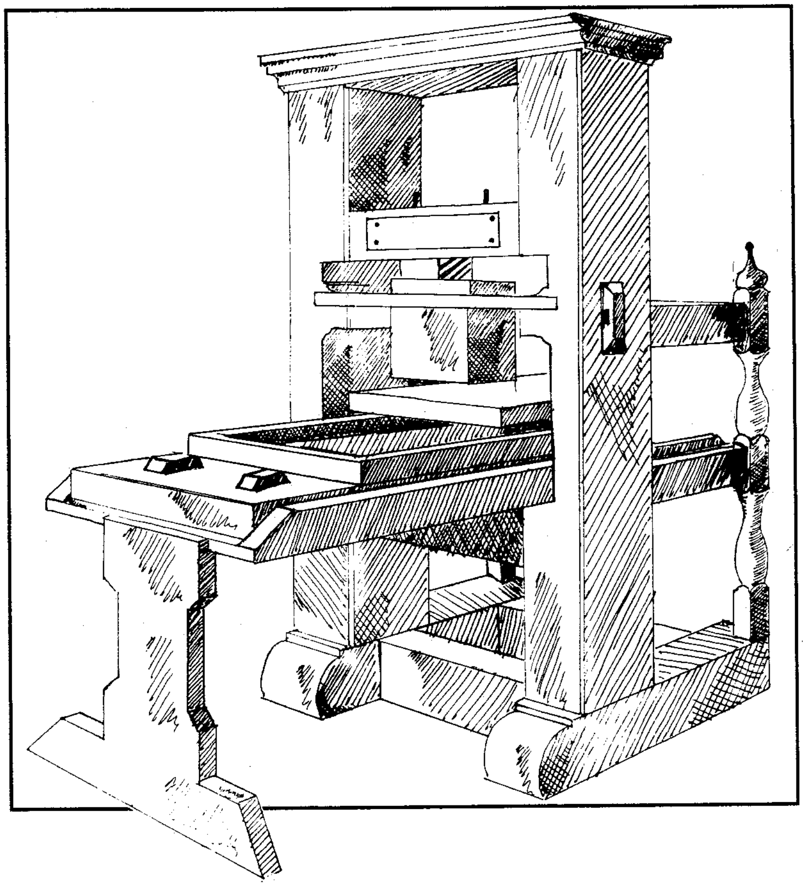
TUGBOAT

The Communications of the TEX Users Group



Volume 12, Number 4, December 1991
1991 TUG Conference Proceedings

TeX Users Group

Memberships and Subscriptions

TUGboat (ISSN 0896-3207) is published four times a year plus one supplement by the TeX Users Group, 653 North Main Street, P. O. Box 9506, Providence, RI 02940, U.S.A.

1992 dues for individual members are as follows:

- Ordinary members: $60
- Students: $50

Membership in the TeX Users Group is for the calendar year, and includes all issues of *TUGboat* and *TeX & TUG News* for the year in which membership begins or is renewed. Individual membership is open only to named individuals, and carries with it such rights and responsibilities as voting in the annual election. A membership form is provided on page 572.

TUGboat subscriptions are available to organizations and others wishing to receive *TUGboat* in a name other than that of an individual. Subscription rates: North America $60 a year; all other countries, delivery by surface mail $60, by air mail $80.

Second-class postage paid at Providence, RI, and additional mailing offices. Postmaster: Send address changes to the TeX Users Group, P. O. Box 9506, Providence, RI 02940, U.S.A.

Institutional Membership

Institutional Membership is a means of showing continuing interest in and support for both TeX and the TeX Users Group.

For further information, contact the TUG office. An application form for institutional membership may be found in this issue of *TUGboat* on page 573.

TUGboat © Copyright 1991, TeX Users Group

Permission is granted to make and distribute verbatim copies of this publication or of individual items from this publication provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this publication or of individual items from this publication under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this publication or of individual items from this publication into another language, under the above conditions for modified versions, except that this permission notice may be included in translations approved by the TeX Users Group instead of in the original English.

Some individual authors may wish to retain traditional copyright rights to their own articles. Such articles can be identified by the presence of a copyright notice thereon.

Board of Directors

Donald Knuth, *Grand Wizard of TeX-arcana*†
Nelson Beebe, *President**
Christina Thiele, *Vice President/Secretary**
Allen Dyer, *Treasurer**
Malcolm Clark, *President-Elect*, Vice-President for the U.K.*
Bernard Gaulle, *Vice-President for GUTenberg*
Roswitha Graham, *Vice-President for the Nordic countries*
Kees van der Laan, *Vice-President for NTG*
Joachim Lammarsch, *IBM VM/CMS Site Coordinator, Vice-President for DANTE*
Barbara Beeton, *TUGboat Editor*
Lance Carnes, *Small Systems Site Coordinator*
Bart Childs, *DG MV Site Coordinator*
John Crawford, *Prime 50 Series Site Coordinator*
David Fuchs
Regina Girouard
Raymond Goucher, *Founding Executive Director*†
Dean Guenther
Hope Hamilton
Doug Henderson, METAFONT *Coordinator*
Alan Hoenig*
Patrick Ion
David Kellerman, *VAX/VMS Site Coordinator*
David Kratzer
Pierre MacKay, UNIX *Site Coordinator*
Craig Platt, *IBM MVS Site Coordinator*
Hermann Zapf, *Wizard of Fonts*†

* *member of executive committee*
† *honorary*
See the 1991 TUG Membership List for addresses.

Addresses

General correspondence:
TeX Users Group
P. O. Box 9506
Providence, RI 02940

Payments:
TeX Users Group
P. O. Box 594
Providence, RI 02901

Parcel post,
delivery services:
TeX Users Group
653 North Main Street
Providence, RI 02904

Telephone

401-751-7760

Fax

401-751-1071

Electronic Mail

(Internet)
General correspondence:
`TUG@Math.AMS.com`

Submissions to *TUGboat*:
`TUGboat@Math.AMS.com`

TeX is a trademark of the American Mathematical Society.

1991 Annual Meeting Proceedings, Part 2

TEX Users Group
Twelfth Annual Meeting
Dedham, Massachusetts, July 15–18, 1991

TUGBOAT

COMMUNICATIONS OF THE TEX USERS GROUP

TUGBOAT EDITOR BARBARA BEETON
PROCEEDINGS EDITOR HOPE HAMILTON

VOLUME 12, NUMBER 4 • DECEMBER 1991
PROVIDENCE • RHODE ISLAND • U.S.A.

Production Notes

Many thanks are given to the editorial team which tackled proof-reading and copy-editing for these *Proceedings*. In addition to *Proceedings* Editor Hope Hamilton, this team consisted of:

Mimi Burbank
Dian De Sha
Christina Thiele

TEX source files were transmitted via network or diskette from authors to the editorial team and then on to the TUG office for issue assemblage and final changes. PC TEX, μTEX and emTEX were used in the office on an IBM PC-compatible 386 to generate dvi files. These files were then shipped to the American Mathematical Society via telephone line, and final copy was produced on the Society's APS μ5 and AGFA-Compugraphic 9600G (articles by Haralambous and Roth).

One article was produced locally by authors (Kakiuchi, et al.) and camera copy was mailed to the editors for incorporation in the issue. Three other articles (by Williams, Horstmann, and Hoenig) contained figures supplied by the authors and stripped into the final copy by the TUG office.

Trademarks

Many trademarked names appear in the pages of *TUGboat*. If there is any question about whether a name is or is not a trademark, prudence dictates that it should be treated as if it is. The following list of trademarks which appear in this issue may not be complete.

APS μ5 is a trademark of Autologic, Inc.
DOS and MS/DOS are trademarks of MicroSoft Corporation
LaserJet, PCL, and DeskJet are trademarks of Hewlett-Packard, Inc.
METAFONT is a trademark of Addison-Wesley Inc.
PC TEX is a registered trademark of Personal TEX, Inc.
PostScript is a trademark of Adobe Systems, Inc.
TEX and $\mathcal{AMS}$-TEX are trademarks of the American Mathematical Society.
UNIX is a trademark of AT&T Bell Laboratories.

Other Conference Proceedings

Europe

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.]

TEX88 Conference Proceedings. Malcolm Clark, ed. Chichester, England: Ellis Horwood, 1990. [18–20 July 1988, Exeter University, Exeter, England.]

TEX90 Conference Proceedings. Mary Guenther, ed. *TUGboat* 12, no. 1. Providence, Rhode Island: TEX Users Group, 1991. [10–13 September 1990, University College, Cork, Ireland.]

North America

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.]

Conference Proceedings: TEX Users Group Tenth Annual Meeting. Christina Thiele, ed. *TUGboat* 10, no. 4. Providence, Rhode Island: TEX Users Group, 1989. [20–23 August 1989, Stanford University, Stanford, California.]

1990 Annual Meeting Proceedings: TEX Users Group Eleventh Annual Meeting. Lincoln Durst, ed. *TUGboat* 11, no. 3. Providence, Rhode Island: TEX Users Group, 1990. [18–20 June 1990, Texas A&M University, College Station, Texas.]

Automatic Conversion from a Scientific Word Processor to TEX

Cay S. Horstmann
Department of Mathematics and Computer Science
San Jose State University
San Jose, CA 95192-0103
408-942-0461
Horstmann Software Design Corporation
Four North Second Street, Suite 500
San Jose, CA 95113
408-298-0828; FAX: 408-298-6157
Internet: horstman@sjsumcs.sjsu.edu

Abstract

In this paper, we report on our experience with a utility which converts files written with the ChiWriter scientific word processor into TEX files. With this converter, it is feasible to write a manuscript in a "what-you-see-is-what-you-get" (WYSIWYG) fashion, with all fonts, special symbols, mathematical formulas, and tables displayed correctly on the screen during editing, and to translate the document into TEX for publication. This method has several advantages over typing straight TEX code. The word processor is easier to learn, and it is easier to revise material that is displayed on the screen without codes. We describe design decisions and limitations of our approach.

Features of Our Word Processor

The CHI2TEX converter described in this article, as well as ChiWriter, its source word processor, are commercial products, available from Horstmann Software and its international distributors. Many of the issues raised here apply to the design of conversion software from another scientific word processor as well, and some observations are valid for general purpose word processors. In the following, we will refer to ChiWriter and CHI2TEX as "our word processor" and "our converter".

Our word processor has the same capabilities as most other word processors: cut and paste, search and replace, spell checking, etc. The program operates in graphics mode. Characters in fonts such as bold, italic, Greek, and math are displayed correctly on the screen. A number of features differentiate it from general purpose word processors. Multiple superscripts and subscripts (e.g., x_1^2, x^{n_k}) are supported and correctly displayed. Mathematical formulas, such as fractions or integrals, can be entered as easily as any other text. There is no separate "equation mode" and no code language for formula entry. No separate preview step is required to view the formulas in the doucment.

An older version of our word processor (ChiWriter version 3) employs a very simple imaging model. It essentially simulates a "golf ball" style typewriter. The cursor can be moved vertically in half-line steps and characters can be placed anywhere on the screen. The user must piece together fractions, roots and integral symbols from building blocks. While this is quite intuitive for the typist and requires essentially no learning curve, it is tedious to revise formulas entered in this way. For a review of this program, see Milne.

It was quite a challenge to write a converter that is able to scan mathematical formulas in this pictorial representation and translate them into the logical structure required by TEX. Our scanning algorithm translates most formulas surprisingly well; and, with a bit of foresight, formulas can be entered to be translated reliably.

The current version of the program (version 4) supports automatic formatting of mathematical structures. For example, when editing a fraction, the numerator and denominator are continuously centered and the fraction bar expands or shrinks to the correct length. Because the word processor is aware of the structures, no guessing is required for conversion of mathematical structures and tables.

User Acceptance

Users who wrote their document using the ChiWriter word processor, then translated it to TEX and shipped a paper copy of the word processor document together with the TEX file, were generally happy. Publishers would have preferred a higher quality TEX file but resigned themselves to a one-time cleanup. The advantage of this approach is clear. The publisher doesn't have to rekey the text or cope with an alien word processor format, and the author doesn't have to spend much time proofreading since the text, mathematical symbols, and special fonts remain untouched by human hands.

We would have preferred it if users could have shipped a disk with their word processor file to the publisher and have had the publisher enter the corrections arising out of copy-editing into the word processor file before conversion to TEX. The word processor file could have been handed back to the author, preserving the changes for future revised editions. Unfortunately, publishers are reluctant to learn yet another word processing system.

Users unfamiliar with TEX expected that the converter and TEX could be used like a printer driver. They were very disappointed because they had hoped they could completely avoid learning TEX. However, some knowledge of TEX is required to produce a professional looking document with our converter. Some users abandoned TEX as a result; most others learned enough Pidgin TEX to succeed.

