticism of  $\text{\LaTeX}$ . It is a criticism of all those of us who have sat idly by and who have not nailed our friendly local typographers and document designers to the wall to help us develop new, appropriate styles.

Like most other groups, NTG operate a list server. While this is really of use only to those on electronic networks, the NTG augments this by providing some style files and macros on pc disk.

## 2.5 The UK TeX Group

This group, which has not yet adopted a catchy title like DANTE or GUTenberg, has been formed in the last year or so. It is still being run by a non-elected organising committee, although this should have been rectified by October 1989, when the annual meeting will be held in Exeter. Despite its rather low-key approach, it has organised three meetings since December 1988: all were one-day affairs, but used different venues each time — an attempt to diminish the domination of London as a location (for reports see *TeXline* 8:17–18; 8:18–20; 9:20–22). It is perhaps a hallmark of the strength of the group that there have been dominant themes for the meetings (except the initial prototype). The intention is to have four one-day meetings each year, at a variety of locations. Historically though, the very first UK “TeX” meeting was probably the METAFONT meeting organized in Oxford on March 20th, 1987, by Charles Curran (for a report, see *TeXline* 5:8–9).

It is difficult to separate the UK group entirely from the activity of the Aston TeX archive.<sup>1</sup> Thanks to the far-sightedness of Aston University and the efforts of Peter Abbott, an entire VAX 11/780, together with a considerable amount of disk space, is dedicated to electronic TeXware. This archive is probably the largest single collection of electronic TeX-related material in the world, and represents a unique resource. The archive is accessible over various electronic networks, and although it does not share the delight of using the otherwise ubiquitous IBM list servers, has comparable power and ease of use. A recent major re-organisation has made navigation through the archive somewhat easier. However a recurrent problem is the character corruption which occurs at one of the major gateways into and out of the JANET network. This has the effect of losing braces entirely and corrupting at least two other characters, in a two-to-one mapping. This minor problem greatly compromises any chance of using the archive in a global area network. It is hoped that this problem will soon be removed.

The user group is also considering how to make the archive accessible to those without JANET access, either through guest accounts, or other types of electronic (or even non-electronic) communication. Aston has also been the focus of UKTeX, a regular listing similar to TeXhax. Although begun as a UK-specific listing, it now has a world-wide membership. An interesting feature of UKTeX is its regular Friday evening appearance (or, at least, transmission). Other listings might take note.

As a spin-off from part of the archive activities, an introductory book is being prepared. This is not a TeX book, but a background book which aims to be of use to implementors of TeX. It seeks to explain the inter-relationships between the various TeX tools. Although this information already exists, it is available in a wide variety of (often) inaccessible sources — like early editions of *TUGboat*, or Stanford reports.

The UK was also the site of TeX88 (or TeXeter), the 1988 European TeX conference. Joachim Lammarsch, of DANTE, says that it was here that “the idea of a German TeX society came into my mind”. He then went on to try to bring together German users into a formal group. It was also at Exeter that the first moves were made towards a UK grouping. TeX88 was a very good, friendly meeting, attended by about the same number of people as the TUG meeting which followed in Montreal (August 1988).

It is curious that no books have yet appeared from the UK on TeX or *LATeX*, although rumours abound. Addison-Wesley, the nominal publisher of TeX material is known to have turned down proposals for such books from the UK, and it seems that it will be left to other publishers such as Oxford University Press, John Wiley and Sons, or even Ellis Horwood Publishers, to publish these. On the other hand, the newsletter *TeXline* has been produced and distributed since late 1984. Now in its ninth edition, it has gradually grown from about eight pages to a peak of forty pages (edition eight). Like the *Cahiers*, it covers a wider range of topics than *TUGboat*.

## 3. What of the Rest of Europe?

Curiously, although Italy was the site of the first TeX conference in Europe (1985), there appears to be no real focus for TeX users there. Equally, the lack of a user group in Eire is not inhibiting the hosting of the first TUG meeting in Europe, to be held in Cork in September 1990. Although there is at least one journal being produced with TeX in Spain, whatever individuals exist do not seem to

---

<sup>1</sup> See next article, by Peter Abbott –Ed.

have been drawn together yet.<sup>2</sup> In Israel (defining Europe rather widely), there are a number of active TeX workers, but again, no clear focus.

Clearly something is required. We will assume that user groups are useful phenomena. One of the useful suggestions made has been that a list server could be set up, specific for any national group. The list server could be almost anywhere, provided there is adequate electronic link up. Again this divides the world into haves and have nots, but provided we remain aware of the problem, it can probably be handled.

There is also the problem of Eastern Europe. The problem is mainly one of currency restrictions, rather than availability of computing hardware. The IBM pc (or clone) appears to have penetrated widely, although laser printers (and consumables) appear less readily available. It is difficult for TeX users in eastern Europe to join TUG (or to buy TeXware). Nevertheless, there is a significant TeX presence in Poland, in Jugoslavia, and probably in the German Democratic Republic and the Soviet Union. The 1985 European TeX Conference Proceedings contain a paper about the problems involved in transferring TeX to an indigenous East European computer (Bień and Kołodziejska 1985), and since then some very sound work has been done in creating a Polish TeX, which, among other concerns, handles the national characters which are not easily obtained through standard TeX — like the ogonek and the circle accent.

#### 4. What are the Concerns of TeX Users in Europe?

Of course the concerns are as diverse as the concerns of TeX users elsewhere (perhaps more so than users in North America). But some key themes do keep cropping up.

##### 4.1 National Characters

The various national characters, including the accented characters, pose certain problems. The first of these is that TeX does not hyphenate beyond the point at which the first control sequence in a word occurs.<sup>3</sup> Another minor problem is the absence of the ogonek accents and the non-explicit presence of the circle accent. European languages use different conventions to indicate “quotes”, for example, French uses «guillemets», and German tends to use quotes like „this“.

##### 4.2 Hyphenation

The hyphenation algorithm used by TeX may be used by other languages, but the patterns have to be created for each language. Although this has been done for a good many languages, this does require an added degree of knowledge and research on the part of the ordinary user.

##### 4.3 L<sup>A</sup>T<sub>E</sub>X

The TeX tool of choice in Europe is L<sup>A</sup>T<sub>E</sub>X. It is interesting to speculate why — perhaps the slightly later diffusion of TeX and L<sup>A</sup>T<sub>E</sub>X to Europe meant that by the time it arrived, L<sup>A</sup>T<sub>E</sub>X was a stable product. I suspect that L<sup>A</sup>T<sub>E</sub>X is actually used in North America much more than we commonly acknowledge. Perhaps one of the reasons that TUG courses have only taken off very slightly in Europe is TUG’s tendency to offer courses on TeX rather than L<sup>A</sup>T<sub>E</sub>X. TUG’s perception of TeX has tended to be of TeX rather than L<sup>A</sup>T<sub>E</sub>X, or even METAFONT.

##### 4.4 L<sup>A</sup>T<sub>E</sub>X Customisation — The Language

As noted earlier, to use L<sup>A</sup>T<sub>E</sub>X outside the restriction of English requires some work. This has been completed for German, Dutch and French (at least). This information should be properly documented to make it easier for other language groups to emulate. There is no such thing as a TeX or L<sup>A</sup>T<sub>E</sub>X problem which is “only of interest to our own language group”.

##### 4.5 L<sup>A</sup>T<sub>E</sub>X Customisation — The Typography

L<sup>A</sup>T<sub>E</sub>X styles are being generated to accommodate European styles, for publishers based in Europe,

<sup>2</sup> At the TUG conference I learned that a Spanish group was in the process of being organised.

<sup>3</sup> In a major announcement at the Stanford meeting, Knuth outlined a proposed revision of TeX, to v. 3.0, which would include looking at the hyphenation problems —Ed.

but also for everyday needs. Many of us feel that the "received" L<sup>A</sup>T<sub>E</sub>X styles are rather "loose", with excessive white space for European expectations. It is clearly necessary to document any new styles so that others can use them (please).

#### 4.6 Education

It is not enough to assume that T<sub>E</sub>X courses in English are sufficient. As T<sub>E</sub>X, and especially L<sup>A</sup>T<sub>E</sub>X spread out, we acquire users who do not necessarily understand, far less speak, English, as it is not the *lingua franca* that T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X are.

#### 4.7 Networking/List Servers

This is an area which will burgeon and penetrate everywhere, no matter what controls and directions we attempt to give it. It is difficult area to coordinate. Keeping track of what software is available where, or even which version is current, represents a real problem.

#### 4.8 Standards

This is by no means unique to Europe, but there is plenty of evidence of awareness of international standards in the document processing area, such as SGML (the Standard Generalized Markup Language), ODA (Office Document Architecture), and so on. There also seems to be a marked willingness to adopt and adapt to these standards — and to cooperate with their user groups or allied organisations. Papers in which such standards have played an important role are to be found in the *Cahiers*, in all the European T<sub>E</sub>X conferences, and also in T<sub>E</sub>Xline.

### 5. Contacts

DANTE:	DANTE Research Center of the University of Heidelberg Im Neuenheimer Feld 293 D-6900 Heidelberg 1 FRG Bitnet: dante@dhdurz1
GUTenberg:	GUTenberg c/o IRISA Campus Universitaire de Beaulieu F-35042 Rennes Cedex France Uucp: gut@irisairisa.fr Bitnet: ucir001@frors31
Nordic Group:	Roswitha Graham Royal Institute of Technology S-100 44Stockholm Sweden Sunet (Internet): roswitha@admin.kth.se
NTG:	Kees van der Laan Rekencentrum RUG Landleven 1 NL-9700 AV Groningen The Netherlands Bitnet: cgl@hgrrug5
UK group:	UK T <sub>E</sub> X User Group c/o Computing Service Aston University

Aston Triangle  
Birmingham B4 7ET  
UK  
Janet: abbott@uk.ac.aston

## 6. Acknowledgements

I am grateful to the many people who have contributed directly and indirectly to this article. Although too numerous to be named individually, I must note in particular the helpful and generous contributions of Bernard Gaulle, Joachim Lammarsch, Kees van der Laan and Roswitha Graham.

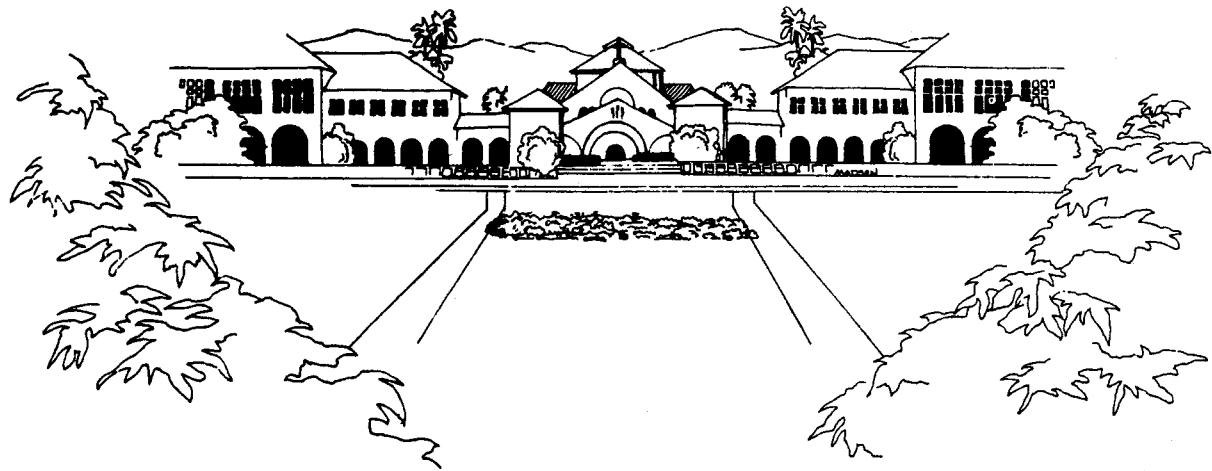
## Bibliography

The following is a very partial and incomplete list of *TeX*- and *LATeX*-related books which have been published in Europe.

- Appelt, Wolfgang. *TeX für Fortgeschrittene*. Bonn: Addison-Wesley, 1988.
- Brüggeman-Klein, Anne. *Einführung in die Dokumentenverarbeitung*. Stuttgart: B.G. Teubner, 1989, 200pp.
- de Bruin, Rob, Cornelis G. van der Laan, Jan R. Luyten and Herman F. Vogt. *Publizeren met LATeX*. Amsterdam: CWI Syllabus 19, 1988, 196pp.
- Désarménian, Jacques, ed. *TeX for Scientific Documentation*. Second European Conference, Strasbourg, France, June 1986. Berlin: Springer-Verlag, 1986, 204pp.
- Kopka, Helmut. *LATeX: Eine Einführung*. Bonn: Addison-Wesley, 1988.
- Larsen, Steen. *LATeX på Dansk*. Copenhagen: UNI-C, 1989, 111pp.
- Lehtonen, Ari. *AM-S-TeX Ja Matemaattisen Tekstin Käsittely*. [*AM-S-TeX and Mathematical Text Processing*]. Jyväskylä: Jyväskylän Yliopisto, 1988, 53pp + 6pp + 10pp + 33pp + 7pp.
- Lucarella, Dario, ed. *Proceedings of the First European Conference on TeX for Scientific Documentation*. Reading, Mass.: Addison-Wesley, 1985, 204pp.
- Nenonen, Liina. *PC TeX-opas*. [*PC TeX Manual*]. Jyväskylä: Jyväskylän Yliopiston Monistuskeskus, 1988, 76pp.
- Saarinen, Kauko. *TeX-alkeisopas*. [*TeX Beginners' Manual*]. Jyväskylä: Jyväskylän Yliopiston Monistuskeskus, 1989, 57pp. + appendices.
- Schwartz, Norbert. *Einführung in TeX*. Bonn: Addison-Wesley, 1988.
- Séroul, Raymond. *Le petit livre de TeX*. Paris: InterEditions, 1989, 317pp.
- Suhonen, Timo. *LATeX-opas*. [*LATeX Manual*]. Jyväskylä: Jyväskylän Yliopiston Monistuskeskus, 1988, 63pp.
- Wonneberger, Reinhard. *LATeX Kompaktführer*. Bonn: Addison-Wesley, 1988, 141pp.

## References

- Bień, Janusz S., and Hanna Kołodziejska. "TeX for RIAD Computers." Pp. 133–140 in *Proceedings of the First European Conference on TeX for Scientific Documentation*, Dario Lucarella, ed. Reading, Mass.: Addison-Wesley, 1985.
- Clark, Malcolm. "METAFONT meets at Oxford." *TeXline* 5:8–9, 1987.
- Clark, Malcolm. "Nordic TeX." *TeXline* 9:25–27, 1989.
- Clark, Malcolm. "Nederlandse TeX Gebruikergroep." *TeXline* 9:29–30.
- Clark, Malcolm. "Réflexions sur le Congrès GUTenberg." *TUGboat* 10(2):150–153.
- Hewlett, Carol. "UK TeX Users' Group." *TeXline* 8:17–19, 1988.
- Osborne, David. "Notes on 1st meeting, UK TeX Users Group." *TeXline* :20–22, 1989.
- Rahtz, Sebastian P. Q. "A UK TeX Users Group?." *TeXline* 8:20–22, 1988.
- Romberger, Staffan, and Yngve Sundblad. "Adapting TeX to Languages that Use Latin Alphabetic Characters." Pp. 27–40 in *Proceedings of the First European Conference on TeX for Scientific Documentation*, Dario Lucarella, ed. Reading, Mass.: Addison-Wesley, 1985.



**T<sub>E</sub>X Users Group**  
**Stanford University, August 11–14, 1985**  
**Terman Engineering Center Auditorium**

# The UKTEX Archive at the University of Aston

PETER ABBOTT

The Computing Service  
University of Aston  
Aston Triangle  
Birmingham B4 7ET  
England  
[AbbottP@Aston.Ac.Uk](mailto:AbbottP@Aston.Ac.Uk)

## ABSTRACT

A viable archive should contain material which is useful, accessible and easily implemented on the target system. The growth of electronic communication, electronic mail in particular, has brought into being computer-based archives, removing the restrictions of geographical or political boundaries.

This paper describes the growth of the Aston Archive, the Archivists and how we manage the archive. I shall describe our approach to make implementation across a wide range of systems consistent and how we answer questions such as "Does the archive contains fonts for device Y" or "How do I use printer X".

I hope to answer the question "Is the Archive Useful?"

### 1. Definition of an Archive

If the proverbial "man in the street" were asked the question "What is an archive?", the most likely reply would be "A collection of books or records". A popularly held view is of the 19th-century room filled with books collecting dust, or records of births, deaths and marriages. In the United Kingdom for many centuries, parish records formed the basis of archives and were usually kept in the Parish Church — a situation which no longer applies, I might add. Chamber's Student Dictionary (Revised edition 1980) defines archives as "the place in which government records are kept: public records". The University is mainly funded by government monies and the archive does contain public records, so the definition seems relevant.

I suppose that the modern concept of an archive was created with the advent of the digital computer, and it was only a matter of time before "collections" of computer programs and similar material acquired the title ARCHIVE. In fact, the common term used when preserving the filestore of a computer against corruption for future use is archiving, and a number of products on the open market make use of the word. The major growth in electronic communication in the last few years has opened up the concept of a central archive to serve large areas of a country or even the globe.

There are a number of well-known archives in the world, and several which serve the TeX community. I shall, of course, be concentrating on the one located at Aston University in the United Kingdom.

### 2. The System at Aston

Aston University is located in the City of Birmingham on a 35-acre city-centre campus. Birmingham has two universities, and the older one (carrying the title "Birmingham University") owes much to Chamberlain and is classed as a "red-brick" University.<sup>1</sup> The Aston Campus does not conform to the standard city-centre mould, and its campus (albeit small) would not disgrace a country-park approach. The Computing Service which hosts the archive is located on the fourth floor of the Main Building,

<sup>1</sup> 'Red-brick' — The Victorians were fond of using red terra cotta bricks for buildings and universities founded in Victorian times have acquired the name "red-brick".

which was designed in 1933 and built after the second World War. As a result, the accommodation arrangements are generous for the hardware and acceptable for the staff and users.

To all intents and purposes, the equipment used to support the archive is immaterial, other than for the unusual problems that arise; the present computer system is a VAX processor running the VMS operating system.

### 3. Archive History

In the summer of 1987, a number of interested users in the United Kingdom discussed (via e-mail) the idea of a UK-based version of *TeXhax*, whereby local problems could be aired and informal contact made between like users. At that time, Aston were interested in text publishing and the Vice Chancellor expressed an interest in *TeX* (having spent some years at Stanford).

I agreed to set up a distribution list and to collate and distribute the queries. The first few issues were experimental, with a number of hiccups and problems, but eventually a pattern emerged of a weekly digest normally posted on a Friday afternoon beginning at 1700 hours local time.

As a result of the digest software, a large number of questions started arriving at Aston; these mainly asked if 'x' was available, but also frequently offered items which might be useful to others. Like Topsy, it grew and grew and grew.

For many months, the digest was from the UK for the UK, but as it became more widely known, requests for subscription arrived from many areas of the world including the North American continent. This brought more material for the archive and requests for distribution both via e-mail and by more conventional means (such as magnetic tape). The archive and digest are now considered as originating in the United Kingdom with almost universal distribution.

All of this activity consumes a large part of my "free" time and I became increasing worried that the archive was becoming disorganised with duplicate material. At a meeting in November 1988 at Nottingham, I appealed for help in preparing an article for *University Computing*, a publication of the Inter University Computer Committee (IUCC); this has since developed into a book (Clark and Abbott 1989; I have brought copies of part of the book with me to the meeting). I also approached a number of people with a view to assistance with the archive.

#### 3.1 Archivists

##### *Adrian Clark*

Adrian (who prefers the alias *Alien*) is a researcher in the image-processing field, being especially interested in algorithmic and software aspects. His interests in *TeX* are hence mainly concerned with graphics — particularly the inclusion of grey-level pictures ("halftones") in documents. He is the author of a number of articles in *TUGboat* concerning halftone output and aspects of the implementation of *TeX* under VAX/VMS. Adrian is responsible for several areas of the archive, including contributed *IATeX* and *BjBTEx* styles, VMS implementations, and halftones. He also developed the Aston mail-server.

##### *Malcolm Clark*

Malcolm really needs no introduction but he has been producing books in *TeX* since 1984; is editor of *TeXline*, a newsletter of the *TeX* community, since 1985; organised *TeX88*, the 3rd European *TeX* Conference at Exeter University; is the editor of the *TeX88* Conference Proceedings (which we are still awaiting). He is currently TUG's European Co-ordinator and also Secretary of the British Computer Society's Electronic Publishing SIG; teaches TUG courses in the US and Europe; carefully watches SGML, PostScript and ODA; currently completing *A plain TeX Primer* — an introduction to *TeX*, and *HyperTeX* — a Hypertext guide to *TeX* and *IATeX*. Responsible for a tiny portion of the Aston Archive.

##### *Charles Curran*

Charles has worked in the Systems Development section at Oxford University Computing Service for the last fourteen years. His main interests are operating systems and document architectures. He supports the UNIX services, which are provided on Suns and Convex supercomputers. He is currently chairman of the UK Convex Users Group. He first became familiar with *TeX* about eight years ago,

when he installed it on a local VMS VAX and also on the sadly maligned Perq. He has also spent a fair amount of time with METAFONT. It is with founts and things METAFONT that he is primarily concerned in his work on the Archive.

*David Osborne*

Dave joined the Cripps Computing Centre, University of Nottingham, in 1984 as systems programmer. His special interests are UNIX and text processing. He helped in setting up TeX on the Centre's VAX 11/750 some time in 1986, and has since worked on the Centre's TeX implementation on an ICL Clan 7 running UNIX System V. His areas of Texnical interest are general, and mainly related to the implementation side of things, partly from a UNIX system manager's point of view. Dave joined TUG in 1988.

His responsibilities in Archive management include: the Beebe drivers, Andrew Trevorrow's software, Common TeX, and PC items.

*Sebastian Rahtz*

Sebastian is a lecturer in the Department of Electronics and Computer Science at the University of Southampton, where he has a special interest in humanities computing, teaching courses in text-processing for computer scientists and computing for archaeology M.Sc. students. He originally got involved with computers for publication of an archaeological report, and his research interests of typesetting, generic markup, hypertext and graphical databases continue to reflect a preoccupation with publication.

He has been using TeX since 1985, and has produced a number of books. The archaeological aspect has produced a particular interest in the problems of graphical inclusion in TeX, and the integration of PostScript. Within the Aston Archive, he attempts to cover the UNIX implementations, some of the PostScript problems and some of the graphical systems.

*Philip Taylor*

Phil has been with the University of London for almost eighteen years, having migrated from Westfield College (University of London) to Bedford College (London), and thence to Royal Holloway and Bedford New College (RHBNC, University of London), where he is presently a member of the Computer Centre. His interest in TeX was evoked by a chance encounter at British Petroleum, where he first saw a sample of TeXset text, and was amazed to learn that it had been produced on a system identical to that at RHBNC. He rushed back to College with a tape, spent the next month pestering various colleagues at KQC for assistance with his naïve attempts to produce TeX output, and has never looked back since. He is currently visiting Canada (yet again) to collaborate with the Professors Gibson and Gibson, at the Universities of Guelph and Waterloo, in TeXsetting their book on nutrition. When not TeXsetting, he spends most of his time falling off horses.

Within the archive group, Phil is responsible for advice on VAX/VMS file aliasing, and for maintenance of the VAX/VMS list-server.

Areas of interest: Primitive and Plain TeX; do-it-yourself macro design; TeX esoteria and arcana.  
*Bêtes noirs:* L<sup>A</sup>T<sub>E</sub>X, UNIX & Wimps.

*Niel Kempson and Brian Hamilton Kelly*

Niel and Brian joined the archivists in May 1989 to take over the VMS part from Adrian, who was having difficulty in coping with it in addition to the mail server. Adrian still retains the mail server.

Brian (note the double-barrelled surname, *without* a hyphen) is a Senior Research Officer in the Software Engineering Group at the Royal Military College of Science, which is now part of Cranfield Institute of Technology: he is currently working for an M.Phil. connected with code generation derived directly from graphical representations of systems' designs (although he spends more of his time on TeXware!)

He has spent the whole of his working life in software, since 1964, the majority of it with the MoD (until the privatization of RMCS), but also worked for a short spell in 1966 at Imperial College. He has had an interest in computerized typesetting since 1968, and in type and typography going back to his schooldays. (Ah! The joys of an Adana hand press!) He has been using L<sup>A</sup>T<sub>E</sub>X since 1985, and his

first TeXware project was to rewrite Rose's original DVI driver for the DEC LN03 wholly in WEB, to overcome the bugs in the PL/I part of it. (His program, DVitoLN03 is not to be confused with Rose's later DVitoLN3 offering.)

He would like to see the authors of TeX- and METAFONT-ware pay much more attention to writing for ease of maintenance, and feels that every C-programmer should have three men with whips standing over him ensuring that good software engineering precepts are followed! Furthermore, he would like to see every TeX macro package re-written and documented through Frank Mittelbach's `doc` style — any volunteers??!

Niel is by profession a chartered electrical engineer, but is currently on sabbatical leave at RMCS Shrivenham, studying for a PhD. His academic interests are centered around the use of multivariate analysis to identify different types of communications signals.

As with most of the post-graduate students at RMCS, he soon discovered the joys of using L<sup>A</sup>T<sub>E</sub>X to produce reams of mathematical equations. He soon became involved with the maintenance of TeXware at RMCS, specializing in the Unix systems, DEC's language sensitive editor and translating BiBTeX into C for use on the IBM PC.

#### 4. JANET

It is fair to say that the *only* way that we can work in maintaining and developing the archive and its facilities is by utilising modern technology. In this case I mean JANET — the (U.K.) Joint Academic Network. JANET is a private network operated by the Joint Network Team under a crown agency. The recurrent funding is met centrally and all universities have free and unfettered access. Polytechnics, research councils and other bodies deemed "suitable" are also connected (they generally have to pay a once-off connection charge).

The archivists have an account on the system at Aston and use JANET to carry out allotted tasks. Archive users are not permitted at present to access the archive interactively (archivists enjoy this perk), but Network Interface File Transfer Protocol (NIFTP) and e-mail access are both provided.

##### 4.1 Gateways

Whilst this provides a service to the connected community, there are wider implications for its use. Gateways to other networks exist at various points in the United Kingdom, all linked via a backbone. The well-known gateways are:

Internet/Arpa/Nsfnnet	Uk.Ac.Nsfnnet-Relay
Earn/Bitnet/Netnorth	Uk.Ac.Earn-Relay
uucp	Uk.Ac.Ukc

plus others including commercial networks such as the British Telecom PSS service. The main trunks operate at 2 megabits and the Aston-Manchester link is currently 48K.

#### 5. Structure of the Archive

We have decided to sub-divide the archive into the following main topic areas:

- tex
- metafont
- latex (subdirectory for slitex)
- amstex
- digests (texhax, uktex, texmag)
- bibtex
- utils
- fonts
- etc
- drivers
- docs
- langs
- tools (for use by the 'team', not the general public)

## 5.1 Areas Covered

There are two aspects to areas covered:

- What material should be accepted?
- What systems should it cover?

In dealing with the first area, I must mention the United Kingdom National Public Archive at Lancaster University. This archive contains public-domain software mainly for MS-DOS PCs and Apple Macs. At one time it carried some of the PC software related to  $\text{T}_{\text{E}}\text{X}/\text{IAT}_{\text{E}}\text{X}$ , but this has now been removed and all enquiries are re-directed to Aston.

The archivists are only interested in working with material directly related to  $\text{T}_{\text{E}}\text{X}$  and related software, and a long debate has taken place over utilities such as the bootstrap for MS/DOS PCs. In the end, it was agreed that these would need to be stored at Aston as well as Lancaster, because not all subscribers (principally those on the other side of a gateway) could obtain the items from Lancaster.

We are working towards the situation whereby *all* needs for an implementation — from the smallest PC to the largest mainframe — can be obtained, preferably by e-mail, but in the interim supplemented by other media.

## 6. Who Uses the Archive?

### 6.1 Contributors

It might appear that we are working in a vacuum at Aston, but this is far from the truth — contributions are received from many countries and organisations. It is impossible to name the majority, but I must mention the following:

- Pierre MacKay (Univ. of Washington/Seattle), who regularly sends an update tape of the UNIX versions
- Michael DeCorte (Clarkson Univ., New York), who regular mails updates from his collection
- Max Calvani (International School for Advanced Studies (ISAS), Italy), who can supply material which avoids the EARN-Relay and the corruption that occurs there
- Andrew Trevorrow (freelance  $\text{T}_{\text{E}}\text{X}$  programmer, Australia)
- Peter Flynn (Academic Projects Manager University College Cork, Ireland)
- Graham Toal (self-employed consultant, Edinburgh), who provided the early mail server to the Archive

plus, of course, all the others too numerous to mention.

### 6.2 Types of Access

As of August 1989 there are only two means of access: NIFTP and mail. Users on JANET can use NIFTP to transfer items from the archive to their local system. *All* users can access the archive via the mail server, and I am pleased to be able to report that the VAX/VMS problem of STREAM\_LF files causing untold havoc has been overcome. There might be one or two files still to convert, but the end is in sight.

There are no interactive access facilities, although it is hoped to introduce a processor dedicated to the service in the near future. This will provide not only Kermit and Zmodem facilities, but also (hopefully) a secure environment for mail-boxes and bulletin-board services.

## 7. Digest Support

The digest is the major vehicle for information dissemination, and until access for external users becomes available, is also distributed via paper mail. This service is provided on a VAX 11/750 independent of the archive service, and it is intended to merge this with the archive as soon as the separate processor becomes available.

## 8. Non-Connected Users

There are many commercial users, most of whom do not have a connection or access to JANET, so for this group of users the advent of the independent service for the archive will open new opportunities.

The University has dial-in lines which will give access from any telephone, provided that the user has a suitable modem.

## 9. Is it Useful?

I think the following statistics will answer the question:

For the period November 1988 to end July 1989

No. of searches	26
No. of requests for a Directory listing	975
No. of requests for files	2,988
Files sent	3,986
Help sent	142
Italian help	6
Danish help	21
Invalid requests	195

## Bibliography

- Clark, Adrian, and Peter Abbott, eds. *The T<sub>E</sub>X Companion*. In preparation.
- Clark, Malcolm. *HyperT<sub>E</sub>X*. Exeter and London: T<sub>E</sub>Xpert Systems, 1989.
- Clark, Malcolm. *A plain T<sub>E</sub>X Primer*. [Forthcoming].
- Clark, Malcolm, ed. *T<sub>E</sub>X88 Conference Proceedings*. Chichester: Ellis Horwood, 1989.
- Mittelbach, Frank. "The doc option." *TUGboat* 10(2):245–273, 1989.

# With $\text{\LaTeX}$ into the Nineties

FRANK MITTELBACH

Fachbereich Mathematik Universität Mainz Staudinger Weg 9 D-6500 Mainz <code>schoepf@dmznat51.bitnet</code>	Electronic Data Systems (Deutschland) GmbH Eisenstraße 56 D-6090 Rüsselsheim <code>qzdmgn@drueds2.bitnet</code>
---	---

RAINER SCHÖPF

Institut für Physik Universität Mainz Staudinger Weg 7 D-6500 Mainz <code>schoepf@dmznat51.bitnet</code>
--

## ABSTRACT

During the last three years,  $\text{\LaTeX}$  has spread widely, even into such new fields as business applications. The fact that there are new classes of users forces one to reconsider the  $\text{\LaTeX}$  implementation and some of its features. Within a few years,  $\text{\LaTeX}$  2.09 alone will not be sufficient to satisfy the increasing needs of its users. As a consequence one of the important characteristics of the  $\text{\LaTeX}$  concept — the possibility of exchanging documents — is in danger of being sacrificed on the altar of local changes and enhancements.

Starting from these considerations and from our experiences of several years of  $\text{\LaTeX}$  support, we will present a proposal for a re-implementation of  $\text{\LaTeX}$ . This new version would not only preserve the essential features of the present user interface (in order to be compatible with old  $\text{\LaTeX}$  files), but also take into account already formulated requests, as well as future developments.

### 1. Introduction

During the last several years,  $\text{\LaTeX}$  has become a widely used tool for typesetting documents. Its advantages over WYSIWIG systems as well as over plain  $\text{\TeX}$  — logical design and high-level commands for formatting issues — allow even the inexperienced typesetter (author) to easily produce readable output. Especially for author groups, the concept of logical design is essential to ensure uniformity of their work.

At least in theory,  $\text{\LaTeX}$  is a markup language which enables the users to specify their input as logical parts, e.g., to mark a text fraction as a “quotation” or as a “labeled list” instead of supplying formatting commands such as “indent the next three lines and start with a bullet”.

The translation into formatting commands, i.e.,  $\text{\TeX}$  primitives, is restricted from use (again, at least in theory) in the source file. It will take place instead, unnoticed by the user, in one of several style files which should be applicable to the same input source producing different layouts.  $\text{\LaTeX}$  itself provides four different prototype styles (`article`, `report`, `book` and `letter`) which should be used for different types of input sources, i.e., should not be interchanged. This contradiction at a first glance (one can't switch between `article` and `report` for example) is the first (of many) misunderstandings for  $\text{\LaTeX}$  users and amateur style designers.

Since there is only one officially supported document style for each type of input,  $\text{\TeX}$  users with plain  $\text{\TeX}$  experience often find themselves unable to produce a desired layout with  $\text{\LaTeX}$ . So, after some unsuccessful attempts, they return to plain  $\text{\TeX}$  even if they then have to face other problems (such as cross-references, etc.) which are easily provided in a markup language such as  $\text{\LaTeX}$ .

In this case the root of the problem is the fact that most of the  $\text{\LaTeX}$  interfaces are poorly documented so that even  $\text{\TeX}$  experts might have trouble designing an offset layout style for example, which could be used instead of the standard report style. The result is that  $\text{\LaTeX}$  documents all look alike because all existing document styles are either unusable (because of many bugs), or they are only variants of the prototype styles, without noticeable differences.

## 2. The $\text{\LaTeX}$ User Interface — or Essential Features of $\text{\LaTeX}$

For every type of document  $\text{\LaTeX}$  provides a set of high-level formatting commands which themselves are defined with the help of internal macros. This internal code is highly modular, often ingenious,<sup>1</sup> and allows a wide variety of layouts if the style designer is sufficiently familiar with the interfaces.

On the surface,  $\text{\LaTeX}$  demonstrates a thought-out concept too: similar situations require similar syntax, unusual cases are hidden behind optional arguments ... But that isn't all. As a standard in all applications (but with different layouts)  $\text{\LaTeX}$  provides:

- automatical generation of contents listings
- a powerful cross-referencing mechanism with symbolic names. This allows insertion, deletion and movement of text blocks without re-arranging the equation numbers, etc.
- the possibility of typesetting only parts of the document without losing cross-references, counter-values, etc.
- a general float mechanism for automatic placement of figures, tables, etc. independently of each other (but with each class preserving its order)
- in conjunction with  $\text{BIB\TeX}$  and  $\text{Makelindex}$ , it has powerful tools for automatic creation of index and bibliography entries<sup>2</sup>
- fully implemented size changing commands for several types of fonts
- a general mechanism for switching page layouts (running headings, etc.)
- ...

So, why not use  $\text{\LaTeX}$ ? This question leads us to our next topic.

## 3. Limitations of $\text{\LaTeX}$ Version 2.09

Limitations to the current  $\text{\LaTeX}$  version can be divided into several groups, which we will discuss separately, giving examples as we go along.