Other users were reluctant to fix conversion errors in the original word processor file, changing them in the TEX file instead. As a reason, several cited the amount of time required to enter the word processor, making the change there and running the document through the converter before executing the TEX program and the previewer. Some of those users finally abandoned our word processor and became TEX experts.

Most users wrote with the word processor as long as possible. Upon completion of the document, they performed a trial conversion and then corrected converter errors and added tags as required by the submission style of the publisher. These changes were made in the word processor file. Additional markup was performed by the publisher in the TEX file.

Conclusion

Many potential TEX users are justifiably concerned about the drudgery of entering TEX codes in an ASCII file. Our conversion utility, which translates files written in a scientific word processor to TEX, offers a number of advantages. The learning curve for the word processor is not as steep as for raw TEX. Fonts, special symbols, and mathematical structures show up correctly on the screen. This eases editing and revising. Typical TEX keyboarding errors, such as omitted backslashes or mismatched $ signs, are reduced. Documents can be translated into different dialects of TEX. A special font is translated directly to TEX code to access any features not provided by the word processor or converter.

There are several disadvantages. The conversion pass takes time. The user must cope with converter errors and limitations in addition to TEX problems. Sometimes the converter's actions are difficult to predict. The converter cannot detect math mode with perfect accuracy, and the user must occasionally work around the converter's guesses. The code generated by the converter contains a few nonstandard macros which may need to be modified by publishers.

Most users of this system felt that we are on the right track. They need TEX output, either for high quality printing or for submitting documents. They find that the problems of the conversion pass are far outweighed by the convenience of not having to manually enter the TEX codes, and the ease of making revisions in the WYSIWYG screen display.

Bibliography

Adobe Systems Inc., *PostScript Language Reference Manual.* Addison-Wesley, 1985.

Knuth, Donald E. *The TEXbook*. Reading, Mass.: Addison-Wesley, 1986.

Milne, J. S. "Four Word Processors with TEX Capability", *Notices Amer. Math. Soc.* 37, pages 1018–1022, 1990.

On the Logical Structure of Mathematical Notation

Dennis S. Arnon
Xerox PARC
3333 Coyote Hill Road
Palo Alto, CA 94304 USA
415-494-4425; FAX: 415-494-4241
arnon@parc.xerox.com

Sandra A. Mamrak
Department of Computer and Information Science
The Ohio State University
Columbus, OH 43210 USA
614-292-2770; FAX: 614-292-9021
mamrak@cis.ohio-state.edu

Abstract

We show how the logical structures of a realistic class of mathematical formulae can be recovered from Plain TEX source representations, using the Centaur system, a tool for Language-Based Environments.

Introduction

A major current trend in structured document representation and processing is to distinguish the *logical* and *layout* structures of (the instances of) a given family of documents. Both ODA (Office Document Architecture) and SGML (Standard Generalized Markup Language) [3] offer tools, much akin to context-free grammars, for specifying either or both of these structures for a document class ("document type") of interest.

In general, we may say that document logical structure expresses the author's (and hopefully the reader's) organization of the material being presented, independently of how the words, formulae, and illustrations of the work are actually to be turned into marks on paper or screen. Document layout structure expresses how primitive "glyphs" (font characters, illustrations, images) are positioned and juxtaposed on display surfaces, and how a hierarchy of groupings of them (e.g., "paragraph blocks," "pages") can be identified. Both structures are usually thought of as trees, possibly with cross-links between nodes.

These general remarks specialize well to mathematical formulae, i.e., to mathematical notation. The author and reader of a technical document think about a formula in terms of its logical structure. Communication between them is achieved via a representation of the formula as a layout structure; this of course must be imaged (printed, displayed) for it to actually play its communicative role.

The logical structure of formulae is also the basis for computational applications, such as symbolic mathematical computation, that operate on their meanings, i.e., that manipulate (effective representations of) the objects *denoted* by the formulae. For example, a program to symbolically invert a matrix of polynomials would typically require a logical structure representation of the matrix, and not a layout representation. Beyond computation, the majority of information-retrieval applications one might imagine for a database of mathematical formulae (such as an online table of integrals) would use logical structure.

The high-quality mathematical typesetting that has been brought about by systems such as TEX has whetted the appetites of computational mathematicians for WYSIWYG symbolic computation, also sometimes called "direct manipulation," that provides the ability to interact directly with the pleasant-to-look-at (imaged) layout structures of formulae as they appear on the screen. The catch is that the manipulations that are desired require logical structures. And while it is now straightforward to generate layout from logical structure, going in the reverse direction is generally hard.

Building on these observations, signficant effort has been devoted recently to building WYSIWYG symbolic math systems in which logical structures of formulae are always held as the primary representation: Layout structures are generated when needed, and links back to the logical structure are then maintained to enable desired subunits of logical structure to be inferred from (visible) selections of subunits of layout structure. (See, for example, [4] and [1].)

Nonetheless, there are numerous situations in which one starts without a logical structure representation of a formula of interest, and would like to obtain one. In this paper we shall suppose that we begin with a Plain TEX representation of a formula from a simple class of combinations of elementary functions and integrals. We then show that by using contemporary tools of Language-Based Environments we can do a reasonable job of recovering logical structure from TEX.

In the next section, we briefly discuss the Centaur system for Based Environments, which we have used. Then we specify the concrete syntax of TEX that we parse, and the abstract syntax (logical structure) we translate it into. We mention the restrictions we are forced to impose on the TEX syntax we can accept. Finally we show some examples. It should be clear that the logical structures we obtain are suitable, with minor transliteration, for input to such symbolic computation systems as Mathematica or Maple. We hope it will also be clear that we could also "unparse" our logical structures into SGML, EQN, or virtually any other concrete syntax for the logical structure of mathematical notation.

Centaur

Centaur [2] is a meta-tool for the generation of language-based environments. From a grammatical specification of a (context-free) language and executable specifications of its formal semantics, parsers, type checkers, and interpreters for it can be automatically generated. Centaur is written in Lisp and Prolog and usually runs under X-Windows.

In the next section we shall see a grammar for our class of formulae. Nonterminals in the concrete syntax rules are enclosed in angle brackets. Literal strings to match in the input stream are enclosed in double quotes. Underneath each rule is a specification of the portion of abstract syntax tree that gets built when that rule is recognized. The last part of the grammar defines the "signatures" of the abstract syntax tree, i.e., what arities they have and what "phyla" ("sorts," "types") their children must have. Finally the phyla themselves are given; each is simply a set of abstract syntax operators.

Grammar for a Class of Formulae

The appendix shows the grammar for each of the example expressions. Here are some properties to note:

1. Variables are restricted to single letters; integer constants are restricted to single digits.

2. Non-character integrands must be single chars or TEX subformulae, i.e., enclosed in braces. Also, the args of special functions (currently sin, cos, log, exp, prime) must be characters or subformulae. These requirements simplify the grammar and parsing.

3. Multiplication is denoted by asterisk. This avoids three shift/reduce conflicts from yacc.

4. All integrals are represented by instances of a single abstract syntax operator. Formatting routines need to handle this appropriately (e.g., not print the "d" for a null (defaulted) variable of integration).

5. We assume that prime of an expression means derivative with respect to its main variable, and that there is some clear way to know what the main variable is (e.g., the expression has only one variable). It is the user's job to enclose the argument of prime in parentheses to prevent ambiguity. Similarly the args of sin and cos must be in parentheses.

6. Exponential function must be done as exp, not e to the x.

Examples

The following are examples of expressions accepted by our concrete grammar.

$$\int_{t+u*5}^{x} \frac{e^{-x^2}+e^{x^3}}{s*x^2+c^2*x}dx$$

$$\int \frac{e^{-x^2}+e^{x^3}}{s*x^2+c^2*x}dx$$

$$\int \frac{x}{(x-1)*(x+2)}dx$$

$$\int_{1}^{z} f*x*g*hdx$$

$$\int_{1}^{a}\int_{1}^{b}\int_{1}^{c} e^{x+y+z}dxdydz$$

$$\int_1 (a+b+c)*(e^{-x^2}+e^{x^3})dx$$

$$\int \frac{\exp -x^2+\exp x^3}{\sin x^2+\cos x^2}dx$$

$$\int \sin\log x dx$$

$$(\int \sin\log x dx)'$$

$$(\frac{-(x*\cos(\log(x)))}{2}+\frac{x*\sin(\log(x))}{2})'$$

$$(\frac{2*x^5}{5}-\sqrt{\frac{2*x^5*\log x}{5}}+\frac{x^5*\log x^2}{5})'$$

$$\int_{t+u*5}^{x}\frac{\int_{t+u*5}^{x}\frac{e^{-x^2}+e^{x^3}}{s*x^2+c^2*x}dx}{s*x^2+c^2*x}dx$$

Figure 1 shows the abstract syntax tree for the last expression.

```
integral(
  quotient(
    integral(
      quotient(
        sum(power(e, power(negate(x), 2)), power(e, power(x, 3))),
        sum(times(s, power(x, 2)), times(power(c, 2), x))), x,
      sum(t, times(u, 5)), x),
    sum(times(s, power(x, 2)), times(power(c, 2), x))), x,
  sum(t, times(u, 5)), x)
```

Figure 1: Abstract syntax tree

References

[1] D. Arnon, R. Beach, K. McIsaac, and C. Waldspurger. Caminoreal: an interactive mathematical notebook. In *Proc. Intl. Conf. on Electronic Publishing, Document Manipulation, and Typography*, pages 1–18. Cambridge University Press, April 20-22 1988. (J.C. van Vliet, ed.) ISBN 0-521-36294-6.

[2] P. Borras et. al. Centaur: the system. In *Proceedings of the SIGSOFT'88, Third Annual Symposium on Software Development Environments*, 1988. Boston, Massachusetts.

[3] *Information Processing—Text and Office Systems—Standard Generalized Markup Language (SGML)*, October 1986. ISO 8879-1986 (E).

[4] Neil Soiffer. The design of a user interface for computer algebra systems. Technical Report UCB/CSD 91/626, Computer Science Division (EECS), University of California, Berkeley, April 1991.

References

[1] D. Arnon, R. Beach, K. McIsaac, and C. Waldspurger. Caminoreal: An interactive mathematical notebook. In *Proceedings of the International Conference on Electronic Publishing, Document Manipulation, and Typography,* pages 1–18. Cambridge University Press, April 20–22, 1988. (J.C. van Vliet, ed.) ISBN 0-521-36294-6.

[2] P. Borras et. al. Centaur: The system. In *Proceedings of the* SIGSOFT'88, *Third Annual Symposium on Software Development Environments,* 1988. Boston, Massachusetts.

[3] *Information Processing — Text and Office Systems — Standard Generalized Markup Language (SGML),* October 1986. ISO 8879-1986 (E).

[4] Neil Soiffer. The design of a user interface for computer algebra systems. Technical Report UCB/CSD 91/626, Computer Science Division (EECS), University of California, Berkeley, April 1991.