### 3.1 Implementation Disasters

This is where most of our examples are located. In a way it is also the group of problems which are the easiest to solve: just learn from the mistakes Leslie Lamport made but use all his good ideas.

#### *Fragile commands*

Maybe the most troublesome concept in this category is the classification of commands into *robust* and *fragile* ones. At the bottom of page 23 the  $\text{\LaTeX}$  book says:

The argument of a sectioning command may be used for more than just producing the section title; it can generate a table of contents entry and a running head at the top of the page. [...] When carried from where it appears in the input file to the other places it is used, the argument of a sectioning command is shaken up quite a bit. Some  $\text{\LaTeX}$  commands are *fragile* and can break when they appear in an argument that is shaken in this way. Fragile commands are rarely used in the argument of a sectioning

<sup>1</sup> Clearly not all parts of  $\text{\LaTeX}$  are well implemented. But the overall concept is wisely chosen and this isn't affected by design or implementation bugs in its modules.

<sup>2</sup> At a site with an up-to-date  $\text{\LaTeX}$  installation, both programs should be available. Unfortunately, this is often not the case.

command. [...] On the rare occasions when you have to put a fragile command in a section title, you simply protect it with a `\protect` command.

And later on (p. 24):

An argument in which fragile commands need `\protect` will be called a *moving* argument. Commands that are not fragile will be called *robust*. For any command [...] I will indicate whether it is robust or fragile. Except in special cases [...] a `\protect` command can't hurt, so it is almost always safe to use one when you're not sure if it's necessary.

Isn't that easy? Unfortunately not, because a broken command will produce a totally unintelligible error message and, to make the chaos perfect, not only could this error occur at a different location, it is also possible that the error will not vanish when the missing `\protect` is finally discovered. This is hell for novice L<sup>A</sup>T<sub>E</sub>X users, but even experts are shaken quite a bit if they commit this sin.

The `\protect` is used to suppress unwanted expansions which are the reasons for the errors mentioned above. This is the wrong design decision: it would be better to suppress all expansions by default, and allow the user to expand single macros if there is need for it.

#### L<sup>A</sup>T<sub>E</sub>X error recovery

This part of the implementation can be summarized in a single statement: there is none. Actually there are several situations where the L<sup>A</sup>T<sub>E</sub>X error routine is triggered but the recovery mechanism isn't very powerful. On its own, this poses no problems because one can adopt the philosophy "understand the error, correct the source and re-run", but unfortunately the error help messages aren't very enlightening:

You're in trouble here. Try typing <return> to proceed.  
If that doesn't work, type X <return> to quit.

Well, we certainly know we're in trouble when we see a whole page of error messages coming from T<sub>E</sub>X's stomach,<sup>3</sup> and instructions on how to turn off the computer and go home: that isn't what the user expects to get when he or she enters 'H' (for help) after a L<sup>A</sup>T<sub>E</sub>X error.

The error messages themselves are often simply wrong; for example, the input

```
\begin{center}
  ...
\end{center}
```

produces the error

Something's wrong--perhaps a missing \item

In nine out of ten cases this error isn't caused by a missing `\item` so the user doesn't know what to correct — in this case the `center` environment expects text, i.e., something in horizontal mode, and not just a blank line. Looking in the manual helps a little bit, because there the error messages are explained in greater detail, but all in all the error messages produced by L<sup>A</sup>T<sub>E</sub>X are a mess from the user's point of view.

#### The generic list environment

The generic list environment is one of the central modules of the L<sup>A</sup>T<sub>E</sub>X implementation. It is used internally by most standard environments provided by L<sup>A</sup>T<sub>E</sub>X; even environments such as `center` are handled as a special kind of list (with an empty `\item` command).

To allow for such a variety of applications, the list environment has nearly 20 parameters and switches which can be manipulated to change the layout. Additionally the default values for these parameters (as defined in the document style file) depend on the level of nesting; that is, the document style may provide different default spacing for lists within lists.

<sup>3</sup>L<sup>A</sup>T<sub>E</sub>X's tendency to produce horrible-looking error listings is actually a T<sub>E</sub>X problem which should be listed under "T<sub>E</sub>Xnical limitations". A large macro package such as L<sup>A</sup>T<sub>E</sub>X is bound to have many expansion levels and there is no possibility of suppressing the intermediate part in the stack history when T<sub>E</sub>X spots an error. In our opinion there should be a T<sub>E</sub>X `\tracing...` command to control the amount of history listing.

But there are also a few implementation problems:

- An actual conceptual bug was the decision to add the value of `\parskip` to all vertical spacing parameters, even when it is used in places where no paragraph ends.  
This means that changing this parameter influences the layout in unexpected places, which in turn means that other parameters must be adjusted unnecessarily to compensate for this undesired side effect.
- There is also the problem that the `\parskip` parameter is user-accessible while the affected parameters are only changeable in a style file. The user can change the `\topsep` parameter, for example, but its default value, define `d` in the style file, will be restored later on.
- The resetting of parameters to their default values if nothing else is specified is somewhat arbitrary. Some of the important parameters (i.e., the penalty values for page breaking before and after the list) get their values from the surrounding list which is more than a nuisance for a style designer.<sup>4</sup>
- Another implementation decision makes it impossible to define lists with page-wide labels, i.e., with labels placed on a line by themselves. Those lists cannot be nested properly (see, for example, comments in the article about the implementation of the extended theorem environment [8]).

As `LATEX` is used in more and more fields the need for an even more general list environment is increasing every day.

### 3.2 Design Limitations

It is certainly not possible to draw a sharp line between implementation disasters and design limitations. The former are problems introduced when the macros were written, the latter come from decisions or omissions made during the design phase. Nevertheless, these two phases often go with each other.

#### *Font selection*

One of these limitations was Leslie Lamport's decision to take over from plain `TEX` the method of selecting different typefaces: he arranged fonts in a two-dimensional grid, one dimension specifying the size, the other one the typeface. This was reasonable at the time it was implemented as there were only a few different typefaces for use with `TEX`. In the meantime, however, more and more fonts have become available, rendering the above design decision inadequate. For example, it is not possible for the user to switch between Knuth's fonts and the resident fonts of a PostScript printer in a transparent way.

#### *The list environment*

In spite of its generality, the list environment has some conceptual weaknesses:

- There is no possibility of generating a run-in list, i.e., a list where the first item runs into the preceding text. This feature is provided in `AMS-TEX`.
- More generally, since the layout of the standard list types is fixed by the document style selected, there is no way for the user to select different kinds of layouts for the same type of lists (e.g., enumerated or itemized lists) without defining his own environments. It would be better to provide a way to specify attributes such as "compact", "stream" (see, for example [17]), "run-in", etc.
- The vertical spacing before and after lists is controlled by the same parameter.
- The vertical space preceding the first item does not depend on the length of the last line of the preceding paragraph (as is the case for displayed equations).

#### *Attribute concept*

`LATEX` does not allow the user to specify attributes as, for example, Script-DCF does. However, this concept is worth being considered for at least partial inclusion in `LATEX` (see previous subsection).

---

<sup>4</sup>As an unpleasant result the `LATEX fleqn` style option (which causes displayed equations to be typeset flush left instead of the usual centering) favours pagebreaks just before displayed equations. (This bug is corrected in the `LATEX` version of 24 May 89.)

#### *Text-producing arguments*

The  $\text{\TeX}$  mechanism for scanning macro arguments fixes the  $\backslash\text{catcodes}$  of the token it reads. As a result, certain  $\text{\LaTeX}$  commands (such as  $\text{\verb}$ ) cannot be used inside  $\text{\parbox}$ ,  $\text{\footnotes}$ , etc. This can be avoided using a more elaborate scheme; see for example the  $\text{\footnote}$  implementation in plain  $\text{\TeX}$  [3, p. 363] or the article about chapter mottos [18].

#### *Support for mathematics*

In addition to the math features provided by plain  $\text{\TeX}$ ,  $\text{\LaTeX}$  offers only the  $\text{\eqnarray}$  and  $\text{\array}$  environments. For typesetting mathematical papers with  $\text{\LaTeX}$  this is certainly not sufficient.  $\text{\AMS-\TeX}$  provides the necessary features; however, inclusion of these into  $\text{\LaTeX}$  has not been done yet.<sup>5</sup> Instead, various people have written unpublished style options, each implementing limited subsets.

#### *array and tabular*

$\text{\LaTeX}$ 's  $\text{\tabular}$  environment is a sophisticated tool for typesetting alignments. That means that you need not be a  $\text{\TeX}$  master to "know how to make ruled tables" [3, p. 245]. On the other hand several easy-to-implement features are not provided. See for example the new implementation described in [5, 6].

#### *Pictures*

To draw picture diagrams  $\text{\LaTeX}$  offers mostly basic positioning commands that should actually be used internally to define high-level interfaces. Examples of such interfaces are given in [13, 16].

#### *The output routine*

The  $\text{\LaTeX}$  output routine is a very complex and sophisticated module that offers a wide variety of possibilities for page make-up. However, certain layout decisions, such as special placement of footnotes and floats, cannot be implemented in style files without a re-definition of  $\text{\LaTeX}$ 's internal macros. This poses problems of compatibility. See also Section 4.

### **3.3 Unknown Interfaces**

As we have already noted, many interfaces are not properly documented. This results in unnecessarily complicated solutions to certain layout problems. Even worse: sometimes people are led to the conclusion that there is no solution at all!

Such problems can often be solved easily by the internal  $\text{\LaTeX}$  macros used in the right way. As an example, consider an offset layout where all section headers are to extend into the left margin that itself is wide enough to hold them. This is provided by the generic sectioning command  $\text{\@startsection}$  that allows an explicit indentation parameter to be specified. If you give a negative value to it, you end up with section headers that stick out to the left.

### **3.4 $\text{\TeX}$ nical Limitations**

In this group we gather limitations caused by the  $\text{\TeX}$  program itself. Probably the most severe limitation comes from  $\text{\TeX}$ 's insert mechanism. After prematurely ejecting a page (so that the output routine can look at its contents), inserted material is removed from the main vertical list and gathered into certain boxes. It is therefore not possible to re-insert the page contents unchanged. This makes it nearly impossible to balance the height of several columns when insertions such as footnotes or floats are involved [9]. Other  $\text{\TeX}$ nical problems arise from the way  $\text{\TeX}$  inserts penalties, or breaks paragraphs into lines.

## **4. New Demands**

When asked their opinion, many  $\text{\LaTeX}$  users reply:

" $\text{\LaTeX}$  is a very fine thing but this (...) and that is missing."

---

<sup>5</sup>This situation has now been remedied; the American Mathematical Society will be releasing an  $\text{\AMS-\TeX}$  style file which will be an option for use with any  $\text{\LaTeX}$  style file. There will also be versions of  $\text{\book.sty}$  and  $\text{\article.sty}$ , implementing book and journal styles defined by the AMS. This information comes courtesy of Regina Girouard, Composition Services Manager of the AMS –Ed.

If we ignore those requests that can be satisfied by reading the manual, we are left with three groups of wishes:

- Features that can be implemented in the current version of L<sup>A</sup>T<sub>E</sub>X. Here we have the problem that there are too few people who know the internal structure and interfaces sufficiently well to do it. These interfaces are poorly documented, and misunderstandings and misuse of these can lead to catastrophic results. Even if this is done correctly there remains the problem of compatibility of the different style options.
- Features where the necessary interfaces are missing. Here one needs to change the internal macros of L<sup>A</sup>T<sub>E</sub>X to implement them.
- Totally new requirements that lead to large-scale changes to the code or even to the concepts. Even worse: some things cannot be implemented at all because T<sub>E</sub>X itself is not able to handle them.

Instead of listing the numerous requests we will give a few examples.

#### 4.1 Easy Implementable Features

We have already talked about the offset style problem. Other examples for requests which can be solved easily in a style file are special *pagestyles* (like the one Leslie Lamport used in his book), numbering conventions (e.g., equations within theorems . . . ), and special heading layouts (e.g., centered headings or \chapter without the word "Chapter").

A very important issue is the support of national languages. Standard L<sup>A</sup>T<sub>E</sub>X does not offer anything in this regard. However, this is easily implemented as a style option as the file *german.sty* shows. This file constitutes the German standard to which all German language sites agreed in 1987. See [12] for a description of its features.

Certain demands arise in business applications; for example, the need to stamp every page of the document with date, time, name of the input file, and possibly security information.

#### 4.2 Features Implementable with Moderate Effort

As an example of this second group, take the new implementation of the *array* and *tabular* environments described in [5, 6]. In addition to the possibility of creating beautiful ruled tables, this new implementation allows the user to specify the layout of a column in one place.

Or consider the challenge posed by David F. Rogers in [15]. He asked for a figure placement macro that would fulfill at least three of five requirements, and states that "L<sup>A</sup>T<sub>E</sub>X's floating insert commands also do not work". This is only partly true. Two requirements are automatically fulfilled in standard L<sup>A</sup>T<sub>E</sub>X; the remaining one (numbered figures must be inserted after the first reference to the figure) can be easily implemented by changing only *one* line of code in *one* internal macro of the L<sup>A</sup>T<sub>E</sub>X output routine: in the macro \c@addtocurcol, the call to \c@addtotoporbot has to be changed to a call to \c@addtobot. Of course, a style option which implements this change has to take care of the float parameters too, since their default settings tend to discourage bottom floats.

As another example take a two-column layout where all footnotes appear at the bottom of the right column. Again this can be solved by re-defining *only* the internal macro \c@makecol.

#### *Support for PostScript printing devices*

"We have to acknowledge the importance of the *de facto* standard, POSTSCRIPT. [...] We must be aware of the way in which PostScript compatibility is crucial if we are to be taken seriously by the rest of the world" [2, p. 153]. The question of converting the *.DVI* file contents to PostScript has already been taken care of, but here we are concerned with the problem of incorporating PostScript fonts. It is easy to change *lfonts.tex* to use the fonts built into PostScript printers instead of the ones by Knuth. However, this must be done at dump time, and is therefore not selectable by the user. Instead, one needs a font selection scheme that allows dynamic switching of fonts during a T<sub>E</sub>X run, as outlined in [10].

#### 4.3 Hard Problems

Now that we have seen that complex page make-up can be handled easily in L<sup>A</sup>T<sub>E</sub>X, one might wonder which demands (in our opinion) belong to the third group. Well, page make-up for example; *really*

complex demands such as the layout used by *Scientific American*: three columns, figures spanning one to three columns with captions placed in the neighbouring column if necessary. As already mentioned, TeX's \insert primitive cannot be used for such a task. Doing *everything* by hand is possible (TeX is a Turing engine, as Leslie Lamport remarked) but there are limitations in space and time.<sup>6</sup>

As we have already noted, we think that balancing of columns belongs to the third group if footnotes are involved, despite the fact that the *TeXbook* [3, p. 417] implicitly says that this is possible. We would be only too happy to see an algorithm which did *not* break under normal (i.e., unrestricted) conditions.

## 5. Proposal for a New Implementation

The current L<sup>A</sup>T<sub>E</sub>X version essentially consists of the following files:

- lplain.tex — plain TeX features used by L<sup>A</sup>T<sub>E</sub>X
- lfonts.tex — font definitions
- hyphen.tex — hyphenation patterns
- latex.tex — most L<sup>A</sup>T<sub>E</sub>X features or the internal macros to build them in a style file
- \*.sty — document style files and document style options

The first four of these files are all loaded when a format file is dumped. This means that a large amount of TeX's internal memory is used to store the definitions contained in these files, even though there are only very rare occasions when they are all used together. If we consider adding more and more features to L<sup>A</sup>T<sub>E</sub>X, we are led to ask: which parts of L<sup>A</sup>T<sub>E</sub>X are essential for most document types (called the "kernel") and which are only used by special applications (called "the peripheral part").

### 5.1 The L<sup>A</sup>T<sub>E</sub>X Kernel

Before we can talk about re-writing the kernel, we have to separate it from the peripheral parts. The following modules are considered part of the kernel:

- Basic features from plain TeX (defined in lplain.tex)
- Font selection
- Constants used by the rest of the program
- Program control structure
- Semi-parameter concept
- Basic error handling routines
- Spacing, line and page breaking
- File handling
- Counter management
- Cross-referencing
- Environment handling
- Basic math commands
- Generic list environment and their basic applications
- Box making commands (including parbox and minipage)
- array and tabular
- The interface for the theorem environment
- Generic sectioning commands
- The interface to generate tables of contents, lists of figures, etc.
- Bibliography
- Floats
- Footnotes

---

<sup>6</sup>However, we are working on this project at Mainz, hoping that we can prove that this problem belongs to the second group.

- The output routine

Some of these parts need to be revised, others must be re-implemented completely: the list environment, the writing to `.aux` files to remove the `\protect` feature (already done), the font selection code (already done), the counter mechanism (done), hierarchical references, `array` and `tabular` (done), or a conceptionally new output routine (certainly the hardest part). All the above parts should be better modularized to obtain a higher degree of flexibility.

To provide better control by means of the styles or the extensions, a number of hooks, such as `\everypar`, will have to be introduced: think of `\everylist`, `\everysection`, `\everybegindocument`, `\everyenddocument`, etc.

## 5.2 Peripheral Features

The peripheral parts should not be included in the format file. They can easily be moved to a number of files that are loaded on demand during a `LATEX` run.

We consider the following parts to be peripheral:

- Higher math. The only features currently available are `eqnarray` and `array`. A new implementation should provide the same features as `AMS-TEX`, each of them loaded separately.
- New `verbatim` with unlimited size of verbatim text and a command to input verbatim text from a file (done).
- Enhanced `picture` environment (conceptionally done, partly implemented: `epic`, `PICTEX`)
- Tabbing. Improvements should be discussed, e.g., push/pop over several tabbing environments.
- Support for special draft options showing symbolic labels, index entries where they appear, time and date stamps, etc.
- Index: more exactly, the interface to an index program such as `MakelIndex`.

As we mentioned earlier, a `LATEX` installation is complete only if it provides `BIBTEX` and `MakelIndex`. The integration of these programs, especially of `MakelIndex`, i.e., the interfaces, should be improved.

## 5.3 Document Styles

### *A standard mark-up concept (SGML)*

For a re-implementation of `LATEX` one also has to reconsider the standard document styles. As described in the introduction, there are different types of documents, e.g., books, manuals, articles, reports, letters, proceedings, etc. These types cannot be interchanged since each has its own logical concepts: letters do not possess chapters whereas there is no need for an opening or a signature in books. Given one type of document (e.g., report), there are many different ways to do the formatting. The `report` style of current `LATEX` is therefore only one out of many instances of the “meta” report style.

The important point here is that all report-type styles must use the same logical concepts so that a properly written `LATEX` document is independent of the specific style instance used. A German site (say, the University of Mainz) may decide to provide a special style (called, say, `uni-mainz-report`) to format reports according to its own conventions. But a report written at Stanford can then be typeset at Mainz using this style *without* changing the `LATEX` input file — but of course the document will come out with different spacing, etc.

Therefore an important task of a new implementation is to reconsider the logical concepts used so far, and then decide to drop or change some, or to add new ones. For example, currently there isn’t any difference between the prototype styles `report` and `book`. Obviously this cannot be correct. This is of course a question being debated by the experts.

### *International language support*

There is one point to make here: from what we have said above one must not infer that the textual representation of headers (e.g., “Contents”, “Litteratura”) must be fixed by the style. In this respect we disagree with Leslie Lamport. To the contrary: since it may well be that a site wants to typeset documents written in different languages, but all in the same style, it is only reasonable to provide a way to change header names. Actually we propose to make this a logical concept of the specific meta style. This means that there must exist commands such as `\chaptername` and `\refname` that can be changed by the document (using `\renewcommand`) or by style options. It is perfectly allowed (though

not necessarily reasonable) for a specific style to ignore these commands, and to typeset all headers in the same way.

## 6. Institutional Considerations

Leslie Lamport has copyrighted L<sup>A</sup>T<sub>E</sub>X and fixed the status quo. While this is the best way to ensure uniformity over a large group of installations and users, one runs the risk that further developments and enhancements will become incompatible. Therefore our proposal for a new implementation is bound to fail if it becomes only one more among many others (with only a new name). We feel that such a project should only be undertaken if it is supported by an institution which can channel future developments.

### 6.1 Maintenance

From the size of the L<sup>A</sup>T<sub>E</sub>X source code, it is clear that it must be maintained. This not only means that there must exist a person who will correct any bugs found; this is only part of the story, and not necessarily the most important one. Another necessary task is to develop the software as new demands arise. Software design is a difficult job. It is even more difficult to revise design decisions made earlier, because in addition to software development aspects, one has to consider questions of compatibility.

For the maintenance of a package the size of L<sup>A</sup>T<sub>E</sub>X, one needs a group of people who can respond to demands and wishes, and decide what can and should be done. This means that this group must include document style as well as L<sup>A</sup>T<sub>E</sub>X specialists. These people need not necessarily be the implementors themselves. They have to set the standards, not write the macros. Nevertheless, they need to be sufficiently familiar with L<sup>A</sup>T<sub>E</sub>X.

### 6.2 Suggestion for a L<sup>A</sup>T<sub>E</sub>X Standardization Group

Certainly this is only possible if Leslie Lamport is willing to share control over L<sup>A</sup>T<sub>E</sub>X. Another important point is to guarantee that this group continues to perform its task even if the individual members change. The logical point to organize this would be the T<sub>E</sub>X Users Group. We think that TUG should form a committee to discuss these problems.

## 7. Update

During and after the conference at Stanford, discussions were held with Leslie Lamport concerning the issues raised in this paper. From these it became clear that he originally expected L<sup>A</sup>T<sub>E</sub>X to be superceded by some other document preparation system within a few years of its appearance. This has not happened and now he too sees a need for its further development.

He is in agreement with the principles contained in this paper concerning the future of L<sup>A</sup>T<sub>E</sub>X but does not wish to be directly involved in their implementation. During the discussions the following two-stage development plan was suggested:

1. Re-design the style file interface and document it: this would involve the publication of a manual on the design of style files. The resulting L<sup>A</sup>T<sub>E</sub>X version would be 2.10. This version may contain some minor enhancements visible to the user, but every input file that uses only features documented in the current L<sup>A</sup>T<sub>E</sub>X manual would work with version 2.10.
2. Produce a completely new implementation of L<sup>A</sup>T<sub>E</sub>X, version 3.0, according to the principles outlined in this paper. This would be upwardly compatible in the sense that it would be possible to process any existing document with the addition of a style option.

The timetable for this work cannot be fixed at present since it is not yet clear how much of our time we shall be able to spend on this project, nor what support will be forthcoming from various parties.

## Bibliography

- [1] Bartlett, Frederick H. "Automatic Page Balancing Macros Wanted." *TUGboat* 9(1):83, 1988.
- [2] Clark, Malcolm. "International Standards and T<sub>E</sub>X." *TUGboat* 10(2):153-156, 1989.
- [3] Knuth, Donald E. *The T<sub>E</sub>Xbook. Computers and Typesetting* Vol. A. Reading, Massachusetts: Addison-Wesley, 1986.

- [4] Lamport, Leslie. *L<sup>A</sup>T<sub>E</sub>X: A Document Preparation System. User's Guide and Reference Manual.* Reading, Massachusetts: Addison-Wesley, 1985.
- [5] Mittelbach, Frank. "A new implementation of the `array-` and `tabular-`environments." *TUGboat* 9(3):298–314, 1988.
- [6] Mittelbach, Frank. "A new implementation of the `array-` and `tabular-`environments — addenda." *TUGboat* 10(1):103–104, 1989.
- [7] Mittelbach, Frank. "The `doc` option." *TUGboat* 10(2):245–273, 1989.
- [8] Mittelbach, Frank. "An Extension of the L<sup>A</sup>T<sub>E</sub>X theorem environment." *TUGboat*, 1989 [forthcoming].
- [9] Mittelbach, Frank. "An environment for multi-column output." *TUGboat*, 1989 [forthcoming].
- [10] Mittelbach, Frank, and Rainer Schöpf. "A new font selection scheme for T<sub>E</sub>X macro packages — the basic macros." *TUGboat* 10(2):222–238, 1989.
- [11] Mittelbach, Frank, and Rainer Schöpf. "A new font selection scheme for T<sub>E</sub>X macro packages — the L<sup>A</sup>T<sub>E</sub>X interface." *TUGboat*, [forthcoming].
- [12] Partl, Hubert. "German T<sub>E</sub>X." *TUGboat* 9(1):70–72, 1988.
- [13] Podar, Sunil. Enhancements to the Picture Environment of L<sup>A</sup>T<sub>E</sub>X. Dept. of Computer Science, S.U.N.Y. at Stony Brook, Technical Report 86-17, version 1.2, July 14, 1986.
- [14] Price, Lynne A. "SGML and T<sub>E</sub>X." *TUGboat* 8(2):221–225, 1987.
- [15] Rogers, David F. "A Page Make-up Challenge." *TUGboat* 9(3):292–293, 1988.
- [16] Schöpf, Rainer. "Drawing histogram bars inside the L<sup>A</sup>T<sub>E</sub>X picture environment." *TUGboat* 10(1):105–107, 1989.
- [17] Wonneberger, Reinhard. "Stream lists and related list types for L<sup>A</sup>T<sub>E</sub>X." *TUGboat* 6(3):156–157, 1985.
- [18] Wonneberger, Reinhard. "Chapter Mottos and Optional Semi-Parameters in General and for L<sup>A</sup>T<sub>E</sub>X." *TUGboat* 7(3):177–185, 1986.
- [19] Zalmstra, Joost, and David F. Rogers, "A Page Make-up Macro." *TUGboat* 10(1):73–81, 1989.

# T<sub>E</sub>Xreation — Playing Games with T<sub>E</sub>X's Mind

ANDREW MARC GREENE

Project Athena  
Room E40-342  
and  
Student Information Processing Board  
Room W20-557  
Massachusetts Institute of Technology  
Cambridge, MA 02139  
*amgreen@athena.mit.edu*

*Please address all correspondence to the Project Athena address.*

## ABSTRACT

T<sub>E</sub>X can be used to write applications that have little or no connection with document preparation. Games such as ANIMALS and BATTLESHIP are just a few of the recreational uses to which T<sub>E</sub>X can be put — mostly to show that it *can* be done, but also to provide an entertaining medium for both experimentation and presentation of programming techniques that can be used in more serious macro packages. Database management is exemplified by the program ANIMALS, and array handling is developed in BATTLESHIP.

### 1. Introduction

At some point, most T<sub>E</sub>X users find it necessary to extend the language of T<sub>E</sub>X to perform some task. Whether the task is as simple as defining a macro to alleviate repeated typing of some lengthy string, or as complicated as rewriting output routines, we sit at our keyboards alternately cursing and blessing Donald Knuth.

Few, if any, of us ever write a program in T<sub>E</sub>X that has nothing to do with typography. T<sub>E</sub>X is slower than C, more obtuse than assembler, and harder to trace than BASIC. Nevertheless, writing programs in T<sub>E</sub>X is possible and will occasionally yield results that are useful in "real" T<sub>E</sub>X programs (or macro packages).

This paper will present two such programs, both of which are games. ANIMALS, a simple "artificial intelligence" program, resulted in a set of T<sub>E</sub>X database management routines. BATTLESHIP, the classic game of naval battle on a grid, was a perfect candidate for implementation of array handling and indexed variables in T<sub>E</sub>X.

### 2. The Game of ANIMALS

ANIMALS was written in response to a dare from a friend at the Student Information Processing Board at MIT. It is a simple expert system, in which the computer asks questions and tries to guess which animal the user has selected based on the user's responses. An annotated listing of ANIMALS appears in Appendix A.

#### 2.1 Rules

The user thinks of an animal which the computer will attempt to guess. On each round, the computer asks a yes/no question, which the user must answer truthfully. Eventually the computer will take its guess; if it is correct then the program ends, otherwise the computer will amend its database to include the new animal and a question distinguishing the new animal from the original guess.

These rules imply that the database should be arranged in a binary tree, such as the sample in Figure 1. Since T<sub>E</sub>X doesn't have random access files, this would be difficult to implement. However,

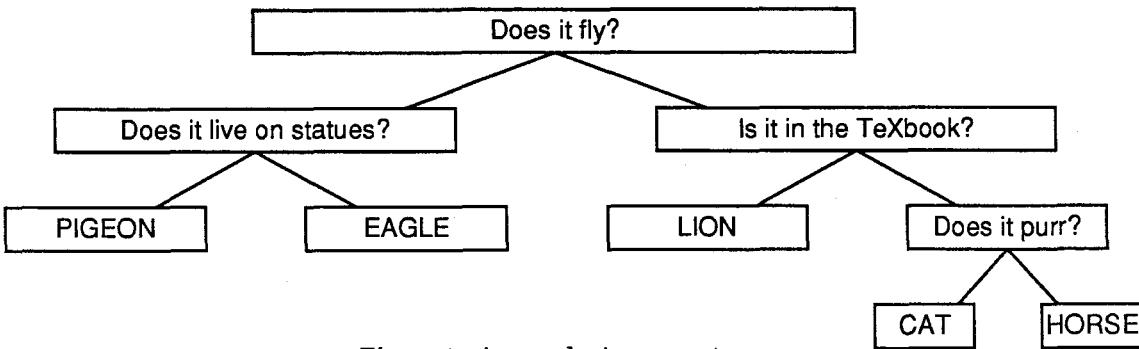


Figure 1: A sample ANIMALS tree

it should be noted that progress is always towards the bottom of the tree; therefore it is possible to simulate a random-access database whose records will always be read in a certain order using *T<sub>E</sub>X*'s sequential file operators *\openin*, *\read*, and *\closein*.

## 2.2 Database Routines

For ANIMALS, three fields are required for each record: the question to ask, the record to go to for a "Yes" response, and the record to go to for a "No" response. These are stored in the data file as separate lines of text, with the record number prepended to each record. Thus, the data file for the sample tree in Figure 1 begins:

```

1
Does it fly?
2
3
2
Does it live on statues?
4
5

```

What is needed next is a routine to go to a specific record in the file, keyed by the record number. The *\Scan* routine does just that, assuming that *T<sub>E</sub>X* is already aligned at the start of a record. Once *T<sub>E</sub>X* is at the correct record, the *\Query* routine deals with extracting the data and selecting the next record.

This solves the problem of reading a quasi-random-access database. If the database needs to be modified, however, we run into difficulties. *T<sub>E</sub>X* allows a file to be open only for input or output, but not both. Furthermore, modifying a variable-length field within a file would be impossible, even if we could modify (rather than replace) an existing file.

The solution, of course, is to read in the original data from the beginning of the data file, copying each field to a temporary file. For ANIMALS, we find that one record needs to be replaced and two new ones added. Therefore, the record number is watched and, when the record to be replaced is reached, a modified version is output. Finally, the two additional records are output to the temporary file, which contains the revised database.

The process is now repeated in the other direction. The original file is replaced by a line-by-line copy of the temporary file. Since *\openout* overwrites the original file, this is an effective way to "modify" that original file.

Users of UNIX have another option to this way of modifying the data file, although it violates the spirit of this exercise. They can run, in the background, *tail -f /tmp/shell.tex : /bin/csh* and output editor commands to */tmp/shell.tex*. This is cheating, however, since the goal is to write seemingly useless programs *entirely* in *T<sub>E</sub>X*.

	A	B	C	D	E	F	G	H	I	J
1										
2	<b>B</b>									
3	<b>B</b>									
4	<b>B</b>		<b>C</b>	<b>C</b>	<b>C</b>	<b>C</b>				
5	<b>B</b>				<b>D</b>	<b>D</b>	<b>D</b>			
6	<b>B</b>									
7										
8										
9							<b>S</b>			
10							<b>S</b>			

Figure 2a: A typical BATTLESHIP setup

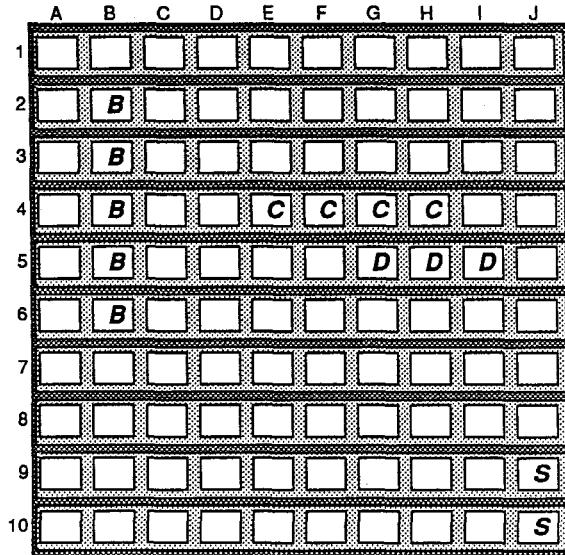


Figure 2b: Box representation

### 2.3 Other Database Applications in *TEX*

Despite the intentions of the author to avoid presenting any useful (i.e., typographic) code in this paper, there is one program that ought to be mentioned.

The author is in charge of "CokeComm" at the Student Information Processing Board at MIT. CokeComm is a debit-based system in which members deposit money, Coca-Cola products are delivered in bulk, and members mark a space on a list whenever they take a can. This calls for a program that can maintain balances and print out new sheets, properly formatted.

The code for *coke.tex*, which appears in Appendix B, should be compared to the *ANIMALS* code. The comments in *coke.tex* are intended to illustrate how to splice the database routines into other programs.

## 3. BATTLESHIP

**BATTLESHIP** is the most well-known variation of a game also known as "Salvo" or "Naval Battle." The implementation described in this section is based on the Milton Bradley version. **BATTLESHIP** was written as an excuse to experiment with arrays and indexed variables in *TEX*.

### 3.1 Rules

This version is simplified slightly by having fixed ship positions. The computer places four ships, each of a different length, in a ten-square grid. The ship designations and their lengths are "Carrier" (5 squares), "Battleship" (4), "Destroyer" (3), and "Submarine" (2). The ships are aligned either vertically or horizontally, so that each takes up the appropriate number of adjacent grid spaces. A typical arrangement is shown in Figure 2a.

Once the ship positions have been entered, the second player begins. On each turn, the player selects a grid location at which to shoot. The computer responds with the result — either the designation of the damaged ship or the text "You missed." If all cells of a particular ship are damaged, the ship is considered sunk; when all ships in the fleet are sunk the game is over.

It is obvious that the naval grid is a two-dimensional array. Unfortunately, *TEX* does not support array variables. The command string *\array* refers to typesetting a collection of elements in row-column format, surrounded by large delimiters. This is not what we want.

### 3.2 The Wrong Way

One's first impulse might be to implement arrays as nested boxes, as depicted in Figure 2b. An array variable might be declared with *\newbox\MyGrid* and initialized appropriately. In the diagram, *\box\MyGrid* is the large, dark *\vbox*, inside of which are ten light grey *\hboxes*, inside each of which

are ten white \hboxes containing the array elements.

There are a number of disadvantages to this approach. An admittedly trivial objection might be that there are only 256 box registers, and some of those are reserved. Any programmer who needs more than 250 different arrays would have difficulties.

A more realistic objection arises when one considers how to extract element  $(i, j)$  from \MyGrid. \MyGrid needs to be \unvcopy'ed; the first  $i - 1$  boxes need to be thrown away. The next box needs to be copied into a scratch box register, which is then \unhcopied. Again we throw away  $j - 1$  boxes, saving the next box, and tossing out  $18 - i - j$  more boxes, before returning the saved value of \MyGrid<sub>ij</sub>.

If that seems convoluted, consider how to *modify* the value stored in position  $(i, j)$ . The neglected subarrays from the last paragraph now need to be stored up and re-combined correctly. An English description of how this can be accomplished is left as an exercise to the masochistic reader.

### 3.3 A Better Way

First, let's examine the simpler problem of a one-dimensional array. What is needed is a way to refer to a unique value pointed to by the two fields *ArrayName* and *Index*.

Fortunately, TeX has a pair of primitives (\csname and \endcsname) that allow us to do just that. Everything between the \csname and \endcsname is evaluated and formed into a control sequence. This control sequence can then be used like any other.

One needs to exercise caution in setting this up. If the *Index* is a \count, then it needs to be forcibly expanded into characters by using \the. If it's a constant (or a macro), however, using \the will cause a TeX error. The solution, of course, is to examine the category of the unexpanded *Index* and only use the \the if *Index* isn't already a constant.

In TeX code, that results in:

```
\def\elt#1#2{\csname #1.\if\relax #2\the\fi #2\endcsname}
```

The \if\relax #2 handles the category testing by comparing \relax, a control sequence (which is considered to be \char256 and have \catcode=16), and #2, the *Index*. If these match, then *Index* is a control sequence and the \if should insert a \the to force expansion of #2 to a constant. The period after the #1 serves to separate the *ArrayName* from the *Index*.

There are a number of advantages to this method. First, \csname allows digits to be part of the control sequence, which is normally not allowed. This protects the array elements from direct access.

Second, since TeX has to do minimal evaluation for any array reference, this method is the fastest for both reading and writing array elements.

Third, the contents of the array can be anything, and need not be the same between elements. One control sequence can be a macro, another can be a \count register, and yet another can be a \vbox.

Fourth, the *Index* of the array can be *any string*, including alphabets. Space is allocated only for the elements that are used, and arrays can be indexed on, for example, the last names of students in a class.

### 3.4 Using \elt

Reading an array element is as simple as \elt{MyGrid}{17}. Writing to an array element is a bit more tricky, since \def\elt{MyGrid}{17}{value} will result in redefining \elt to be MyGrid and typesetting the string 17value. Using \expandafter, we can have the \elt macro expanded into the control sequence that we want to define:

```
\expandafter\def\elt{MyGrid}{17}{value}
```

To save typing \expandafter, it is convenient to define \put as follows:

```
\def\put#1#2#3{%
\edef\AINAME{\elt{#2}{#3}}%
\expandafter #1\AINAME}
```

\put is called with three arguments. The first is the version of \def to use (e.g., \def, \edef, \outer\def). The second and third are the *ArrayName* and *Index*, respectively. \put first finds the

*Array-Index Name* and stores it in `\AINAME`. It then constructs the correct version of the `\expandafter` technique, causing the token after the third argument to be taken as the value.

This will be problematic once an element has already been assigned a value, since the `\edef` will expand the desired element all the way to that value. Therefore, we need to copy over the definition of `\elt` into the definition of `\put`, which also saves us the effort of using `\AINAME`:

```
\def\put#1#2#3{%
\expandafter #1\csname #2.\if\relax #3\the\fi #3\endcsname}
```

### 3.5 Two-Dimensional Arrays and the Naval Battle

It is not difficult to modify this code to use two index variables and act as a two-dimensional array, which is what the **BATTLESHIP** program, whose listing is in Appendix C, does. The program starts off by defining `\put` and its counterpart, `\get` (which is a better name than `\elt` now that we are not putting a call to `\elt` in `\put`). It then initializes the ten-square grid to all "Z" (for zero). This version does not feature random ship placement, so the four ships are hard-coded in the next section.

The macro `\damage` will record damage to the ship whose counter is passed; it then uses `\string` to use the name of the counter as the name of the ship in the message.

The main loop follows. One of its distinguishing features is the use of `^C` to prevent the user from confusing **TeX** by inputting something other than a coordinate in the form `<letter><digit>`, with `<letter>` in the range A to J and `<digit>` any digit from 0 to 9. Another useful point is that `\csname` will make an unknown control sequence expand to `\relax`, which is how **BATTLESHIP** checks for invalid coordinates.

### 3.6 Potential Applications

There are a number of uses for array variables in **TeX**. The most significant of these is a combination of an array of records with the database routines discussed in Section 2. A file could be read into an array, manipulated as a truly random-access database, and then written out (over the original file) at the end of the session. Code to do this, as well as all the code in the appendices, is available for anonymous FTP from 18.72.1.4 (`gevalt.mit.edu`).

## 4. Conclusion

There are other games that offer interesting challenges to the **TeX** programmer. For example, a full implementation of **BATTLESHIP**, not to mention any card game, would require a fairly good pseudo-random number generator, using **TeX**'s simple integer arithmetic facilities.

Why program games, or any non-typographical code, in **TeX**? First, as can be seen from the **COKECOMM** application, routines written for games can find use in "real" **TeX** programs. Second, one is more likely to experiment if the end result is actually fun. Third, the results are more interesting to other people, who can learn from one's experiment, as this paper testifies.

And, besides, it's nice to have something to show people who still use Scribe.

## Appendix A: ANIMALS

```
1 % -----begin: animals.tex-----
2 % Animals (in TeX, no less!!!)
3 %
4 % This is the program that uses a binary tree of questions to
5 % guess the type of animal of which the user is thinking.
6 %
7 % Andrew Marc Greene
8 % <amgreen@athena.mit.edu>
9 % Student Information Processing Board, MIT
10 % March-April 1988
11 %
12 % Cleaned up April 1989
13 %
14 % Moral support (i.e., "You can't do that! Show us!")
15 % provided by the Student Information Processing Board
16 % of MIT.
17 %
18 % Instructions on running this program:
19 %
20 % tex animals
21 %
22 % Think of an animal. The program will try to guess your animal.
23 %
24 % You will be asked a whole bunch of yes/no questions. This is a
25 % spartan implementation, so answer with a capital Y or N. When
26 % the program finishes going through its tree, it will either have
27 % guessed your animal or it will ask you to enter a question that
28 % it can ask to differentiate between your animal and its guess.
29 % It will then ask you which one is 'yes.'
30 %
31 % Here's where I declare all my variables, etc.
32 %
33 % "curcode" is the current index into the data file.
34 % "temp" is a temporary holding variable.
35 % "lc" is a loop counter
36 % "ifamg" is a general-purpose flag. amg are my initials.
37 % "ifreloop" controls loop repetitions.
38 % "ifmainlooprep" controls repetitions of the main loop.
39 % "inp" is the input file.
40 % "outp" is the output file.
41 % "amgY" and "amgN" are character constants. Why I did it this way I
42 % don't remember.
43 %
44 \newcount\curcode\curcode=1\newcount\temp\temp=0\newcount\lc
45 \newif\ifamg\newif\ifyn\newif\ifreloop\newif\ifmainrep
46 \newread\inp\newwrite\outp\def\foo{ }
47 \def\amgY{Y}\def\amgN{N}
48 %
49 % The data file consists of records stored in the following format:
50 %
51 % Record Number <newline>
52 % Question <newline>
```

```

53 % If-Yes-Goto-Record Number <newline>
54 % If-No-Goto-Record Number <newline>
55 %
56 % The following routine scans the data file until it reaches the
57 % record requested in \curcode
58 %
59 \def\Scan{
60 {\loop
61   \global\read\inp to \thisrecnum
62   \ifnum\thisrecnum=\curcode\amgfalse\else\amgtrue\fi
63   \ifamg
64     \read\inp to \foo % Discard unwanted record
65     \read\inp to \foo
66     \read\inp to \foo
67 \repeat}
68 %
69 % The following routine displays the question and waits for a Y or N
70 % answer
71 %
72 \def\Query{
73 {\read\inp to \question
74 \immediate\write16{}
75 \message{\question}
76 \GetYN
77 \ifyn
78   \read\inp to \foo\global\curcode=\foo\read\inp to \foo
79 \else
80   \read\inp to \foo\read\inp to \foo\global\curcode=\foo
81 \fi
82 }
83 %
84 % The following routines deal with the user's input
85 % \vread (verbatim read) ignores <newline>s and makes <space>s normal
86 % \GetYN gets input and repeats until it gets a Y or N response.
87 %
88 \def\vread#1{\catcode`\^^M=9\catcode`\\ =12\global\read-1 to #1}
89 \def\GetYN{
90 {\loop
91   \vread{\bar}
92   \def\baz{\bar}
93   \reploopfalse
94   \if\amgY\baz\global\yntrue\else
95   \if\amgN\baz\global\ynfalse\else\replooptrue\fi\fi
96   \ifreploop
97   \immediate\write16{Hey, you! Answer Y or N, please.}
98   \message{Please enter Y or N -->}
99   \repeat
100 }
101 %
102 % The following routine is called if the "Goto-Record" is -1,
103 % meaning that the program didn't guess correctly and is clueless.
104 % It gets the new animal and the differentiating question, and
105 % modifies the data file. Actually, it makes a modified copy of
106 % the file, then copies the temporary new one over the old outdated

```

```

107 % one.
108 %
109 \def\NewAnimal{
110 \immediate\write16{Well, I'm stumped. What animal did you have in mind?}
111 \vread{\usersanimal}
112 \immediate\write16{OK. What question would let me tell the difference?}
113 \vread{\userquery}
114 \immediate\write16{Is the answer to that question Yes or No if I ask about}
115 \message{\usersanimal?}
116 \curcode=-1
117 \GetYN
118 \Scan
119 \read\inp to \lastcode\lc=\lastcode
120 \closein\inp
121 %
122 % Open up the files. These names are system-dependent. *FLAG*
123 %
124 \openin\inp=/mit/amgreen/TeXhax/animals.dat
125 \immediate\openout\outp=/tmp/animals-new.dat
126 %
127 % Read through the inp file, copying all records that don't need to
128 % be changed, outputting modified versions of the changed ones (and
129 % discarding the old), and appending the new records.
130 %
131 {\loop
132 \read\inp to \foo
133 \amgtrue
134 \ifnum\foo=\temp\amgfalse\fi
135 \ifnum\foo=-1=\amgfalse\fi
136 \ifamg\immediate\write\outp{\foo}
137 \read\inp to \foo\immediate\write\outp{\foo}
138 \read\inp to \foo\immediate\write\outp{\foo}
139 \read\inp to \foo\immediate\write\outp{\foo}
140 \amgtrue
141 \else\ifnum\foo=\temp
142 \immediate\write\outp{\foo}
143 \immediate\write\outp{\userquery}
144 \immediate\write\outp{\number\lc}
145 \global\advance\lc by 1
146 \immediate\write\outp{\number\lc}
147 \read\inp to \animal\read\inp to \foo\read\inp to \foo
148 \amgtrue
149 \else
150 \lc=\lastcode
151 \ifyn\WriteUsers\WriteAnimal
152 \else\WriteAnimal\WriteUsers
153 \amgfalse\fi
154 \fi\fi
155 \ifamg
156 \repeat}
157 \immediate\write\outp{-1}
158 \immediate\write\outp{\number\lc}
159 \closeout\outp
160 \closein\inp

```

```

161 %
162 % Now copy the temporary file over the original one
163 %
164 % These filenames are also system-dependent.          *FLAG*
165 %
166 \openin\inp=/tmp/animals-new.dat
167 \immediate\openout\outp=/mit/amgreen/Texhax/animals.dat
168 {\loop
169   \read\inp to \foo
170   \immediate\write\outp{\foo}
171   \amgtrue
172   \ifeof\inp\amgfalse\fi
173   \ifamg
174   \repeat}
175 }
176 %
177 % This routine is called by NewAnimal and writes the record for
178 % the user's new animal
179 %
180 \def\WriteUsers{
181 \immediate\write\outp{\number\lc}
182 \immediate\write\outp{Is it \usersanimal?}
183 \immediate\write\outp{0}
184 \immediate\write\outp{-1}
185 \global\advance\lc by 1
186 %
187 % This one writes the modified old animal
188 %
189 \def\WriteAnimal{
190 \immediate\write\outp{\number\lc}
191 \immediate\write\outp{\animal}
192 \immediate\write\outp{0}
193 \immediate\write\outp{-1}
194 \global\advance\lc by 1
195 \openin\inp=/mit/amgreen/Texhax/animals.dat % *FLAG*
196 %
197 % Now we get into the main routine.
198 % It simply repeats the scan-query loop until it gets a 0 (right answer)
199 % or a -1 (wrong answer, I'm stumped), and calls the appropriate routine.
200 %
201 \loop
202   \temp=\curcode
203   \Scan\Query
204   \mainlreptrue
205   \ifnum\curcode=0
206     \immediate\write16{Thank you for using Animals. I'm glad I got it right.}
207     \mainlrepfalse
208   \else
209     \ifnum\curcode=-1\NewAnimal\mainlrepfalse\fi
210   \fi
211 \ifmainlrep
212 \repeat
213 %
214 % Ah, the joys of a job well-done. We can now exit to the system, knowing

```

```

215 % that the world is a slightly better place for our efforts.
216 %
217 % The following line of code, probably the most profound in the entire
218 % program, sums up this philosophy of life in four characters. The
219 % Puritan work ethic is embodied in this amazingly meaning-laden
220 % command designed by Donald Knuth.
221 %
222 \bye
223 -----end: animals.tex-----

```

## Appendix B: CokeComm

```

1 -----begin: coke.tex-----
2 %
3 % CokeComm program for SIPB
4 %
5 % Andrew Marc Greene
6 % <amgreen@athena.mit.edu>
7 % Student Information Processing Board, MIT
8 % March 1989
9 %
10 \newif\ifamg
11 %
12 % Macros for typesetting each person's entry on the list
13 %
14 \def\person#1#2#3#4{%
15   \vbox{%
16     \hbox{%
17       \hbox to 1.5in{\strut#1\hfill}\hbox to .5in{\hfill#3\quad}%
18       \hbox to .7in{\hfill$#4\quad}\bubbles}%
19       \hbox{\hbox to 2.7in{\strut\tt #2\hfill}\bubbles}\hrule}%
20 %
21 %
22 \newbox\fivebubbles
23 \setbox\fivebubbles=\hbox{$\cal{0}\cal{0}\cal{0}\cal{0}\cal{0}$}
24 \def\five{\copy\fivebubbles}
25 \newbox\bubblebox
26 \setbox\bubblebox=\hbox{\five\five\five\five\five\qquad\five\five\five}
27 \def\bubbles{\copy\bubblebox}
28 %
29 \hsize=8in\hoffset=-.75in
30 %
31 \font\title=cmbx10 at 17.2667pt
32 \font\coltit=cmbx12
33 %
34 \headline={\hfil}
35 {\centerline{\title CokeComm Sheet}}
36 \bigskip{}
37 \vbox{%

```

```

38 \strut\hbox to 1.5in{\coltit Name\hfill}\hbox to .5in{\coltit cans\hfill}%
39 \hbox to .7in{\coltit Balance\hfill}\hbox to \wd\bubblebox{\coltit
40 Soda\hfill {\sl --Fill in Circle--}\hfill Juice}}\hrule}}
41 \footline={\hfil}
42 %
43 \def\flush{\immediate\closeout3\closein2}
44 %
45 % Here's an old friend... (from Animals)
46 %
47 \def\vread#1#2{\catcode`\\=12\global\read#1 to #2}
48 %
49 %
50 % ----- END PREAMBLE -----
51 %
52 % Last changed 20-Mar-89
53 %
54 \def\NextRecord{
55 \vread{2}{\pname}
56 \vread{2}{\obalance}
57 \vread{2}{\ocans}
58 }
59 %
60 \immediate\openout3=coke.dat
61 \immediate\openin2=oldcoke.dat
62 \newcount\balance\newcount\cans\newcount\numb\newcount\dollars\newcount\rcount
63 \loop
64 \immediate\write16{-----Next Person-----}
65 \ifeof2\amgfalse\else\amgtrue\fi
66 \ifamg
67 \NextRecord
68 \balance=\obalance
69 \cans=\ocans
70 \immediate\write16{\pname (\uname)}
71 \message{Total Deposits:}
72 \vread{-1}{\adddep}
73 \advance\balance by \adddep
74 \message{Enter sodas: }
75 \vread{-1}{\sodas}
76 \advance\cans by \sodas
77 \numb=\sodas
78 \multiply\numb by 35
79 \advance\balance by -\numb
80 \message{Enter juices: }
81 \vread{-1}{\juices}
82 \advance\cans by \juices
83 \numb=\juices
84 \multiply\numb by 45
85 \advance\balance by -\numb
86 \edef\nbalance{\the\balance}
87 \dollars=\balance
88 \divide\dollars by100
89 \multiply\dollars by100
90 \numb=\balance
91 \advance\numb by-\dollars

```

```

92 \divide\dollars by100
93 \def\sep{.}
94 \ifnum\numb<0 \multiply\numb by -1
95 \else
96 \ifnum\numb<10 \def\sep{.0}\fi
97 \fi
98 \def\bal{\the\dollars\sep\the\numb}
99 \edef\ncans{\the\cans}
100 \immediate\write3{\pname}
101 \immediate\write3{\uname}
102 \immediate\write3{\nbalance}
103 \immediate\write3{\ncans}
104 \immediate\write16{\pname : \uname : \ncans : \nbalance}
105 \person{\pname}{\uname}{\ncans}{\bal}\penalty-100
106 \repeat
107 \immediate\closeout3
108 \bye
109 -----end: coke.tex-----

```

### Appendix C: BATTLESHIP

```

1 -----begin: battle.tex-----
2 % Battleship in TeX
3 %
4 % Andrew Marc Greene
5 % MIT Project Athena
6 % and
7 % Student Information Processing Board
8 % Version 1.0 April 1989
9 %
10 % Battleship is a registered trademark of the Milton Bradley Corp.
11 %
12 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
13 %
14 % Array-handling code (modified to handle two-dimensional arrays)
15 %
16 % \put takes four arguments:
17 %       the variant on \def
18 %       the array name
19 %       the two index values
20 %
21 \def\put#1#2#3#4{\expandafter #1%      \def -- but first find the AINAME
22 \csname #2%                          begin the \csname and use the arrayname
23 .\if\relax #3\the\fi #3.%           first index
24 \if\relax #4\the\fi #4\endcsname }%   second index and end the \csname
25 %
26 %
27 % \get takes three arguments:
28 %       the array name

```

```

29 %           the two index values
30 %
31 \def\get#1#2#3{\csname #1%
32 .\if\relax #2\the\fi #2%
33 .\if\relax #3\the\fi #3%
34 \endcsname}
35 %
36 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
37 %
38 % \say is a useful shorthand for "output to screen"
39 %
40 \def\say#1{\immediate\write16{#1}}
41 %
42 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
43 %
44 % Initialize the array to all 'Z'
45 %
46 \newcount\idx\newcount\idy
47 \idx=0\idy=0
48 \loop
49 \edef\Idx{\ifcase\idx A\or B\or C\or D\or E\or F\or G\or H\or I\or J\fi}
50 \put{\def}{MyGrid}{\Idx}{\idy}{Z}%
51 \advance\idx by 1
52 \ifnum\idx=10
53 \advance\idy by 1
54 \idx=0
55 \fi
56 \ifnum\idy<10
57 \repeat
58 %
59 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
60 %
61 % Display welcome message
62 %
63 \say{Welcome to Battleship.}
64 \say{(Battleship is a trademark of Milton Bradley)}
65 \say{%
66 \say{This version uses fixed-position ships.}
67 %
68 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
69 %
70 % Position the ships. Future versions will use a random-number
71 % generator to provide a different game each time.
72 %
73 \put{\def}{MyGrid}{D}{4}{C}%
74 \put{\def}{MyGrid}{E}{4}{C}           Carrier has 5 spaces
75 \put{\def}{MyGrid}{F}{4}{C}
76 \put{\def}{MyGrid}{G}{4}{C}
77 \put{\def}{MyGrid}{H}{4}{C}
78 \put{\def}{MyGrid}{B}{1}{B}%
79 \put{\def}{MyGrid}{B}{2}{B}           Battleship has 4
80 \put{\def}{MyGrid}{B}{3}{B}
81 \put{\def}{MyGrid}{B}{4}{B}
82 \put{\def}{MyGrid}{G}{7}{D}%
83 \put{\def}{MyGrid}{D}{7}{D}           Destroyer has 3

```

```

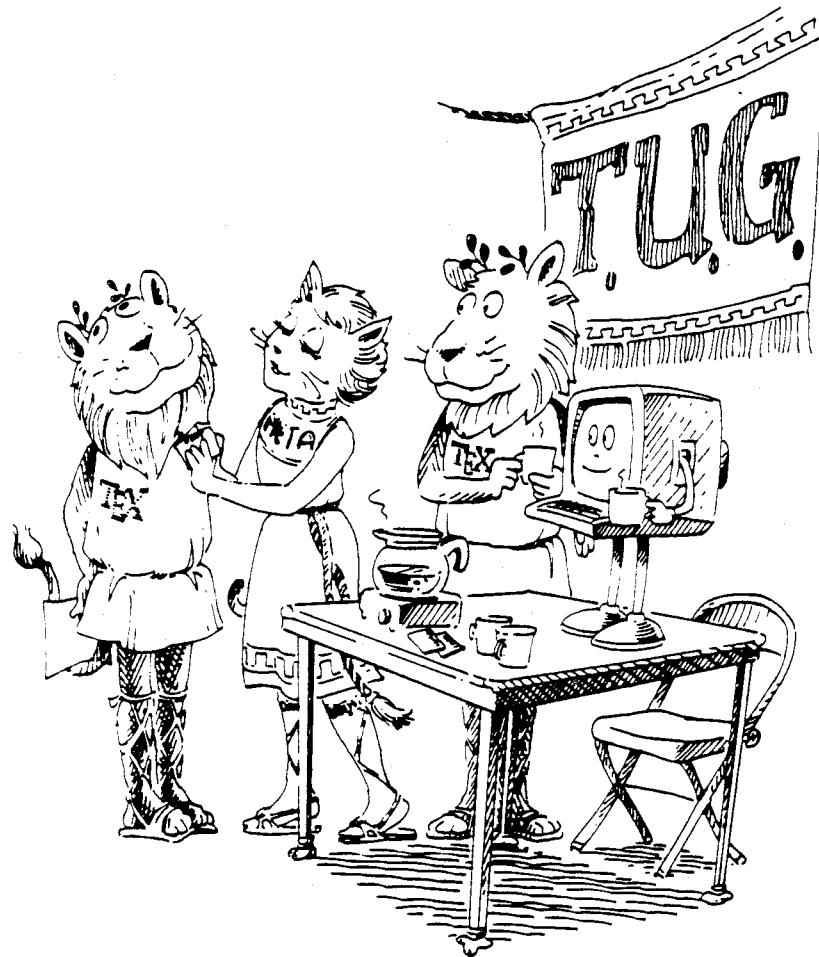
83 \put{\def}{MyGrid}{G}{8}{D}
84 \put{\def}{MyGrid}{G}{9}{D}
85 \put{\def}{MyGrid}{J}{6}{S}%
86 \put{\def}{MyGrid}{J}{7}{S} Submarine has 2
87 %
88 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
89 %
90 % Initialize some counters
91 %
92 \newcount\turns\turns=0% Number of turns it takes to win
93 \newcount\hits\hits=14% Number of hits it takes to win
94 \newcount\carrier\carrier=5% Number of hits to sink each ship
95 \newcount\battleship\battleship=4
96 \newcount\destroyer\destroyer=3
97 \newcount\submarine\submarine=2
98 %
99 %
100 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
101 %
102 % Routines to handle damage to a ship
103 %
104 % \gobble (as defined in the TeXbook, page 308)
105 %
106 \def\gobble#1{}% Remove one token
107 %
108 % \damage takes the name of a counter and damages that ship:
109 %
110 \def\damage#1{\advance#1 by -1% Lose one 'hit point'
111 \ifnum #1=0% If there are no more,
112 % Print a 'sank' message:
113 \say{You sank my \expandafter\gobble\string #1!}
114 \else% Otherwise, a 'hit' message:
115 \say{You have hit my \expandafter\gobble\string #1.}
116 \fi}
117 %
118 % Note that the above messages used \expandafter\gobble\string #1
119 % to get the name of the counter and strip the \escapechar off the
120 % front of it. The resulting string (because of the way we named
121 % our counters) is the name of the appropriate ship.
122 %
123 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
124 %
125 % Give the player a chance to specify a coordinate.
126 % If it's already been used, or is out-of-bounds, \say an error.
127 % Otherwise increment \turns and print an appropriate message;
128 % If it is a hit, decrement \hits;
129 % Whether it is a hit or not, blot the space.
130 % If \hits is non-zero, keep going.....
131 %
132 %
133 \def\defab#1#2#3^{(\def\aa{#1}\def\bb{#2})}
134 \loop
135 \message{Your turn: }
136 \read-1 to \usrinp

```

```

137 \expandafter\def\usri{^C
138 \edef\c{\get{MyGrid}{\a}{\b}}
139 \if\c\relax
140 \say{Sorry, that's not a valid coordinate.}
141 \else
142 \if\c X\say{Sorry, you already shot there.}
143 \else\advance\turns by 1
144 \if\c Z\say{You missed.}
145 %
146 % Drat! The user hit one of our ships! Record damage to the correct one.
147 %
148 \else\if\c C\damaged{\carrier}
149 \else\if\c B\damaged{\battleship}
150 \else\if\c D\damaged{\destroyer}
151 \else\if\c S\damaged{\submarine}
152 \fi% submarine
153 \fi% destroyer
154 \fi% battleship
155 \fi% carrier
156 %
157 % And record that there was a hit someplace:
158 %
159 \advance\hits by -1
160 \fi% End of the hit-or-miss section
161 %
162 % Record that this space has been shot:
163 %
164 \put{\def}{MyGrid}{\a}{\b}{X} End of the 'shoot here' section
165 \fi%
166 %
167 \fi% End of the {in}valid spot section
168 %
169 % We've finished a cycle. If there is any part of the fleet left,
170 % we go around again:
171 %
172 \ifnum\hits>0\repeat
173 %
174 % Otherwise, we display the player's score and exit.
175 %
176 \say{You have destroyed my fleet. It took you \the\turns\gobble1 turns.}
177 \bye
178 -----end: battle.tex-----

```



TEX Users Group  
Tufts University, July 20-23, 1986  
P. T. Barnum Auditorium

# A Permuted Index for $\text{\TeX}$ and $\text{\LaTeX}$

WILLIAM R. CHESWICK

AT&T Bell Laboratories  
Room 2C416  
600 Mountain Ave  
Murray Hill, NJ 07974  
`ches@research.att.com`

## ABSTRACT

The Permuted Index for  $\text{\TeX}$  and  $\text{\LaTeX}$  was written to help  $\text{\TeX}$  designers find the right command among the extensive list of  $\text{\TeX}$ , plain  $\text{\TeX}$ , and  $\text{\LaTeX}$  commands.

To prepare the index, a one-line definition was written for each command. These definitions were run through the UNIX `ptx` filter, which created an entry indexed by each key word in each definition. UNIX tools were used to massage the text, and convert the `ptx` troff-style output into  $\text{\TeX}$  macro calls.

This paper discusses some technical aspects of preparing the Index, and some of the problems encountered.

## 1. Introduction

It is easy to tell when someone is having trouble with a typesetting package: they are surrounded by heaps of nearly-identical printout. Their goal is often simple: to lower a headline or adjust spacing. When the struggle lasts more than a day, it becomes an Epic Battle.

Epic Battles seem to be a common result when a programmer attempts to stretch typesetting software beyond the novice examples. They often occur when the “safe driver” winds up on dangerous curves. The problems can be daunting, especially when the user has no  $\text{\TeX}$ nician available. *The  $\text{\TeX}$ book* [Knuth] covers everything, of course. But it is actually three books in one: a beginner’s manual and two levels of reference manual guarded by dangerous-bend signs. Some dangerous curves are vital, others extremely arcane and usually irrelevant. Which ones should be read, and which ignored, and by what level of user?

Epic Battles are not confined to  $\text{\TeX}$  — I have seen proficient `troff` users in the same state. It seems that typesetting is a fundamentally difficult task, and these battles come from an incomplete understanding of complex typesetting programs, languages, and macro packages.

### 1.1 My problem

My early Epic Battles lasted as long as two days. (I was a manager at the time, and could ill-afford the time. I took it anyway.) One day I was creating a  $\text{\LaTeX}$  style file for a newsletter, including a table of contents. When I wrote out my contents line to a file,  $\text{\LaTeX}$  added lots of strange goo to my string. I had to tell  $\text{\TeX}$  to stop expanding my macros and just write out the stuff I wanted. There *had* to be a command in the  *$\text{\TeX}$ book* somewhere, but how could I find it? I was the  $\text{\TeX}$ nician, and  $\text{\TeXHaX}$  wasn’t available. It turns out that I wanted the `\string` command. It was hiding behind a double-dangerous curve on page 22 in the  *$\text{\TeX}$ book*. (I found it in an example dug out of `latex.tex`.) At this point I started thinking about a permuted index for  $\text{\TeX}$  commands.

```
% a silly definition file
frog    green amphibian found near water
gnat    black insect found in the air
gnu     animal found on land
toad    green amphibian found on land
```

gnat—	black insect found in the	air. ....	gnat
	frog—	green	frog
	toad—	green	toad
	gnu—	animal found on land.	gnu
	gnat—	black insect found in the air.	gnat
	frog—	green amphibian found near water.	frog
	gnat—	black insect found in the air.	gnat
	gnu—	animal found on land.	gnu
	frog—	green amphibian found near water.	frog
	toad—	green amphibian found on land.	toad
	gnat—	insect found in the air.	gnat
	gnu—	animal found on land.	gnu
	toad—	green amphibian found on land.	toad
	frog—	near water.	frog
	frog—	green amphibian found on land.	toad
	gnat—	water. ....	frog

Figure 1: Some sample definitions and a permuted index of their definitions

### 1.2 A permuted index

A permuted (or keyword-in-context) index contains an alphabetical listing of each keyword in each command's definition. Figure 1 shows a short, silly example. The UNIX manual [Unix] has a permuted index for locating the correct text filter or command. There wasn't one for  $\text{\TeX}$ , so I decided to write one. I even had a shot at the first definition:

```
\string:      don't mess with the following text I am writing out
```

## 2. Building the Index

The Permuted Index is generated from a file of definitions. The file is processed by a few UNIX filters to create a file of  $\text{\TeX}$  macros. These are read into a  $\text{\LaTeX}$  file to create the Permuted Index. The index consists of three chapters summarizing the commands for  $\text{\TeX}$ , plain  $\text{\TeX}$ , and  $\text{\LaTeX}$ , and a combined permuted index of all these commands. It is well over one hundred pages long.

### 2.1 Command definitions

The first chore in creating the Index was to collect a complete list of commands. I had decided to index plain  $\text{\TeX}$  and  $\text{\LaTeX}$  commands as well as  $\text{\TeX}$ . Most of the  $\text{\TeX}$  commands came from `tex.web`. The plain  $\text{\TeX}$  commands were extracted from `plain.tex`, including a couple of internal commands that I have found useful. Barbara Beeton supplied a list that was a useful cross-check. `latex.tex`, `lplain.tex`, and `lfonts.tex` supplied the  $\text{\LaTeX}$  commands.

It has taken a long time to create and correct the definitions. I have tried to keep the style uniform and the definitions useful. A definition should give a fair description of the command's semantics using consistent keywords. For example, commands that manipulate tokens should have the word "token" in the definition. A symbol's definition contains the word "symbol", and math mode symbols have "math symbol" in the definition. Wording should be consistent: does `\parskip` define the "space", the "separation", or the "glue" between paragraphs?

```

frog----green amphibian found near water
    ↓
    s/^(\.*\_)____/\com{\1}{/
    s/^\([^.?!]\)$/\1./
    s/$/}/
↓
\com{frog}{green amphibian found near water.}

```

Note: the four underbars signify  
a single tab character

These are the sed commands that  
create the macro call.

Figure 2: Generation of the command description macros

I battered my copy of the *TeXbook* hunting down definitions, and annoyed several mathematicians about the math symbols. I suspect real TeXperts could improve on some definitions, and novices could suggest keywords that should appear in some definitions.

Most command definitions are easy. For example:

```
\time| current time of day
```

Some are not. Consider `\expandafter`, which has the following definition in the *TeXbook* (p. 213):

TeX first reads the token that comes immediately after `\expandafter`, without expanding it; let's call this token *t*. Then TeX reads the token that comes after *t* (and possibly more tokens, if that token has an argument), replacing it by its expansion. Finally TeX puts *t* back in front of that expansion.

My definition:

```
\expandafter| expand the token following the next token
```

It doesn't have to be complete, just enough to guide the careening driver.

Proficient TeXperts (and others) can certainly find definitions that miss the mark. I welcome corrections and suggestions.

## 2.2 The command definition input files

The definitions for L<sup>A</sup>T<sub>E</sub>X, plain TeX, and TeX are stored on separate files. The definitions are used in three different ways:

1. The filters in Figure 2 generate the `com` macros in three separate files. These are used in the chapters that summarize the command definitions.
2. They are processed by `ptx` to create the file with the index macro calls.
3. Lines beginning with `%%%def` are extracted and the `%%%def` is stripped off. These are special definitions that are read directly into the document before the commands and index are read.

Figure 3 demonstrates the processing steps on a very simple command definition file. Notice that a single command definition permutes to five index entries.

Each definition file has a simple format. There is one line per definition. Each line has a command, followed by a tab character, followed by the definition. Lines beginning with `%%%def` are special definitions, described below. Other lines beginning with a `%` are ignored. Figure 4 shows some actual TeX definitions.

```

% This is a sample input file
% Comments and special lines have '%' in column one
% Note: four underbar characters stand for a single tab character
frog____green amphibian found near water

↓
egrep -v "^\%" Strip comments and special entries from the file.

frog____green amphibian found near water

↓
awk -F"[\_\_]" \
    '{ print $1 "{ type }" \
        "----" "$2 }' \
    type=P Format for the ptx -r option. type is set to "P" in this example.

frog{P}____frog--- green amphibian found near water

↓
sed 's/^\([^.!?\])$/\1./' Suffix period if no punctuation present.

frog{P}____frog--- green amphibian found near water.

↓
ptx -r -f -t -w 200 -i eign

...xx "" "frog--- green" "amphibian found near water." "" frog{P}
...xx "" "" "frog--- green amphibian found near water." "" frog{P}
...xx "" "frog---" "green amphibian found near water." "" frog{P}
...xx "" "frog--- green amphibian found" "near water." "" frog{P}
...xx "" "frog--- green amphibian found near" "water." "" frog{P}

↓
sed 's:^\xx "(\[^"]*\") "(\[^"]*\") " \
      "(\[^"]*\") "(\[^"]*\") " \
      (\[^\{\}*\}){\(\[^"]*\})$: \
      \\\ptx{\2}{\3}{\5}{\6}: Convert from troff to TeX-style macros. Discard unused macro parameters. (The actual command is on a single line. It is broken up here to accommodate this annotation.)

\\ptx{frog--- green}{amphibian found near water.}{frog}{P}
\\ptx{}{frog--- green amphibian found near water.}{frog}{P}
\\ptx{frog---}{green amphibian found near water.}{frog}{P}
\\ptx{frog--- green amphibian found}{near water.}{frog}{P}
\\ptx{frog--- green amphibian found near}{water.}{frog}{P}

```

Figure 3: Sample processing of a command file

```

|\-\|    discretionary hyphen
|\|/|    italic correction
\%%\def\showspace{\tt\char`\\ }
+showspace+    space character
|\above|      fraction with rule thickness
|\abovedisplayshortskip|    extra glue above displays following short lines
|\abovedisplayskip|    extra glue above displays
|\abovewithdelims|    fraction with specified rule and delimiters
|\accent|      put an accent over the next character

```

Figure 4: Sample command definition input file

### 2.3 Editing the Index

The machine-generated permuted index in the UNIX manual was extensively hand-edited. Knuth said in the *TEXbook* that he prefers hand-editing of index entries. But I expect to make many revisions to the command definition file, and hand edits to my ptx output would be lost. I must make do with automated editing. There are a few things that help.