Appendix

```
definition of texMath is
rules
<markedTexExpr> ::= "$$" <texExpr> "$$" ;
<texExpr>

<markedTexExpr> ::= "\(" <texExpr> "\)" ;
<texExpr>

<markedTexExpr> ::= "$" <texExpr> "$" ;
<texExpr>

<texExpr> ::= "\int" <null_limit> <null_limit> <integrand> <null_name> ;
integral(<integrand>, <null_name>, <null_limit>.1, <null_limit>.2)

<texExpr> ::= "\int" <null_limit> <null_limit><integrand> "d" <name> ;
integral(<integrand>, <name>, <null_limit>.1, <null_limit>.2)

<texExpr> ::= "\int" "_" <lowerLimit><null_limit><integrand> <null_name> ;
integral(<integrand>, <null_name>, <lowerLimit>, <null_limit>)

<texExpr> ::= "\int" "_" <lowerLimit> <null_limit> <integrand> "d" <name> ;
integral(<integrand>, <name>, <lowerLimit>, <null_limit>)

<texExpr> ::= "\int" <null_limit> "^" <upperLimit><integrand> <null_name>;
integral(<integrand>, <null_name>, <null_limit>, <upperLimit>)

<texExpr> ::= "\int" <null_limit> "^" <upperLimit><integrand> "d" <name>;
integral(<integrand>, <name>, <null_limit>, <upperLimit>)

<texExpr> ::= "\int" "_" <lowerLimit> "^" <upperLimit><integrand>  <null_name>;
integral(<integrand>, <null_name>, <lowerLimit>, <upperLimit>)

<texExpr> ::= "\int" "_" <lowerLimit> "^" <upperLimit><integrand> "d" <name> ;
integral(<integrand>, <name>, <lowerLimit>, <upperLimit>)

<integrand> ::= <bracedTexExprOrChar> ;
<bracedTexExprOrChar>

<lowerLimit> ::= <bracedTexExprOrChar> ;
<bracedTexExprOrChar>

<upperLimit> ::= <bracedTexExprOrChar> ;
<bracedTexExprOrChar>

<texExpr> ::= "\frac" <numerator><denominator> ;
quotient(<numerator>, <denominator>)

<numerator> ::= <bracedTexExprOrChar> ;
<bracedTexExprOrChar>

<denominator> ::= <bracedTexExprOrChar> ;
<bracedTexExprOrChar>
```

```
<texExpr> ::= <texFactor> ;
<texFactor>

<texExpr> ::= <texExpr> "+" <texFactor> ;
sum( <texExpr>, <texFactor>)

<texExpr> ::= <texExpr> "-" <texFactor> ;
difference( <texExpr>, <texFactor> )

<texFactor> ::= <texPower> ;
<texPower>

<texFactor> ::= <texFactor> "*" <texPower> ;
times(<texFactor>, <texPower>)

<texFactor> ::= <texFactor> "\over" <texPower> ;
quotient(<texFactor>, <texPower>)

<texPower> ::= <texUnary> ;
<texUnary>

<texPower> ::= <texPower> "^" <texUnary> ;
power(<texPower>, <texUnary>)

<texUnary> ::= <texTerm> ;
<texTerm>

<texUnary> ::= "-" <texTerm> ;
negate(<texTerm>)

<texTerm> ::= <bracedTexExprOrChar> ;
<bracedTexExprOrChar>

<texTerm> ::= "(" <texExpr> ")" ;
<texExpr>

<texTerm> ::= "\sin" <bracedTexExprOrChar>   ;
sin(<bracedTexExprOrChar>)

<texTerm> ::= "\cos" <bracedTexExprOrChar>   ;
cos(<bracedTexExprOrChar>)

<texTerm> ::= "\log" <bracedTexExprOrChar>   ;
log(<bracedTexExprOrChar>)

<texTerm> ::= "\exp" <bracedTexExprOrChar>   ;
exp(<bracedTexExprOrChar>)

<texTerm> ::= "\sqrt" <bracedTexExprOrChar>   ;
sqrt(<bracedTexExprOrChar>)

<texTerm> ::= <bracedTexExprOrChar> "\prime"   ;
derivative(<bracedTexExprOrChar>)
```

```
<texTerm> ::= <bracedTexExprOrChar> "'"  ;
derivative(<bracedTexExprOrChar>)

<bracedTexExprOrChar> ::= <char> ;
<char>

<bracedTexExprOrChar> ::= <bracedTexExpr> ;
<bracedTexExpr>

<bracedTexExpr> ::= "{" <texExpr> "}" ;
<texExpr>

<char> ::= <name> ;
<name>

<char> ::= <digit> ;
<digit>

<name> ::= %LETTER ;
name-atom (%LETTER)

<digit> ::= %DIGIT ;
digit-atom (%DIGIT)

<null_name> ::=  ;
null_inst ()

<null_limit> ::=  ;
null_inst ()


abstract syntax
integral -> EXP NAME EXP EXP;
quotient -> EXP EXP ;
power -> EXP EXP ;
sum -> EXP EXP ;
difference -> EXP EXP ;
times -> EXP EXP ;
negate -> EXP ;
sin -> EXP ;
cos -> EXP ;
log -> EXP ;
exp -> EXP ;
sqrt -> EXP ;
derivative -> EXP ;
name -> implemented as IDENTIFIER ;
digit -> implemented as STRING ;
null_inst -> implemented as SINGLETON ;

EXP ::= integral quotient power sum difference times negate sin cos log
exp sqrt derivative NAME digit null_inst ;
NAME ::= name ;

end definition
```

Math into BLUes

Part I: Mourning

Kees van der Laan
Hunzeweg 57, 9893PB, Garnwerd, The Netherlands
+31 5941 1525
Internet: cgl@rug.nl

Abstract

TEXing mathscripts is not simply typing. Math has to be translated into TEX commands. First the motivation for this work is given. Next traditional math page makeup is summarized along with the macroscopic math TEX commands. After answering "Why is TEXing mathscripts difficult?", an anthology of TEXfalls and their antidotes is discussed. In part II, suggestions are given in order to lessen the difficulties.

Prelude

My assistance was called for in TEXing a mathscript. Part of the mathscript was typed, and contained TEX commands; but it did not compile. Inspection revealed it never could have. It occurred to me that at least three typists had been involved, mixing the use of LATEX, AMS-TEX and macros from other sources. Furthermore, the TEXscript showed various 'TEXfalls' (from "pitfalls"). I define these as "correct encoding which yields neither the required nor customary layout." Also 'pseudo-guru' involvement could be felt, which I would define as a too-complicated use of TEX, inhibiting the intelligibility of the TEXscript. There is an appropriate quote to be found on page 373 of *The TEXbook* in the "Dirty Tricks" chapter.

> Always remember, however, that there's usually a simpler and better way to do something than the first way that pops into your head.

Not only was I looking over the shoulder of a typist, I was also inspecting a math book TEXed by a mathematician (Temme, 1990). The book looks good and examples from it are used here in order to show other ways of TEXing.

In Part I, attention is paid to:

- traditional math page make-up;
- some advanced math examples;
- what makes TEXing mathscripts difficult; and
- an anthology of TEXfalls with antidotes.

Part II of this series will deal with:

- (cross)referencing;
- hyphenation of long formulae; and
- what ought to be done to lessen the difficulties.

Part II will be published in the Proceedings of EuroTEX91, Paris, Cahiers GUTenberg, #10 – 11, 147 – 170.

With respect to the future TEXing of math I don't consider Mittelbach's (1990:11) criticisms too severe. First, the spacing around atoms can be adapted via casts. Second, the lack of hyphenation for subformulae is similar to verbatim; in general, hyphenation is avoided in boxes. Math hyphenation has been conscientiously avoided in displays as well. For in-line math, it is true that *sub*formulae are not hyphenated automatically. It is not that relevant however, because in-line math should be short, and should not be complicated (read 'nested').

For you and me. Most, if not all, math TEXfalls[1] have been envisioned by the Grand Wizard himself and references to those or related issues are indicated by '*TB*' (*The TEXbook*) followed by page or exercise number. Other terms used in this paper include the following. 'Mathscript' denotes a mathematics manuscript. 'TEXscript' denotes a TEX formatted compuscript, especially the one for which my assistance was asked. 'TEXnigma' is a computer system with TEX installed. 'TEXnowledge' means knowledge of TEX. A 'TEXist' is a TEX typist. 'DEK' is Donald E. Knuth. 'BLUe' is DEK's

[1] The TEXfalls discussed herein are not specific to plain TEX, AMS-TEX, or LATEX. They illustrate basic pitfalls in encoding math. Sources include the inspected TEXscript, the Temme book and some pitfalls I stumbled upon myself. LATEX is rather superficial with respect to math. Formula classes are not even mentioned, which is dead wrong, but understandable from the viewpoint of descriptive markup.

unwary B.L. User (Ben Lee User of *The TeXbook* fame). 'TeXfalls' has already been described.

Math Page Makeup

Swanson (1986) is a good source for information on traditional math markup. In publications, math is either part of the running text or is displayed. In displays, indentation on all sides is active, and formulae are sometimes aligned, for example, at the = symbol.

TeX requires math within text to be surrounded by $ signs:[2] `$<math>$`. Displayed math is tagged by `$$` signs:[3] `$$<displayed math>$$`. For the general multi-line display, plain TeX provides the macro `\displaylines` (*TB* 194, 362), and for aligned formulae the macro `\eqalign` (*TB* 190, 362). Displays are centered by default and that is all there is to TeXing math, from an outer level point of view.

- The following example of a Pascal triangle:

$$\begin{gathered}1\\ 1\quad 1\\ 1\quad 2\quad 1\\ 1\quad 3\quad 3\quad 1\\ \cdot\quad\cdot\quad\cdot\quad\cdot\quad\cdot\end{gathered}$$

is obtained via:

```
$$\displaylines{1\cr
  1\quad1\cr
  1\quad2\quad1\cr
  1\quad3\quad3\quad1\cr
  \hbox to 7em{$\cdot$\hss
    $\cdot$\hss$\cdot$\hss
        $\cdot$\hss$\cdot$}}$$
```

The example demonstrates two levels of formatting math: (1) the inner level, where the triangle has to be defined unambiguously (detailed TeX commands), and (2) the outer level, where the triangle is positioned within the text (`$$` signs meaning display) and subject to the style of the publication series.[4]

On the phone one would say: "Pascal's triangle; you know; a 1 with two 1's below it, and a 1,2,1 below that, and a 1,3,3,1 below that, etc., all centered." However, for formatting (read 'encoding'), more precise information is needed than when describing math by phone, in order to eliminate ambiguity. A computer-based formatting system is not yet that intelligent.

Right- or left-aligned formula numbers can be provided by the tags `\eqno` and `\leqno` (*TB* 187, 362). Individual lines in a multi-line display can be numbered; therefore, the macros `\eqalignno` and `\leqalignno` are provided (*TB* 192, 362).

- In summary, all plain TeX's math page makeup macros (with essential ways of numbering formulae) are demonstrated in the following templates:

$$\sin 2x = 2\sin x\, \cos x \tag{TB 186}$$

$$\displaylines{F(z)=a_0+{a_1\over z}+{a_2\over z^2}+\cdots+{a_{n-1}\over z^{n-1}}+R_n(z),\cr \hfill n=0,1,2,\dots\,,\cr \hfill F(z)\sim\sum_{n=0}^\infty a_nz^{-n},\quad z\to\infty\qquad\qquad\hfill \text{(TB ex19.16)}}$$

$$\begin{aligned}\cos 2x&=2\cos^2 x-1\\ &=\cos^2 x-\sin^2 x\end{aligned}\tag{TB 193}$$

$$\begin{aligned}\cosh 2x&=2\cosh^2 x-1 &&\text{(TB 192)}\\ &=\cosh^2 x+\sinh^2 x\end{aligned}$$

which are obtained via:

```
$$\sin2x=2\sin x\, \cos x
     \eqno({\rm TB\ 186})$$
$$\displaylines{F(z)=
a_0+{a_1\over z}+{a_2\over z^2}+\cdots
   +{a_{n-1}\over z^{n-1}}+R_n(z),\cr
           \hfill n=0,1,2,\dots\,,\cr
\hfill F(z)\sim\sum_{n=0}^\infty a_nz^{-n},
       \quad z\to\infty\qquad\qquad\hfill
       \llap{(TB\ ex19.16)}\cr}$$
$$\eqalign{\cos2x&=2\cos^2x-1\cr
                &=\cos^2x-\sin^2x\cr}
  \eqno({\rm TB\ 193})$$
$$\eqalignno{
\cosh2x&=2\cosh^2x-1&({\rm TB\ 192})\cr
       &=\cosh^2x+\sinh^2x\cr}$$
```

It was difficult to get the above example `\eqalign`, labeled TB 193, to work correctly in two-column format. It would left-justify rather than center the formula because of insufficient space left by the wide label. Deactivating the glue '`\,`' before the `\vcenter` in the body of `\eqalign` forced TeX to center the formula (see *TB* 189).

One can also use the general `\halign` macro.

- From example 22.9 of *The TeXbook*, we have:

[2] Expensive!

[3] Even more expensive!

[4] What should be displayed is left to the discretion of the author but it should serve clarity of exposition. Swanson (1986) advises displaying any math which is longer than half a line.

$$\begin{aligned} 10w + {}& 3x + 3y + 18z = 1, && (\ 9)\\ 6w - {}& 17x \qquad - 5z = 2, && (10) \end{aligned}$$

obtained via:

```
$$\openup1\jot\tabskip=0pt plus1fil
\halign to\displaywidth{\tabskip=0pt
$\hfil#$&$\hfil{}#{}$&
$\hfil#$&$\hfil{}#{}$&
$\hfil#$&$\hfil{}#{}$&
$\hfil#$&${}#\hfil$\tabskip=0pt plus1fil&
\llap{#}\tabskip=0pt\cr
10w&+& 3x&+&3y&+&18z&=1,&( 9)\cr
 6w&-&17x& &  &-& 5z&=2,&(10)\cr}$$
```

I consider `\cases`, `\(p)matrix`, `\overbrace`, and `\underbrace` to be *parts* of formulae (*TB* 177):

- `\(p)matrix` as formula part

$$A = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{pmatrix}$$

obtained via:

```
$$A=\pmatrix{
    a_{11}&a_{12}&\ldots&a_{1n}\cr
    a_{21}&a_{22}&\ldots&a_{2n}\cr
    \vdots&\vdots&\ddots&\vdots\cr
    a_{n1}&a_{n2}&\ldots&a_{nn}\cr}$$
```

Am I BLUe?

Before diving into the difficulties which arise when encoding math, the following are some examples which show what TEX can do for you.[5]

- Selections from chapters 16–18 in *The TEXbook*:

$$\mathbf{S}^{-1}\mathbf{TS} = \mathbf{dg}(\lambda_1, \ldots, \lambda_n) = \mathbf{\Lambda}$$

$$\sum_{k=1}^{\infty} \frac{1}{2^k} = 1, \quad \sqrt{1 + \sqrt{1 + \sqrt{1 + x}}}$$

$$\frac{f(x + \Delta x) - f(x)}{\Delta x} \to f'(x) \quad \text{as} \quad \Delta x \to 0$$

$$\underbrace{\{\overbrace{a, \ldots, a}^{k\ a\text{'s}}, \overbrace{b, \ldots, b}^{l\ b\text{'s}}\}}_{k+l \text{ elements}}, \quad 2 \uparrow\uparrow k \overset{\text{def}}{=} 2^{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}\Big\}{\scriptstyle k}$$

- The Cardano solution of $x^3+px = q$, with $p, q \geq 0$ reads:

$$\sqrt[3]{\sqrt{p^3/27 - q^2/4} + q/2} - \sqrt[3]{\sqrt{p^3/27 + q^2/4} - q/2}$$

- Derivatives:[6]
  $\dot{y}\,\ddot{y}\,\dddot{y} \quad y'\,y''\,y''' \quad \partial_x y\, \partial_x^2 y, \partial_x^3 y$
- Bessel equation:

$$z^2 w'' + zw' + (z^2 - \nu^2)w = 0$$

  with solutions:
  $J_{\pm\nu}(z)$, Bessel function of the first kind,
  $Y_{\pm\nu}(z)$, Bessel function of the second kind (Weber),
  $H_\nu^{(1)}(z)$, and $H_\nu^{(2)}(z)$, Bessel function of the third kind (Hankel).
- Primed summation symbols are used in Chebyshev expansions:

$$\sum_{k=0}^{n}{}' a_k T_k(x) \overset{\text{def}}{=} .5\,a_0 + a_1 x + a_2 T_2(x) + \cdots \\ + a_n T_n(x).$$

- Hypergeometric function:

$$M_n(z) = {}_{n+1}F_n\left(\begin{matrix} k + a_0, k + a_1, \ldots, k + a_n \\ k + c_1, \ldots, k + c_n \end{matrix}; z\right)$$

- From Swanson (1986:40):

$$\begin{aligned} &\int_U \delta(I)\mu(I) \leq \\ &\sum_{\mathcal{D}}\sum_{\mathcal{D}_{I'}}\Bigl[\int_J \alpha(J')\mu(J') - \alpha(J)\mu(J) \\ &\qquad\qquad - \int_J [\{s(\alpha\eta)(J')\}/\eta(J')]\mu(J')\Bigr] \\ &+ \Bigl[\sum_{\mathcal{D}}\sum_{\mathcal{D}_{I'}} |\alpha(J) - [\{s(\alpha\eta)(J)\}/\eta(J)]|\mu(J)\Bigr] \\ &\quad \times \Bigl[\sum_{\mathcal{D}}\sum_{\mathcal{D}_{I'}} |\alpha(J) - [\{s(\alpha\eta)(J)\}/\eta(J)]|\eta(J)\Bigr] \end{aligned}$$

- Magic squares (Dürer's 4-by-4 with dotted lines):

2	7	6
9	5	1
4	3	8

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

- Calculation flow of autocorrelation function a_f (inspired by *TB* 358, ex18.46). $\mathcal{F}$ denotes Fourier transform and $\mathcal{F}^{-}$ the inverse Fourier

[5] NTG (Dutch Users Group) uses this section, along with other interesting examples, in their info/demo package for potential members.

[6] Kerning an extra point in superscripts was pointed out by Daniel Olson.

transform:

$$\begin{tikzcd} f \arrow[r, "\otimes"] \arrow[d, "\mathcal{F}"] & a_f \\ \mathcal{F}(f) \arrow[r, "\times"] & \left(\mathcal{F}(f)\right)^2 \arrow[u, "\mathcal{F}^-"'] \end{tikzcd}$$

- Some matrix icons (Wilkinson, 1965):

□ ◺ = ◺◹ or $AL = LH$

▯ = ▯◹ or $A = QR$

- Rhombus scheme (Schwarz, et al., 1972:166):

$$\begin{tikzcd} 1^{st}\mathrm{RS} & e_k^{(s)} \arrow[dl, dash] \arrow[dr, dash] & \\ q_k^{(s+1)} \arrow[dr, dash] & & q_{k+1}^{(s)} \arrow[dl, dash] \\ & e_k^{(s+1)} = \frac{q_{k+1}^{(s)}}{q_k^{(s+1)}} e_k^{(s)} & \end{tikzcd}$$

$$\begin{tikzcd} 2^{nd}\mathrm{RS} & q_k^{(s)} \arrow[dl, dash] \arrow[dr, dash] & \\ e_{k+1}^{(s+1)} \arrow[dr, dash] & & e_k^{(s)} \arrow[dl, dash] \\ & q_k^{(s+1)} = q_k^{(s)} + (e_k^{(s)} - e_{k+1}^{(s+1)}) & \end{tikzcd}$$

- Continued fractions:

$$1 + \mathop{\Phi}_{k=1}^{n} \frac{a_k}{b_k} \overset{\text{def}}{=} 1 + \cfrac{a_1}{b_1 + \cfrac{a_2}{b_2 + \ddots + \cfrac{a_{n-1}}{b_{n-1} + \cfrac{a_n}{b_n}}}}$$

with (space saving) variant notations:

$$= 1 + \frac{a_1 \rfloor}{\lfloor b_1} + \frac{a_2 \rfloor}{\lfloor b_2} + \cdots + \frac{a_n \rfloor}{\lfloor b_n}$$

$$= 1 + \frac{a_1}{b_1 +}\,\frac{a_2}{b_2 +}\cdots\frac{a_n}{b_n}$$

- Reduction to Hessenberg form via lower triangular similarity transformation (Wilkinson, 1965:357):

$$\overset{\text{A}}{\begin{pmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{pmatrix}} \overset{\text{N}}{\begin{pmatrix} 1 & & \\ 0 & 1 & \\ 0 & \times & 1 \end{pmatrix}}$$

$$= \overset{\text{N}}{\begin{pmatrix} 1 & & \\ 0 & 1 & \\ 0 & \times & 1 \end{pmatrix}} \overset{\text{H}}{\begin{pmatrix} \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \end{pmatrix}}$$

- Partitioning (Wilkinson, 1965:291):

$$P_r = \left(\begin{array}{c|c} I_{n-r} & 0 \\ \hline 0 & I - 2v_r v_r^T \end{array}\right)$$

- Braces and matrices (Wilkinson, 1965:199):

$$\begin{matrix} & \overbrace{\qquad\qquad\quad}^{p} \quad \overbrace{\qquad\quad}^{n-p} \\ \begin{matrix} p\left\{\begin{matrix} \\ \\ \\ \\ \end{matrix}\right. \\ n-p\left\{\begin{matrix} \\ \\ \\ \end{matrix}\right. \end{matrix} & \begin{pmatrix} \times & \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times & \times \end{pmatrix} \end{matrix}$$

- Matrices, braces, (dotted) partitioning and icons (space efficient variant):

$$\begin{matrix} & \overbrace{\qquad\quad}^{p} \; \overbrace{\qquad}^{n-p} \\ \begin{matrix} p\left\{\begin{matrix} \\ \\ \\ \\ \end{matrix}\right. \\ np\left\{\begin{matrix} \\ \\ \\ \end{matrix}\right. \end{matrix} & \left(\begin{array}{cccc:ccc} \times & \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times & \times \\ \hdashline 0 & 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times & \times \end{array}\right) \\ & \underbrace{\qquad\quad}_{◹} \; \underbrace{\qquad}_{▯} \end{matrix}$$

The above examples resemble 'macho' behavior (showing off with TEX). I agree with that, but in practical situations I would like to use constructs which are as simple as possible.

What's Wrong, Doc?

Mathscripts differ from TEXscripts.

- The output[7]:

$$x = 1 + \left(\frac{y^2}{k+1}\right)^{1/3}.$$

looks different from:

```
$$x=1+\left({y^2\over k+1}
     \right)^{\!\!1/3}.$$
```

Because of this disparity, the problem is how to get a correct TEXscript, starting from just a mathscript. This is difficult because of the complexity of math typesetting, and the inherent complexity of TEX, if not because of the bewildering and confusing variety of TEX-based products.[8]

[7] Note that the kind of parentheses and the kind of division notation have to be specified as well.

[8] In this paper we restrict ourselves to plain TEX, and assume that no fancy, friendly, WYSIWYG user interface is available.

First, one has to find the appropriate format command from nearly a thousand.[9] In *The TeXbook* the following chapters are devoted to math formatting: 16 (11pp), 17 (21pp), 18 (23pp), 19 (14pp), 22 (242, ex22.9/11), 24 (up to 281, 15pp), 26 (5pp); Appendices A (33pp), B (6pp), F (13pp), G (7pp). Add to these the required general TeXnowledge of how to use TeX for non-complex documents and general page makeup, how to format tabular material (matrices, commutative diagrams), how to handle output routines, and how to use non-default fonts, and no-one would consider TeX to be trivial.[10]

Second, content and context-dependent extras have to be added, as demonstrated throughout this paper.

Third, once the TeX language is mastered, the difficulty of locating and correcting errors — misconceptions as well as typos — remains.[11] So add chapter 27 of *The TeXbook* to the above, just for completeness.

Once you have coped with everything that is mentioned above, you are still faced with true (La)TeX driver bugs and LaTeX's inconsistency. I was trapped by LaTeX's quote environment when I tried to get the opening quote to hang out. It did not work, not even after inserting `\null`.

Spivak (1986) has dealt with TeXing math in his delightful book, but alas, it is not a proper extension. My own perspective is to look for what is needed and to extend plain TeX in a compatible way, keeping overhead as low as possible. Plain TeX already provides enough TeXfalls.

The Bad News

The material in this section started as a list of pitfalls and grew into a general discussion with antidotes. (If readability for BLUe is reduced below par, I pitfailed.)

I would like to start by mentioning the nasty small white space on a new line after a heading. This creature can be killed by providing a comment symbol `%` immediately after the heading command (just a warm-up for the unwary[12]).

Too many. The 'too many' pitfall is a serious problem. It occurs when using many incompatible products which are partly, or not at all, understood.

In the typing project for which my assistance was asked, TeXed chapters showed different approaches: AMS-TeX was used in one, LaTeX in another, etc. This demonstrated the involvement of several typists and the lack of a common approach. The document also did not compile, showing that TeXing is one thing; getting it correct — if only just those braces — is quite another. This is especially true for typists not familiar with programming. Apart from the above, encoding was done inconsistently: AMS-TeX was used for some math symbols not available in plain, such as $\gtrless$. Commands like `\frac`, and `\overset` were used along with their plain functional equivalents. Obviously one typist was AMS-TeX oriented, while the others were not.

In short, the TeXscript was far from correct, suffered from too many tools, and otherwise was full of horrible TeXfalls. The Temme book didn't suffer from these TeXfalls, as it used plain TeX and only a few extra symbols.

I incurred the following problem when preparing this paper. This paper uses `ltugproc.sty`, and therefore LaTeX. In LaTeX `\eqalign`, etc. are not available, so I defined them. But, I did not think of redefining `\centering`, which has a different meaning within LaTeX than within plain TeX and as a consequence, `\eqalignno` did not produce the desired result. Another TeXfall was that `\eqalign` did not center in two-column format when `\eqno` was used as well! I had to first deactive the glue item '`\,`' of `\eqalign`. (For an explanation see *TB* 189.)

However, for all those mathematicians who practise self-publishing, it pays to encode as simply as possible.[13] Understanding the basics and adding a few macros will do, especially for those who otherwise have to rely on Word*whatever*. This is demonstrated by the Temme book, and as far as I understand it, it is also the attitude of the Grand Wizard himself.

[9] Cheswick (1990) has provided a KeyWord In Context with all the TeX and LaTeX commands. This is handy when in doubt as to whether a command is already in use.

[10] Beeton (1990) states that it was the intent of the AMS-TeX project to "simplify input of complex mathematical expressions."

[11] The TeXist's task has been silently increased by the parsing and correcting of the TeXscript in order to provide the author with proofs.

[12] This is overlooked in the Dutch course book on LaTeX, and also in the Dutch `brief` style, where the addressee label on the subsequent page headings is preceded by white space.

[13] This means that the tools should be powerful and mixing similar tools should be kept to a minimum.