For one thing, not all words in a definition are useful keywords. By default, ptx consults a file named `eign` for a list of uninteresting words. The default file was not useful, since it included words like “left” and “right”, which are certainly important keywords for the Index. So in the end, I built a list based on the original file and careful examination of the Index.

I built a script to locate pairs of adjacent lines that define the same command. In such cases, something is usually wrong: perhaps a command is defined twice, or something is amiss in the definition.

Finally, there is a filter to remove lines that are sorted in a non-obvious way. Ptx sorts on the ASCII character set, but it isn’t obvious to a human where entries like % should appear. Also, actual command names tended to clutter the listing. Here are some entries that were rejected:

\$— dollar sign symbol.	.....	\$
%— percent sign (%)	.....	%
\above— fraction with rule thickness.	.....	\above
'— acute accent (é).	.....	'

I can print a full list the rejected entries to make sure nothing important is ignored.

### 2.4 Macros

The `\ptx` macro formats each definition. It used to be fairly complex when I tried to wrap long definitions around on a line. The problem led me deep into *TEX*’s paragraph-formatting algorithm, a wonderful but dangerous territory. After an Epic Battle, I gave up. A fairly simple version of the macro now truncates definitions.

Since the Index has to display the name of every *TEX* command within a line of text, it needs a good embedded verbatim environment. I took the macro from the *TEXbook*’s `manmac.tex` that uses pairs of vertical bars to delimit verbatim text. It works fine in straight text, but fails in some cases when used in an argument to a macro. Since all definitions in the Index are formatted by macros, this was quite a problem. The failure stems from the argument processing: *TEX* pre-scans the arguments with varying attention to the text’s meaning. The *LATEX* `\verb` command demonstrates the problem in Figure 5.

The first `\mac` call works. The second and third die from unmatched curly braces. The fourth doesn’t work because *LATEX*’s `\verb` is not defined with `\long`. I am not sure why the fifth call fails. It appears to be trying to process the `\newif` command. (Other `\newxxx` definitions fail as well.) Also, a few commands caused problems with my filters: double quotes would confuse the patterns that match the troff macro fields; commands containing a blank would cause unwanted permutations by ptx.

It took an Epic Battle to convince me that I couldn’t figure out a solution to the parameter

```

\long\def\mac#1{
    \begin{flushleft}
        #1
    \end{flushleft}

\mac{This is a test of string: \verb|\string|}
\mac{This is a test of string: \verb|{}|!}
\mac{This is a test of string: \verb|{|!}
\mac{This is a test of par: \verb|\par|!}
\mac{This is a test of newif: \verb|\newif|!}

```

Figure 5: Attempted verbatim in a macro call

problem using verbatim. I now use pairs of plus signs to delimit the names of macro calls that display the offending commands.

### 3. Results

The Permuted Index has been well-received. It clearly meets a need for many T<sub>E</sub>X users. I found that it was useful in its own preparation, a good sign.

The `\string` command now has the following definition in the Index:

`|\string| expand a control sequence into character tokens`

This is a reasonably accurate definition. Would it have helped me three years ago? I am not sure. Does a precise definition help the user find the command? Or should some commands be less precise but contain more familiar words to aid the novice.

Constructing the Index has taken far more work than I expected. Even so, further work is needed on several problems:

1. Better definitions are needed for some commands, especially those with difficult semantics.
2. I'd like to improve definitions for the novice.
3. I'd still like to solve the problems of line wrap in the index.
4. Foreign language versions could be useful. Also, other macro packages could be included.

### 4. Update

The Conference attendees made a number of useful suggestions. First, I am indebted to a number of people for improved definitions, especially Barbara Beeton and John Hobby. Here are a few suggestions that I will try to adopt in the next release of the Index:

1. The command names will be included in the Index, as well as listed separately as they are now.
2. The separate list of command names should be hidden in the back of the index: they are not as useful as I thought.
3. Several simple keywords, like 'page' and 'paragraph', don't yield as many entries as they should.
4. Simpler words are probably preferable to more accurate ones. For example, 'space' should probably be used where 'glue' is more precise.

## **5. Availability**

The Permuted Index is available as Computer Science Technical Report 145 from Computer Science Research at Bell Laboratories. Send inquiries to [neera@research.att.com](mailto:neera@research.att.com) or

Neera Kuckreja  
Room 2C551  
AT&T Bell Laboratories  
600 Mountain Ave  
Murray Hill, NJ 07974

After some review and revision it will be published in the *T<sub>E</sub>Xniques* series, which will be available from the T<sub>E</sub>X User's Group.

At present, the source code is not available. For now, I would like to have some idea of the actual number of copies in use. I will consider special requests for the source for foreign language and other macro package versions.

## **Bibliography**

Knuth, Donald E. *The T<sub>E</sub>Xbook*. Reading, Mass.: Addison-Wesley, 1984.

*UNIX Time-Sharing System Programmer's Manual*, Vol. 1. 10th ed. 1989.



**TEX** Users Group  
University of Washington  
August 23 - 26, 1987

# LATEX Memos and Letters

STEVE SYDORIAK

Los Alamos National Laboratory  
Group C-2, MS B253  
Los Alamos, New Mexico 87545  
sxs@lanl.gov

## ABSTRACT

Letters and memos at Los Alamos National Laboratory (LANL) are formatted in accordance with rules established in the Laboratory's *Office Procedures Manual*. LATEX style files were written to let people produce letters and memos without worrying about a complicated set of rules. Macro and template files are distributed through the Laboratory's Change Control system. A testbed of several hundred test files is used to minimize bugs in the distributed versions.

There is a choice of Computer Modern fonts or PostScript fonts. Memos and letters can be printed in Roman or typewriter typefaces. When called for, classification labels will be printed on every page. Headers on pages following the first page are compiled from information found on the first page. Letters can handle multiple addresses. Default options are provided where applicable, and error messages warn users about missing information fields.

## 1. Introduction

In 1984, a Publication Strategy Team was formed at Los Alamos National Laboratory to make decisions about future directions to be taken in text formatting and publication.<sup>1</sup> One of the decisions reached was to choose LATEX, TEX, and troff as the text formatters that would be supported by the Computing and Communications Division. LATEX was picked as the formatter of choice due to its ease of use and declarative interface. LATEX, written by Leslie Lamport, is a document preparation system based on TEX.

To encourage people to use LATEX, the Computing and Communications Division embarked on a project to provide LATEX style files for commonly used Laboratory documents, such as memos, letters, and reports. The formats of these document types are well established at LANL, and have been formalized in the *Office Procedures Manual* (OPM).

## 2. Approach

We started with the memo style because it is the most frequently used document type. We wanted to provide an easy to use, well-documented utility, capable of producing memos in accordance with the OPM. I became the designated LATEX macro writer. At the time I knew how to use and install TEX, but I had no experience with LATEX or with writing TEX macros.

My hope was to be able to write the LATEX style file using nothing but LATEX commands. I began by using LATEX's picture environment to format the memo header and the list environment to format the opening and closing lists in the memo. When I began to work on the user interface and second-page headers, however, it became obvious that I was going to have to learn a lot about writing macros in TEX. LATEX by itself is not well equipped to handle the loops, the ifs, and the token manipulation necessary for writing a sophisticated macro package. To increase my limited knowledge of TEX, I took

<sup>1</sup>This work was performed under the auspices of the U.S. Dept. of Energy.

TUG's courses on writing TeX macros, L<sup>A</sup>T<sub>E</sub>X macros, and output routines.

Studying L<sup>A</sup>T<sub>E</sub>X's macro files was essential in writing the memo macros. The macro files that I borrowed from include *latex.tex*, *1fonts.tex*, *letter.doc*, *article.doc*, and *art10.doc*. I began by copying over half of *letter.doc*, Lamport's letter style file, into the memo style file. This was useful for defining basic elements of the memo such as page layout, paragraphing, and many of L<sup>A</sup>T<sub>E</sub>X's environments. The article style files provided the figure, table, and bibliography environments. Code taken directly from the letter and article styles provided over one quarter of the final memo style file.

The *latex.tex* file is a large collection of macros that define elements of L<sup>A</sup>T<sub>E</sub>X common to all L<sup>A</sup>T<sub>E</sub>X styles. Some of *latex.tex*'s macros, such as *\@item* and *\@outputpage*, were lacking one or two features needed for memos. In these cases, I copied the macros into the memo style file, making and commenting changes to the necessary lines. During debugging stages, faulty interaction between *latex.tex* and the memo style file was common. It was helpful to temporarily move selected *latex.tex* macros into the memo style file, where *\typeout* commands could be used to follow execution.

After the memo style file had been distributed to LANL's TeX community, I began to work on the letter style file. Most of the memo style transferred directly to the letter style, and new features were added where needed.

It took about three person-months of work to complete the memo style file, and another month or two to finish L<sup>A</sup>T<sub>E</sub>X letters. A month or so of this time was devoted to learning TeX inside and out. There was perhaps a two-week investment in getting familiar with L<sup>A</sup>T<sub>E</sub>X's macro files. Several weeks were spent working on the testbed for memos and several more weeks on the testbed for letters.

In the sections below, I will talk about many of the features and some of the programming tricks used in the memo style file. I will then talk about some of the features unique to the LANL letter style file.

### 3. L<sup>A</sup>T<sub>E</sub>X Memos

#### 3.1 Layout of a L<sup>A</sup>T<sub>E</sub>X file

The format of a memo's source file is similar to the general layout of any L<sup>A</sup>T<sub>E</sub>X document.

```
\documentstyle{memo}
```

The preamble, commands such as

```
\to{...}
\from{...}
\headerfonts{...}
\enclosures{...}

.
.

\begin{document}
\opening
.
.
.
\closing
\end{document}
```

The *\documentstyle* command is used by L<sup>A</sup>T<sub>E</sub>X to load in the *memo.sty* style file, which defines all the commands used to print a memo. An optional argument asks for an eleven or twelve point font rather than the default ten point size.

The philosophy of the preamble is to free the user from having to know anything about the layout of the opening or the closing of the memo. The preamble contains all the commands that affect the opening and closing of the memo. These commands may be entered in any order. The style file ensures that everything will be printed in the right place. One doesn't need to know, for example, whether the distribution list comes before or after the list of enclosures.

All of the preamble commands are optional. When a preamble command is omitted from the preamble, one of three things happens. One, that part of the memo, such as a list of enclosures, may not be required in a memo, so it is simply omitted from the memo. Two, there may be a default for that command. For example, if no font is given for the body of the memo, the Computer Modern fonts will be used. Three, if the omitted command is required, such as the \to command, the memo will be printed with that field left blank. A warning message will be printed in the log file that tells the user how to fix up the input file.

The \opening command prints the header on the first page of the memo, including the information that tells who the memo is going to, the sender's mail stop, and so forth.

The body of the memo comes after the \opening command. One can use any of the standard L<sup>A</sup>T<sub>E</sub>X commands within the body of the memo.

The \closing command prints the part of the memo that comes after the body of the memo. This includes the sender's initials, the distribution list, the list of enclosures, and so on.

### 3.2 Opening of a Memo

Here is an example of the beginning of a memo file that uses all of the fields that can be printed in a memo header. Figure 1 shows the top of the printed memo from this example.

```
\documentstyle{memo}
\to{A. S. Harris, X-7, MS B257}
\from{S. R. Groves, X-7, MS B257}
\thru{L. S. Steele, X-5, MS B567}
\subject{Interruption of Building Schedule}
\reference{Memo, Maestas to Platz, March 12, 1989, CT-6-88}
\reference{Subject OPM-1-3, Office Procedures Manual}
\mailstop{B257}
\telephone{7-4555}
\symbol{X-7}
\serialnumber{111}
\headerfonts{postscript}
```

## Los Alamos

Los Alamos National Laboratory  
Los Alamos, New Mexico 87545

## memorandum

TO: A. S. Harris, X-7, MS B257

DATE: June 19, 1989

THRU: L. S. Steele, X-5, MS B567

MAIL STOP/TELEPHONE: B257/7-4555

FROM: S. R. Groves, X-7, MS B257

SYMBOL: X-7-111

SUBJECT: INTERRUPTION OF BUILDING SCHEDULE

REFERENCES: 1. MEMO, MAESTAS TO PLATZ, MARCH 12, 1989, CT-6-88  
2. SUBJECT OPM-1-3, OFFICE PROCEDURES MANUAL

A series of unfortunate events has interrupted the building schedule of the RTX project, causing us to request a postponement of the work shown

Figure 1: Opening and first paragraph of a memo

Notice that the subject line is always printed in uppercase. In this example, two references were mentioned. When one reference is given, the word "REFERENCE" is printed rather than "REFERENCES" (see Figure 2). The \thru, \serialnumber, and \reference information is optional. When the \serialnumber is not given, the \symbol will be printed by itself, without the trailing dash.

When the \thru command is not used, a different style of header is printed. Figure 2 shows an example of a memo printed in the “thru-less” style. It is worth noting that the user does not have to be aware that there are two different styles of memo headers. The style file selects the right header on the basis of the information given in the preamble.



TO: A. S. Harris, X-7, MS B257 DATE: August 1, 1989  
FROM: S. R. Groves, X-7, MS B257 MAIL STOP/TELEPHONE: B257/7-4555  
SUBJ: X-7-111  
SUBJECT: INTERRUPTION OF BUILDING SCHEDULE  
REFERENCE: MEMO, MAESTAS TO PLATZ, MARCH 12, 1989, CT-6-88  
A series of unfortunate events has interrupted the building schedule of the RTX project, causing us to request a postponement of the work shown on Job Order J87940.  
The recent period of inclement weather has delayed delivery of the metal torus being trucked in from

Figure 2: Opening of a memo with roman typeface and no \thru

### 3.3 Fonts

There are four possible arguments to the `\headerfonts` command: `tex`, `postscript`, `lafonts`, and `memopaper`. Each does its best to mimic the pre-printed memo paper that secretaries push into a typewriter. When `TEX` fonts are used, the Computer Modern fonts closest in appearance and size are used. While this provides only a rough approximation of the correct fonts, this option is the default for the `\headerfonts` command. Computer Modern fonts allow printing of a memo with a bare-bones installation of `TEX`.

When \headerfonts{postscript} is used, PostScript fonts are used to print the header. The hollow letters used to print "Los Alamos" are created by using a \special command to include PostScript code into the dvi file. In order for this to work, ArborText's DVIPS must be used to create a PostScript file. At Los Alamos we have standardized on DVIPS as the PostScript driver. All the commonly used systems (VAX/UNIX, VAX/VMS, Sun, Apollo, and IBM PC) have site licenses for DVIPS.

The best looking fonts for Los Alamos headers come from The Metafoundry in Dublin, Ohio; these are accessed by using the \headerfonts{lafonts} command. We bought a site license for these fonts so that all our users have access to them. If a user has installed these fonts, a header that is indistinguishable from the pre-printed memo paper can be printed.

Users also have \headerfonts{memopaper} available to print their memos on pre-printed memo paper. In this case a piece of memo paper is put into the printer. Of course, there is usually a problem when trying to print onto a pre-printed piece of paper: the L<sup>A</sup>T<sub>E</sub>X output will very rarely line up just right with the "TO:", the "FROM:", and so forth. To overcome this problem, the user can specify a horizontal and vertical correction to match any particular combination of printer and pre-printed header paper.

### **3.4 The Body of the Memo**

Any command mentioned in the Lamport's L<sup>A</sup>T<sub>E</sub>X manual can be used in the body of a LANL memo. The first version of our memo style followed Lamport's letter style in leaving out the table, figure, and bibliography environments. Some of the first calls I got were from people asking why they could not use the figure and bibliography environments in their memos. So I added them into the second version.

There are two styles of memos at LANL. The commonly used one is single-spaced with block

paragraphs. There is another one for memos of ten lines or less that is double-spaced with indented paragraphs.

For the body of the memo, a user can specify either PostScript or Computer Modern fonts. The user can also choose between the Roman or typewriter typefaces. Some users feel (strongly) that a memo that looks like it came out of a typewriter looks more professional than one that came out of a printer. Others feel that the typewritten look is humbler and less extravagant looking, therefore better for obtaining grants and funds. So I provided the typewriter typeface.

One of the more difficult challenges in writing LANL's style files was to accommodate the typewriter typeface. In regards to everything from hyphenation to vertical spacing to list formats, *TeX* and *L<sup>A</sup>T<sub>E</sub>X* are designed with a variable spaced font in mind. To truly emulate a typewriter, it is necessary to have all the lines spaced evenly down the page and characters spaced evenly across the page. This requires an *\if* statement and parallel code at nearly every turn.

### 3.5 Closing of a Memo

Here is the input that creates a memo with most of the options that can be printed in the closing of a memo. Figure 3 shows the closing of the printed memo from this example. Not shown in this example are command calls that print "Enc. a/s" and "Attachments a/s."

```
\originator{srg}
\typist{jak}
\signature{Betty J. Donaldson}
\signer{bjd}
\approval{R. J. O'Conner \\ ST Division Leader}
\cy{T. J. Benton, WX-5, MS G780 \\ A. L. Salazar, X-7, MS B257}
\enc{Graph, Dan K. Lookce, TP-3, MS B882}
\enc{Drawing No.\ 3988-R}
\attachments{Graph, Fallout vs.\ Image Time, TP-3, MS B881}
\attachments{Memo, Maestas to Platz, March 12, 1989, CT-6-88}
\distribution{K. C. Jordan, C-5, MS B775 \\
    M. K. Solomon, TP-1, MS B233 \\
    A. J. Gomez, TP-2, MS B234 }
```

*L<sup>A</sup>T<sub>E</sub>X*'s list environment provides an easy set of tools for formatting the various types of lists used in the memo macros. The list environment does not keep track of widow and orphan lines, however, so code was added to *L<sup>A</sup>T<sub>E</sub>X*'s *\citem* definition to keep the first two and last two items of a closing list together on the same page. There are also page breaking commands that allow the user to begin a specified closing list on a new page.

### 3.6 Classification Labels

At Los Alamos some memos are classified secret, confidential, or unclassified.<sup>2</sup> When using a typewriter, paper is used that has the classification marked on the top and bottom of the page. If *TeX* fonts are being used in the memo, the font cmss10 scaled *\magstep5* is used to approximate the pre-printed classification paper. If PostScript fonts are used, a *\special* command uses PostScript code to simulate the tall, skinny, and bold font used on pre-printed classification paper. DVIPS's overlay facility is used to overlay the classification labels on each page. A 29-point Helvetica font is compressed horizontally and then overstruck to give the desired look.

## 4. Token Formats

Many of the lists used by the memo commands can have multiple entries. The list of items can be specified in one command call with double backslashes separating items on the list. A list of items can also be built up by multiple calls to the same command. For example:

```
\cy{R. T. Smith, C-2, MS B263 \\ Files}
```

<sup>2</sup>For more on treating classification labels, see Pollari's article in the 1988 TUG Conference Proceedings, pp. 43, 48-49, in *TeXniques* 7 -Ed.

We will not need your services until the torus has arrived and has been installed by our technicians. Work may then proceed as outlined in the original set of milestones.

Betty J. Donaldson

BJD-SRG:jak

APPROVED BY:

R. J. O'Conner  
ST Division Leader

date

Enc. 1. Graph, Dan K. Lookce, TP-3, MS B882  
2. Drawing No. 3988-R

Attachments:

1. Graph, Fallout v.s. Image Time, TP-3, MS B881  
2. Memo, Maestas to Platz, March 12, 1989, CT-6-88

Distribution:

K. C. Jordan, C-5, MS B775  
M. K. Solomon, TP-1, MS B233  
A. J. Gomez, TP-2, MS B234

Cy: T. J. Benton, WX-5, MS G780  
A. L. Salazar, X-7, MS B257

Figure 3: Last paragraph and closing of a memo

and

```
\cy{R. T. Smith, C-2, MS B263}  
\cy{Files}
```

are equivalent. A macro named `\@toksinput` takes care of building up these lists. When a command such as the `\cy` above is called, it in turn issues a command of the form:

```
\@toksinput{cy}{R. T. Smith, C-2, MS B263 \\ Files}
```

`\@toksinput` then increments a counter that keeps track of how many `\cy`'s there are on the list. The list items are appended to a token variable that has the form:

```
\@tokscy={R. T. Smith, C-2, MS B263\\Files}
```

## 5. Parsing Token Variables

The construction of second-page headers will illustrate the parsing of variables that have embedded newlines. Second-page headers mention everybody's name that the memo is addressed to. On page one, the addressee's name, group, and mail stop are given, separated by commas. On the second-page headers however, only the name is listed. For example, if the addresses are specified by:

```
\to{A. B. Carroway, Z-1, MS B234 \\  
D. E. Fritz, Y-2, MS B345}
```

the second-page header appears as in Figure 4. The internal representation of the "TO" field for this example would be the token variable:

```
\@toksto={A. B. Carroway, Z-1, MS B234\\D. E. Fritz, Y-2, MS B345}
```

The parsing of this token variable is done by macros based on one of Knuth's dirty tricks mentioned in the *TeXbook* (1984:375). This dirty trick is called `\btest`. It searches for an asterisk in a token variable and returns true or false depending on the success of the search.

The recent period of inclement weather has delayed delivery of the metal torus being trucked in from Brown and Armbruster Metalworks, Inc., in

Figure 4: Second-page header

Two clones of \btest are needed to parse \@toksto; one parses for the double backslash token and the other parses for a comma. Besides reporting the success of the search, the macros store the parts of the token variable that come before and after the search token.

To parse \@toksto, a loop is executed until all the people's names have been extracted from \@toksto. First the macro \atnewlinepull is called (see below) with \@toksto as its argument. The tokens before the \\ in \@toksto are put into the token variable \@tksbeforenewline and the tokens after the \\ are put into \@tksafternewline.

```
\def\@atnewlinepull#1{\let\@nlssave=\\\let\\=\relax
  \expandafter\@atnewlinepullone\the#1\\@empty
  \if@newline\expandafter\@atnewlinepullthree\the#1@empty
  \else\fi\let\\=\@nlssave}%
\def\@atnewlinepullone#1\\{\@tksbeforenewline={#1}%
  \futurelet\next\@atnewlinepulltwo}%
\def\@atnewlinepulltwo#1@empty{\@tksafternewline={#1}%
  \ifx\@empty\next\@newlinefalse
  \else\@newlinetrue\fi}%
\def\@atnewlinepullthree#1\\{\@tksbeforenewline={#1}%
  \futurelet\next\@atnewlinepullfour}%
\def\@atnewlinepullfour#1@empty{\@tksafternewline={#1}%
  \ifx\@empty\next\@newlinefalse
  \else\@newlinetrue\fi}
```

For example, if one has the expression:

```
\@toksto={A. B. Carroway, Z-1, MS B234\\D. E. Fritz, Y-2, MS B345}
```

then executing the command \atnewlinepull\@toksto will cause the names to separate into "before" and "after" components:

```
\@tksbeforenewline={A. B. Carroway, Z-1, MS B234}
```

and

```
\@tksafternewline={D. E. Fritz, Y-2, MS B345}.
```

A macro named \atcommapull (see below) is then called with \@tksbeforenewline as its argument.

```
\def\@atcommapull#1{\expandafter\@atcommapullone\the#1,\@empty}%
\def\@atcommapullone#1,{\@tksbeforecomma={#1}%
  \futurelet\next\@atcommapulltwo}%
\def\@atcommapulltwo#1@empty{\ifx\@empty\next\@commafalse
  \else\@commatrue\fi}%
\def\@atcommapullthree#1@empty{\@tksbeforenewline={#1}%
  \futurelet\next\@atcommapullfour}%
\def\@atcommapullfour#1@empty{\@tksafternewline={#1}%
  \futurelet\next\@atcommapullfive}%
\def\@atcommapullfive#1@empty{\@tksbeforenewline={#1}%
  \futurelet\next\@atcommapullsix}%
\def\@atcommapullsix#1@empty{\@tksafternewline={#1}}
```

The \atcommapull macro definition is a bit simpler than \atnewlinepull, since it is not necessary to determine \@tksaftercomma. Executing \atcommapull\@tksbeforenewline in the above example leaves:

```
\@tksbeforecomma={A. B. Carroway}
```

which is saved into another token variable named `\@toksheader`. This finishes the first time through the loop. After a second time through the loop,

```
\@toksheader={A. B. Carroway\\D. E. Fritz}
```

which can then be used to print the second-page header.

The height of the second-page header is calculated and is used to set the values of `\topskip`, `\headsep`, and `\textheight`. These values change between the first page and the second page, so they are set to their new values after outputting page one in L<sup>A</sup>T<sub>E</sub>X's `\@outputpage` macro.

## 6. L<sup>A</sup>T<sub>E</sub>X Letters

L<sup>A</sup>T<sub>E</sub>X letters are quite similar to L<sup>A</sup>T<sub>E</sub>X memos. Most of the information needed on the first page header of a letter is entered into the file the same way as for memos. The closing portion of a letter is virtually identical to that of a memo.

Nevertheless, L<sup>A</sup>T<sub>E</sub>X letters do have several features that memos do not offer. For example, it is possible to adjust the width and height of the body of a letter to give it a balanced look. Two interesting features in L<sup>A</sup>T<sub>E</sub>X letters are the ability to print multiple addresses and the ability to print mailing labels.

### 6.1 Multiple Addresses

In letters, a single address is specified like this:

```
\to{John Binnington, Manager\\
    Technical Information Division\\
    Brookhaven National Laboratory\\
    Associated Universities, Inc.\\
    Upton, Long Island, NY 11973}
```

A single address is printed on the left side of the header, as shown in Figure 5. For multiple addresses, one `\to` command is used for each address. If two addresses are called for, they will both be put on the left side of the header. When three or more addresses are called for, the addresses are split into two columns. Half of the addresses go in the left column and the other half go in the right column. If there is an odd number of addresses, the extra address goes in the left column. Figure 6 shows a letter with five addresses:

**Los Alamos**  
Los Alamos National Laboratory  
Los Alamos, New Mexico 87545

DATE: June 15, 1989  
IN REPLY REFER TO: IS-DO-82-144  
MAIL STOP: P360  
TELEPHONE: (505) 667-4355  
(FTS) 843-4355

John Binnington, Manager  
Technical Information Division  
Brookhaven National Laboratory  
Associated Universities, Inc.  
Upton, Long Island, NY 11973

Dear Sir:

We are happy to announce the dates for the third annual Library Management Meeting, March 23-25, 1983. The meeting will be held here at the

Figure 5: Letter with one address

The user can override a multiple address layout by using `\lefttto` and `\righttto` to specify which addresses should go in which column.

# Los Alamos

Los Alamos National Laboratory  
Los Alamos, New Mexico 87545

DATE: June 15, 1989  
IN REPLY REFER TO: IS-DO-82-144  
MAIL STOP: P360  
TELEPHONE: (505) 667-4355  
(FTS) 843-4355

John Binnington, Manager  
Technical Information Division  
Brookhaven National Laboratory  
Associated Universities, Inc.  
Upton, Long Island, NY 11973

R. R. Dickison, Manager  
Library System  
Oak Ridge National Laboratory  
Oak Ridge, TN 37830

Juanita L. Garcia  
Library Science Specialist  
Technical Library  
Sandia National Laboratories  
Albuquerque, NM 87115

Roy Nielson, Head  
Library Department  
Lawrence Berkeley Laboratory  
University of California  
Library, Bldg. 50, Room 134  
Berkeley, CA 94720

Wayne Snyder, Manager  
Technical Information  
Batelle-Northwest  
Pacific Northwest National  
Laboratory  
P.O. Box 999  
Richland, WA 99352

Ladies and Gentlemen:

We are happy to announce the dates for the third annual Library Management Meeting, March 23-25, 1983. The meeting will be held here at the

Figure 6: Letter with five addresses

## 6.2 Mailing Labels

The way I do mailing labels is similar to the way Lampert does them. I have added macros so that the user can print onto labels of any size, so long as the labels are in a two-column format. If the following is specified:

```
\maillabelheight{1in}  
\maillabeltopmargin{.5in}
```

the labels will be printed on label paper that has one-inch high labels, with the top label being one half inch from the top of the paper.

The easiest way to print labels is to print them onto a regular piece of paper with a laser printer. If you like the way the labels look, you can then copy them onto any label paper that can be used in a copying machine. Most letters only have one address and a return address. After Xeroxing and peeling the two labels off, you can turn the label paper end to end and Xerox one more set of labels. This leaves you with a bunch of unusable labels in the middle of the label paper. To overcome this problem, there is another command that lets the user specify how many labels have been peeled off the sheet of labels. The command \skiplabels{2} will print the first label on the third label of the mailing label paper.

## 6.3 When an Inch Is Not an Inch

Unfortunately, if you try to print a sheetfull of one-inch high labels on Apple LaserWriter or QMS-800 PostScript printers that we have around here, the printing at the bottom of the page will be higher on the page than the labels. This is because these two printers do not accurately print a vertical inch: if you ask for 9 inches, these printers will actually print 8.8 inches. To compensate for this we use:

```
\maillabelheight{1.01in} % For our QMS-800  
\maillabeltopmargin{.6in} % For our QMS-800
```

The newer printers that we have, such as the Varityper, the Dataproducts, and the NeXT printers do not need this correction.

#### 6.4 Printing and Parsing Address Tokens

From an internal point of view, the token variable for addresses is different from the other token variables. An \endto token is inserted after each address. For example, if the following two addresses are called for:

```
\to{Person One}\Town One}
\to{Person Two}\Town Two}
```

the information is stored in a token variable like this:

```
\@toksto{Person One}\Town One\endto Person Two\Town Two\endto}
```

Address tokens are parsed to retrieve individual lines in the addresses for two reasons. For one thing, the second-page header uses, by default, the top line of each address. For another, if a line of an address is too long to fit on a single line, the remainder will be indented on the next line (see Figure 6). This is accomplished by printing each line of an address as an item in L<sup>A</sup>T<sub>E</sub>X's list environment.

To print the addresses, a loop within a loop is executed. The outer loop calls a macro named \toendtopull that is quite similar to \tonewlinepull, described above. \toendtopull stores the next address to be printed in a token variable named \tksbeforeendto. An inner loop then pulls out individual lines from \tksbeforeendto by calling \tonewlinepull. Without going into gruesome details, the general form of this is as follows:

```
\begin{list}{...}
  % Goes thru longloop once per address.
  \longloop
  \if (there's another address)
    ...
    % extract next address
    \item
    % Goes thru innerloop once per printed line.
    \innerloop
    ...
    % extract next line
    ...
    % Print the next line.
    \the\tksbeforenewline
    ...
  \if (there are more lines in this address)
    \par
    \innerrepeat
  \longrepeat
\end{list}
```

The macros \longloop and \innerloop are clones of plain.tex's \loop definition. The trick is to have different names for all the macro names used by \loop, \longloop, and \innerloop.

#### 6.5 The mlb File

Ordinarily, the aux file would be used to store the mailing label information for printing at the end of the job. However, the figure, table, and bibliography environments also use the aux file in a way that is incompatible with mailing labels. So I defined a new file suffix mlb to indicate the file that is used for printing mailing labels.

### 7. UNIX Tools

I use the Emacs editor on a Sun workstation with ArborText's Preview to write and debug the style files. Emacs makes it easy to write editor macros for doing such things as commenting out lines or adding various kinds of \typeout commands.

Sccs, UNIX's source code control utility, is a great help for keeping track of incremental changes in the style files. Sccs allows you to recover previous versions of a file. There were several times when

attempted methods went awry. Sccs made it easy to go back to the latest version that worked and to try a better method.

During development of the *sty* file, I indent to indicate structure, and I comment as much as possible. When distribution time comes around, the *sty* file is copied to the *doc* file. I use sed, UNIX's stream editor, to automatically strip unwanted indentation and comments from the *sty* file.

## 8. Testing

I have 180 test files for the memo style and 220 for the letter style. I try to keep the individual tests simple, so there's not too much to look at in any test. I test all of Lamport's *LATEX* commands in addition to all of the commands specific to the memo or letter style. I print the purpose of each test in four places: in a comment at the top of the file, at the top and bottom of the printed output, and in the log file.

The test directory is divided into sub-directories to group various kinds of tests. For example, there are directories for testing headers, page breaks, error messages, and classification labels. There is a UNIX C shell script that will run *LATEX* on all the test files. Another script collates all the log files, filtering out almost everything except filenames and error messages. Most *dvi* files can be previewed on the Sun, although some *dvi* files contain PostScript code that cannot be previewed on a Sun.

When the style file gets to an almost debugged stage, small changes are often made that affect the output of only a few of the test files. To avoid having to re-preview all the test files, I wrote a utility DVIDIFF based on UNIX'S diff command that compares *dvi* files. The differences are printed out in clumps of ASCII lines corresponding to places in the *dvi* file that are ASCII, and lines of hex corresponding to places in the *dvi* file that are binary. When a difference between two *dvi* files is found, DVIDIFF shows where to look in the printed output.

## 9. Version 2.09 ≠ Version 2.09

Using *LATEX* commands in my style files turned out to be a pain in the neck. When I upgraded *LATEX* from version 2.09 <9 Mar 1987> to version 2.09 <4 Aug 1988>, *LATEX*'s definition of \parbox changed, and vertical space disappeared from my memo output in several places. It was not so difficult to fix up the problem, but it did point out the vulnerability of *LATEX*. Many users are not even aware when their version of *LATEX* changes, and therefore cannot be expected to look around for new versions of the memo and letter style files. It sure would be nice if there were a trip*LATEX* to bring *TEX*'s stability to the world of *LATEX*.

## 10. Documentation and Distribution

At LANL, we have a group that writes local documents. They have written documents that explain how to use *LATEX* memos and letters. The documents include instructions for usage and installation, a list of error messages, and sample memos and letters.

Files are distributed for all the commonly used operating systems. The distributed files are the *sty* file, the *doc* file, a sample input file that uses most of the available commands, and a *readme* file with installation instructions.

At LANL, software changes for the commonly used operating systems are regulated through the Change Control system. The Computing and Communications Division distributes a monthly *ICN Change Bulletin* that announces new utilities and new versions of existing utilities. Software changes on mainframes are installed on a given day of the month. System managers of workstation networks have more flexibility as to when they want to install software.

The Change Control system makes it easy to announce software, since one *Bulletin* article announces a new version of the style file for most of the major system. The files are available to most users through our Common File System, which can be accessed by most of our computers. The exception to this rule is the IBM PCs. Style files for IBM PCs are distributed on diskettes through an organization called PC STORE.

## 11. Conclusions

*LATEX* memos and letters are useful utilities to have in an organization. Almost everybody writes a memo or a letter once in a while. *LATEX* memos and letters are welcomed by *LATEX* users and encourage

others to learn  $\text{\LaTeX}$ .

$\text{\LaTeX}$  is a popular document system for many reasons. The user works on the content and leaves the formatting to  $\text{\LaTeX}$ ; the user doesn't have to continually fuss with fonts and vertical spacing;  $\text{\LaTeX}$  macros are consistent across document types; most desired document features are there and well documented.

These strengths were compelling enough to make it worthwhile for us to provide our users with  $\text{\LaTeX}$  memos and letters. Of course, there is a price: it takes quite a bit of effort to taylor a style file to the numerous  $\text{\LaTeX}$  commands and environments. On the plus side, however,  $\text{\LaTeX}$  encourages the macro writer to go the extra nine yards in writing a complete and easy-to-use command set.

## Bibliography

Knuth, Donald E. *The TeXbook*. Reading, Mass.: Addison-Wesley, 1984.

Lamport, Leslie.  *$\text{\LaTeX}$ : A Document Preparation System*. Reading, Mass.: Addison-Wesley, 1984.

# Inserts in a Multiple-Column Format

GARY BENSON, DEBI ERPENBECK, AND JANET HOLMES

Gary Benson  
MS D417  
Los Alamos, NM 87545  
`gsb@lanl.gov`

Debi Erpenbeck  
MS D418  
Los Alamos, NM 87545  
`dje@lanl.gov`

Janet Holmes  
MS D417  
Los Alamos, NM 87545  
`jah@lanl.gov`

## ABSTRACT

A few years ago, we approached the problem of placing inserts (or "objects") automatically in a multiple-column format. Among the issues we encountered were enabling users to place an object wherever they desired following a textual reference (the "place-it-here" option); placing objects that spanned columns into a page while maintaining text flow in a one-pass system; and creating a text-flow pattern that followed the style of the physics journal *Physical Review*. We use a stack-like method of storing objects to create solutions to these problems.

### 1. Introduction

In October 1986 we became intrigued by the problems caused when inserts or objects, such as figures and equations, must be placed within a multiple-column TeX format. Such text/figures/equations formats are common in documents we process at Los Alamos National Laboratory in our composition section. Therefore, we felt we had to find a (hopefully simple) way to solve the figure-placement (or insert-placement) problem within TeX. The goal of our first attempts was to place inserts at the top and bottom of pages in a three-column report format.

We were able to complete the TeX coding required to solve this first problem in about forty hours and produced the desired results. However, reports designed by various groups at Los Alamos required different insert-placement rules — frequently ones that would not permit the simple top or bottom insertion. At this point, therefore, our thoughts turned to the larger problem of placing inserts either after a textual reference or down and to the right of a reference. Because of other work assignments, we produced only design notes and sketches at first. In February 1989, however, we received a request from the editors working on a journal at Los Alamos for TeX macros that would direct the placement of inserts according to the style of the *Physical Review* in a two-column format. This gave us the opportunity to work with insert-placement rules considerably more complex than our original problem.

### 2. Definition of the Problem

The *Physical Review* format requires that several basic rules be followed. These are:

1. Material must be easily included into multiple-column (in this case two-column) text.
2. The insert must always follow the textual reference to it, which means that, depending on its position within the article, it could conceivably fall at the top of a page, at the bottom, or anywhere

0 — the page number that is currently being processed  
user specified — in which case the program will try to place the insert on the specified page  
*Note:* this does not apply to “place-it-here”-type objects; see item 6 below.

4. `\endmulcolinsert`. Shows where the inserted object ends and places the object on the stack.
5. `\balanceatthispoint`. This macro balances the material that TeX is going to put on the current page between two columns and makes a top insert of the balanced material.
6. `\refinsertadj=a dimension value`. Use this macro to position a “place-it-here” object after its reference. We found that, in some cases, an object is not placed correctly (for example, between the word “Figure” and the number “1” with each on a separate line). This macro is a “fudge factor” to ensure proper placement in such cases, and goes before the `\beginmulcolinsert` macros.

The above macros required a few additions to format articles according to the *Physical Review* specifications. Notably, *Physical Review* uses reading flow indicators, or “reader bars,” to guide the reader’s eye in the direction of text when the text flow is interrupted by a figure or equation that spans two columns. We show an example of *Physical Review* page layout using these reader bars in Sec. 4, examples 3 and 4. The additional macros supporting a *Physical Review*-style format are given below.

1. `\phyrevtrue`. This draws reading flow indicators when a `\balanceatthispoint` macro is encountered. It draws \_\_\_\_\_ to indicate that the text above the line is a section and draws \_\_\_\_\_ to indicate that the text below the line is a section.
2. `\phyrevfalse`. This cancels the effect of `\phyrevtrue`.
3. `\noclubtextlines{n}`, where  $n$  is the number of lines of text (“club lines”) allowed to sit by themselves in a column below an insert. At Los Alamos, we usually require five or more lines to appear in such a position, so the `\noclubtextlines` argument would be 5.
4. `\copy\upreadbarbox`. This sequence of commands creates an upward-pointing reader bar at the position specified in the text.
5. `\copy\downreadbarbox`. Likewise, this sequence of commands creates a downward-pointing reader bar at the position specified in the text.

### 3.4 Design and Implementation Concerns

The coding for the macros listed in Section 3.3 required about 100 hours. Up to ten objects can be stored on the stack at one time. Only versions of TeX that have their registers doubled can process code using these macros; i.e., Sun and VAX systems.

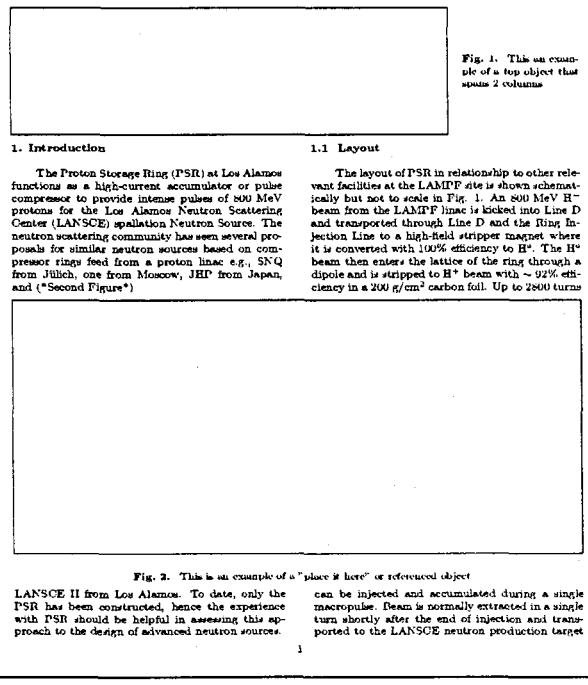
Early in the design we realized that users may mistake features of these macros for bugs and that, to avoid user confusion, we’d better document what the code does *not* do as well as what it does do. For example, take the case of an object that is referenced in the right column but spans two columns. This situation will process correctly only if there are no multiple-column-spanning objects already in place that had been referenced in the left column. (This is a one-pass system, remember.) With this macro package, the program will place the object referenced in the right column on the next page.

We divided the processing code so that it would be easy to introduce new coding and allow fault (bug) detection. Some of the bugs we suspected were not bugs, but resulted from our inability to foresee results. There were, as usual, a number of typos and “why did I do that?”-type problems. However, there are still several elusive bugs that need to be found. For example, the following conditions occurring together in a document cause some text to be dropped:

1. The first page of the document contains a `\balanceatthispoint`, which creates an object containing text and graphics that is placed on the stack as a top object. The next object is an equation.
2. The second page of the document contains two top and two bottom objects. `\noclubtextlines` is set to 5 to avoid club text at midpage.
3. The third page, like the first, contains a `\balanceatthispoint`, creating an object containing text and graphics that is placed on the stack as a top object, and an equation follows as the next object. However, text is suddenly missing from the top of this page. We’ve worked around this situation by placing the two top and two bottom objects from the second page within a full-page object, but this fix seems clumsy.

## 4. Examples

### 4.1 Example 1: A Simple Example



This is a simple example of a page with two inserted objects. The first is a top object and is placed automatically at the top of the page. The second is a "place it here" (or referenced) object that is placed about midway on the page after its reference in the text. (Note that the object is placed correctly after its reference.) The reference point is flagged by "(Second Figure)". The first part of the TEX code for this example is given below.

The first object's coding was placed before the start of the text. By coding this way, a compositor can group all the top and bottom objects in a report together in one place. Compositors like this feature because they tend to think of object placement in terms of "Fig. 1 at the top of page 5, Picture 3a at bottom of page 7," etc. However, because the number of objects that can be placed on a stack at any one time is ten, a report with more than ten inserts will require more than one grouping of top and bottom objects. Only the referenced objects must be embedded within text.

```
%  
%      ex1.tex  
%  
\vsize=50pc          % Set the vsize so macros know its value  
\hsize=39pc          % Set the hsize  
\input mulcol2        % Load the mulcol/inserts macros  
\input ex_macros       % Load the macros for the examples  
\parskip=0pt  
  
\beginmulcol{2}{18.5pc}    % Begin multiple columns  
%  
% First figure, a top object  
%  
\beginmulcolinsert{t}{2}{1}{1}  
\lfigure{10pc}{30pc}{This an example of a top object that spans 2  
columns}{}  
\endmulcolinsert  
  
\leftline{\bf I.--INTRODUCTION}  
The Proton Storage Ring (PSR) at Los Alamos functions as a high-current  
accumulator or pulse compressor to provide intense pulses of 800 MeV  
protons for the Los Alamos Neutron Scattering Center (LANSCE) spallation  
Neutron Source. The neutron scattering community has seen several  
proposals for similar neutron sources based on compressor rings feed from a  
proton linac e.g., SNQ from J\" ulich, one from Moscow, JHP from Japan, and  
(*Second Figure*)
```

A portion of the log for example 2

```
The insert starts in the left column
and only spans the left column
>> Placing bottom insert number 4
The insert starts in the right column
and spans only the right column
>> Removing insert 3, from stack
>> Removing insert 4, from stack
>> [Finished page 2]
[3]
>> Processing insert number 9, to be placed on page 3
>> Processing insert number 10, to be placed on page 3
>> Processing insert number 11, to be placed on page 3
>> Processing insert number 12, to be placed on page 4
>> Processing insert number 13, to be placed on page 4
>> Processing insert number 14, to be placed on page 4
>> Insert 9, and reference moving to page 4
>> Insert 10, and reference moving to page 4
>> Insert 11, and reference moving to page 4
>> Insert 8, referenced in left column
    and spans the right column
>> Placing referenced insert number 8

Overfull \vbox (0.38889pt too high) has occurred while \output is active
>> Removing insert 8, from stack
>> [Finished page 3]
[4]
>> Insert 11, will not fit in left column
    Moving it to page 5, as a top insert and
    leaving the reference on the current page
>> Placing top insert number 7
The insert starts in the right column
and spans only the right column
>> Placing top insert number 6
The insert starts in the left column
and only spans the left column
>> Placing top insert number 5
The insert starts in the left column
and spans the right column
>> Placing bottom insert number 12
The insert starts in the left column
and only spans the left column
>> Placing bottom insert number 13
The insert starts in the right column
and spans only the right column
>> Placing bottom insert number 14
The insert starts in the left column
>> Insert 9, referenced in left column
    and spans the right column
>> Placing referenced insert number 9
```

#### 4.3 Example 3: What's Nice About the Fizzrev Shuffle.

This example illustrates the use of the macros to produce *Physical Review*-style documents and illustrates the advantage of using this style.

The example is broken up into two sections. The first section represents the output without using the *Physical Review* style; the second section shows the result of using this style. The example makes up a “before-and-after” sequence.

### Before

#### 5.1.8 Determination of Peak Position by a Linearized Gaussian Fit

This procedure transforms the Gaussian-shaped peak into a line and then fits a line to the transformed peak. The slope and intercept of the fitted line are related to  $\sigma$  and  $\sigma$ . The background continuum under the peak is first subtracted so that

$$\sigma^2[y(x)] = y(x) - \frac{1}{4} \left[ k^2 \frac{B_k}{N_k} - (x - k)^2 \frac{B_k}{N_k^2} \right] \quad (5-16)$$

where

$$k = \frac{2(x - x_0)}{(x_1 - x_0)}$$

the fit is made only to the Gaussian-shaped peak.

Transformations that linearize the Gaussian function have been applied only recently to determine the parameters of gamma-ray peaks (Ref. 7). The simplest of a class of similar transformations is the function

$$Q(x) = I_n \frac{y(x-1)}{\mu(x-1)} = \frac{2}{\sigma^2} x - \frac{2x_0}{\sigma^2} \quad (5-11)$$

where  $y(x)$  is the number of counts in channel  $x$ . The last expression in Equation 5-11 is correct if  $y(x)$  is the usual Gaussian function. The linear function  $Q(x)$  has a slope  $m$  and intercept  $b$  given by

$$m = 2/\sigma^2 \quad (5-12)$$

$$b = -2x_0/\sigma^2$$

Solving for  $\sigma^2$  and  $x_0$  gives

$$\sigma^2 = 2/m$$

$$x_0 = -b/m. \quad (5-13)$$

Equation 5-14 gives the expression for the slope  $m$  and the intercept  $b$  of the line fit to the set of points  $(x, [y(x)])$  by the weighted-least-squares method. (\*5-14\*)

The estimated variance of  $Q(x)$  is a function only of the uncertainties  $w(x)$  (\*5-14\*)

$$\sigma^2[Q(x)] = \sigma^2[y(x-1)] - \sigma^2[y(x+1)] \quad (5-15)$$

where  $w(x) \equiv w(y)/x$ .

If the background continuum is small enough to ignore, then the variance becomes

$$\sigma^2[Q(x)] \approx 1/w(x). \quad (5-16)$$

If the background continuum is subtracted by the straight-line procedure later shown in Section 5.3, the expression for  $\sigma^2[y(x)]$  is given by (\*5-17\*)

Also,  $y_1(x)$  is the total counts in channel  $x$  and the meanings of the other parameters are given in Section 5.3.3.

For a linear fit there are simple expressions for the estimated variance  $s^2$  of  $m$  and  $b$ :

$$s^2(m) \approx \frac{1}{\Delta} \sum \frac{1}{w_i^2} \quad (5-18)$$

$$s^2(b) \approx \frac{1}{\Delta} \sum \frac{x_i}{w_i^2}$$

Although the fitting procedure just described may seem somewhat complex, the fit can be performed by a short computer program in only a few seconds. The Gaussian function should be fit to avoid problems with non-Gaussian tails and inappropriate data. If the data are Gaussian, the value of  $\chi^2/\nu$  is 1.0. When at least four or five values of  $Q(x)$  are used in the fit, the results are more than adequate to determine the peak parameters needed for the energy calibration. Unfortunately, it is very difficult to estimate the statistical uncertainty in  $\sigma$  using this fitting procedure. However, experience indicates that for peaks of reasonable precision, the values of  $\sigma$  are good to  $\sim 0.1$  channel or better.

In automated operation, a test should be used to automatically determine whether a Gaussian function adequately describes the data. The reduced chi-square statistic  $\chi^2/\nu$  provides such a test. For the linear fit of  $Q(x)$  versus  $x$ , (\*5-18\*)

where  $m$  and  $b$  are computed from Equation 5-14 and  $n$  is the number of values of  $Q(x)$  in the fit. For good fits,  $\chi^2/\nu$  should be  $\sim 1.00$ . (See Ref. 5 for a very readable discussion of the properties of  $\chi^2/\nu$ .) For low-precision peaks (up to  $\sim 10,000$  counts/channel),  $\chi^2/\nu$  is really  $\sim 1.00$  for peaks of

The points of reference of the objects (equations) being inserted are flagged with “(\* eq. no. \*)” — for example, (\*5-14\*).

On the first page of the “before” example, the equations flow correctly up until Eq. 5-14. This equation did not appear after its reference because of its size; thus, it was moved to the top of the next page. Equation 5-17 is referenced in the right column and placed two lines below its reference. This could have been corrected by use of the `\refinsertadj` dimension register. Equation 5-19 was too large to fit after its reference and thus was moved to the top of the next page.

The only two-column-spanning equations that were set correctly in the “before” example are Eqs. 5-22 and 5-24, which occur on page 3.

$$b = \frac{1}{\Delta} \left( \sum \frac{x_i}{w_i^2} \sum \frac{Q_i}{w_i^2} - \sum \frac{x_i}{w_i^2} \sum \frac{x_i Q_i}{w_i^2} \right) \quad (5-14)$$

$$m = \frac{1}{\Delta} \left( \sum \frac{1}{w_i^2} \sum \frac{x_i Q_i}{w_i^2} - \sum \frac{x_i}{w_i^2} \sum \frac{Q_i}{w_i^2} \right)$$

where

$$\Delta = \sum \frac{1}{w_i^2} \sum \frac{x_i^2}{w_i^2} - \left( \sum \frac{x_i}{w_i^2} \right)^2$$

$w_i^2$  = estimated variance of  $Q(x)$ .

$$\chi^2/\nu = \frac{1}{n-3} \left\{ \sum \frac{1}{w_i^2} [Q_i - (mx_i - b)]^2 \right\} \quad (5-19)$$

where  $c_2 = -1/2\sigma^2$

$c_1 = x_0/\sigma^2$

$x_0 = \ln y_0 - x_0^2/2\sigma^2$ .

A fit of Equation 5-20 to the set of points  $(x_i, \ln y_i)$  produces values of  $c_0, c_1$ , and  $c_2$  that give the parameters of the Gaussian:

$$x_0 = -c_1/c_2$$

$$\sigma = \sqrt{-1/c_2}$$

In  $y_0 = y_0 - c_2^2/4c_0$ . (5-21)

qualitatively good shape. As the maximum number of counts per channel increases,  $\chi^2/\nu$  increases even though the peak shape remains the same. The increase in  $\chi^2/\nu$  does not necessarily mean the fit is inadequate for determining energy calibration or for testing resolution. At low precision, the goodness of fit is dominated by counting statistics; at high precision it is dominated by the inevitable small deviations of the peak shape from a true Gaussian shape. Resolution is determined by the computed value of  $\chi^2/\nu$ . Experience will determine acceptable value of  $\chi^2/\nu$  for a given range of peak precision.

Figure 5.9 shows a low-precision spectral peak from a germanium detector (FWHM  $\sim 19$  keV) with the fitted Gaussian function superimposed upon it. The lower portion of the figure shows the plot of  $Q(x)$  versus  $x$  for the upper two-thirds of the peak along with the fitted line and the computed peak parameters.

#### 5.1.9 Determination of Peak Position Using a Parabolized Gaussian Fit

The natural logarithm of the Gaussian function is parabolic as is strikingly apparent when high-energy peaks are viewed using the logarithmic display of an MCA. The natural logarithm of the Gaussian (Equation 5-9) gives

$$\ln y = c_2 x^2 - c_1 x - c_0 \quad (5-20)$$

The fitted curve is a parabola that opens downward and whose axis is parallel to the  $x$ -axis. The procedure described here determines  $y_0$  in addition to  $\sigma$  and  $\sigma$ , the two parameters obtained from the linear fit to the linearized Gaussian. Therefore, the full-energy-peak area can be determined using Equation 5-8.

Figure 5.7 shows a parabolized Gaussian fit to a high-energy spectral peak from the 122-keV gamma ray of  $^{67}\text{Co}$ . The same high-quality germanium detector was used in the figure as in Figure 5.3(a). At low energies, charge collection in germanium detectors is more complete than at high energies, with a resultant decrease in low-energy tailing. Comparison of Figures 5.3(a) and 5.7 shows that the tailing in the 122-keV peak of Figure 5.7 is even less than the small tailing of the 1332-keV

peak in Figure 5.3(a).

The expressions for the weighted quadratic least-squares fit are included for the convenience of possible users. The expressions are given in determinant form and involve eight sums, indicated by S1, S2, ..., S8. (\*5-22\*)

$$c_0 = \frac{1}{\Delta} \begin{vmatrix} S6 & S2 & S3 \\ S7 & S3 & S4 \\ S8 & S4 & S5 \end{vmatrix} \quad (5-22)$$

$$c_1 = \frac{1}{\Delta} \begin{vmatrix} S1 & S6 & S3 \\ S2 & S7 & S4 \\ S3 & S8 & S5 \end{vmatrix} \quad (5-23)$$

$$c_2 = \frac{1}{\Delta} \begin{vmatrix} S1 & S2 & S6 \\ S2 & S3 & S7 \\ S3 & S4 & S8 \end{vmatrix} \quad (5-24)$$

$$\Delta = \begin{vmatrix} S1 & S2 & S3 \\ S2 & S3 & S4 \\ S3 & S4 & S5 \end{vmatrix} \quad (5-25)$$

$$S1 = \sum \frac{1}{w_i^2}, \quad S2 = \sum \frac{x_i}{w_i^2}, \quad S3 = \sum \frac{x_i^2}{w_i^2}, \quad S4 = \sum \frac{x_i^3}{w_i^2}$$

$$S5 = \sum \frac{x_i^4}{w_i^2}, \quad S6 = \sum \frac{\ln y_i}{w_i^2}, \quad S7 = \sum \frac{x_i \ln y_i}{w_i^2}, \quad S8 = \sum \frac{x_i^2 \ln y_i}{w_i^2}.$$

As usual, the sums are over all the points fit. The  $y_i$  values have the background continuum subtracted. The remaining expressions required for the fitting procedure are

$$s^2 = s(\ln y_i)$$

$$s(\ln y) = \frac{s^2(y)}{y \ln y} \quad (5-23)$$

where  $s(y)$  is given by Equation 5-17. The expression for  $\chi^2/\nu$  is given in Equation 5-24. The expression for  $\chi^2/\nu$ , the goodness-of-fit statistic, is (\*5-24\*)

$$\chi^2/\nu = \frac{1}{n-3} \left\{ \ln y_i - (c_2 x_i^2 - c_1 x_i - c_0) \right\} \quad (5-24)$$

where  $n$  is the number of points fit and  $c_2, c_1$ , and  $c_0$  are the values computed from Equation 5-22.

peak is taken as the peak position for purposes of energy determination.

The remarks made in the previous section about the portion of the peak to be fit and about trends in  $\chi^2/\nu$  values apply equally well here. The quadratic fits put considerable demands on the computer, and occasionally the six significant decimal digits provided by many 16-bit processors run-

large fitting codes are used to analyze complex spectra with overlapping peaks. The codes decompose the peaks with functions that have a basic Gaussian form, with tailing functions added to describe the peak shape more accurately. An iterative nonlinear least-squares procedure is used to fit the data. The residual of the Gaussian component of the fitted

large fitting codes are used to analyze complex spectra with overlapping peaks. The codes decompose the peaks with functions that have a basic Gaussian form, with tailing functions added to describe the peak shape more accurately. An iterative nonlinear least-squares procedure is used to fit the data. The residual of the Gaussian component of the fitted

```

%      material in TeX's memory
\begin{mulcolinsert}{r}{2}{1}{pg.no.} % r=right here, 2=spans two columns,
 1=starts
                           % in 1st column, pg.no.=on this page #
the object
combination
\end{mulcolinsert}           % End of the object and single column mode

```

In addition to the objects discussed above, the *fizzrev* macro package formats a journal's title page, figure captions (including figures inserted with the *psfig* macro package), three levels of section headings, and textual references and numbers for equations (using a placeholder instead of an equation number). The line lengths of the title and running heads are both user-adjustable. See the listing below for definitions of available macros in the *fizzrev* package.

Using the two-column macros from inside one's own macros makes writing macros like `\refrule` easier; neither the macro designer nor the user needs to worry about the details of getting out of two-column mode to center the rule. The following coding was required to implement the reference rule:

```

\def\refrule{                                % Begin def
  \phyrevfalse                            % Don't draw reading flow indicators here
  \begin{mulcolinsert}{r}{2}{1}{0}          % r=the rule is an insert, 2=spanning
                                         % two columns, 1=starting in the
                                         % left column, 0=on current page
  \vskip2pc                                % Leave some white space
  \centerline{\vrule width16.5pc depth0pt height.5pt} % Center the line
  \vskip1.5pc                                % Leave more white space
  \end{mulcolinsert}                         % End of insert
  \phyrevtrue                               % Set indicators back on
}

```

#### 4.5 Fizzrev Macro Definitions

Following each definition, is an example of how to use the macro in a paper. See the *Fizzrev Users Guide* for complete descriptions.<sup>1</sup>

<code>\abstract{}</code> :	formats the abstract. The argument is the text of the abstract. <code>\abstract{This is an abstract.}</code>
<code>\affil{}</code> :	for the name of the author's affiliation. The argument is the name of the affiliation. <code>\affil{Los Alamos National Laboratory}</code>
<code>\authors{}</code> :	used when there is more than one author/affiliation pair. <code>\authors{Jane Doe}\affil{Sandia National Laboratory}</code>
<code>\autoauthorfalse</code> or <code>true</code> :	switch for user control of running author headline. It is set true in the <i>fizzrev</i> package to enable automatic generation of a running author headline on even-numbered pages. <code>\autoauthorfalse</code> will cause the macros to prompt the user for a running author headline.
<code>\autotitlefalse</code> or <code>true</code> :	switch for user control of running title headline. It is set true in the <i>fizzrev</i> package to enable automatic generation of a running title headline on odd-numbered pages. <code>\autotitlefalse</code> will cause the macros to prompt the user for a running title headline.

---

<sup>1</sup> Available from the authors.

<code>\beginmc{}{}</code> :	abbreviation for the multi-column <code>\beginmulcol</code> macro. <code>\beginmc{2}{20.5pc}</code>
<code>\binsertr{}{2}{1}{2}</code> :	abbreviation for the multi-column <code>\beginmulcolinsert</code> macro. <code>\binsertr{r}{2}{1}{2}</code> “Place it right here, span 2 columns, start in 1st column, and put the object on page 2.”
<code>\bttpl</code> :	abbreviation for the multi-column <code>\balance</code> macro.
<code>\einsert:</code>	abbreviation for the multi-column <code>\endmulcolinsert</code> macro.
<code>\endmc:</code>	abbreviation for the multi-column <code>\endmulcol</code> macro.
<code>\eq:</code>	used to reference an equation in the text and number it. ‘‘See Equation <code>\eq{\qa}</code> . $A + B = C$ <code>\eqno(\qa)</code> .’’ <code>\qa</code> is just a placeholder, the <code>\eq</code> macro will evaluate the placeholder to the current number stored in the equation counter.
<code>\fig{}{20pc}{20.5pc}{This is a caption.}{1}</code> :	used to set the caption and leave space for the figure. Argument 1 is the height of the figure, argument 2 is the width of the figure, argument 3 is the caption excluding a label (i.e., Fig. 1), and argument 4 is a switch for a box outline (set to 1 for on, 0 for off). <code>\fig{20pc}{20.5pc}{This is a caption.}{1}</code>
<code>\figcount:</code>	used to control the figure number. <code>\figcount=3</code> to get a figure labelled “Fig. 4”.
<code>\getheaderfooterinfo{}{35}{Title}{Author}</code> :	provides the information for the header and the footer. Argument 1 is the volume number, argument 2 is the issue number, argument 3 is the title, and argument 4 is the author’s name. <code>\getheaderfooterinfo{12}{35}{Title}{Author}</code>
<code>\it:</code>	used for standard italics text. <code>\it{emphasis}</code>
<code>\jtp:</code>	used to set a journal titlepage and page parameters. See also <code>\ptp</code> .
<code>\letter{}:</code>	used for figures that are lettered, i.e., Fig. 1A. The argument is the needed letter. <code>\letter{A}</code>
<code>\ninepoint:</code>	for text and math at nine point. Shouldn’t be needed but it is available. <code>\ninepoint This text will be in 9-point.</code>
<code>\page{}:</code>	used instead of <code>\pageno</code> macro. The argument is the first page of the journal or the paper. <code>\page{1}</code>
<code>\phyrevfalse or true:</code>	switch that controls the drawing of reading flow indicators, true for on, false for off. Set to true in the <code>fizzrev</code> package.
<code>\postscriptfig{}{20pc}{20.5pc}{figure1.ps}</code> :	for use with PSFIG macro package. Argument 1 is the height of the figure, argument 2 is the width of the figure, argument 3 is the figure caption, excluding a label (i.e., Fig. 1), and argument 4 is the name of the PostScript file. <code>\postscriptfig{20pc}{20.5pc}{This is a caption.}{figure1.ps}</code>
<code>\ptp:</code>	for a paper’s title page. Will set up the <code>\vsize</code> and the headline size for a paper.
<code>\recvd{}:</code>	argument is a date. The word “Received” and the opening and closing parentheses are added by the macros and centered. <code>\recvd{3 March 1989}</code>

\refrule:

centers a rule before the references, leaving white space before and after the rule.

\rm:

for times roman font. Ten point default font for text.

\rauthorhsize:

the hsize for the running author headline. Defaults to 32 picas.  
\rauthorhsize=28pc

\rtitlehsize:

the \hsize for the running title headline. Defaults to 27 picas.  
\rtitlehsize=28pc

\section{*{}{}*:

for a level one -heading. Argument is the heading, will be centered in bold and set in all caps automatically. Macro will leave the appropriate vertical spacing. \section{This is a section heading}

\subsection{*{}{}*:

for a level-two heading. Argument is the heading, will be centered in bold. Macro will leave the appropriate vertical spacing. User must type the argument using mixed case.  
\subsection{This is a subsection heading}

\subsubsection{*{}{}*:

for a level-three heading. Argument is the heading and it will be centered in italics. Macro will leave the appropriate vertical spacing. User must type the argument using mixed case.  
\subsubsection{This is a subsubsection heading}

\tenpoint:

the default point size used for a paper. Shouldn't be needed but it is available.

\titlehsize:

the \hsize used for the title on the titlepage of the paper. Defaults to 38 picas. \titlehsize=36pc

# PHYSICAL REVIEW A GENERAL PHYSICS

THIRD SERIES, VOLUME 39, NUMBER 2

JUNE 16, 1989

## Transport coefficients of hard-sphere mixtures: Theory and Monte Carlo molecular-dynamics calculations for an isotropic mixture

Jerome J. Heywood,  
Los Alamos National Laboratory  
(Received 2 March 1989)

The text has been rearranged, deleted, and edited with the author's permission. The thermal transport properties of mixtures can be formulated in a number of ways, depending on the choice of driving forces and the assumptions made about the interactions between the components. Here we treat mixtures in mixtures based on the driving force  $-\nabla(\mu_i) = -\nabla(\mu_i/\tau_i)$ , with  $\tau_i$  the mean lifetime of the specific chemical potential, to obtain the Green-Kubo expression and the Rehing theory for the corresponding transport coefficients which seem most amenable to molecular dynamics evaluation. The transport properties of a two-species mixture with a 1.0 diameter ratio of 1.0, at a volume fraction of 0.1 (and a corresponding volume), calculated by a Monte Carlo code and compared to the Green-Kubo formulae, are compared with the predictions of the Rehing theory. The long-time behavior of the Green-Kubo autocorrelation functions for shear viscosity, thermal conductivity, thermal diffusion, and thermal diffusivity, for the same mixture, are in good agreement with the predictions of the Rehing theory. Except for viscosity, the conductivities and diffusivities in terms of the transport coefficients fit the data very well. We obtain values, relative to Rehing, of  $1.016 \pm 0.007$  for shear viscosity,  $1.215 \pm 0.009$  for thermal conductivity,  $1.267 \pm 0.026$  for thermal diffusion, and  $1.117 \pm 0.008$  for thermal diffusivity.

### I. INTRODUCTION

The theoretical study of the transport properties of mixtures has had a rather limited history in recent years, involving at least in part the need to understand the behavior of the multicomponent fluid mixtures which are important in a great variety of natural and industrial systems. This interest has been manifested in both analytic theory and computer simulations. Our aim in the present series of papers is to focus the main areas of these developments on simple model mixtures so as (1) to determine a comprehensive assessment of analytic theories and the extent to which it can describe transport processes, and (2) to explore the power and limitations of computer simulations with respect to transport in mixtures.

There are now some improvements in the kinetic theory of fluids in that an empirical form of transport theory, valid beyond the low-density limit, has been developed only for such models. For single component fluids, the high-density theory developed by Rehing<sup>1,2</sup> was based from the outset on a hard-sphere model. Extensions to soft interactions have been proven elusive, although ad hoc methods for applying the

Rehing theory to arbitrary systems exist, viz. the so-called Modified Rehing Theory.

The Rehing theory was revised by van Beijeren and Ernst<sup>3,4</sup> who resolved certain ambiguities in the earlier version. The extension of the theory to mixtures<sup>5</sup> resulted in the omission of the first applications to binary mixtures by Thomas and to ternary mixtures by van Beijeren and Grubisic.<sup>6</sup> The van Beijeren-Ernst development, the so-called Revised Rehing Theory, yielded a theory consistent with the Onsager reciprocity relations in all cases.

Exotic calculations for the Revised Rehing theory were made by van Beijeren, Ernst, and Kijlstra<sup>7</sup> in a series of papers giving the general development of the method definition constant,<sup>8</sup> and the thermal diffusion constant.<sup>9</sup> Our principal aim is to test that theory quantitatively.

In a test of the accuracy of the Rehing theory for a single component fluid, Allende et al.<sup>10</sup> have used the molecular dynamics method to evaluate the dynamic time-correlation functions which occur in the Green-Kubo theory of transport coefficients. It was through these studies that the long-time tail of the velocity autocorrelation function was first observed and explained hydrodynamically,<sup>11</sup> subsequently being compared<sup>11-17</sup> with the theoretical predictions

### 39 TRANSPORT COEFFICIENTS OF HARD-SHPERE MIXTURES: THEORY...

One can readily obtain the inverse relations in a similar manner:

$$L_{\alpha\alpha} = L_{\alpha\beta}, \\ L_{\alpha\beta} = L_{\alpha\alpha} + \sum_i \mu_i L_{i\alpha}, \\ L_{\alpha\beta} = L_{\alpha\alpha} + \frac{1}{2} \sum_i \mu_i \tau_i L_{\alpha\alpha} + \frac{1}{2} \sum_i \sum_j \mu_i \mu_j L_{ij}, \quad (1)$$

Similarly, the double-prime not is found to be,

$$L_{\alpha\alpha}'' = L_{\alpha\beta}, \\ L_{\alpha\beta}'' = L_{\alpha\alpha} + \sum_i \mu_i L_{i\alpha}, \\ L_{\alpha\beta}'' = L_{\alpha\alpha} + \frac{1}{2} \sum_i \mu_i \tau_i L_{\alpha\alpha} + \frac{1}{2} \sum_i \sum_j \mu_i \mu_j L_{ij}, \quad (2)$$

which can also be readily inverted.

### II. MICROSCOPIC THEORY

The microscopic calculation of transport coefficients is based on Green-Kubo theory. Unfortunately, the literature on the subject is unworthy for the most part so that considerable care must be exercised in the use of so-called standard equations. For example, the theory of the pair distribution function developed by Ornstein<sup>22</sup> although some minor correction is needed in this case also. Because Ornstein derived expressions for the transport coefficients based on the prime states and forces, we shall further transform these to obtain expressions appropriate to the mainstream and double-prime systems.

#### A. Dynamical System.

We consider a fluid system of the temperature  $T$  contained in cubic volume  $V = L^3$ . The system consists of  $N$  particles,  $N_1$  having mass  $m_1$ ,  $N_2$  having mass  $m_2$ , ..., and  $N_n$  having mass  $m_n$ .