On the other hand, I welcome the approaches taken by *TUGboat* (see Whitney and Beeton, 1989), and the AMS, (see the *AMS-TEX User's Guide*, 1990) where, on top of a common style, similar LATEX and TEX procedural markup 'user interface' commands have been built. This provides freedom for authors to submit either LATEX or plain TEX manuscripts, while the publisher can easily integrate these sources into one publication. This indicates that some TEXfalls can be taken care of by the publishing house.

Separation of concerns. *The* pitfall for author publishing, or self-publishing systems is insufficient awareness of the requirements in other fields. Not only does the author have to worry about the content, the organization, the power of the examples, the use and spelling of the language, and consistency, etc., but he also must worry about conventions of math typesetting, the computer system, and typing skills, not to mention proofreading.

First gains. The typographic markup pitfall reflects the temptation to specifically format how elements should *look*, instead of tagging the elements with the purpose of *identifying* them — a matter of abstraction and separation of concerns. One can think of the various headings: chapter, definition, theorem, and the like, where the formatting can be postponed and provided separately in style or format files. This pitfall can also be classified as the portability pitfall: submitting an article to another journal needs adaptation of the copy when descriptive markup is not used. I encountered the following in the TEXscript:

```
{\bf Summation of infinite
     series of complex functions.}
{\bf Theoretical background.}
\vskip1truecm
{\it1.2.1\underbar{Ten standard
            definitions}}.
\vskip1truecm
\underbar{Definition 1.}
```

As can be seen, a lot of typographical detail had been supplied. Agreed, it is generally available in the mathscript because authors are accustomed to supplying the section numbers and indicating bold and underlining. Using formats provided by the publisher does pay, because it is then the concern of the publisher to achieve the correct results.

To the same category of pitfalls belongs the typing of commands for creating extra white space along with each display, especially when the mathscript is full of crowded formulae. Instead of repeatedly typing `\vskip`'s, use can be made of the `\everydisplay` command, along with the assignment of new values to `\abovedisplayskip`, `\belowdisplayskip`, and their shorter variants.

Similarily, more white space can be placed between lines within a display. One does not have to modify the code because `\openup` is a cummulative command. Just say, for example, `\openup1\jot`, and interline spacing is increased by the given amount. AMS-TEX's `\spreadlines` is a disguise for this assignment, I presume.

Redefinition of existing commands is also a pitfall. First, trivially, because the original meaning of the command might be lost by accident. Second, when the command is to be customized — i.e., `\proclaim` *into* `\theorem` — one cannot simply use `\proclaim` *within* `\theorem` because `\proclaim` is an outer command. You must either remove `\outer` from `\proclaim` or copy the body of `\proclaim` into `\theorem`

Emptiness. The spacing pitfall is a difficult morass.[14] Once the automatic spacing is overruled by explicit spacing commands, the inconsistency pitfall opens up. A nice list of rules for spacing between symbols in math is given in Swanson (1986, chapter 3).

In math mode, spaces in the source file are ignored. Before and after each formula, space is inserted of size `\mathsurround`, which defaults to `0pt` in plain TEX (*TB* 162, 353). Within a formula the spacing is context-dependent, and determined by the class of the math character. Some symbols, those of the binary class for example, get extra spacing *around* them. Punctuation symbols have spacing *after* the symbol. The math character classes are given on page 154 of *The TEXbook*. For each class, the precise spacing values, related to the context, are given in the table on page 170 of *The TEXbook*.

Alas, in spite of — or because of — the above mentioned automatic mechanisms, extra space has to be inserted or deleted now and then.

- Some examples of additional positive and negative spacing in math mode:

$$(\lambda)_2\,{}_2F_1,\quad \int\!f(x)\,dx,\quad \Gamma_{\!2}+\Delta^{\!2},\quad \sum^\infty_{n=-\infty}\!\!\!\cos nt$$

are TEXed as (*TB* 168):

```
$$(\lambda)_2\,{}_2F_1,              \quad
         \int\!f(x)\,dx,             \quad
 \Gamma_{\!2}+\Delta^{\!2},          \quad
\sum^\infty_{n=-\infty}\!\!\!\cos nt$$
```

[14] Once in a while I think of TEX as dealing essentially with flexible spaces.

In the Temme book I found $n!n^2$ encoded as `$n!n^2$`, as opposed to $n!\,n^2$, which is correctly encoded as `$n!\,n^2$`.

Negative kerning after integral signs was not used either, especially with double integrals. The integral signs are spread too wide and stood too far away from the integrand. Summation symbols with large limits would also have benefitted from negative kerning.

Another aspect of spacing is ${}_1\phi_0(a;-;q,z,)$. The empty symbol ␣ could have been used by using `{\tt\char'040}`.

And what about placeholders? For example, the source `$\bigl(f,K_n(\cdot,y)\bigr)$` yields $\bigl(f,K_n(\cdot,y)\bigr)$? Introduce spaces around the placeholder via '`\,\cdot\,`'.

- Also of interest are expressions in exponents or indices. The Temme book contained:

$$e^{-z\sinh t+\nu t}$$

which does not look nice because of suppression of space around the operator. Introduce explicit thin spaces before and after the binary operator, or use parentheses around the argument of the function. Why not format $\exp\,(-z\sinh t+\nu t)$, in agreement with the general advice to use `\exp` for non-simple exponents? For other situations where `\exp` cannot be used, `\hbox{$...$}` can be considered as a sub- or superscript, yielding the correct spacing.[15]

In the TEXscript I encountered:

```
\wit  %meaning white space
$$|t| \quad < \quad | \quad x \quad -
  \quad (x + 1)^{\frac12}
  \quad (x - 1)^{\frac12} \quad |,$$
\wit
```

Spacing between formulae was not understood and done inconsistently. Unnecessary extra white space was introduced in too many places by the insertion of hundreds of `\,`, `\quad`, and `\qquad`'s.

On the use of `\(q)quad`, the best quote may be found on page 166 of *TB*:

> The traditional hot-metal technology for printing has led to some ingrained standards for situations like this, based on what printers call a "quad" of space. Since these standards seem to work well in practice, TEX makes it easy for you to continue the tradition: When you type '`\quad`' in plain TEX format, you get a printer's quad of space in the horizontal direction. Similarly, '`\qquad`' gives you a double quad (twice as much); this is the normal spacing for situations like the F_n example above.

A little further down on page 166 of *The TEXbook* the reader's attention is drawn to a different approach which is needed in alternating math and text in a paragraph.

```
$F_n=F_{n-1}+F_{n-2}$, for $n\ge2$.
```

Consistency can be enhanced by defining a document element, and subsequently using the element via its name. For example, the real part of z can be obtained in math mode via `\Re z`, once we have defined

```
\def\Re#1{{\rm Re}\,#1}
```

In the Temme book this was implemented via `{{\cal R}\,#1}`, which is especially handy when real parts of quantities are used in formulae. In the TEXscript I also encountered the following subtle examples which, after correction, read

$$C_\nu^\lambda(-z)=\cos\pi\nu\,C_\nu^\lambda(z)-\sin\pi\nu\,D_\nu^\lambda(z),$$

where '`\,`' (extra space) had to be inserted after the arguments of the trigonometric functions. In the Temme book, similar situations were circumvented via parentheses, $\cos(\pi\nu)C$, via `$\cos(\pi\nu)C$`; no extra space had to be inserted after the closing parentheses (*TB* 170).

Class unawareness. Several examples are provided below which demonstrate the unawareness of mathematical characters belonging to one of the eight classes, (*TB* 154).

- An example on page 171 of *TB* shows:

  `$|-x|$`, `$\left|-x\right|$`,
  and `$\lfloor-x\rfloor$`

  with the results $|-x|$, $\left|-x\right|$, and $\lfloor-x\rfloor$.[16] In the Temme book I found $\gamma^\star(a,x)$, as well as $\gamma\star(a,x)$. Do you see the difference?

Innocent braces. The pitfall here is that braces are not harmless but yield a formula of class 0 within math mode!

- Compare the following results:

  $a+b$, $a{+}b$ and a+b

  with their respective source code:

  `$a+b$` `$a{+}b$` `a+b`

[15] Petrycki (1991) also mentions difficulties with math spacing in TEX. The spacing around growing parentheses and the lack of spacing in sub- and superscripts is unacceptable.

[16] Why $|-x|$, and not just $|x|$? Furthermore, norm fences don't belong to the opening or closing class.

The first + is of the class binary and takes spacing according to the table on page 170 of *TB*, and in the second, the + is reduced to the class zero, and takes only `\mathsurround` spacing (default in plain TEX is 0pt).

- Analogously:

  $a = b\,,\qquad a{=}b\qquad$ a=b

  are obtained via:

  ```
  $a=b$     $a{=}b$     a=b
  ```

TEXperts frequently use braces, especially in alignments where empty formulae are to be used now and then. Another occurrence of *harmful* braces is given by \cnt={1} and the like, yielding an error message.

The general issue is that in math mode braces have the extra function of creating a subformula — not only delineating a scope — with the resulting subformula being an atom of class ordinary (*TB* 154, 158, etc.), and therefore BLUe must understand the various atom classes.

Whoops. What about the following?

```
\def\Inn{{\raisebox{1pt}
        {{\hbox{{$\in$}}}}}}
```

The concept of a binary operator was not accounted for, yielding the wrong spacing. TEX did not know that the raised `\in` had to be considered as an operator and therefore it was reduced to class ordinary, taking `\mathsurround` spacing.

- In the Temme book, I found:

  $$2\pi i\, {\rm Res}_{s=e^{i\pi}} f(s) = -2\pi i\, e^{i\pi z}$$

  TEXed via:

  ```
  2\pi i\, {\rm Res}_{s=e^{i\pi}}f(s)
      =-2\pi i\, e^{i\pi z}.
  ```

- I prefer the `Res` operator (in display and in agreement with Swanson, 1986) to look like this:

  $$\mathop{\rm Res}_{s=e^{i\pi}} f(s) = -e^{i\pi z}$$

  encoded as:

  ```
  \mathop{{\rm Res}}_{s=e^{i\pi}}
        f(s)=-e^{i\pi z}
  ```

An example of spacing which has to be suppressed is in *<name>*, (coded as `${<}name{>}$`). The relational operators are not used as such, and are forced by the braces into class ordinary. The latter example is taken from the BNF notation of programming languages, denoting meta-linguistic variables. (DEK uses $\langle$ and $\rangle$ in his syntax, with similar use as (and).)

Just a comma. The number 3,14 innocently encoded as `$3,14$`, would yield $3,14$. The correct formatting is `$3{,}14$` (*TB* 134). Braces are needed again. The comma belongs to the punctuation class of math symbols and the surrounding braces — creating a subformula — reduce it to class ordinary, which requires no extra spacing. As part of the text, the number could have been obtained via `3,14`, with no `$`'s around it.

Binary operator vs. punctuation mark. A dot is used for a (binary) multiplication operator and as a punctuation mark. Three dots in a row don't yield an ellipsis. The formatting of the ellipsis is context-dependent: they can be at the axis of the formula, or at the baseline. Moreover, they can be vertical, or diagonal.

Multiplication in mathematics can be denoted by: $a \times b$, $a \cdot b$, or implicitly by a thin space $a\,b$, which has to be marked up explicitly. If you simply enter `a b`, the space will be gobbled up by TEX when it's in a good mood. Typists, and those used to the old typewriter, err by using '`x`' for `\times`, and by using the punctuation dot '`.`' instead of `\cdot` (with the binary multiplicator operator raised above the baseline).[17]

The real issue is that the handwritten symbol must be recognized from the context: is it a punctuation mark, an operator, or a significant space?

Colons: is there a difficulty? A colon as a *punctuation* symbol can be obtained via the `\colon` command, and as a *relation* symbol via '`:`', (*TB* 134).

- Examples of colons:

  $$f\colon A\to B,\quad \{x:x>5\}$$

  are obtained via:

  ```
  $$f\colon A\to B,\quad \{x:x>5\}$$
  ```

The Temme book used ':' throughout.

CMR fonts. Text in displays and standard function names traditionally use roman fonts, Swanson (1986, Table IV).

Sinus hyperbolicus ($\sinh x$) was encoded as `$\hbox{sin }hx$`, demonstrating bad handwriting by the author, and incorrect encoding. The TEXist was not familiar with the hyperbolic function names and therefore could not compensate for the bad handwriting. I also encountered the following badly encoded examples:

[17] To this category of misuses I also include 1 vs. l, 0 vs. o, U vs. ∪ etc. For more examples of these erroreous similarities, see "A Manual for Authors of Mathematical Papers" (AMS, 1973).

```
\cos\,\alpha
h^\lambda_\nu (z) \, : = _2F_1( ... )
\hbox{ for Re }\, z \, > \, 0
```

Chameleons. Regarding the chameleon pitfall, I mention those situations where TEX can't determine the correct sizing from the context. TEX provides facilities for automatically formatting the right size, given the context. TEX provides, for example, the correct sized openings and closings for a matrix, when these are specified by `\left...` and `\right`.[18] A TEXfall occurs when the context does not prompt for the possible need for another size (while BLUe expects TEX to do everything correctly).