The system is assumed to evolve with time under the Newtonian equation of motion subject to a two-body, central interaction potential  $u_{ij}(r)$  acting between particles of species  $i$  and  $j$  at a separation  $r$ . If we denote the particle positions by  $r^N = \{r_1, r_2, \dots, r_N\}$  and the particle velocities by  $v^N = \{v_1, v_2, \dots, v_N\}$ , then the potential energy is the sum of pairwise additive terms,

$$U_2 = \sum_{i,j} u_{i,i,j}(r_{ij}), \quad (3)$$

in which we denote the species index of particle  $i$  by  $s_i$ , and in which  $r_{ij} = |r_{ij}|$ . To simplify the notation, however, for the  $i,j$ -interaction, we write hereafter,

$$\phi_{ij}(r_{ij}) = u_{i,i,j}(r_{ij}), \quad (4)$$

For the case of hard-disk or hard-sphere mixtures, which will be our principal concern, the interaction potential is singular, in which  $\sigma_{ij}$  is the diameter of species  $i,j$ . In the discussion which follows, we shall speak of the hard-sphere interaction to indicate the 2- or 3-dimensional case. Only in Sec. IV and Sec. VI will we speak of the three-dimensional case.

In numerical calculations, the system is typically for from macroscopic sizes. In order to minimize surface effects, we employ the usual periodic boundary conditions in which the  $N$ -particle system is replicated throughout space through translation by  $DL$  for all  $d$ -directions  $D$  having integer components. Thus the total potential energy of the system becomes

$$U(r^N) = \sum_i \phi_{ii}(r^N), \\ U_{\alpha\beta}(r^N) = \tau_{ij} \phi_{ij}(r_{ij} + DL), \quad (5)$$

in which the  $i,j$ -prime denotes the omission of the  $j-i$  term for  $\beta = \bar{\beta}_i$ , as in Eq.(3).

In order to make contact in the face of periodic boundary conditions, a "friction" definition of  $\tau_{ij}(t)$  to be the integral of the velocity  $v_{ij}(t)$ . This definition is sometimes called the "intrinsic checkerboard" description in which the  $v_{ij}(t)$  move continuously throughout the periodic cells. It is to be distinguished from the discontinuous definition whereby the particle remains within the "primary" or  $\beta = \bar{\beta}_i$  cell.

#### B. Green-Kubo Formulas.

Ornstein<sup>22</sup> derived the Green-Kubo formulae for a multi-component mixture. These can be written in the usual form

$$L_{\alpha} = L_{\alpha}^{(KK)} + L_{\alpha}^{(KZ)} + L_{\alpha}^{(KD)} + L_{\alpha}^{(AD)}, \quad (6)$$

in which

$$L_{\alpha}^{(AD)} = \lim_{t \rightarrow \infty} \lim_{N \rightarrow \infty} L_{\alpha}^{(AD)}(t), \\ L_{\alpha}^{(AD)}(t) = \int_0^t ds \rho_{\alpha}^{(AD)}(s), \quad (7)$$

with  $A, B \subset \{1, \bar{\beta}\}$ , for transport coefficient  $\alpha \in C \subset \{ub, uu, uv, \eta\}$  representing the optical diffusion constant, the thermal diffusivity constant, the thermal conductivity, and the shear viscosity, respectively. Here  $\lim_{N \rightarrow \infty}$  denotes the thermodynamic limit of large system size and  $\rho_{\alpha}(t)$  are super-correlation functions of the form,

$$\rho_{\alpha}^{(AD)}(t) = \frac{1}{\rho_{\alpha}} \langle \tilde{J}_{\alpha}^{(A)}(0) \cdot \tilde{J}_{\alpha}^{(D)}(t) \rangle, \quad (8)$$

and coupling theory, as developed by Rouse et al.,<sup>13,15,17</sup> and comparing the monostream and double-prime systems in addition to the two hydrodynamic theories of Pomeau.<sup>32</sup> Because the form of the mode-coupling results depends on the form of the starting hydrodynamic equations, it is evident that the long-time tails for a given set of time-correlation functions will have different forms when expressed in terms of the different transport coefficients.

Wood<sup>33</sup> has studied the case of a binary mixture in 3-dimensions, assuming that the equation of state and transport coefficients are known. In addition to conduction, except for a non-uniform state in the presence of forces and forces, he also obtains relatively simple expressions for the tails for the double-prime system and a somewhat more complicated expression for the monostream system. In this section we display the mode-coupling results for the latter.

The general form of the long-time tail for transport coefficient  $L_{\mu}$  is

$$\mu_{\mu}(t) \sim k_1 t^{-d/2}, \quad (10)$$

It is perhaps simplest to write the coefficient  $L_{\mu}$  in terms of the double-prime transport coefficients rather than the monostream coefficients. For mutual diffusion, then, the coeffi-

cients is

$$k_{11} = \frac{d}{d\beta(\mu_{\mu}(\eta_1, \eta_2))} \sum_{n=1}^{\infty} \frac{n \eta_1 L_{\mu}^{(n)} / c_v T}{[4\pi(\eta_1 + \eta_2)/\rho]^{d/2}} \quad (11)$$

in which

$$\begin{aligned} \eta_1 &= \frac{1}{2} (\mu_{\mu} L_{11}^{(1)} + L_{\mu\mu}^{(1)}) / c_v T \pm D^{1/2}, \\ D &= \frac{1}{2} \left( \frac{L_{\mu}^{(1)}}{c_v T} - \mu_{\mu} L_{11}^{(1)} \right)^2 + \frac{\mu_{\mu}}{c_v T} (L_{11}^{(1)})^2, \\ \mu_{\mu} &= \left( \frac{(\mu_{\mu_1} \mu_{\mu_2})}{\rho c_v} \right)_{T, \mu}, \end{aligned} \quad (12)$$

and in which  $c_v$  is the specific constant-pressure heat capacity. For thermal diffusion, the coefficient is

$$\begin{aligned} k_{11} &= \frac{d}{d\beta(\mu_{\mu}(\eta_1, \eta_2))} \frac{c_v T}{L_{\mu\mu}^{(1)}} \\ &\quad \sum_{n=1}^{\infty} \frac{n \eta_1 L_{\mu}^{(n)} / c_v T}{[4\pi(\eta_1 + \eta_2)/\rho]^{d/2}}, \end{aligned} \quad (13)$$

For thermal conductivity, the long-time tail coefficient is

$$\begin{aligned} k_{11} &= \frac{c^2}{d\beta(\mu_{\mu}(\eta_1, \eta_2))} \frac{1}{L_{\mu\mu}^{(1)}} \\ &\quad \left( \frac{c_v T}{d\mu_{\mu} \beta(\mu_{\mu}(\eta_1, \eta_2))} \frac{1}{L_{\mu\mu}^{(1)}} \right)^2, \\ &\quad \sum_{n=1}^{\infty} \frac{n \eta_1 L_{\mu}^{(n)} / c_v T}{[4\pi(\eta_1 + \eta_2)/\rho]^{d/2}}, \end{aligned} \quad (14)$$

in which  $c$  is the adiabatic sound speed,

$$c^2 = \left( \frac{\partial p}{\partial \rho} \right)_{T, \mu}, \quad (15)$$

and  $\nu$  is the acoustic damping constant;

$$V_1 = \frac{1}{\rho} \left[ \frac{2(d-1)}{d} \eta + \zeta \right] + V_{11}, \quad (16)$$

in which

$$X = \frac{T}{\rho} \left( \frac{\partial p}{\partial T} \right)_{\mu, \mu}, \quad (17)$$

is related to the usual volume coefficient of thermal expansion and

$$\mu_{\mu} = \left( \frac{\partial(\mu_1 \mu_2)}{\partial \rho} \right)_{T, \mu}. \quad (18)$$

By virtue of Eq.(2) relating the double-prime transport coefficients to the monostream coefficients, we expect these expressions to provide the monostream long-time tails in terms of the monostream coefficients.

### III. MONTE CARLO, MOLECULAR DYNAMICS CALCULATIONS

In this section, we discuss the evaluation of the transport coefficients directly from the Green-Kubo formulas for a 50-50 binary mixture of hard spheres having diameter  $a$ , but with a shear viscosity  $(d=3)$ , which includes having a shear viscosity  $\eta = 0.1$  and a thermal conductivity  $\kappa = 0.1$ . Because the thermodynamic and transport variables have a trivial dependence on temperature for hard spheres, we specify the state through the volume relative to

We have used the highest "Bloemberg approximation" (i.e., highest degree Stirling polynomial expansion) for which the

long-time tails of the correlation functions, as can be seen from Table II. The values of  $L_{\mu\mu}(35t)$  above quote standard deviations from unity, even though the tails are long compared to the Bloemberg decay times. The additional contributions for longer times are small for shear viscosity, but increasingly large for mutual diffusion (7%), thermal diffusion (11%), and thermal conductivity (16%). The agreement with the conclusion of Alder and Wainwright<sup>17</sup> for the long-time conductivity of a single-component hard-sphere fluid did agree with their observations for shear viscosity.

The present results suggest that further study of the parameter space for hard-sphere mixtures is in order. Calculations for other densities and for a number of other mass and diameter ratios are now in progress.

### ACKNOWLEDGMENT

The author is grateful to W. W. Wood of Carroll College for extensive discussions of the molecular dynamics problem and for his results for the long-time tails of the thermal conductivity. He is also grateful to J. M. Kincade of the State University of New York at Stony Brook, E. O. D. Cohen

### TRANSPORT COEFFICIENTS OF HARD-SPHERE MIXTURES: THEORY...

FORTRAN package was designed, namely the tenth approximation.

In similar fashion, we determine the long-time tails of the time-correlation functions from these double-prime transport coefficients, obtaining the required thermodynamic quantities. For example, to obtain  $\mu_{\mu}$  we use

$$\begin{aligned} \left( \frac{\partial \mu_{\mu}}{\partial \rho} \right)_{T, \mu} &= \sum_n \left( \frac{\partial \mu_{\mu}}{\partial \rho} \right)_{T, (\mu_1, \mu_2)} \left( \frac{\mu_{\mu}}{\partial \rho} \right)_{T, \mu}, \\ \left( \frac{\partial \mu_1}{\partial \rho} \right)_{T, \mu} &= \frac{1}{J_2} \left( \frac{\partial p}{\partial \rho} \right)_{T, \mu}, \\ \left( \frac{\partial \mu_2}{\partial \rho} \right)_{T, \mu} &= \frac{1}{J_2} \left( \frac{\partial p}{\partial \rho} \right)_{T, \mu}, \\ J_2 &= \frac{m_1 m_2}{(m_1 + m_2)^2} \sum_n n \left( \frac{\partial p}{\partial \rho} \right)_{T, (\mu_1, \mu_2)}, \\ \mu_{\mu} &= \mu_{\mu} \frac{\pi a^2}{6} \rho, \end{aligned} \quad (19)$$

which can be readily evaluated.

### A. Numerical Results

Our calculations include the study of systems of 100, 500, 1000, 1500, and 3000 particles. The parameters for each system are given in Table I, including the time-step per trajectory  $\tau_{\text{step}}$  and the number of trajectories  $N_{\text{traj}}$ . Observe that for the larger systems the trajectories are relatively short on the basis of the number of collisions per particle. For certain systems the time-step per trajectory is not particularly large, the calculation for the larger systems, especially the 3000-particle system, are quite less expensive when applied to the calculation of time-correlation functions for the transport properties studied here.

Our results consist of nine time-dependent transport coefficients, either evaluated directly from, or in the limit of certain differences, from the corresponding microscopic version of the Einstein relation, as in the case of the thermal conductivity and the viscosity. We define reduced transport coefficients relative to the Bloemberg values (in the Scalin approximation) by adding a caret over the symbol, e.g.,

$$L_{\mu} = L_{\mu}/L_{\mu}^B. \quad (20)$$

The Bloemberg transport coefficients are evaluated as outlined above, yielding

$$\begin{aligned} L_{\mu}^B &= 3.0197430 \frac{m_1}{(m_1 + m_2)^{1/2} \rho^2}, \\ L_{\mu_1}^B &= 0.0540925 \frac{m_1}{(m_1 + m_2)^{1/2} \rho^2}, \\ L_{\mu_2}^B &= 0.0021154 \frac{m_1 (m_1 + m_2)^{1/2}}{\rho^2}, \\ L_{\mu\mu}^B &= 0.2293762 \frac{m_1}{(m_1 + m_2)^{1/2} \rho^2}, \end{aligned} \quad (21)$$

### TRANSPORT COEFFICIENTS OF HARD-SPHERE MIXTURES: THEORY...

of Rockefeller University, and M. López de Haro of Instituto de Investigaciones en Materiales, Universidad Nacional Autónoma de México for numerous discussions and for providing the evaluation of the Bloemberg transport coefficients and

their help in modifying these programs to yield the desired form of the coefficients.

This work was supported by a contract with the U. S. Department of Energy, Office of Basic Energy Sciences, Division of Chemical Sciences.

### REFERENCES

- D. K. Bloemberg, *Svenska Vetenskapsakad. Handl.* **63**, 4 (1922).
- J. Chapman and T. G. Cowling, *The Mathematical Theory of Non-Uniform Gases*, Cambridge University Press, Cambridge, 1952, Second edition.
- H. van Beijeren and M. H. Ernst, *Physica* **68**, 437 (1973).
- H. van Beijeren and M. H. Ernst, *Physica* **70**, 235 (1973).
- M. H. Ernst and K. E. Gubbins, *J. Chem. Phys.* **60**, 268 (1974).
- M. López de Haro, E. O. D. Cohen, and J. M. Kincade, *J. Chem. Phys.* **78**, 2744 (1983).
- J. M. Kincade, M. López de Haro, and E. O. D. Cohen, *J. Chem. Phys.* **78**, 4509 (1983).
- J. M. Kincade, E. O. D. Cohen, and M. López de Haro, *J. Chem. Phys.* **60**, 937 (1974).
- B. J. Alder and T. E. Wainwright, *Phys. Rev. Lett.* **16**, 98 (1967).
- B. J. Alder and T. E. Wainwright, *Phys. Rev. A* **1**, 18 (1970).
- J. J. Hepenbäck and W. W. Wood, *Phys. Rev. A* **30**, 1648 (1984).
- J. J. Hepenbäck and W. W. Wood, *Phys. Rev. A* **32**, 412 (1985).
- J. R. Dorfman and E. O. D. Cohen, *Phys. Rev. A* **6**, 776 (1972).
- J. R. Dorfman and E. O. D. Cohen, *Phys. Rev. A* **12**, 293 (1975).
- M. H. Ernst, E. H. Haage, and J. M. J. van Leeuwen, *Phys. Rev. A* **20**, 2055 (1979).
- M. H. Ernst, E. H. Haage, and J. M. J. van Leeuwen, *J. Stat. Phys.* **16**, 7 (1976).
- M. H. Ernst, E. H. Haage, and J. M. J. van Leeuwen, *J. Stat. Phys.* **10**, 25 (1976).
- B. J. Alder, D. M. Ornstein, and T. E. Wainwright, *J. Chem. Phys.* **53**, 3813 (1970).
- T. R. Kirkpatrick, *Phys. Rev. Lett.* **53**, 1735 (1984).
- I. M. de Schepper, A. F. Hoffmann, and H. van Beijeren, *Phys. Rev. Lett.* **57**, 1715 (1986).
- O. Jacucci and I. R. McDonald, *Physica A* **60**, 607 (1975).
- C. Holstein and U. Debye, *Mol. Phys.* **37**, 95 (1979).
- A. A. Clifford and E. Dickinson, *Mol. Phys.* **34**, 873 (1977).
- D. L. Jolly and R. J. Bearman, *Mol. Phys.* **41**, 137 (1980).
- M. Schoen and C. Holstein, *Mol. Phys.* **53**, 33 (1984).
- M. Schoen and C. Holstein, *Mol. Phys.* **52**, 1029 (1984).
- R. Vogelsang and C. Holstein, *Phys. Rev. A* **30**, 3487 (1987).
- D. MacCormick and D. J. Evans, *Phys. Rev. A* **34**, 2133 (1986).
- S. R. de Groot, *Thermodynamics of Irreversible Processes*, North-Holland, Amsterdam, 1951.
- W. W. Wood, *Mode-Coupling Theory for the Long-Time Tail for the Transport Coefficients of Mixtures*, (in preparation).
- S. R. de Groot and P. Mazur, *Nonequilibrium Thermodynamics*, North-Holland, Amsterdam, 1962.
- H. S. Green, *J. Math. Phys.* **2**, 344 (1961).
- Y. Pomeau, *J. Chem. Phys.* **57**, 2800 (1972).
- J. J. Hepenbäck and W. W. Wood, *J. Stat. Phys.* **34**, 453 (1981).
- J. M. Kincade, (private communication).
- J. M. Kincade and J. J. Hepenbäck, *The Mutual Diffusion Constant of Binary Isotropic Hard-Sphere Mixtures: Molecular Dynamics Calculations using the Green-Kubo and Smoluchowski Methods*, 1986.

FIG. 1. Time-dependent thermal conductivity, relative to the Bloemberg value, for a 50-50 binary hard-sphere mixture having mass ratio of 0.1 and diameter ratio of 1.0 at a volume of three times close packing, as a function of the time  $t$ , relative to the time  $t_0$  for which  $L_{\mu\mu}(t_0)$  is a 500-particle system. The curve shows the second moment to be zero for the system. The error bars represent ±1 standard deviation about the mean.

### IV. CONCLUSIONS

We list here what seem to be the most important conclusions that can be drawn from this work.

1. The transport coefficients for mixtures can be defined in a nearly exact way by virtue of the fact that the separation of the hard-sphere model does not affect the transport properties, even when subject to the Onsager condition. From the point of view of molecular dynamics calculations, the monostream choice of fluxes and forces, with driving force of  $-T^{-1}\nabla T$  and  $-T\nabla(\mu_1/\mu_2)$ , yields Green-Kubo expressions which do not require knowledge of the partial specific volumes for the system and are, therefore, much easily applied.

2. For the mass and diameter ratios studied here at a volume of three times close packing, the long-time behavior of the autocorrelation function for shear viscosity, thermal conductivity, and mutual diffusion is found to agree with predictions of the mode-coupling theory. Except for the strong evidence for agreement in the case of one-component self-diffusion,<sup>12,17</sup> the present evidence seems to be the first to support the mode-coupling theory for the long-time autocorrelation function. Whether this result holds for other densities and for other mass and diameter ratios remains to be seen.

3. The results for the hard-sphere transport coefficients for the isotropic mixture show significant deviations from the Bloemberg theory. These deviations arise principally from the

## 5. What Next?

1. We need to simplify the TeX code for these macros and make it smaller. It would be nice if PCs and Macintoshes could use it.
2. We want to start processing flowing text as objects.
3. There is a need for one- and two-column footnotes.
4. Document the code.
5. Port the "place it here" code to three columns so it could be used for other formats.

## 6. A Comment on the Future of TeX

TeX must be enhanced with multiple-column formatting, object-placement options, and other related commands if it is to remain viable in the years ahead. WYSIWYG systems (such as Interleaf and Frame) that combine WYSIWYG with batch formatting in an effort to allow users to use tags for Standard Generalized Markup Language (SGML), will be tough competition for TeX as it now stands. In addition to these features, math formatting will soon be offered by Frame.

With content tagging (SGML) soon to be a way of life for most of us as we enter the age of knowledge information processing, a good batch formatting system in the public domain would be of great value. We would like that system to be TeX. We have a lot of TeX experience to draw upon. However, the program can't perform the tasks its competitors are beginning to be able to address.

Some of the other enhancements we would like to see include:

1. An input file that could be buffered so that processing could be done on the contents of the buffer. A command such as `\inputbuffersize2000` would establish the size of the buffer in bytes.
2. The ability to do string searches on the contents of the buffer and process the buffer up to the point a string match was found. It would also be useful if string searches could be done on the contents of any box and the match point returned. Commands such as `\vsplit\mybox` to `\findstring{string arg}` would then be possible.
3. The ability to write the contents of the unprocessed buffer to another file.
4. The ability to write contents of any box or delineated string to an ASCII file.
5. The ability to create multiple `dvi` output files.
6. The ability to assign a variable as a real, as well as a dimension or count register. Full floating-point arithmetic would be available on these variables.

# T<sub>E</sub>X Macros for COBOL Syntax Diagrams

MARY MCCLURE

Unisys Corporation  
19 Morgan Avenue  
Irvine, CA 92718-2093

## ABSTRACT

COBOL syntax diagrams have a unique format that has evolved into an industry-wide standard. This format is particularly difficult to accommodate without treating the diagram as artwork. When a manual contains over a hundred syntax diagrams, as several of our manuals at Unisys do, the production process becomes quite unwieldy.

However, T<sub>E</sub>X's math mode can be exploited to allow inclusion of COBOL syntax diagrams within the document itself. This paper presents macros that typeset COBOL syntax diagrams. The paper is divided into two parts: the first demonstrates how to use a set of macros to create the diagrams, and the second part lists and explains the actual macro definitions.

### 1. The Diagrams

COBOL diagrams are composed of four basic kinds of elements: items that are required, items that are optional, items that offer a choice, and items that can be repeated. In addition, COBOL diagrams can contain reserved words, which are displayed in uppercase, and programmer-supplied information, which is displayed in lowercase.

These elements can be combined to form diagrams that are quite complicated. But first, let's look at each element by itself. The composition of the macros used to create these elements is discussed later in this paper (see Section 2).

#### 1.1 Required Elements

Required elements are underlined. For example, the **VALUE** clause looks like this:

VALUE IS literal

and can be coded using simply the \req macro:<sup>1</sup>

```
\syntax{%
    \req{VALUE} IS literal
}
```

If several required items occur in a row, they must be identified individually. For example,

```
\syntax{%
    \req{MOVE} \req{CORRESPONDING} identifier-1 \req{TO} identifier-2
}
```

produces

MOVE CORRESPONDING identifier-1 TO identifier-2

---

<sup>1</sup> COBOL syntax diagrams always begin with the \syntax macro.

## 1.2 Optional Elements

Optional elements are enclosed in square brackets. For example, the **IF** statement looks like this:

**IF** condition [**THEN**] statement-1 [**ELSE** statement-2]

and is coded using both the **\req** and **\option** macros, like this:

```
\req{IF} condition \option{!THEN!} statement-1 \option{!\req{ELSE}}
                                statement-2!}
```

The function of the exclamation points (!)<sup>2</sup> in this example is not intuitively obvious. They are necessary to delimit optional elements that are composed of more than one item. When a series of items appears in an **\option** macro, it is a good idea to stack them one atop another in the macro call to keep track of where one item ends and another begins. For example, the **RECORD** clause contains a stack of optional items, of which one or none can be chosen:

**RECORD** CONTAINS [integer-1 **TO**] integer-2  
[ ASCII  
 COMPUTATIONAL  
 COMPUTATIONAL-2 ]  
[ CHARACTERS  
 WORDS ]  
[ DISPLAY ]

The coding for the **RECORD** clause gets a little more complicated and looks like this:

```
\syntax{%
  \req{RECORD} CONTAINS \option{!integer-1 \req{TO}!} integer-2
  \option{!ASCII!
    !COMPUTATIONAL!
    !COMPUTATIONAL-2!
    !DISPLAY!
  } \option{!CHARACTERS!
    !\req{WORDS}!
  }
}
```

## 1.3 Choice Elements

Elements that offer a choice are very similar to optional elements. However, instead of being enclosed in square brackets, they are enclosed in curly braces to indicate that one of the items within must be chosen. For example, one form of the **OPEN** statement looks like this:

**OPEN** {INPUT } file-name **REEL - NUMBER** {literal }  
          {OUTPUT } data-name }

and is coded thusly:

```
\syntax{%
  \req{OPEN} \choice{!\req{INPUT}!
    !\req{OUTPUT}!
  } file-name \req{REEL-NUMBER} \choice{!literal!
    !data-name!
  }
}
```

As in optional elements, each of the items in a choice element must be delimited by exclamation points.

## 1.4 Elements That Can Be Repeated

When an element can occur more than once in the syntax of a command, it is followed by a series of dots called an ellipsis. For example, the **ADD** command can operate on any number of variables:

**ADD** {identifier } ...**TO** identifier-n  
          {literal }

<sup>2</sup> The exclamation point was chosen as the delimiter because exclamation points are not members of the ANSI-standard COBOL character set.

The ellipsis is produced with the `\repeatable` macro. The **ADD** command is coded as:

```
\syntax{%
  \req{ADD} \choice{!identifier!
    !literal!
  } \repeatable \req{TO} identifier-n
}
```

### 1.5 Formatting Commands

Because COBOL syntax diagrams are quite complicated, their length often exceeds a single line. Left to its own devices, TeX will break the diagram at some point between two elements. For example, the **MULTIPLY** command is automatically broken between the seventh and eighth elements:

```
MULTIPLY{identifier-1 } BY {identifier-2 } GIVING identifier-3 [ROUNDED] [, identifier-4 [ROUNDED] ]...
```

You can insert `\par` commands to override TeX's line-breaking algorithm and instead force the line to break earlier in the diagram. For example, if you wanted to break the above diagram into two approximately equal parts, you would code:

```
\syntax{%
\req{MULTIPLY}\choice{%
  !identifier-1!
  !literal-1!
}
\req{BY}\choice{%
  !identifier-2!
  !literal-2!
}

\req{GIVING}
\par
  identifier-3
  \option{%
    !\req{ROUNDED}!
  }
  \option{%
    !, identifier-4
  }
  \option{%
    !\req{ROUNDED}!}!\}\repeatable
}
```

producing:

```
MULTIPLY{identifier-1 } BY {identifier-2 }
GIVING
  identifier-3 [ROUNDED] [, identifier-4 [ROUNDED] ]...
```

A `\par` command can be used only to produce a line break between two elements — it cannot be used in the middle of the `\option` or `\choice` macros. If you want a line break inside an `\option` or `\choice` macro, you should use the `\midbreak` macro. For example, the **RECORD** portion of an **FD** statement is quite lengthy:

```
[ RECORD {CONTAINS integer-3 CHARACTERS
  IS VARYING IN SIZE [[FROM integer-4 ] [TO integer-5 ] CHARACTERS] [DEPENDING ON data-name]
  CONTAINS integer-6 TO integer-7 CHARACTERS}
```

A line break is needed in the second item of the choice element. If you insert a `\midbreak` command after the **CHARACTERS** item, you obtain a more satisfactory diagram:

`[ RECORD { CONTAINS integer-3 CHARACTERS  
 IS VARYING IN SIZE [[FROM integer-4 ] [TO integer-5 ] CHARACTERS ]  
 [ DEPENDING ON data-name-1 ]  
 CONTAINS integer-6 TO integer-7 CHARACTERS } ]`

## 2. The Macros

Before we can define the macros, we must declare a few variables, define the font, and make the exclamation point an active character so that it can be used as the delimiter in the `\option` and `\choice` macros:

```

\newif\ifmchoice
\newif\ifmoption
\newif\ifmreq
\newif\ifmrepeat
\newif\ifstarted

\font\cobfont=CMB10
\catcode`!=\active

```

### 2.1 The `\syntax` Macro

All COBOL syntax diagrams must begin with the `\syntax` macro. `\syntax` sets up the environment and adds some white space before and after the diagram:

```

\long\def\syntax#1{%
\begin{group}
\cobfont
\textfont1=\cobfont
\mathcode`-=012D
\let!=\startorstop
\baselineskip=12pt
\lineskip=2pt
\parindent=0pt
\pretolerance=10000
\medskip
#1
\medskip
\end{group}
}%

```

The macro definition must be preceded by `\long` so that `\par` commands can occur within the diagram to force line breaks.

The macro loads the font defined as `\cobfont`, defines `\cobfont` to be the font accessed when TeX is in math mode, and changes the `\mathcode` of the hyphen. By default, a hyphen maps to a minus sign in the Computer Modern Math Italic font when it is encountered in math mode (Knuth 1984:153–154, 344, 351). By changing the `\mathcode`, we map the character to the hyphen in the normal text font. This is necessary at Unisys because some of our reserved words in COBOL contain hyphens and they look strange when the hyphen is displayed as a minus sign. Your site may encounter similar problems with other characters — if anything ends up something other than you expected, you should check the character's `\mathcode` and modify it to something more appropriate.

All of this font wizardry is local to the group, so if TeX enters math mode outside of a syntax diagram, it uses the default cmti 10 point font and maps characters using the definitions of `plain.tex`.

The `\syntax` macro also defines the active character `!` to be a call to the `\startorstop` macro, described below in Section 2.5.

The settings of `\baselineskip`, `\lineskip` and `\parindent` control amounts of white space. `\baselineskip` determines white space between vertically stacked items in a choice of optional elements. `\lineskip` determines white space between lines of a diagram when the diagram is too long

to fit on a single line. And `\parindent` determines how far the diagram is indented from the left margin.

The `\pretolerance` command is necessary to tell TeX it is OK to break lines and create underfull `\hboxes`. When TeX's own line-breaking algorithm analyzes COBOL syntax diagrams, it finds breakpoints only when math mode is turned off. This conveniently occurs between each element of the diagram, but TeX calculates the badness of each of these breakpoints to be so extreme that it ignores them all unless `\pretolerance` is set very high.

Finally, the `\syntax` macro uses the `\medskip` macro of `plain.tex` to surround a COBOL syntax diagram with a certain amount of white space.

## 2.2 The `\req` Macro

All the `\req` macro does is underline an item. All that is required is to enter math mode and use TeX's `\underline` command. However, sometimes TeX encounters the `\req` macro when it is already in math mode, so some logic is required to determine if math mode should be turned on and off.

```
\def\req#1{%
  \ifmmode\relax\else\mreqtrue$\fi
  \underline{#1}%
  \ifmreq$\mreqfalse\fi
}%
```

## 2.3 The `\option` Macro

The main function of the `\option` macro is to enclose the parameter text in square brackets ([ ]). The parameter text can be quite complicated and can contain calls to the `\req` macro or the `\choice` macro. The parameter text *always* contains at least two exclamation points (!) to delimit items in the optional element.

```
\long\def\option#1{%
  \begingroup
  \startedefalse
  % Local to the group.
%
  \ifmmode\relax\else\moptiontrue$\fi
  % As with \req, math mode
  % may or may not need to be
  % entered.
%
  \left\lbrack
  \vcenter{%
  \vbox{%
  \cbofont
  #1
  }%
  }%
  \right\rbrack
  % Load the desired font.
%
  \right\rbrack
  % Right square bracket.
  \ifmoption$\moptionfalse\fi
  % End math mode if need be.
  \endgroup
}%
```

The commands `\left` and `\right` allow TeX to determine how tall the square brackets need to be. These two commands are what make TeX so ideal for COBOL syntax diagrams (Knuth 1984:148). By using them, you make TeX stretch and shrink the brackets to correctly enclose the items, so you don't have to worry about the effect of adding or deleting items as the syntax of a COBOL command changes.

The entire parameter text is enclosed in a `\vbox` so that the `\vcenter` command can be used to center the text within the square brackets.

## 2.4 The \choice Macro

The \choice macro is exactly like the \option macro except that it encloses the parameter text in curly braces ({} ) rather than square brackets.

```
\def\choice#1{%
  \begingroup
  \startedfalse
  \ifmmode\relax\else\mchoicetrue$\fi
  \left\lbrace
    \vcenter{%
      \vbox{%
        \cbofont
        #1
      }%
    }%
  \right\rbrace
  \ifmchoice$\mchoicefalse\fi
  \endgroup
}%
```

## 2.5 Exclamation Points and the \startorstop Macro

The format of COBOL syntax diagrams requires that if more than one item occurs in an optional or choice element, the items must be stacked one on top of the other. Since TeX is designed to stack a series of \hboxes one atop another, this requirement is easily met by enclosing each item in an \hbox. But typing \hbox{ and } around each item gets a little tedious and takes up extra space on the line; you can make the exclamation point (or any character you choose) an active character and let TeX do some of the work.

If you make the exclamation point an active character and then assign it to be a control sequence that calls a macro, you can use that macro to determine if the exclamation point denotes the beginning or the end of the item. For example:

```
\catcode'!=\active
\let!=\startorstop

\def\startorstop{%
  \ifstarted
    \egroup
    \startedfalse      % Order of commands is important here. Flag
  \else                % should be turned on INSIDE the hbox and turned
    \hbox\bgroup       % off OUTSIDE the hbox.
    \startedtrue
  \fi
}%
```

Thus when the construction !RECORD IS! is encountered, it is transformed into \hbox{RECORD IS}.

Using \begingroup and \endgroup in the \option and \choice macro makes the value of the \started flag always local to the group. This enables nesting of elements, for example:

```
\option{!LABEL \choice{!RECORD IS!
  !RECORDS ARE!} STANDARD!}
```

If the value of \started is not local to the group, the exclamation point before **LABEL** is correctly identified as the starting point, and \started set to true. But when TeX encounters the exclamation point before **RECORD**, the \ifstarted command is evaluated as true and TeX attempts to end an \hbox when actually it is supposed to start a second \hbox!

In fact, the whole concept of using a single character for a macro call can be carried to extremes. What if, instead of requiring the user to remember that he needs to use the \option macro to get

the square brackets in his diagram, we allow him to just type an opening square bracket where the optional element begins and a closing square bracket where the optional element ends? The following commands would allow this:

```
\catcode`[=\active  
\catcode`\]=\active  
  
\let[=\option\bgroup  
\let]=\egroup
```

This works great, but the analogous situation of allowing curly braces to be used for choice elements creates complications. Once the curly braces are made active, they can no longer be used as they were originally intended to define macros, delimit parameters in macro calls, and generally serve as beginning and end or group markers. So then some other characters, perhaps parentheses, must be redefined to take on the traditional function of the curly braces, and then what will you do when you want to use parentheses in *their* normal context?

Pretty soon things become quite confusing to a person familiar with the traditional workings of TEX. But if you can keep your character codes straight, all this re-defining of character functions might be helpful to a person sitting down to code COBOL syntax diagrams who is totally unfamiliar with TEX. Users might find it helpful to be able to type:

```
\beginsyntax  
  
_RECORD_ CONTAINS  
  
[!integer-1 _TO_]  
  
integer-2  
  
[!ASCII  
!COMPUTATIONAL!  
!COMPUTATIONAL-2!  
!DISPLAY!]  
  
[!CHARACTERS!  
!_WORDS_]  
  
\endsyntax
```

instead of what was previously described in this paper to obtain a syntax diagram for the **RECORD** clause.

## 2.6 The `\repeatable` Macro

Like the `\req` macro, the `\repeatable` macro is quite simple. It boils down to entering math mode, if need be, and calling the `\ldots` macro of `plain.tex` to create the ellipsis indicating repeatability:

```
\def\repeatable{  
  \ifmmode\relax\else\mrepeattrue\fi  
  \ldots  
  \ifmrepeat\mrepeatfalse\fi  
}%
```

change tasks. The solution to this awkward method was to transfer the entire procedure to within the editor. Hence the editing environment, used the most in the production of a *T<sub>E</sub>X* document, becomes the environment that controls the entire process.

Once the basic sequence of producing, previewing, and printing a document was simplified to a few keystrokes, more sophistication was soon desired. The following is a partial list of significant features the authors believed important enough to include in the initial editing macro package.

- capability of inputting various kinds of information easily and efficiently
- instantaneous graphical display of fonts and font information
- a complete context-sensitive help system for *T<sub>E</sub>X* and the editor
- automatic text reformatting and *T<sub>E</sub>X* control code modification
- representing *T<sub>E</sub>X* macros as modified ASCII characters

## 2. Editing Environment

The text editor of choice is KEDIT, the PC version of the IBM mainframe editor XEDIT. KEDIT is an extremely powerful and versatile editor. Together with the procedural language REXX, practically any task can be simplified to the touch of a key. The principle advantage of KEDIT, because it is programmable, is that it can emulate most text editors (*not* word processors). Users of this interface will not need to learn a new editor — a fate on par with a root canal gone awry. KEDIT can be made inanely simple, such as EDLIN, or as sophisticated and complex as the user desires. Thus, the *T<sub>E</sub>X* interface is completely uncoupled from the editing process.