- An example of context-dependent sizes, as inspired by Spivak (1986:55):

  ```
  $$||\alpha(\sqrt a+\sqrt b)||
       \leq|\alpha|.
     (||\sqrt a+\sqrt b||).$$
  ```

  yields the result:

  $$||\alpha(\sqrt a+\sqrt b)|| \leq|\alpha|.(||\sqrt a+\sqrt b||).$$

- Better encoding would be:

  ```
  $$\bigl\|\,\alpha(\sqrt{\mathstrut a}+
           \sqrt{\mathstrut b}\,)\,\bigr\|
    \leq|\alpha|\,
    \bigl\|\sqrt{\mathstrut a}+
     \sqrt{\mathstrut b}\,\bigr\|$$
  ```

  with resulting:

  $$\bigl\|\,\alpha(\sqrt{\mathstrut a}+\sqrt{\mathstrut b}\,)\,\bigr\| \leq|\alpha|\,\bigl\|\sqrt{\mathstrut a}+\sqrt{\mathstrut b}\,\bigr\|$$

In this example, the norm fences are made larger and all `\sqrts` have been told to have arguments of `\mathstrut` size (Ascender and descender invariance!). Moreover, the multiplication dot can be replaced by a thin space.

- Another use of the vertical bar occurs in set notation (*TB* p175, ex18.21):

  $$\bigl\{\,x^3\bigm|h(x)\in\{-1,0,+1\}\,\bigr\}.$$

  obtained via:

  ```
  $$\bigl\{\,x^3\bigm|h(x)\in%
          \{-1,0,+1\}\,\bigr\}.$$
  ```

This not only demonstrates the correct size of the outer braces and the vertical bar, but also exhibits awareness of the binary operator function of the vertical bar, with the appropriate spacing by default. Set notations in the Temme book had not been marked up via the use of `\mid|` or its variants. For nested parentheses the `\big`, etc., representations were not used. The old technique with square brackets for the outer parentheses was used: $[\ln(z+1)]^m$.

Note that it looks better to introduce some spacing along with the outer braces. If an author wants these kinds of results he has to indicate that in the mathscript. I expect these kinds of issues will not be touched upon or will be handled inconsistently in a document of nontrivial size.

Triads. The 'three dots in a row,' or ellipsis, are heavily used in mathematical notation.

- For example:

  $$x_1+\cdots+x_n,\quad x_1x_2\ldots x_n$$

  is obtained by using the `\cdots` and `\ldots` commands.

The issue is not to type '...', but to use the `\cdots` or `\ldots` command, respectively.

When the ellipsis is followed by a punctuation dot, a small extra space '`\,`' has to be specified: `1+x+x^2+\cdots\,`. An ellipsis is often used in a fixed context.

- For example: for i = 1, 2, ..., n. can be obtained via:

  ```
  for $i=1$,~2, $\ldots\,$,~$n$.
  ```

Such sentences are candidates for abbreviation into a macro such as `\for in`:

```
\def\for#1#2{
     for $#1=1$,~2, $\ldots\,$,~$#2$},
```

maintaining consistency. It also reduces the number of keystrokes. Note that `\dots` is not substituted for `\ldots\,`, and '`\,`' is needed. The pitfall of confusing the use of `\dots` and `\ldots` is also circumvented by the use of the `\for` abbreviation. The Temme book was inconsistent in the use of `\cdots` and `\ldots`.

In order to facilitate looking up shortcuts, Wichura (1990) has provided some macros which yield a table consisting of a math-writing-column and a corresponding TEX-input-column — a fancy tool suited for typists. This is not enough to solve the typist's problems, though it might help.

Real life. Other dots are also used: vertical dots in matrices, (*TB* 177), and diagonal dots, (*TB* 177, ex18.45). Triple-dotted letters are particularly captivating.

With respect to the continued fraction example provided in the 'Am I BLUe' section, Swanson (1986) just uses an ellipsis. Her variant notations also differ somewhat. Those given here originate from Peter Henrici (1977). His Φ symbol is the absolute space saver.

The auxiliary symbol `\cf`, the Φ, must be made robust, so that it can be used with other styles

[18] TEX automatically adapts the correct sizing for (or `\{` , when using `\left` or `\right`.

as well, yielding appropriate sizes. TEX provides \mathchoice for this purpose. For example, \cf should have been defined as:

```
\def\cf{\mathop{\mathchoice{%
\grkop\Phi}{%Magnified
\Phi}{\Phi}{\Phi}      }}
```

Accents differ. Accents are treated differently in math mode than in horizontal mode. DEK decided to provide different math mode accent commands (*TB* 135). It is unclear to me why the commands have not been overloaded.

Embellishments.

- We have barred letters:

  $$\bar z,\ \overline z,\ \bar P,\ \overline P,\ \bar h,\ \hbar,\ \overline{AB}$$

  obtained via:[19]

  ```
  $$\bar z,\ \overline z,\
  \bar P,\ \overline P,\
  \bar h,\ \hbar,\
  \overline{AB}$$
  ```

It is easy to forget that \bar provides a bar of fixed length, and looks strange when used over capitals. It is wrong to use it with subformulae in general; this holds for all accents. The Temme book used \bar over capitals. Note also the use of \vec a, for $\vec a$, \vec A, for $\vec A$, (A accented with an arrow), and \overrightarrow A, for $\overrightarrow{A}$, or \overrightarrow{AB}, for $\overrightarrow{AB}$, (see *TB* 136, 359) — once again a source of confusion and inconsistency. Swanson (1986) advises the use of boldface characters for vector notation.

Normalized functions are often denoted by dotted letters, and because a dot is a tiny blot of ink, more pronounced dots are needed. I encountered the use of bulleted letters, which look awful. Something like a bold dot is needed. Bold dotted P — $\boldsymbol{\dot{}}\!\dot P$ — and simple dotted P — $\dot P$ — are obtained via \bdot P and \dot P, with the use of:

```
\def\bdot#1{{\bf\dot{{\mit#1}}}}
```

Note the extra pair of braces, another TEXfall.[20] The Temme book even used triple-dotted letters, to denote the Schwarz derivative, $\dddot{x}$. The above was done via:[21]

```
\def\dddot#1{%
\setbox0=\hbox{$#1$}
\vbox to\baselineskip{\vss\hbox{%
 ${\mkern1mu.\mkern-2mu.\mkern-2mu.}
               \phantom{\char'177}$}
     \kern-ht0
     \copy0}}%end dddot
```

Alas, we both committed a TEXfall. The above approach is based upon rule 12 in Appendix G-12 page 443 of *The TEXbook*, and should be compatible with \dot, and \ddot. It is not compatible at the moment and Appendix G-12 should be revisited. (My TEXnigma could not reproduce Malcolm's results when fed his experimentally found kernings.) In math literature I found $\ddot y$.

Prime-ry. Characters are primed in math mode to denote derivatives. For instance, y' is encoded by y^\prime or via the more natural shorthand y'. The ' character is active and overloaded (context-dependent).[22] The TEXfall is that symbols which have limits can't be simply primed in display.

- Compare the following examples:

  $$\overset{\prime}{\sum} \quad \sum{}' \quad \sum_{k=0}^{\overset{n}{\prime}} \quad \sum_{k=0}^{n\ \prime} \quad \sum_{k=0}^{n}{}' \quad \mathop{\sum{}'}_{k=0}^{n} \quad \mathop{\sum{}'}_{k=0}^{n}$$

  obtained via:

  ```
  \def\acclap#1{
     \raise\hgtsig\hbox to0pt{$#1$\hss}}
  \newdimen\hgtsig
  \setbox0=\hbox{$\displaystyle{\sum}$}
  \hgtsig=\ht0\relax
  \advance\hgtsig by -1.75ex
  $$\sum'                            \quad
  {\sum}'                            \quad
  \mathop{\sum'}^n_{k=0}             \quad
  \mathop{{\sum^n_{k=0}}'}           \quad
  \mathop{\sum\mathstrut'}^n_{k=0}\quad
  \mathop{{\sum}'}^n_{k=0}           \quad
  \mathop{{\sum}\acclap'}_{k=0}^n $$
  ```

DEK's solution is given in ex18.44 of *The TEXbook*. Primed summation symbols are used in Chebyshev expansions, for example.

The double primed summation symbol can be obtained in a similar fashion. As mentioned by Knuth, the problem is to center the lower limit with respect to the summation symbol proper.[23]

$\frac{1}{2}$'s never have the right size. BLUe invariably goes wrong when TEXing 'halves.'

[19] AMS fonts also provides \hslash similar to \hbar but with the bar inclined.

[20] The extra braces are necessary in order to keep the font changes local.

[21] Clark (1987) used a fixed vertical kerning and therefore his code is not robust.

[22] The accent itself can be obtained within math mode via \mathchar".

[23] Scripting the primed operator would violate this.

- The following example:

$$D^\lambda_0(z)=4a_\lambda\, z\, {}_2F_1(\textstyle\lambda+{1\over2},{1\over2};{3\over2};z)$$

is TEXed via:

```
$$D^\lambda_0(z)=4a_\lambda\, z\,
{}_2F_1(\textstyle\lamda+{1\over2},
         {1\over2};{3\over2};z)$$
```

In the Temme book, I encountered the above notation, and also $F(1/2,1/2;3/2;z^2)$. Later, I stumbled upon $\int^{\frac{1}{2}\pi}$ along with the more usual $\int^{\pi/2}$. The latter is also recommended by Swanson (1986).

- In the TEXscript,[24] I found:

$$D^\lambda_0=-{\textstyle\sin{\pi\nu\over2}}\, C^{\scriptscriptstyle\lambda\over\scriptscriptstyle2}_{\scriptscriptstyle\nu\over\scriptscriptstyle2}$$

via:

```
$$D^\lambda_0=-{\textstyle
\sin{\pi\nu\over2}\,
C^{\scriptscriptstyle\lambda\over
              \scriptscriptstyle2}
 _{\kern-1pt{\scriptscriptstyle\nu\over
              \scriptscriptstyle2}}}$$
```

The general point is to kern and force the right style. Another example of where the right style is coerced occurs when the summation symbol takes stacked limits. Explicit mentioning of \scriptstyle in both operands of the \atop command is needed (*TB* 145).

Knuth (1985), mentions the use of a typographer's '1/2,' especially in recipes, which works better than a mathematician's '$\frac{1}{2}$'.

Various O0$O\emptyset\circ o$'s.

Mathscript O's are overloaded: '$\emptyset$' (the empty set), $f\circ g\colon x\mapsto f\bigl(g(x)\bigr)$ (composition), and the order symbols $o(h^2)$ and $O(h^2)$:

-
  ```
  $\emptyset$,
  $f\circ g\colon
    x\mapsto f\bigl(g(x)\bigr)$,
  $o(h^2)$,
  $O(h^2)$.
  ```

We also have trigonometric and temperature degrees 30° and °K (*TB* 180). Another challenge is a notation for the zero vector, (see *TB* ex18.6).

Backslash penances. Because of the special function of the backslash, people are in trouble when the symbol itself is wanted. In horizontal mode the backslash, as such, can be obtained by selecting the symbol from the tt font, (*TB* 429) position '134 (decimal 92), via {\tt \char'134}. In math, the backslash is used for the setminus (binary) operator and for denoting cosets; the latter requires no space. Compare:

$$A\setminus A=\emptyset \quad \text{and the cosets of } G \text{ by } H\text{: } G\backslash H$$

TEXed by use of \setminus and \backslash (*TB* 436). Needless to say, the mathscript contained several setminus operations, while in the TEXscript the \backslash was used throughout.

Over and over. BLUe is encouraged to treat a fraction as a subformula (*The TEXbook* 140, ex17.3), and to use braces around <*formula* \over *formula*>—a good habit to adhere to throughout. I was trapped when changing \left(and \right) into \bigl(and \bigr). The former notation creates a subformula while the latter does not—this is not robustness!.

Swanson (1986) advises us to consider that the use of slashes when saving space can be achieved while preserving clarity of exposition.

In \buildrel (*TB* 437), \over is overloaded.

Too difficult. Hypergeometric functions sometimes take 'matrices' as arguments. As stated in *TB* (page 178), the use of \(p)matrix in the text of a paragraph yields results which are too big: $M_n(z)={}_{n+1}F_n\bigl({k+a_0,\atop}{k+a_1,\dots,k+a_n\atop k+c_1,\dots,k+c_n};z\bigr)$ is obtained via:

```
$M_n(z)={}_{n+1}F_n
  \bigl({k+a_0,\atop\phantom{kc_1}}
       {k+a_1,\dots,k+a_n\atop
            k+c_1,\dots,k+c_n};z\bigr)$
```

Note the automatic centering 'on the axis' of the last argument. A fuzzy issue involves what to do with empty arguments, especially when several \atop's are used in a row. The general approach is to use \mathstruts. For two \atops the use of \phantom will yield aligned results, as demonstrated in the given example.

The late Yudell Luke used the '|' symbol instead of ';'. For example:

$${}_pF_q\Bigl(\,{\alpha_p\atop\rho_q}\mathpunct{\bigm|}\,z\Bigr)={\Gamma(\rho_q)\over\Gamma(\alpha_p)}\,G^{1,p}_{p,q+1}\Bigl(-z\,\mathpunct{\bigm|}\,{1-\alpha_p\atop 0,1-\rho_q}\Bigr)$$

is obtained via:

```
$${}_pF_q
\Bigl(\,{\alpha_p\atop\rho_q}
  \mathpunct{\bigm|}\,z
\Bigr)=
{\Gamma(\rho_q)\over
   \Gamma(\alpha_p)}\,
G^{1,p}_{p,q+1}
\Bigl(-z\,\mathpunct{\bigm|}\,
  {1-\alpha_p\atop0,1-\rho_q}
\Bigr)$$
```

[24] To be avoided, (Swanson, 1986).

Here the vertical bar is coerced into serving as a punctuation symbol, with some extra spacing added. Also note the lack of spacing in the subscripts and superscripts (*TB* 170).

The TeXscript contained:

```
\begin{dispeqs}
M_n(z):={}_{n+1}F_n
\left(\aligned
k+a_0,&k+a_1,\dots,k+a_n;z\\
      &k+c_1,\dots,k+c_n
      \endaligned\right)
\tag1.2.55
\end{dispeqs}
```

which does not reflect the centering of '; z.' It also demonstrates the use of an unnecesssary 'non-standard' environment, inhibiting intelligibility. The desired result was not obtained, while the TeXscript was encoded with too much, and unnecessary, complexity. (Fortunately, I have never needed to talk about hypergeometric functions by phone.)

Jam Session I

The conclusion of this work is that TeXing math is too difficult for non-TeX-trained typists. The need for more TeX instruction must be taken into consideration by mathematicians as well as those who keyboard the TeXscripts. Both the author and the typist need to have more than a nodding acquaintance with TeX. Education is needed and discipline must to be adhered to. What about a discipline of TeXing?

Bibliography

"A Manual for Authors of Mathematical Papers." Providence, R.I.: American Mathematical Society, 1973. [Booklet, 24pp.]

"AMS-TeX User's Guide," 2nd ed. American Mathematical Society, 1990.

Beeton, Barbara N. "*TUGboat* Production: TeX, LaTeX and Paste-Up." Appendix S of the Minutes of the Fifth NTG Meeting. [Paper presented at the SGML-TeX Conference, Groningen, 1990.]

Cheswick, B. "A Permuted Index for TeX and LaTeX." CST145, AT&T Bell Laboratories, 1990.

Clark, Malcolm. "More Symbols for TeX." TeXline 5, pages 7–8, 1987.

Henrici, P. *Applied and computational complex analysis. II Special functions, integral transformations, asymptotics, continued fractions.* John-Wiley, 1977.

Knuth, Donald E. *The TeXbook.* Reading, Mass.: Addison-Wesley, 1984. [TeX, frozen in version π, 3.1415... is treated in Vol. A of *Computers and Typesetting* from the *ninth* printing onwards, and in the *TeXbook* from the *seventeenth* printing onwards. And if you can't remember the numbers, look for the `\language` command in the index.]

Knuth, Donald E. "Recipes and Fractions." *TUGboat* 6(1), pages 36–38, 1985.

Knuth, Donald E., Tracy Larrabee, Paul M. Roberts. "Mathematical Writing." MAA Notes 14. Stanford: American Mathematical Association, 1989. Also available as STAN-CS-88-1193, January 1988. [With contributions by Palmos Halmos, Leslie Lamport, Mary-Claire van Leunen, Nils Nilsson, Rosalie Stemer, Jeff Ullman, and Herbert S. Wilf.]

Mittelbach, Frank. "E-TeX: Guidelines for Future TeX Extensions." *TUGboat* 11(3), 337–345, 1990.

Petrycki Laurie J. "Comparing TeX and Traditional Typesetting for the Composition of a Textbook." Paper presented at 1991 TUG Annual Meeting — see elsewhere in these proceedings.

Schwarz H. R, H. Rutishauser, E. Stiefel. "Numerik symmetrischer Matrizen." Stuttgart, Teubner, 1972. [ISBN 3-519-12311-8, in German.]

Spivak, Michael D. *The Joy of TeX. A Gourmet Guide to Typesetting with the AMS-TeX Macro Package.* Reading, Mass.: Addison-Wesley, 1986.

Swanson, Ellen. "Mathematics into Type." Providence, RI.: American Mathematical Society, 1986. [Copyediting and proofreading of mathematics for editorial assistants and authors.]

Temme, N. M. "Speciale functies in de mathematische fysica." *Epsilon* 15, 1990. [ISBN 90-5041-019-7, in Dutch.]

Whitney, Ron F., Barbara N. Beeton. "*TUGboat* Authors' Guide." *TUGboat* 10(3), 378–385, 1989.

Wichura, Michael J. "Showing-Off Math Macros." *TUGboat* 11(1), 57–61, 1990.

Wilkinson, J. H. *The Algebraic Eigenvalue Problem.* Oxford: Clarendon Press, 1965.