There are several features within KEDIT that enable it to be so versatile. It allows the user to create synonyms, such that any command can be called by a different name. For example, the term "translate" could be substituted for the command *move*, if that term was more comfortable to use. It can also be abbreviated to any length desired. Using the same example, the "translate" synonym could be specified as "tr", "tran", or "trans!". The ability to define macros and assign them to almost any key- or user-defined command name is what makes KEDIT unarguably superior to non-programmable text editors. These macros can be simple functions used to save key strokes for frequently used commands or a technique to avoid having to go to the editor's command line or to DOS to perform a certain task. The macros are also able to call other macros, such as the *T<sub>E</sub>X* interface, which can all be accessed by hitting a single key.

A rudimentary example of redefining keys in KEDIT is the authors' modification of the opening and closing curly brace and square bracket keys. Whenever the user is in the editing environment, an opening square bracket [ will return a { character. Likewise, a closing square bracket ] will return a } character. This is useful not only for *T<sub>E</sub>X*, where square brackets are not generally used as control characters,<sup>1</sup> but also for C programming. To eliminate confusion, one of the authors has actually switched the keys on his keyboard to signify this modification.

Mansfield Software Group's Personal REXX is an easy to understand yet powerful procedural language written specifically for the IBM-compatible personal computer.<sup>2</sup> Besides an extensive array of commands for file handling, text manipulation, and parsing, the one feature that is primarily used in Personal REXX is windowing. This greatly simplifies the *T<sub>E</sub>X* process by using windows to display various types of helpful information or to control various options.

## 3. *T<sub>E</sub>X* Interface

A single keystroke invokes the *T<sub>E</sub>X* interface. Presently, this is reserved for the F10 key. When this key is hit, two windows will be displayed, as shown in Figure 1.

The top window has five categories of control: **Format**, **Inputs**, **Preview**, **Print**, and **Spell Check**. The particular *T<sub>E</sub>X* formatter (e.g. *TeXT1*, *T<sub>E</sub>X*, *IAT<sub>E</sub>X*, *μT<sub>E</sub>X*...) which can be toggled by hitting the F4 key, will appear in the highlighted box in the upper left; in this case, the formatter is *TeXT1*. The left window shows the function key options for format control, which are self-explanatory.

<sup>1</sup> Note that the authors are referring only to *T<sub>E</sub>X*, and not *IAT<sub>E</sub>X*, where the square bracket is of course as crucial as the curly brace —Ed.

<sup>2</sup> REXX is also available on several other small computer platforms; for descriptions, see Kubik (1989) or Tokicki (1988), both on Amiga REXX.

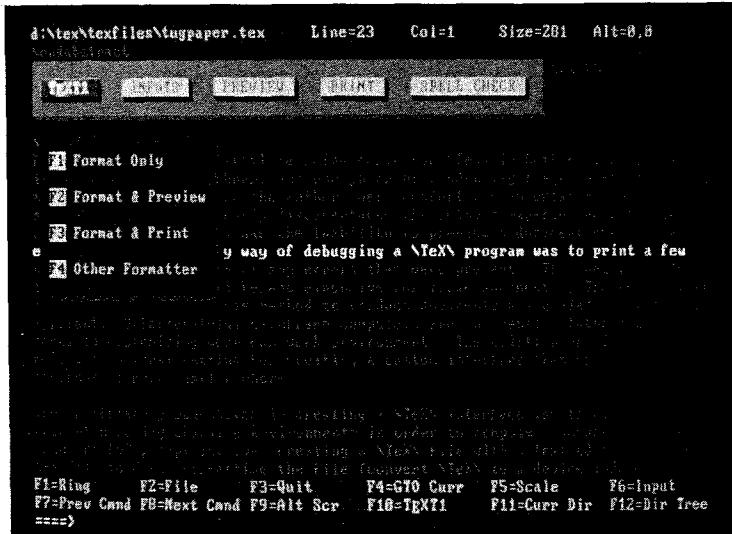


Figure 1: Screen display of the invoked *T<sub>E</sub>X* interface

Control is cursor-selected using the left and right arrow keys. Hitting the right arrow key will highlight the **Inputs** control option and a new lower window displaying six new function key options will appear. The function keys F1 through F6 are used for inputting *T<sub>E</sub>X<sub>T</sub>1* blocks, *T<sub>E</sub>X<sub>T</sub>1* models, font sets, math sets, specific fonts, and tables, respectively. This control option will be discussed in more detail in the next section.

The two control options **Preview** and **Print** are similar in function. The F1 key will list those files in a window that have already been formatted (*dvi* files) and can be cursor-selected to preview or print. The F2 key for the print control option will list those files that have previously been prepared for printing (for the authors' systems, *\*.hp* files). The remaining function keys modify how the output will be presented. Specific to the print control option, an additional window display is located in the lower left that provides a general perspective of how the printer output will appear. The starting and ending page of the document will be shown in the upper right corners of the "pages". The orientation is also clearly displayed as being either portrait (right-side-up) or landscape (sideways). The short paragraph written on the starting page is a summary of options that cannot be easily shown in text-mode. These options include the number of copies per page, the margin offset for odd and even pages, magnification of the print, and whether font information will be echoed prior to printing. Figure 2 shows an example of the print controller option.

The options used for a particular printout can be saved to file. For sets of options that are used frequently, these files can be quickly retrieved by hitting the F12 key and cursor-selecting the file that contains the desired printing options. Once selected, the short paragraph displayed on the starting page will reflect a summary of the new options.

The last control option, **Spell Check**, loads a spell checking utility into memory when the F1 key is hit. The authors have installed Webster's New World Spelling Checker on their systems. Any spell checker, however, can be used to suit the taste of a particular user. The dictionary option, which is activated by hitting the F2 key, will open a window that lists several auxiliary dictionaries that may be cursor-selected and added to the standard dictionary. This is necessary for files such as *T<sub>E</sub>X*, that contain numerous commands or markup which would not normally be accepted by a standard dictionary. The auxiliary dictionary may have a list of "incorrectly" spelled commands that the spell checker will assume to be correct and thereby speed up the process. Under the **Spell Check** window

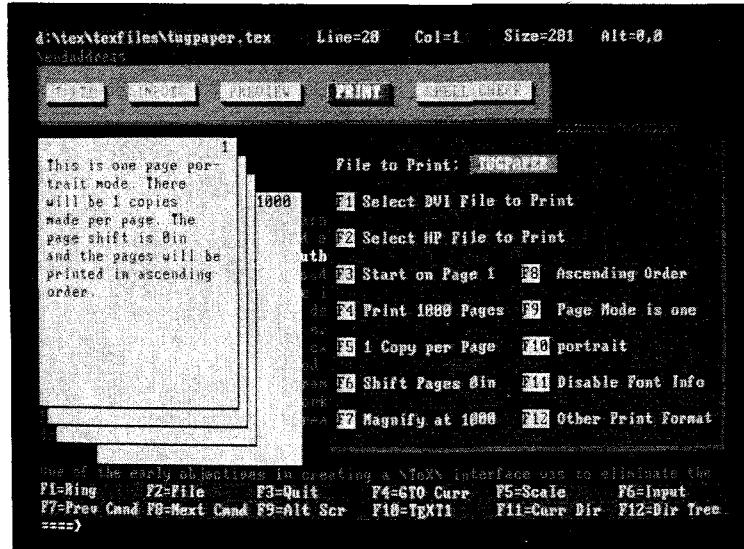


Figure 2: Screen display of the print controller option for the TEx interface

appear several other utilities to "aide" in the writing process. A dictionary and thesaurus may be loaded. Also, RightSoft's RightWriter may be invoked on the TEx source file.

#### 4. Inputting Information

The authors learned TEx and how to use the *TExT1* macros at Washington State University (Pullman, WA), where the *TExT1* macros were developed. The TEx interface at WSU (on an IBM mainframe) was what the authors first modeled their original interface after. The *TExT1* macros have the somewhat unique ability to change the global format from within the TEx source file, see *TExT1* (1987) or Riley (1989). This can best be accomplished by loading basic document control files (blocks) from disk into the TEx file. The document control blocks are simple ASCII files that contain the necessary *TExT1* markup to alter particular formats. This allows document processing to be greatly simplified. From within the editor, it is possible to open a window, display the 37 document component blocks (by name) and cursor-select any formatting block that will automatically be loaded into the file being edited. Figure 3 shows the screen after the window has been opened, displaying the *TExT1* document control blocks.

For example, if a document is being created with a non-standard paper size and margin widths, loading the *page.blk* block into the current file (*page-p* in Figure 3)

```
% Default page dimensions and margins
\pageformat{\pagelength{11in}}      % 792pt = 11in
    \pagewidth{8.5in}                % 612pt = 8.5in
    \topmargin{1in}                  % 72pt = 1in
    \bottommargin{1in}
    \leftmargin{1.2in}                % 86pt = 1.2in
    \rightmargin{1in}
    \bindingadjust{0in}
}%; end pageformat
\normalbottom                      % text height will be the same for each
                                    % page. Bottom lines will be even.
```

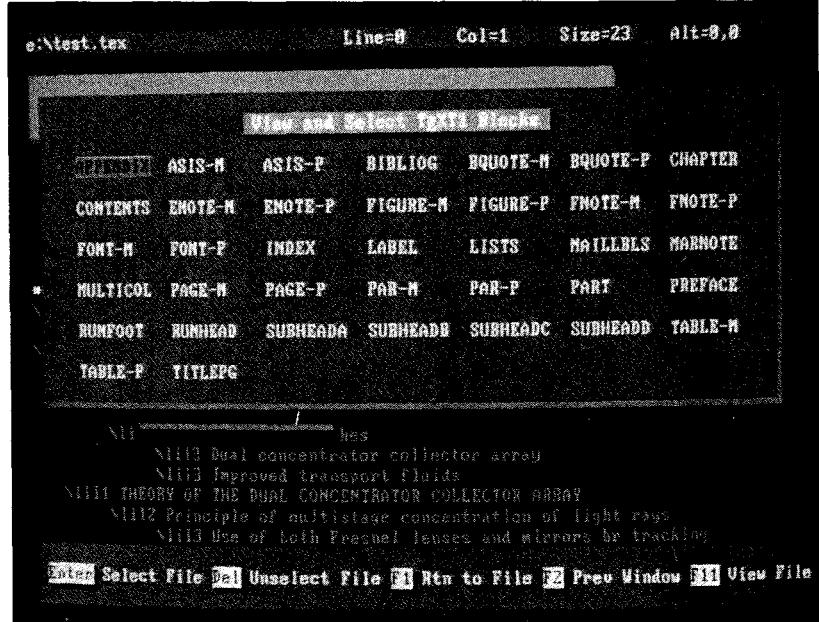


Figure 3: Sample window display of *TEXT1* control blocks that may be selected for insertion

and changing the dimensions of interest (substituting values within the appropriate {} braces), will easily adjust the page format. Markup does not have to be remembered, nor syntax for typing the markup, and time is saved with respect to loading the same file from DOS. The ability to also include document models or style sheets at the touch of a key is equally simple. It is this process that makes creating *TeX* documents a much easier task in comparison to manual insertion of *TeX* control codes.

The window interface allows the user to display and easily select any font available to *TeX*. This is accomplished by displaying all files with an extension of \*.tfm into a window (a scrollable window since there are usually so many files) and then cursor-selecting as many of the font names that are necessary. The macro will read the name, perform a decimal-to-roman numeral conversion on the font size within the file name, and then insert a line into the *TeX* file that correctly loads the font for *TeX*. For example, to load the font cmssi17, a window is displayed with all the font names, much like Figure 3, and the user cursor-selects the cmssi17 font by hitting the Enter key. Upon leaving the window, the current line in the editor will have the following line added after it: \font\cmssixvii = cmssi17 at 17pt (this can be seen in the second line of Figure 5). This not only saves time and frustration looking up what fonts are available, but also introduces a consistent font nomenclature. The authors also like to keep the following types of files located in this input window environment:

- *TEXT1* blocks
- *TEXT1* models/style sheets
- *TEXT1* font sets
- *TEXT1* math sets
- multiple ruled and aligned table formats
- graphical input files (e.g., clip art and scanned images)
- *TeX* Font Metric files (**tfm** files)
  - 75 standard Computer Modern (CM) fonts
  - 63 *TEXT1* fonts
  - resident and cartridge printer fonts
  - down-loadable soft fonts
  - PostScript fonts

Whenever the highlighted section or "cursor" is located on a file name that can be selected for input to the current file, hitting the F11 key will open another window that contains the contents of that file for immediate viewing purposes. This feature is used throughout the interface as well as other editing utilities.

#### 4.1 Graphical Help Facility

The ability to easily display font files in a window for automatic selection led to the desire to have a graphical display of the fonts. This would give the user an idea of what the fonts would look like on an output device without having to use a previewing program or a sample printout. It could also be used to show what default sizes and magnifications were available, and how to invoke them. Two commercially available software packages, ZSoft's Publisher's Paintbrush and PCX Programmer's Toolkit, were used to create this graphical font help system.

The graphics format used is ZSoft's pcx format, a pseudo-standard in the PC arena for graphics. The latest version of ZSoft's Paintbrush package includes a utility called `hp2pcx.exe`. It converts files (both graphics *and* text) produced for the HP laser printer to the pcx format. Thus, it is a trivial process to convert *T<sub>E</sub>X*-generated laser printer output to the pcx format.<sup>3</sup> Another utility, called PCX Programmer's Toolkit from Genus Microcomputing, is needed however, to quickly display the graphical file within the text-mode editing environment. The tool kit contains several useful PCX utilities; one will take a group of pcx files and load them into a library and another will instantaneously display the pcx file to the screen from a library.<sup>4</sup> This provides sample font files for all of the *T<sub>E</sub>X* fonts to be stored in one common library and displayed at the touch of a key from within the editing environment.

The process of generating the standard 75 *T<sub>E</sub>X* font files (plus as many as needed for specific resident printer fonts, soft fonts and the like) was simplified by creating a database of the font names, sample output text, and the sizes of available fonts. *T<sub>E</sub>X* could then produce the entire set automatically. This font database was formatted with *T<sub>E</sub>XT1* using one driver file that contained the necessary `\halign` commands and markup to produce the \*.dvi files. A DOS batch file would convert all the \*.dvi files (using `\magnification=473`) to a temporary \*.hp file (HP LaserJet format, 300 dpi) and then convert the files to the pcx format. The files are loaded into a library and are available for display within the editor. Figure 4 is an example of a graphical font display file.

Besides the graphical font display, the authors have found that other information is easier to comprehend by means of graphical output rather than by using just standard descriptive ASCII text. For example, when loading font sets with *T<sub>E</sub>XT1*, a graphical display file of the particular family shows which faces are available. Also, general *T<sub>E</sub>X* help files are available in graphics format to illustrate quote marks, dashes, special characters, and ruled tables.

### 5. General Help Facilities

Several help facilities have been written to assist the user not only with the *T<sub>E</sub>X* interface but also within the editor and KEDIT/REXX macros. For each screen or window, the function keys (F1--F12) are usually displayed along the bottom of the screen (see Figure 1) with an abbreviated word or phrase describing their function. Complete and separate help menus for key combinations involving the Alt and Ctrl keys will be displayed by hitting the key combinations `Alt-h` and `Ctrl-h`, respectively. Particular to *T<sub>E</sub>X* files (any file with an extension of \*.tex), `Ctrl-\` will determine if the cursor is on a *T<sub>E</sub>X* command; if so, an ASCII help file describing that command will be shown in a window that can be scrolled forward and backward using the PgDn and PgUp keys. When the `Alt-\` combination is hit, a listing of *T<sub>E</sub>X* commands, their correct abbreviations, and short synopsis will be shown in a window. The same help information as for the `Ctrl-\` key combination can then be accessed by hitting the `Enter` key when the highlighted line is located on a particular command.

<sup>3</sup> That is, it is the *T<sub>E</sub>X* output, consisting of the METAFONT-produced down-loaded \*.pk font files in HP's pcl format, that gets converted to the Paintbrush \*.pcx format. This process is *not* available with most graphics conversion utilities, e.g. Hijaak or IMSI's Graphics Transformer. However, `hp2pcx` has never disappointed the authors.

<sup>4</sup> The utility supports standard graphics cards (CGA, EGA, VGA, and Hercules), as well as Super VGA cards (800×600 modes provided by Tseng, Paradise, and Video Seven). The automatic display mode can easily be over-ridden to force an image onto the screen in any desired mode.

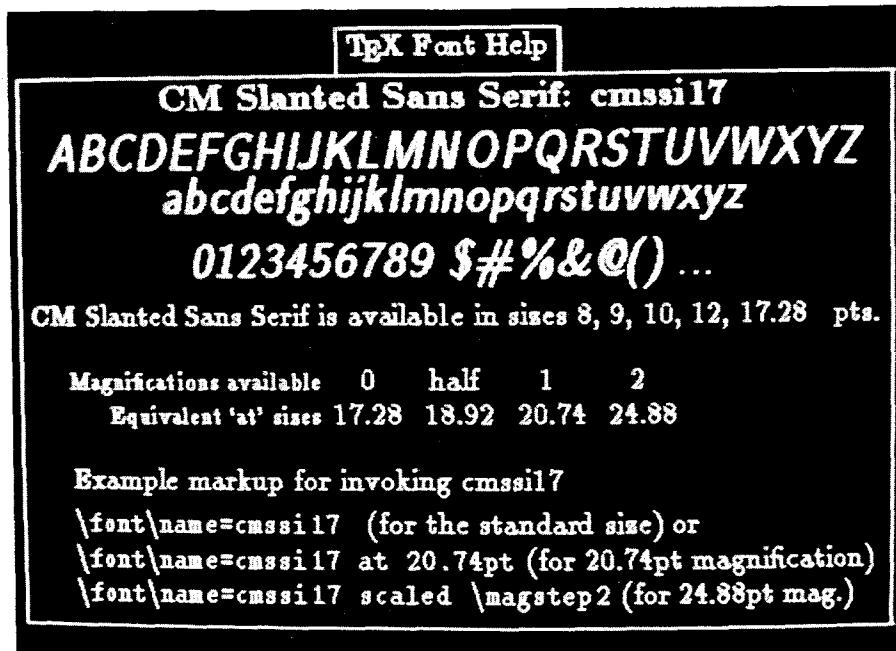


Figure 4: Typical TeX graphical font help display file

### 5.1 Example of a REXX Editing Macro for *TeXT1* Conversion

There is a feature that eases the creation of TeX documents that can serve as an example illustrating the inter-relationship between the editor and REXX, the procedural language. Displaying the text on the monitor in a form that resembles the format after TeX formats the text is a desirable way to edit and view files. For example, a TeX file could have one line of text that will produce three lines of centered text when formatted and printed. It would be more readable to have three centered lines appear in the editor.

Taking this one step further, and using outlines as an example, it would be suitable to have text appear in outline format on the screen with text incrementally indented for each outline level. Also, for editing purposes, it would be advantageous to allow the particular outline level to be easily changed to another level. In one operation, the text will be reformatted for the screen display and the TeX control sequences that format the text will change to alter the final TeX output. This is performed within the editor by hitting one key, Alt-L, that loads the key definitions for list levels and displays the definitions along the bottom of the screen, as shown in Figure 5.

Keys F1 through F7 will create up to seven levels of outline lists, while Alt-F1 through Alt-F7 will, if the cursor is anywhere within an outline level, alter the current indent level. For example, if the outline had a section in the third indent level and the user wanted to alter this to a second level, then hitting Alt-F2 would change \l1l3 to \l1l2<sup>5</sup> and re-format the text with appropriate indentation. This feature allows quick and painless editing of list levels for *TeXT1*.

### 6. Problems and Idiosyncrasies

No system is without its flaws. There are some recommendations on the use of the TeX interface that significantly increase its performance. The way REXX is located in memory, applications should not be made resident while the user is in the KEDIT environment unless they are removed from memory before exiting the editing environment. The availability of LIM Expanded Memory Specification (EMS memory) alleviates the problem of overloading DOS with large macros or window information. To make the system run faster, the macros should be placed in a virtual disk. A significant amount of time is

<sup>5</sup> Here, \l1l3 and \l1l2 stand for \listlevel3 and \listlevel2, the automated list macro for the *TeXT1* macros.

e:\test.tex Line=5 Col=11 Size=25 Alt=3,3  
 \* \* \* Top of File \* \* \*  
 \S1\Design of a High-Efficiency Solar Energy System  
 \S1\atnamesixvii = cnsst17 at 17pt  
 \S1\begin  
 \S1\INTRODUCTION  
**\S1\2 Current design trends in solar energy**  
 \S1\2 Need for higher efficiencies  
 \S1\2 New design approaches  
 \S1\2 Dual concentrator collector array  
 \S1\2 Improved transport fluids  
 \S1\2 THEORY OF THE DUAL CONCENTRATOR COLLECTOR ARRAY  
 \S1\2 Principle of multistage concentration of light rays  
 \S1\2 Use of both Fresnel lenses and mirrors for tracking  
 arrays  
 \S1\2 Method of successive concentrations  
 \S1\2 Comparative analysis of collector arrays  
 \S1\2 Energy gain calculations  
 \S1\2 Effects of adding successive array levels  
 F1 to F7 = \S1\~ where \~ = listlevel number      F8=Bullet      F9=List Resume  
 A1 to A7 = alter listlevel to 1-7      F10=Listbegin F11=Listend F12=Alt Keys  
 →

Figure 5: Screen display of outline list file in list editing mode.

taken if macros have to be constantly read from the hard drive. The *T<sub>E</sub>X* interface has been designed assuming the user has an extended keyboard; the standard keyboard restricts a portion of the features of the interface from being applicable. At this time, there is no elegant accommodation for the standard keyboard.

From an aesthetic viewpoint, the standard PC graphics cards (e.g., VGA and EGA) include the option of setting the number of lines that are displayed on the screen. The authors prefer 28-line mode for several reasons. Many of the utility macros, although still functional in other line modes, simply look best in 28-line mode. Also, the authors have modified many of the default ASCII characters (those above decimal 128) to letters and shapes that are useful for display solely in this mode. These modified characters can be made for other line modes, but, although not a difficult task, it is very time consuming. The entire Greek alphabet, including upper-case letters, has been installed in a modified character set. These Greek letters, which represent simple ASCII numerals, are macros that will print their corresponding character. This "substitution" is useful for typing mathematical equations. Not only will the equation appear more representative of what will be printed, but will also shorten the typed length in the file, thus making it easier to read and debug. The authors have also added character shapes that permit two types of three-dimensional border effects and a descending capital E used in a text-mode *T<sub>E</sub>X*, are shown in Figure 1.

## 7. Portability

The practicality of the *T<sub>E</sub>X* interface would not be appreciated if it could not be easily transferred to other PCs with a wide variety of associated hardware and support software. The first attempt to copy the *T<sub>E</sub>X* interface to another computer proved to be awkward, because the authors had written into the macros several commands that were specific to the directory setup and hardware of the host computer. Some major modifications were immediately implemented. All paths and file identifications were removed from the macros and condensed into a single file, called *config.kex*, from which each macro then calls and retrieves particular information. Therefore, the only file that needs to be altered when copying the KEDIT and REXX files is *config.kex*, which will "personalize" the *T<sub>E</sub>X* interface for individual PCs. In addition, there are REXX functions located in several utility macros that are able to distinguish the hardware setup of the user's computer. For example, the REXX *pcfloppy*

command will return the number of floppy disks available to the system. This command is useful for a PCTOOLS-type of utility that enables the user to easily move to and scan other directories.

## 8. Conclusions

The TeX interface has served the intended purpose the authors were originally trying to achieve: to speed up the creation and modification of TeX documents and to bypass the need to memorize markup. In the process of creating this interface, many additional macros, not necessarily related to TeX, were implemented to complement the editing software. There are a number of projects the authors feel would be extremely useful for the interface but have yet to be accomplished or finished.

1. The TeX formatter, Arbortxt's  $\mu$ TeX, when encountering an error, will not return the line number of the error back to the calling routine. It would be convenient to immediately return to the location of the error in the editing environment so that it can quickly be corrected. However, there is a roundabout solution to this problem. The error line number is written to the \*.log file that is created when the TeX file is formatted. It is possible to read from this log file and retrieve the line number, but this method is exceedingly inefficient. The authors will wait for the next version of  $\mu$ TeX to see if this problem is addressed. Hopefully, by means of setting an environment variable.
2. An example of the auto-reformatting of text has been presented in this paper for *TEXT1*'s list level markup. Similar auto-reformatting will include block quotes, labels, centerlines, hanging paragraphs, justification (right and left), and subheadings.
3. Many of the *TEXT1* font sets are incomplete. For example, a majority of the font sets omit the bold italic font. In the future, all the font sets will contain the six standard text faces: roman, **bold**, *italic*, **typewriter**, **SMALL CAPS**, and **bold italic**. These extra faces will be created using PC-METAFONT.
4. An interesting addition to the input control option of the interface, besides those already mentioned, would include clip art. The integration of graphic pictures and figures that could be cursor-selected and viewed (similar to the font sets) would be very useful. This addition is already in progress.

There is virtually no limit to the TeX interface, other than given that it is a text-mode only editing environment. The foundation has already been made; all new ideas are simply "tacked" onto the option windows and given new function keys to implement them.

## Bibliography

- Kubik, Kim. "AmigaTeX . . . or How Envy Was Resisted and Knowledge Found on the Road to Ööç." *TUGboat* 10:65–67, 1989.
- Riley, Don L. *Using TEXT1: A Set of TeX Macros at Sandia*. Livermore, CA: Sandia National Laboratories, SAND89-8238, 1989.
- Rokicki, Tomas. "The Commodore Amiga: A Magic TeX Machine." *TUGboat* 9:40–41, 1988.
- TEXT1: Reference Manual*. Computing Service Center, Washington State University, 1987.

### Request for Information

The TeX Users Group maintains a database and publishes a membership list containing information about the equipment on which TeX is (or will be) installed and about the applications for which TeX is used. This list is updated periodically and distributed to members with TUGboat, to permit them to identify others with similar interests. Thus, it is important that the information be complete and up-to-date.

Please answer the questions below, in particular those regarding the status of TeX and the hardware on which it runs. (Operating system information is particularly important in the case of IBM mainframes and VAX.) This hardware information is used to group members in the listings by computer and output device.

If accurate information has already been provided by another TUG member at your site, indicate that member's name and the same information will be repeated automatically under your name. If your current listing is correct, you need not answer these questions again. Your cooperation is appreciated.

- Send completed form with remittance (checks, money orders, UNESCO coupons) to:

TeX Users Group  
P. O. Box 594  
Providence, Rhode Island 02901, U.S.A.

- For foreign bank transfers direct payment to the TeX Users Group, account #002-031375, at:

Rhode Island Hospital Trust National Bank  
One Hospital Trust Plaza  
Providence, Rhode Island 02903-2449, U.S.A.

- General correspondence about TUG should be addressed to:

TeX Users Group  
P. O. Box 9506  
Providence, Rhode Island 02940-9506, U.S.A.

Name: _____
Home [ ] Address: _____ Bus. [ ] _____ _____ _____

Qty	1990 Membership/TUGboat Subscription (Jan.-Dec.)											Amount
New (first-time):	[ ] \$35.00 each											
Renewal:	[ ] \$45.00; [ ] \$35.00 - reduced rate if renewed before February 1, 1990											
Mailing charges per subscription:	Canada/Mexico - \$5; Europe - \$10; Other Countries - \$15											
TUGboat back volumes	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989		
Circle volume(s) desired:	v. 1	v. 2	v. 3	v. 4	v. 5	v. 6	v. 7	v. 8	v. 9	v. 10		
Indiv. issues \$18.00 ea.	\$18	\$50	\$35	\$35	\$35	\$50	\$50	\$50	\$50	\$70		

Issues of TUGboat will be shipped via air service outside North America.

Quantity discounts available on request.

TOTAL ENCLOSED: \_\_\_\_\_  
(Prepayment in U.S. dollars required)

### Membership List Information

Institution (if not part of address): \_\_\_\_\_

Date: \_\_\_\_\_

Title: \_\_\_\_\_

Status of TeX: [ ] Under consideration

Phone: \_\_\_\_\_

[ ] Being installed

Network address: \_\_\_\_\_

[ ] Up and running since: \_\_\_\_\_

[ ] Arpanet [ ] BITnet  
[ ] CSnet [ ] uucp  
[ ] JANET [ ] other \_\_\_\_\_

Approximate number of users: \_\_\_\_\_

Specific applications or reason for interest in TeX:

Version of TeX:

[ ] Pascal

[ ] C

[ ] other (describe) \_\_\_\_\_

From whom obtained: \_\_\_\_\_

My installation can offer the following software or technical support to TUG:

Hardware on which TeX is used:

Computer(s)	Operating system(s)	Output device(s)
-------------	---------------------	------------------

Please list high-level TeX users at your site who would not mind being contacted for information; give name, address, and telephone.

_____	_____	_____
_____	_____	_____
_____	_____	_____

# Have You Met Your Mac?

We'd like to suggest a small heresy—that the Apple Macintosh, with our *Textures* software, might just possibly be the best vehicle for T<sub>E</sub>X users right now. Not just the best low-cost alternative—given the choice of any system at any price, you could rightly choose the Macintosh and *Textures*.

If you think the Macintosh is a toy, look at the Macintosh II. It's definitely not for kids—16MHz 68020, memory to 8MB, disk to 300MB, large sharp screens. And *Textures* is not a micro-T<sub>E</sub>X; your T<sub>E</sub>X files will run unchanged.

*Textures* gives you T<sub>E</sub>X at your fingertips—responsive, integrated. Go from editing this copy, through T<sub>E</sub>X, to previewing the finished page—one keystroke, three seconds. Go from page to page—one mouse-click, one second. Scan the whole page for form, proof it at 12 or 14 or 20 points, click the magnifier and check that equation at LaserWriter resolution, instantly. Make your current macro set preloaded in seconds, anytime.

T<sub>E</sub>X is the world's most capable typesetting software, but T<sub>E</sub>X doesn't do pictures. Adobe Illustrator *does* do pictures, with a line quality finer than any technical pen. Or use MacDraw from Claris for technical drawings; learn it in less than one hour. Image Studio from Letraset does halftones, hand-painted or scanned. All world-class programs, all only on the Macintosh. With these tools (and many others), *Textures* does pictures—on screen, on paper, beautifully.

We're convinced that *Textures* and the Macintosh are worth a serious trial from anyone working with T<sub>E</sub>X. We'll make it easy for you to see what we mean, at our risk. If there's a Macintosh in your neighborhood, order a copy on approval. If no Macintosh is nearby, let us arrange a demonstration at a dealer in your area. You'll like what you see.

Call us on it.

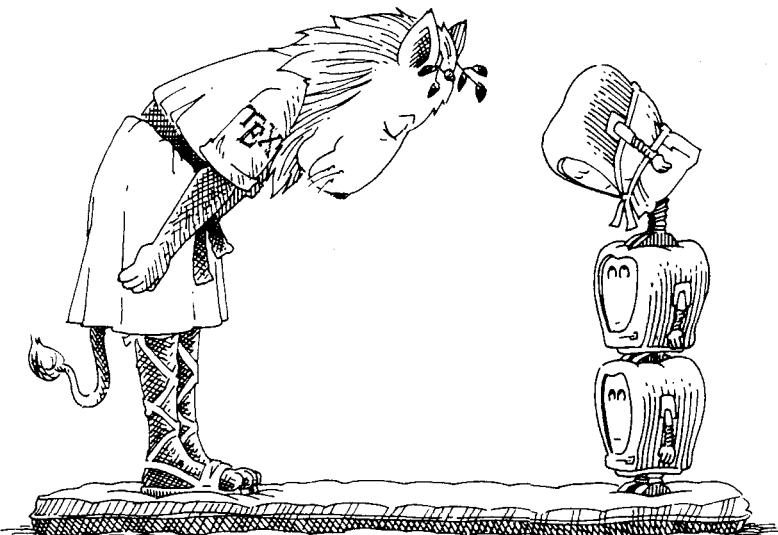
## *Textures*

for the Apple Macintosh

Single Copy \$495  
Educational \$395

Blue Sky Research

534 SW Third Avenue  
Portland, Oregon 97204  
800/622-8398, 503/222-9571  
Telex 9102900911



# ANNOUNCING:

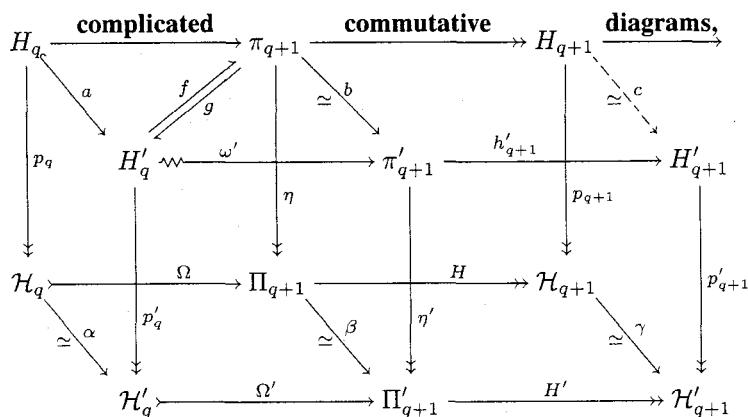
**LAMS-TEX**

*new!*

by  
Michael Spivak,  
author of the  
*AMS-TEX macro package*  
and *The Joy of TEX*

The Synthesis

All the functionality of LATEX, with much greater flexibility, and all features of AMS-TEX, PLUS



		Group C							
		Committee A				Committee B	The T <small>E</small> xplorators Corporation 3701 W. Alabama, Suite 450-273 Houston, TX 77027		
		Unit 1		Unit 2					
complicated tables,		Side 1	Side 2 (Left)	Side 1	Side 2 (Left)				

## AND MUCH, MUCH, MORE!

- Latest *amstex.tex* file (version 2.0)
- LAMS-TEX macro packages
- 300 page Manual (assumes some familiarity with AMS-TEX)
- Fonts for commutative diagrams: .tfm files, .pk files at 118, 180, 240, and 300 dpi, and Meta-Font sources; MAC versions have fonts for TEXtures.
- *dvipaste* program for including tables in files
- *index* program

Single user price \$95; Texas orders add appropriate sales tax. Specify MS-DOS (5 1/4" diskettes) or MAC; for other configurations, write for information. Add \$8 shipping/handling in U.S. and Canada (UPS 2nd day air in U.S., first class to Canada), and \$35 for air shipment elsewhere. Send check or money order to the address cleverly hidden in the table. *Prices subject to change.*



## Publishing Services



### From the Basic

The American Mathematical Society can offer you a basic TeX publishing service. You provide the DVI file and we will produce typeset pages using an Autologic APS Micro-5 phototypesetter. The low cost is basic too: only \$5 per page for the first 100 pages; \$2.50 per page for additional pages, with a \$30 minimum. Quick turnaround is important to you and us ... a manuscript up to 500 pages can be back in your hands in just one week or less.

### To the Complex

As a full service TeX publisher, you can look to the American Mathematical Society as a single source for all your publishing needs.

**Macro-Writing**

**TeX Problem Solving**

**Autologic Fonts**

**Keyboarding**

**Art and Pasteup**

**Camera Work**

**Printing**

**Binding**

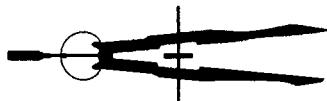
For more information or to schedule a job, please contact Regina Girouard, American Mathematical Society, P.O. Box 6248, Providence, RI 02940 or call 401-272-9500 or 800-556-7774 in the continental U.S.

## DeskJet driver for TeX

With The Toolsmith's DVI driver and the Hewlett-Packard DeskJets you get:

- Print Quality--the precision of laser printers (300 dpi) without the price.
- Speed--with downloaded fonts,  $\approx 1$  page per minute on a DeskJet, faster on a Plus.
- Quiet--non-impact printing and no fan.

You are only limited by the page size and your TeXpertise. \$100 for the DeskJet DVI driver (shipping and any sales tax included). To order or for more information, call or write:



**The Toolsmith**  
P.O. Box 5000  
Davis, CA 95617  
(916) 753-5040

Requires IBM PC or compatible, hard disk, 512K or more RAM, and MS-DOS 2.11 or later. This ad, including the logo, was created on a DeskJet with TeX and METAFONT.

## THE WRITE STUFF, TECHNICALLY SPEAKING

### TeX is the write stuff

*TeX is the powerful publishing system that is guaranteed to make any document you write easier to read... and guaranteed to make that document say the right stuff about you!*

*Micro Programs, Inc. is your source for TeX and related ArborText products for IBM PC and Sun workstations.*

*Call Bob Harris on (516) 921-1351 and get the name of the dealer nearest you.*

**MICRO PROGRAMS, INC.**  
**251 JACKSON AVENUE**  
**SYOSSET**  
**NY 11791**

**Publishing Companion** translates  
**WordPerfect**

to

**TEX**

It doesn't take a TExpert to use TEX.

With **Publishing Companion**, you can publish documents using TEX with **little or no TEX knowledge**. Your WordPerfect files are translated into TEX files, so anyone using this simple word processor can immediately begin typesetting their own documents!

And now, K-Talk introduces **Publishing Companion** version 2.0, which translates **WordPerfect 5.0** files into TEX.

Other word processors are supported using Mastersoft's WordForWord file conversion utility, \$70.

***Special Introductory Offer***

Retail Price .....	\$249
Academic Discount Price .....	\$199

**Introductory Price** ..... **\$179**

This offer good until January 31, 1990. Upgrade from Publishing Companion v.1.XX is \$49.

For the power of TEX with the ease of a word processor, **Publishing Companion** is your "best friend" for desktop publishing.

For more information or to place an order, call or write:



50 McMillen Ave  
Columbus, Ohio 43201  
(614) 294-3535

**DESKTOP PUBLISHING HAS NEVER BEEN SIMPLER  
AND WILL NEVER BE THE SAME**

# MacroTEX

A *T<sub>E</sub>X Macro Toolkit.*

## Clients include:

**MacroTEX:**  
Publishing Companies,  
Research Labs,  
Academic Departments,  
*T<sub>E</sub>X* Users world-wide.

**Macro Writing for Publishing:**  
Addison-Wesley  
MIT Press

Brooks-Cole/Wadsworth  
Prentice-Hall  
Academic Press  
John Wiley and Sons  
Prank and Associates

**Macro Writing for Technical Documents:**  
Shell Research  
American Physical Society  
Technical Typesetting

**Macro Writing for Software Companies:**  
Alpha Software  
Cytel Software  
Intermetrics Corp.  
Saddlebrook Corp.  
Grass Valley Group  
Technical Support Software

**T<sub>E</sub>X-PostScript Interactive Macros:**  
MIT Press, Addison-Wesley

**Teaching T<sub>E</sub>X:**  
Beginning, Intermediate and Advanced Macro Writing;  
DEC, MIT, TUG, NCAR.

## Maximum flexibility

Style files: Generic, Book, Report, Software Documentation, Letter and Note style, each notated and easily adapted to your design.

Macros to help form your own font families, including PostScript font families.

Modular format. Separate files are used for separate functions: listing macros, tables macros, or indexing macros, for example, are called in only when needed.

All or portions of MacroTEX may be added to existing macro files.

Source code included.

## Minimum hassle

All the features you need for document preparation:

- Table of Contents, List of Figures, List of Tables generation
- Multilevel headline and footline, Modular page numbering
- Section heads, Cross-referencing, Listing
- Complete table macros, including tables that continue across pages
- Figure, Table and Program Captions
- Partial page figure that text will wrap around
- Theorem environments, Left-justified equations
- Verbatim that continues across pages, Screen Simulation
- Bibliography, Glossary, Index generation and formatting
- Utilities: Margin control, Footnotes and Endnotes, Diagonal lines, Drop Caps, Margin notes, Mailing labels, Font charts...
- Slides

Single User: \$200, Site Licenses Available.

---

Coming soon: **Post-T<sub>E</sub>X**, our new T<sub>E</sub>X-PostScript package that passes information from PostScript to T<sub>E</sub>X to insert encapsulated PostScript graphics. It passes information from T<sub>E</sub>X to PostScript to position grey screens behind code examples, highlight screen simulations, and position a grey screen in a particular table column entry, as well as and other useful T<sub>E</sub>X-PostScript macros. Post-T<sub>E</sub>X works with both T<sub>E</sub>X and LaT<sub>E</sub>X.

# T<sub>E</sub>Xnology Inc.

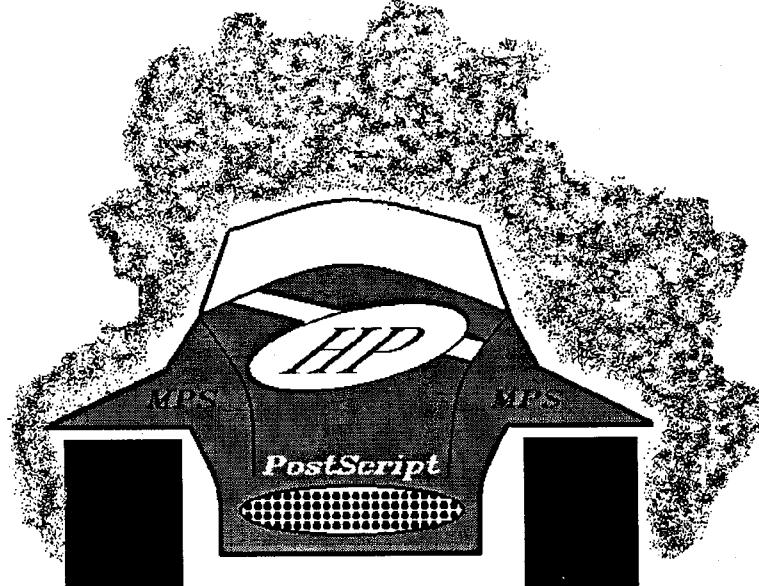
*Amy Hendrickson*

---

57 Longwood Avenue, Brookline MA 02146  
T<sub>E</sub>X, LaT<sub>E</sub>X, and PostScript Consulting

617 / 738-8029

# **T<sub>E</sub>X Plus comes with drivers that get you where you want to go...**



## **in a hurry!**

Our drivers for the HP LaserJet Plus/Series II and PostScript laser printers won't run out of gas in the middle of the race, even under severe memory constraints. Both models include standard equipment such as landscape printing, collating, inclusion of graphics, automatic and interactive font substitution and more, selected from an easy to use menu interface. And (of course) you get a set of CM fonts.

Our PostScript driver provides support for internal and downloaded fonts and includes an .AFM to .TFM translation utility. With the HP driver you'll get a utility which translates HP soft fonts into standard .TFM and .PK files.

T<sub>E</sub>X Plus, which includes both drivers, our T<sub>E</sub>XWRITE editor and CTE<sub>X</sub> (our

version of T<sub>E</sub>X for DOS) is yours for only \$195.00. The T<sub>E</sub>XPRINT drivers can be purchased separately for \$129.00 each. For more information or to place an order, please contact:

**Oregon House Software, Inc.,**

P.O. Box 70,  
12894 Rices Crossing Road,  
Oregon House, CA 95962  
(916) 692-1377

**Micro Publishing Systems, Inc.**

1273 Clyde Avenue,  
West Vancouver, B.C.,  
V7T 1E6 Canada  
(604)926-0500

T<sub>E</sub>X Plus is also available from the T<sub>E</sub>X Users Group.

T<sub>E</sub>X is a trademark of the American Mathematical Society. T<sub>E</sub>X Plus, T<sub>E</sub>XPRINT, T<sub>E</sub>XWRITE and CTE<sub>X</sub> are trademarks of Micro Publishing Systems, Inc. All other product names are the trademarks or registered trademarks of their respective holders.

present

MicroPress



MICRO  
PRESS

67-30 Clyde Street, Rm. 2N  
Forest Hills, New York 11375

(718)-575-1816

TeX for the 90's

# Vector<sup>TM</sup> TeX

Retain all the advantages of TeX and

- Save megabytes of storage.
- Instantly generate any font in any size in any variation (5-70 points).
- Automatically create compressed, slanted, smallcaps, outline or shaded fonts.
- Use either CM-compatible Vector fonts or MicroPress professional typefaces, including Tempo-Roman, Avon Guard, Helvettot,...

Includes the VTEx typesetter, 10 instantly scalable typefaces, VVIEW (arbitrary magnification on EGA, CGA, VGA, Hercules, AT&T,...), VLASER (HP LaserJet), VPOST (PostScript), VDOT(Epson, Panasonic, NEC, Toshiba, Proprinter, Star, Deskjet) and manuals.

**List price .... \$399   Introductory offer .... \$249**

Introductory offer expires on January 1, 1990. S/H \$5. COD add \$5. WordPerfect Interface add \$100. Demo \$3. Site licenses available. Dealers inquires welcome. Professional typefaces and METAFONT sources available for older implementations of TeX. Discounts for PCTeX users.

VTEx IS A TRADEMARK OF MICROPRESS INC.

OTHER PRODUCTS ARE TRADEMARKS OF RESPECTIVE COMPANIES.

THIS AD WAS TYPESET BY VTEx USING SCALABLE FONTS ON A LASERJET.

## Public Domain T<sub>E</sub>X

The authorized and current versions T<sub>E</sub>X software are available from *Maria Code - Data Processing Services* by special arrangement with Stanford University and other contributing universities. The standard distribution tape contains the source of T<sub>E</sub>X and METAFONT, the macro libraries for A<sub>M</sub>S-T<sub>E</sub>X, L<sub>A</sub>T<sub>E</sub>X, SliT<sub>E</sub>X and HP T<sub>E</sub>X, sample device drivers for a Versetec and LN03 printers, documentation files, and many useful tools.

Since these are in the public domain, they may be used and copied without royalty concerns. They represent the official versions of T<sub>E</sub>X. A portion of your tape cost is used to support development at Stanford University.

If you have a DEC VAX/VMS, IBM CMS, IBM MVS or DEC TOPS operating system, you will want to order a special distribution tape which contains "ready-to-run" T<sub>E</sub>X and METAFONT. If you do not have one of these systems, you must perform a more involved installation which includes compiling the source with your Pascal compiler. Ready-to-run versions of T<sub>E</sub>X are available for other systems from various sources at various prices. You may want to examine these before ordering a standard distribution tape.

The font tapes contain GF files for the Computer Modern fonts. While it is possible to generate these files yourself, it will save you a lot of CPU time to get them on tape.

All systems are distributed on 9 track, 1600 bpi magnetic tapes. If both a distribution tape and a font tape are ordered, they may be combined on a single 2400' reel, space permitting.

Your order will be filled with the current versions of software and manuals at the time it is received. If you want a specific version, please indicate that on your order.

Please use the form on the next page for your order. Note that postage, except domestic book rate is based on the item weights in pounds. If you want to place your order by telephone, please call (408) 735-8006 between 9:00 am and 2:00 pm West Coast time. Do not call for technical assistance since no one there can help you.

We normally have a good stock of books and tapes, so your order can be filled promptly — usually within 48 hours.

Make checks payable to *Maria Code - Data Processing Services*. Export orders must have a check drawn on a US bank or use an International Money Order. Purchase orders are accepted.

# T<sub>E</sub>X Order Form

**T<sub>E</sub>X Distribution tapes:**

- Standard ASCII format
- Standard EBCDIC format
- Special VAX/VMS format Backup
- Special DEC 20/TOPS 20 Dumper format
- Special IBM VM/CMS format
- Special IBM MVS format

Tape prices: \$92.00 for first tape,  
\$72.00 for each additional tape.

**Font Library Tapes (GF files)**

- 300 dpi VAX/VMS format
- 300 dpi generic format
- IBM 3820/3812 MVS format
- IBM 3800 CMS format
- IBM 4250 CMS format
- IBM 3820/3812 CMS format

Total number of tapes \_\_\_\_\_

Postage: allow 2 lbs. for each tape

**Documents:**

	Price \$	Weight	Quantity
T <sub>E</sub> Xbook (vol. A) softcover .....	27.00	2	_____
T <sub>E</sub> X: The Program (vol. B) hardcover .....	40.00	4	_____
METAFONT book (vol. C) softcover .....	22.00	2	_____
METAFONT: The Program (vol. D) hardcover ..	40.00	4	_____
Computer Modern Typefaces (vol. E) hardcover	40.00	4	_____
L <sub>A</sub> T <sub>E</sub> X document preparation system .....	27.00	2	_____
WEB language *	12.00	1	_____
T <sub>E</sub> Xware *	10.00	1	_____
BibT <sub>E</sub> X *	10.00	1	_____
Torture Test for T <sub>E</sub> X *	8.00	1	_____
Torture Test for METAFONT *	8.00	1	_____
METAFONTware *	15.00	1	_____
Metamarks *	15.00	1	_____

\* published by Stanford University

**Payment calculation:**

Number of tapes ordered \_\_\_\_\_ Total price for tapes \_\_\_\_\_

Number of documents ordered \_\_\_\_\_ Total price for documents \_\_\_\_\_

Add the 2 lines above \_\_\_\_\_

Orders from within California: Add sales tax for your location. \_\_\_\_\_

**Shipping charges:** (for domestic book rate, skip this section)

Total weight of tapes and books \_\_\_\_\_ lbs.

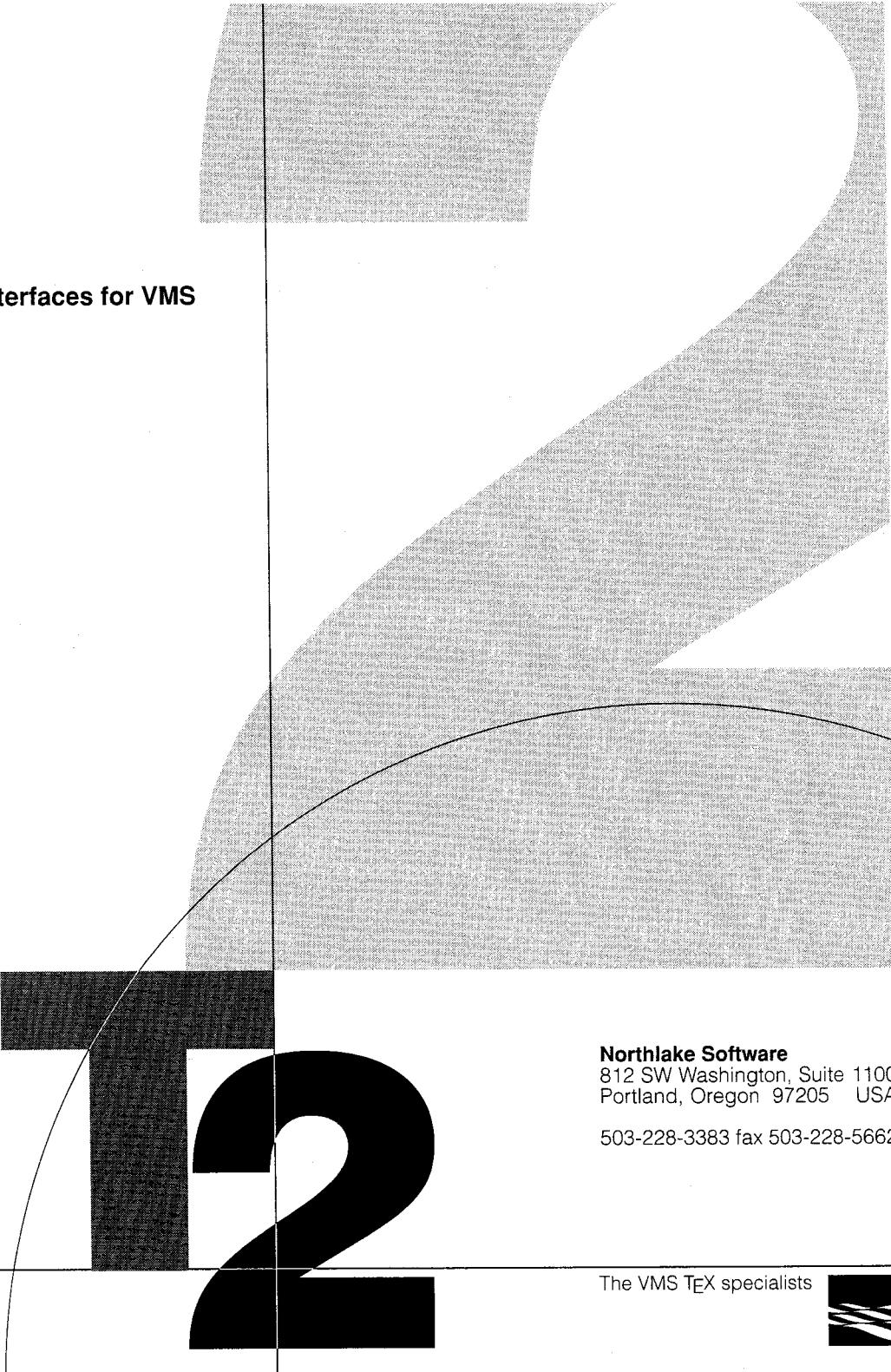
Check	<input type="checkbox"/> domestic priority mail	rate \$1.50/lb.
One	<input type="checkbox"/> air mail to Canada and Mexico:	rate \$2.00/lb.
	<input type="checkbox"/> export surface mail (all countries):	rate \$1.50/lb.
	<input type="checkbox"/> air mail to Europe, South America:	rate \$5.00/lb.
	<input type="checkbox"/> air mail to Far East, Africa, Israel:	rate \$7.00/lb.

Multiply total weight by shipping rate. Enter shipping charges: \_\_\_\_\_

Total charges: (add charges for materials, tax and shipping) \_\_\_\_\_

Send to: Maria Code, DP Services, 1371 Sydney Drive, Sunnyvale, CA 94087.  
Include your name, organization, address, and telephone number.

Are you or your organization a member of TUG? \_\_\_\_\_



**T<sub>E</sub>X Device Interfaces for VMS**

**PostScript**

**LaserJet**

**LN03**

**Northlake Software**  
812 SW Washington, Suite 1100  
Portland, Oregon 97205 USA  
503-228-3383 fax 503-228-5662

The VMS T<sub>E</sub>X specialists



# TEX Users

## Take Note . . .

Computer Composition Corporation offers the following services to those who are creating their technical files using TEX:

- Convert your DVI files to fully paginated typeset pages on our APS-5 phototypesetters at 1400 dpi resolution.
- Files can be submitted on magnetic tape or PC diskettes.
- Provide 300 dpi laser-printed page proofs which simulate the typeset page. (Optional service \$1.50 per page)
- Macro writing and keyboarding from traditionally prepared manuscripts **in several typeface families** via the TEX processing system. Send us your manuscript for our review and quotation.
- Full keylining and camera work services, including halftones, line art, screens and full-page negatives or positives for your printer.
- Quick turnaround (**usually less than 48 hours!**) on customer supplied DVI files of 500 typeset pages or less.
- From DVI files: first 100 typeset pages at \$4.75 per page; 100 pages and over at \$3.50 per page. **Lower prices for slower turnaround service.**

---

*For further information and / or a specific quotation,  
call or write Frank Frye or Tim Buckler*

---



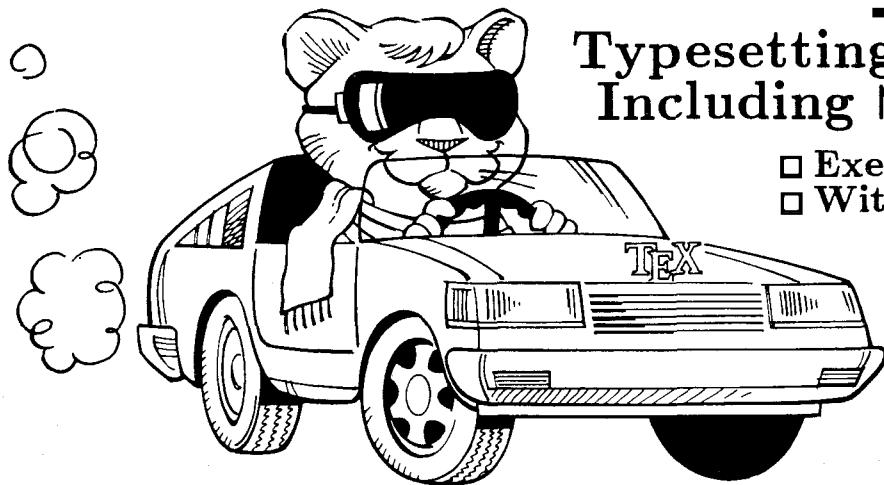
**COMPUTER COMPOSITION CORPORATION**

1401 West Girard Avenue • Madison Heights, MI 48071

**(313) 545-4330    FAX (313) 544-1611**

— Since 1970 —

# TurboTEX



## Typesetting Software Including METAFONT

Executables \$150  
 With source \$300

TurboTEX Release 2.0 software offers you a complete typesetting package based on the TeX 2.95 and METAFONT 1.7 standards: preloaded plain TeX, L<sup>A</sup>T<sub>E</sub>X, INITEX, VIRTEX, and plain METAFONT interfaced to CGA/EGA/VGA/Hercules graphics; TRIP and TRAP certification; Computer Modern and L<sup>A</sup>T<sub>E</sub>X fonts, and printer drivers for HP LaserJet Plus/Series II, PostScript, and dot-matrix printers. New features in the HP LaserJet driver put PCX or TIFF graphics files directly into your TeX documents. This wealth of software fills over 10 megabytes of diskettes, and runs on your IBM PC, UNIX, OS/2, or VAX/VMS system.

**Power Features:** TurboTeX brings big-machine performance to your small computer. TurboTeX breaks the 640K memory barrier under MS-DOS on the IBM PC with our virtual memory sub-system. You'll have the same sized TeX that runs on multi-megabyte mainframes, with plenty of memory for large documents, complicated formats, and demanding macro packages that break other TeX implementations. On

larger computers, TurboTeX runs up to 3 times faster in less memory than the Stanford Pascal distribution.

**Source code:** Order the TurboTeX source in portable C, and you will receive more disks with over 85,000 lines of generously commented TeX, TurboTeX, METAFONT, and printer driver source code, including: our WEB system in C; PASCHAL, our proprietary Pascal-to-C translator; and preloading, virtual memory, and graphics code. TurboTeX meets C portability standards like ANSI and K&R, and is robustly portable to a growing family of operating systems.

**TeX-FAX:** Connects TeX to the FAX revolution. Send perfect TeX output instantly, anywhere, without scanning. With TeX-FAX, any FAX machine in the world becomes your output device! Complete with PC board and software for \$395 (4800 bps) or \$795 (9600 bps).

**Desktop Publishing Interface Option:** Converts TeX output (such as equations or tables) for direct use in programs like Ventura Publisher and Pagemaker. (\$50 for PC).

**Availability & Requirements:** TurboTeX executables for IBM PC's include the User's Guide and require 640K and hard disk. Order source code (includes Programmer's Guide) for other machines. Source compiles with Microsoft C 5.0 or later on the PC; other systems need 1 MB memory and a C compiler supporting UNIX standard I/O. Media is 360K 5-1/4" PC floppy disks; other formats at extra cost.

**No-risk trial offer:** Examine the documentation and run the PC TurboTeX for 10 days. If you are not satisfied, return the software for a 100% refund or credit. (Offer applies to PC executables only.)

### Ordering TurboTeX

Order by phone, FAX, or mail. Terms: Check with order (free media and ground shipping in US), VISA, Mastercard (free media, shipping extra); Net 30 to well-rated firms and public agencies (shipping and media extra). Discounts available for quantities or resale.

Ask for the free, 50-page Buyer's Guide.

### The Kinch Computer Company

PUBLISHERS OF TURBOTEX

501 South Meadow Street  
Ithaca, New York 14850  
Telephone (607) 273-0222  
FAX (607) 273-0484

# Kinch

# TEX TEE. FREE.

*Of course you will want Personal T<sub>E</sub>X, Inc. products for their recognized quality and attractive prices. But on orders of \$300 or more, you'll also get a PC T<sub>E</sub>X T-shirt. Free.*

**PTIJET FOR HP DESKJET.** Full featured printer driver for HP DeskJet, PLUS. Laser quality output.

\$119

**PC T<sub>E</sub>X + PTILASER + PTIVIEW.** T<sub>E</sub>X82, Version 2.9: professional formatting and typesetting results—for amateur prices. Includes INIT<sub>E</sub>X, LaT<sub>E</sub>X, AMS-T<sub>E</sub>X, VANILLA Macro Pak, PC T<sub>E</sub>X and LaT<sub>E</sub>X manuals. Plus a PTILaser device driver, to take full advantage of your laser printer. PLUS the PTIView screen previewer for on-screen previewing of your T<sub>E</sub>X documents and immediate editing. Top performance and low cost make this our most popular package.

\$499

**PC T<sub>E</sub>X + PTILASER.** As above, but without the PTIView screen previewer.

\$399

**PC T<sub>E</sub>X + PTIDOT + PTIVIEW.** This package gives you all the T<sub>E</sub>X and PTIView benefits, together with our dot-matrix device driver for reliable, low cost printing.

\$399

**PC T<sub>E</sub>X + PTIJET + PTIVIEW.** Same as the above package, but with PTIJet instead of PTIDot. Laser quality output.

\$429

**PC T<sub>E</sub>X + PTIJET.** As above, without the PTIView screen previewer.

\$329

**PCMF—METAFONT for the PC.** Lets you design fonts and create graphics. (Not for the novice.) pcMF Version 1.7.

\$195

**PTI LASER HP+, SERIES II.** This device driver for the HP LaserJet Plus and Series II laser printers takes full advantage of the 512K resident memory.

\$195

**PTI LASER POSTSCRIPT.** Device driver for PostScript printer; allows the resident fonts and graphic images to be used in T<sub>E</sub>X in documents.

\$195

**PTI FONTWARE Interface Package.** Software to generate Bitstream outline fonts at any size. (The Interface is necessary to use Bitstream fonts. Fonts are not included—order below).

\$95

**PTI FONTWARE WITH SWISS or DUTCH.** Same as above but includes your choice of either Swiss or Dutch at a special bundled price.

\$179

**BITSTREAM Font Families.** An extensive library of 30 type families, in any size you specify, with true typographic quality.

Each family: \$179



To order, just dial

**(415) 388-8853**

12 Madrona Avenue Mill Valley, CA 94941 FAX: (415) 388-8865 VISA, MC accepted.

Requires: DOS 2.0 or later, 512K RAM, 10M hard disk. T<sub>E</sub>X is an American Mathematical Society TM. PC T<sub>E</sub>X is a Personal T<sub>E</sub>X, Inc. TM. Manufacturers' names are their TMs. Outside the USA, order through your local PC T<sub>E</sub>X distributor. Inquire about available distributorships and site licenses. This ad was produced using PC T<sub>E</sub>X and Bitstream fonts.

# The Joy of $\text{\TeX}$



## A Gourmet Guide to Typesetting with the $\mathcal{AM}\mathcal{S}$ - $\text{\TeX}$ macro package

M. D. SPIVAK, Ph.D.

*The Joy of  $\text{\TeX}$*  is the user-friendly user's guide for  $\mathcal{AM}\mathcal{S}$ - $\text{\TeX}$ , an extension of  $\text{\TeX}$ , Donald Knuth's revolutionary program for typesetting technical material.  $\mathcal{AM}\mathcal{S}$ - $\text{\TeX}$  was designed to simplify the input of mathematical material in particular, and to format the output according to any of various preset style specifications.

There are two primary features of the  $\text{\TeX}$  system: it is a computer system for typesetting technical text, especially text containing a great deal of mathematics; and it is a system for producing beautiful text, comparable to the work of the finest printers.

Most importantly,  $\text{\TeX}$ 's capabilities are not available only to  $\text{\TeX}$  experts. While mathematicians and experienced technical typists will find that  $\text{\TeX}$  allows them to specify mathematical formulas with great

accuracy and still have control over the finished product, even novice technical typists will find the manual easy to use in helping them produce beautiful technical  $\text{\TeX}$ t.

This book is designed as a user's guide to the  $\mathcal{AM}\mathcal{S}$ - $\text{\TeX}$  macro package and details many features of this extremely useful text processing package. Parts 1 and 2, entitled "Starters" and "Main Courses," teach the reader how to typeset most normally encountered text and mathematics. "Sauces and Pickles," the third section, treats more exotic problems and includes a 60-page dictionary of special  $\text{\TeX}$ niques.

Exercises sprinkled generously through each chapter encourage the reader to sit down at a terminal and learn through experimentation. Appendixes list summaries of frequently used and more esoteric symbols as well as answers to the exercises.



ISBN 0-8218-2999-8, LC 85-7506  
290 pages (softcover), 1986  
AMS Indiv. Memb. \$29, AMS Inst.  
Memb. \$32, List price \$36  
To order specify JOYT/T

Free shipment by surface. For air  
delivery, add: 1st book \$5, each  
add'l \$3, max. \$100

**PREPAYMENT REQUIRED.** Order from  
American Mathematical Society  
P. O. Box 1571  
Annex Station  
Providence, RI 01901-1571  
or call 800-556-7774 to use VISA or MasterCard.  
Prices subject to change.

## Updated TeX Products

### AMS-Tex Version 2.0 (Available January 1990)

AMS-Tex, the TeX macro package that simplifies the typesetting of complex mathematics, has been updated to version 2.0. AMS-Tex is intended to be used in conjunction with AMSFonts 2.0 (see below). However, if the purchaser does not need the extra symbols and Euler characters found in AMSFonts, AMS-Tex can be run without AMSFonts. (AMS-Tex 2.0 cannot be used with previous versions of AMSFonts.) AMS-Tex is available on IBM or Macintosh diskettes—either format may be uploaded to many mainframe computers.



### AMSFonts Version 2.0 (Available January 1990)

AMSFonts 2.0 are designed either for use with AMS-Tex 2.0, or for use with Plain TeX. (AMSFonts 2.0 cannot be used with previous versions of AMS-Tex.) There will be two distributions of fonts: one that can be used on PC's as well as mainframes, and one that can be used on a Macintosh with *Textures*. The fonts that will be included on these distributions are:

Font Name	Description	Point Sizes	Font Name	Description	Point Sizes
CMCSC	CM Caps and Small Caps	8-9*	MSAM	Symbols	5-10
CMMIB	CM Math Italic Boldface	5-9*	MSBM	Symbols (w/Blackboard Bold)	5-10
CMBSY	CM Bold Symbols	5-9*	WNCYR	Cyrillic Upright	5-10**
EURB	Euler Cursive Boldface	5-10	WNCYI	Cyrillic Italic	5-10**
EURM	Euler Cursive Medium	5-10	WNCYB	Cyrillic Boldface	5-10**
EUFB	Euler Fraktur Boldface	5-10	WNCYSC	Cyrillic Caps and Small Caps	10***
EUFM	Euler Fraktur Medium	5-10	WNCYSS	Cyrillic Sans Serif	8-10**
EUSB	Euler Script Boldface	5-10			
EUSM	Euler Script Medium	5-10			

\* 10 point is included in the standard TeX distribution.

\*\* Developed by the University of Washington

#### AMSFonts for PC or mainframe

- Font Resolution: 118, 180, 240, 300, 400 dpi (one resolution per order).
- Magnification: All the standard TeX magnifications will be included. The standard magnifications are: 100, 109.5, 120, 144, 172.8, 207.4, and 248.8%.
- Format: Available on both 3.5" and 5.25" diskettes, in either high or low density (the default will be 5.25", high density).

#### AMSFonts for use on a Macintosh with *Textures*

- Font Resolution: 72, 144, and 300 dpi (all resolutions included in each order).
- Magnification: The standard distribution will include fonts at 100% and 120%. An extended distribution, which will contain all the standard TeX magsteps, will also be available.
- Format: The Macintosh fonts will be available on double-sided double-density 3.5" diskettes.

**PRICES:** AMS-Tex: List \$30, AMS member \$27

AMSFonts: List \$45, AMS member \$41

AMS-Tex and AMSFonts: List \$65, AMS member \$59

Shipping and handling: \$8 per order in the US and Canada, \$15 elsewhere.

**HOW TO ORDER:** Prepayment is required. Free upgrades are available for those who have purchased previous versions. When ordering AMSFonts for the PC, specify desired resolution, diskette size, and diskette density.

For more information: Call the AMS at (401) 272-9500, or (800) 556-7774 in the continental U.S. or write to: TeX Library, American Mathematical Society, P.O. Box 6248, Providence, RI 02940.

# TYPESETTING: JUST \$2.50 PER PAGE!

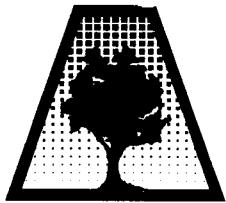
Send us your T<sub>E</sub>X DVI files and we will typeset your material at 2000 dpi on quality photographic paper — \$2.50 per page!

Choose from these available fonts: Computer Modern, Bitstream Fontware™, and any METAFONT fonts. (For each METAFONT font used other than Computer Modern, \$15 setup is charged. This ad was composed with PCT<sub>E</sub>X® and Bitstream Dutch (Times Roman) fonts, and printed on RC paper at 2000 dpi with the Chelgraph IBX typesetter.)

And the good news is: just \$2.50 per page, \$2.25 each for 100+ pages, \$2.00 each for 500+ pages! Laser proofs \$.50 per page. (\$25 minimum on all jobs.)

Call or write today for complete information, sample prints, and our order form. **TYPE 2000, 16 Madrona Avenue, Mill Valley, CA 94941. Phone 415/388-8873.**

— TYPE —  
*2000*



# T<sub>E</sub>X FOR THE HP 9000

ARBORTEXT INC.

ArborText introduces T<sub>E</sub>X for the HP 9000,  
Series 300 workstations.

The new release includes the  
T<sub>E</sub>X Package, Preview for  
the X Window System,  
DVILASER for  
PostScript and  
HP printers.

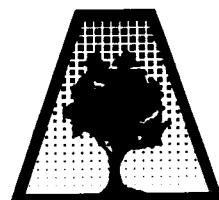
**COMING  
SOON**

- Improved DVILASER/PS  
with expanded support for  
Encapsulated PostScript®

- X11 versions of Preview for Sun and  
Apollo workstations

- Enhanced μT<sub>E</sub>X with support for extended or  
expanded memory

## COMMITTED TO THE FUTURE



ARBORTEXT INC.

535 West William Street, Suite 300, Ann Arbor, MI 48103 • (313) 996-3566 • FAX (313) 996-3573

Apollo is a trademark of Apollo Computer, Inc. HP 9000 is a trademark of the Hewlett-Packard Corp. PostScript is a registered trademark of Adobe Systems, Inc. Sun is a trademark of Sun Microsystems, Inc. T<sub>E</sub>X is a trademark of the American Mathematical Society. The X Window System is a trademark of the Massachusetts Institute of Technology.