Rhapsody in Blue

The following is the source code for the 'Am I Blue' section.

```
\bi
\item Selections from chapters
16\dash 18 in {\sl \TB\/}:
%
$${\bf S^{\rm-1}TS=dg}(\lambda_1,
   \ldots\,,\lambda_n)=\bf\Lambda$$

$$\sum_{k=1}^{\infty}{1\over2^k}=1,\quad
  \sqrt{1+\sqrt{1+\sqrt{1+x}}}$$

$${f(x+\Delta x)-f(x)\over\Delta x}\to
  f'(x)\quad{\rm as}\quad\Delta x\to0$$
%
$$\{\underbrace{\overbrace{\mathstrut a,
  \ldots,a}^{k\;a\mathchar`'\rm s},
    \overbrace{\mathstrut b,\ldots,b}
      ^{l\;b\mathchar`'\rm s}}
      _{k+l\rm\;elements}\},
\quad
 2\uparrow\uparrow k\mathrel{\mathop=
  ^{\rm def}}
  2^{2^{2^{\cdot^{\cdot^{\cdot^2}}}}}
  \vbox{\hbox{$\Big\}\scriptstyle k$}
  \kern0pt}
$$
%
\item {The Cardano solution of $x^3+px=q$,
with $p,q\ge0$}  reads:
\ei
$$ \root3\of{\sqrt{p^3/27-q^2/4}+q/2}-
   \root3\of{\sqrt{p^3/27+q^2/4}-q/2}$$
\bi
\item {Derivatives}:\footnote{Kerning
an extra point in superscripts was
pointed out by Daniel Olson.}\\
$\dot y\,\ddot y\, \dot{\ddot y}\quad
  y'\,y''\,y'''               \quad
\partial_xy\,\partial_x^{\kern1pt2}y,
\partial_x^{\kern1pt3}y$
%

%Some more from analysis
\item {Bessel equation}:
$$z^2w''+zw'+(z^2-\nu^2)w=0$$
with solutions:
\par\noindent
$J_{\pm\nu}(z)$, Bessel function of the
first kind,\par\noindent
$Y_{\pm\nu}(z)$, Bessel function of the
second kind (Weber),\par\noindent
$H_\nu^{(1)}(z)$, and $H_\nu^{(2)}(z)$,
Bessel function of the third kind
(Hankel).
%
\item {Primed summation symbols} are used
in Chebyshev expansions:
\def\acclap#1{
   \raise\hgtsig\hbox to0pt{$#1$\hss}}
\newdimen\hgtsig
\setbox0=\hbox{$\displaystyle{\sum}$}
\hgtsig=\ht0\relax
\advance\hgtsig by -1.75ex
$$\displaylines{\quad
 \setbox0=\hbox{$\displaystyle
 \mathop{{\sum}\acclap'}_{k=0}^n$}
 \dp0=0pt \box0    %Neglect dp size
 a_kT_k(x)\mathrel{\mathop=^{\rm def}}
 {\textstyle.5}\kern1pt a_0+a_1 x+
   a_2T_2(x)+\cdots\hfill         \cr
   \hfill{}+a_nT_n(x).\quad       \cr
}$$
%
\item {Hypergeometric function}:
$$M_n(z)={}_{n+1}F_n\Bigl({k+a_0,
   \atop\phantom{kc_1}}
   {k+a_1,\dots,k+a_n
   \atop k+c_1,\dots,k+c_n};z\Bigr)
$$
%
\item {From Swanson (1986:40)}:
$$\displaylines{
  \int\nolimits_U\delta(I)\mu(I)
                  \leq{}\hfill\cr
{}\sum_{{\cal D}}
         \sum_{{\cal D}_{I'}}
  \biggl[\int\nolimits_J
  \alpha(J')\mu(J')-\alpha(J)\mu(J)
                         \hfill\cr
%CGL278 next lay-out modified
  \qquad\hfill  {}-\int\nolimits_J
    [\{s(\alpha\eta)(J')\}
       /\eta(J')]\mu(J')\biggr]\cr
{}+\biggl[ %CGL278 modified, to align
%                    with double sum
       \sum_{{\cal D}}
       \sum_{{\cal D}_{I'}}
   |\alpha(J)-[\{s(\alpha\eta)(J)\}
   /\eta(J)]|\mu(J)\biggr] \hfill\cr
  \hfill
   {}\times\biggl[
       \sum_{{\cal D}}
       \sum_{{\cal D}_{I'}}
   |\alpha(J)-[\{s(\alpha\eta)(J)\}
   /\eta(J)]|\eta(J)\biggr] \cr} $$
%
%
\item {Magic squares}
(D\"urer's 4-by-4  with dotted lines):
$$
\vcenter{\tabskip0pt\offinterlineskip
 \hrule\halign{\strut\vrule height3ex
    depth1.5ex\relax
    \enspace\hfil#\hfil\enspace\vrule&&
               \strut
    \enspace\hfil#\hfil\enspace\vrule\cr
    2& 7& 6\cr \noalign{\hrule}
    9& 5& 1\cr \noalign{\hrule}
    4& 3& 8\cr}            \hrule}
\quad
%
\def\cdotfill{\cleaders\hbox{%
$\mathsurround0pt\mkern1.5mu{\cdot}
 \mkern1.5mu$}\hfill}%end cdotfill
\def\vdotfill{%
   $\vcenter{%dotted line
   \vbox to5ex{\cleaders\hbox{%
   $\mkern1.5mu{\cdot}\mkern1.5mu$}
    \vfil}}$\hfil}%end vdotfill
%
\vcenter{\tabskip0pt
 \offinterlineskip
 %\hrule replaced by \cdotfill
 \halign{
   \vdotfill
   $\vcenter{\hbox to 5ex{\hss\oldstyle#\hss}
    }$%\hfil\enspace
   \vdotfill&&     %repetition
   %\enspace\hfil
   $\vcenter{\hbox to 5ex{\hss\oldstyle#\hss}
    }$\hfil\enspace
   \vdotfill  \cr%end template
   \multispan4\cdotfill\cr
   16& 3& 2&13\cr
   \multispan4\cdotfill\cr
    5&10&11& 8\cr
   \multispan4\cdotfill\cr
    9& 6& 7&12\cr
   \multispan4\cdotfill\cr
    4&15&14& 1\cr
%No bold oldstyle 1514 available
   \multispan4\cdotfill\cr
}%end \halign
}$$
%
\item{Calculation flow of autocorrelation
function
$a_f$} (inspired by {\sl TB\/} 358, ex18.46).
${\cal F}$ denotes Fourier transform and
${\cal F}\strut^{-}$ the inverse Fourier
transform:
%
\def\lllongrightarrow{\relbar\joinrel%
       \relbar\joinrel\relbar\joinrel%
       \relbar\joinrel\rightarrow}
\def\llongrightarrow{\relbar\joinrel%
        \relbar\joinrel\rightarrow}
\def\normalbaselines{%
           \baselineskip20pt
           \lineskip3pt
           \lineskiplimit3pt}
\def\mapright#1{\smash{\mathop{
   \llongrightarrow}\limits^{#1}}}
\def\lmapright#1{\smash{\mathop{
   \lllongrightarrow}\limits^{#1}}}
\def\mapdown#1{\Big\downarrow
      \rlap{$\vcenter{\hbox{$#1$}}$}}
\def\mapup#1{\Big\uparrow
      \rlap{$\vcenter{\hbox{$#1$}}$}}
$$%Diagram
\matrix{f&\lmapright\otimes&a_f\cr
    \mapdown{{\cal F}}&&\mapup{%
            {\cal F}\strut^{-}}\cr
    \hbox to 0pt{\hss${\cal F}(f)$\hss}
    &\mapright\times\hfil&
    \hbox to 0pt{\hss$\bigl({\cal F}(f)
                    \bigr)^2$\hss}\cr}
$$
%
%%%%%%%%%  Amy's Diagonal line %%%%%%%
\newif\ifminuswd
\newif\ifminusht
\newbox\dotwide
\newbox\dotbox
\newdimen\savehtofdiagline
\newdimen\htofdiagline
\newdimen\wdofdiagline
\newdimen\dotmove
\newdimen\newsaveht
\newdimen\savewdofdiagline
%
\def\diagline #1 #2 wd #3 ht{%
%Amy Hendrickson, improved
%version of TB 6#2, 1985
\global\setbox\dotwide=\hbox{#1}%
\global\setbox\dotbox=
   \hbox to0pt{\hss#1\hss}%
\global\wdofdiagline=#2\relax%
\ifdim\wdofdiagline<1sp
   \global\minuswdtrue
   \global\advance\wdofdiagline
                 by-2\wdofdiagline
\else\global\minuswdfalse%Robustness
\fi% turn neg dim to positive
\global\htofdiagline=#3\relax%
\ifdim\htofdiagline<1sp
   \global\advance\htofdiagline
   by-2\htofdiagline\global\minushttrue
\else\global\minushtfalse%Robustness
\fi
\global\dotmove=1pt%%
\setbox1=\hbox{%
   \global\divide\htofdiagline
    by\the\wdofdiagline\relax}%
\ifminuswd\rlap\bgroup%
\else\bgroup\fi%
\loop\ifdim\htofdiagline>.4pt%CGL mod
\global\divide\htofdiagline by2%
\global\divide\dotmove by2\repeat%
\global\savehtofdiagline=\htofdiagline%
\loop\ifdim\wdofdiagline>0pt%
\hskip\ifminuswd-\dotmove%
      \else\dotmove\fi%
```

```
\ifminusht\lower\else\raise\fi%
      \htofdiagline\copy\dotbox%
\global\advance\htofdiagline
          by\savehtofdiagline%
\global\advance\wdofdiagline
          by-\dotmove\repeat%
\egroup%
}% end diagonal line
%
\def\icmat#1#2{%ICon MATrix(rectangular)
%#1 is ht of icon matrix, e.g. 4ex
%#2 is wd of icon matrix, e.g. 2ex
\vbox to#1{\hrule
           \hbox to#2{\vrule height#1
                         \hfil\vrule}
           \hrule}
}%end icmat
%
\def\icurt#1#2{%IConUpperRightTriangle
%#1 is ht of icon matrix,
%with UT the upper triangular part,
%e.g. 4ex #2 is wd of icon
%(upper triangular) matrix, e.g. 2ex
\vbox to #1{\hrule
   \hbox{\diagline . #2 wd -#2 ht\vrule}%
   \vfil}%
}%end icurt
%
\def\icllt#1#2{%IConLowerLeftTriangle
%#1 is ht of icon matrix,
%with LT the upper triangular part,
%e.g. 4ex #2 is wd of icon
%(lower triangular) matrix, e.g. 2ex
\vbox to #1{\vfil
   \hbox{\vrule\diagline . #2 wd -#2 ht}%
   \hrule}%
}%end icllt
%
\def\icuh#1#2#3{%IConUpperHessenberg
%#1 is size of icon matrix,
%with UH the upper Hessenberg part, e.g. 4ex
%#2 is wd of icon (upper Hesenberg)
%matrix, e.g. 1ex
%#3 is size Lower Left triangular part, #1-#2
%(for simplicity the latter is added,
% could have been calculated, perhaps some
% inconsistency test could be incorporated)
\vbox to #1{\offinterlineskip\hrule%
   \hbox to#1{\vrule height#2 depth0pt
                   \relax\hfil\vrule}%
   \hbox to#1{\diagline . #3 wd -#3 ht
              \hfil\vrule}%
   \hbox to#1{\hfil \vrule width#2
                           height.2pt}%
   }%
}%end icuh
%
\item {Some matrix icons} (Wilkinson, 1965):
$$\eqalign{
\vcenter{\icmat{4ex}{4ex}}\kern2ex
   \vcenter{\icllt{4ex}{4ex}}
&=\kern1ex
   \vcenter{\icllt{4ex}{4ex}}
   \vcenter{\icuh{4ex}{1ex}{3ex}}
&\quad{\rm or} \quad AL=LH \cr
\noalign{\vskip2ex}
\vcenter{\icmat{6ex}{3ex}}\kern1ex
&=\kern1ex\vcenter{\icmat{6ex}{3ex}}
  \kern1ex\vcenter{\icurt{6ex}{3ex}}
&\quad{\rm or} \quad A=QR\cr
}$$

%Rhombus scheme
\newbox\ru %
\newbox\rl %
\setbox\ru=\hbox{%
   \diagline . 4ex wd +2ex ht\relax}%
\setbox\rl=\hbox{%
   \diagline . 4ex wd -2ex ht\relax}%
%
\item {Rhombus scheme}
(Schwarz, et al., 1972:166):

$$\displaylines{%\indent
\vbox{\offinterlineskip\tabskip=0pt
\halign{\hfil$##$%left element
&\hfil$\vcenter{#}$\hfil%left lines
&\hfil$##$\hfil         %middle elements
&\hfil$\vcenter{#}$\hfil%right lines
&$##$\hfil              %right elements
\cr                     %end template
1^{st}{\rm RS}\hfill
         & &e^{(s)}_k&&&    \cr
&\copy\ru& &\copy\rl&       \cr
q^{(s+1)}_k&&&&&q^{(s)}_{k+1}\cr
&\copy\rl& &\copy\ru&       \cr
&     &e^{(s+1)}_k
   &\omit$={q^{(s)}_{k+1}\over q^{(s
+1)}_k}\,e^{(s)}_k$\hfil\hidewidth\cr
\noalign{\vskip1ex}
2^{nd}{\rm RS}\hfill
         & &q^{(s)}_k&&     \cr
&\copy\ru& &\copy\rl&       \cr
e^{(s+1)}_{k+1}&&&&&e^{(s)}_k\cr
&\copy\rl& &\copy\ru&       \cr
&     &q^{(s+1)}_k
   &\omit$=q^{(s)}_k+(e^{(s)}_k-
    e^{(s+1)}_{k+1})$\hfil\hidewidth
                      \cr}%end halign
}% end vbox element
%CGL insertion of qquads will shift
%CGL the box to the left
\qquad\qquad  \qquad
}% end displaylines
$$

\item {Continued fractions}:
```

```
\gdef\cf{\mathop{\grkop \Phi}}
{\newskip\centering
\centering=0pt plus1000pt minus 1000pt
$$\eqalignno{
1+\cf_{k=1}^n{a_k\over b_k}
&{}\buildrel{\rm def}\over=
 1+{a_1\over\displaystyle b_1+
   {\strut a_2\over\strut
    \vrule height3ex width0pt\relax
    \displaystyle b_2 +
     \lower2.0ex\hbox{$\ddots\,
      \lower1.25ex\hbox{$+
      {\displaystyle a_{n-1}\over
        \displaystyle b_{n-1}+
                 {\strut a_n\over
           \displaystyle b_n}}$}
                        $}
    }
   }\cr
\omit {\rm with (space saving)
     variant notations:}\hidewidth\cr
&{}\buildrel{\rm\phantom{def}}\over=
1+{a_1\,
\smash{\vrule depth1ex}\vrule height2ex
\over\strut\vrule\,b_1}+{a_2\,
\smash{\vrule depth1ex}\vrule height2ex
\over\strut\vrule\,b_2}+\cdots+{a_n\,
\smash{\vrule depth1ex}\vrule height2ex
\over\strut\vrule\,b_n}\cr
%
&{}\buildrel{\rm\phantom{def}}\over=
1+
{a_1\over\textstyle\strut
    \vrule height2.5ex width0pt
    b_1\,+\,}
{a_2\over\textstyle\strut
    \vrule height2.5ex width0pt
    b_2\,+\,}
    \cdots
{a_n\over\textstyle\strut
    \vrule height2.5ex width0pt
    b_n}
\cr}%end\eqalignno
$$}%
%
\item Reduction to Hessenberg form via
lower triangular similarity transformation
(Wilkinson, 1965:357):
$$\displaylines{\indent
\bordermatrix{&      &\rm A &      \cr
               &\times&\times&\times\cr
               &\times&\times&\times\cr
               &\times&\times&\times\cr}
\bordermatrix{& &\rm N & \cr
               &1&      & \cr
               &0&1     & \cr
               &0&\times&1\cr}\hfill\cr
\hfill=
\bordermatrix{& &\rm N & \cr
              &1&      & \cr
              &0&1     & \cr
              &0&\times&1\cr}
\bordermatrix{&      &\rm H &      \cr
              &\times&\times&\times\cr
              &\times&\times&\times\cr
              &0     &\times&\times\cr}
}$$
%
\item {Partitioning} (Wilkinson, 1965:291):
$$P_r=\left(\vcenter{
    \offinterlineskip\tabskip0pt
    \halign{
      \vrule height3ex depth1ex width 0pt
      \hfil$\enspace#\enspace$\hfil
      \vrule width.1pt\relax
      &\hfil$\enspace#\enspace$\hfil\cr
      I_{n-r}&0\cr
      \noalign{\hrule height.1pt\relax}
      0      &I-2v_rv_r^T\cr}
          }\right)           $$
%
\item Braces and matrices (Wilkinson, 1965:199):
$$\vcenter{
  \hbox{${\scriptstyle\phantom{n{-}}p}
    \left\{\vrule height4.5ex width0pt
         depth 0pt\right.$}\vglue3ex\relax
  \hbox{${\scriptstyle n{-}p}
    \left\{\vrule height2.5ex width0pt
         depth 0pt\right.\kern2pt$}
         \vglue1ex\relax
      }
\bordermatrix{&\multispan4{\enspace\hfil
$\overbrace{\vrule height0pt width10ex
   depth0pt}^p$}\hfil
              &\multispan3{\enspace\hfil
$\overbrace{\vrule height0pt width7.5ex
   depth0pt}^{n-p}$}\hfil\cr
\noalign{\kern-.25\baselineskip}
&\times&\times&\times&\times&\times&
 \times&\times\cr
&0     &\times&\times&\times&\times&
 \times&\times\cr
&0     &0     &\times&\times&\times&
 \times&\times\cr
&0     &0     &0     &\times&\times&
 \times&\times\cr
&0     &0     &0     &0     &\times&
 \times&\times\cr
&0     &0     &0     &0     &\times&
 \times&\times\cr
&0     &0     &0     &0     &\times&
 \times&\times\cr
}$$

%
\item Matrices, braces, (dotted)
 partitioning and icons
 (space efficient variant):
```

```
%The simplest way is to make the 22-element
%separate, and measure the sizes.
%Subsequently one easily couples these
%sizes to the sizes of the braces.
%Hard things: automatic coupling,
%          vertical dotted lines,
%          inner use \noalign.

\def\vdts{\vbox{\baselineskip4pt
  \lineskiplimit0pt
  \vglue2pt\hbox{.}\hbox{.}\hbox{.}}}%
$$
\vcenter{\offinterlineskip%No interline
%                space in between parts
\halign{\hfil$#$&\hfil$#$\hfil\cr%2-column
%first row with braces, element 11 empty
{}&\hfil\enspace\mathop{\hbox to.9cm%
   {\downbracefill}}\limits^{\vbox{\hbox{
               $\scriptstyle p$}\kern2pt}}
        \enspace\hfil\mathop{\hbox to.6cm%
   {\downbracefill}}\limits^{\vbox{\hbox to
  0pt{\hss$\scriptstyle n-p$\hss}\kern2pt}}%
        \enspace\hfil\cr  % end first row
%Separation between first (border) row and
%second row
\noalign{\vglue1ex}
%first column with braces, 21 element
\vcenter{\vfil
   \hbox{${\scriptstyle p}\left\{\vbox
   to.8cm{}\right.$}\vfil\vglue2ex\vfil
   \hbox{\llap{$\scriptstyle n{}$}%
   ${\scriptstyle p}\left\{\vbox to.5cm{}
   \right.$}\vfil}
&%22-element is the matrix proper
\left(\vcenter{\offinterlineskip
\halign{\hfil$#$\hfil&\hfil$#$\hfil&
\hfil$#$\hfil&\hfil$#$\hfil
\tabskip=.5\tabskip&\vdts#&
\tabskip=2\tabskip
\hfil$#$\hfil&\hfil$#$\hfil&
\hfil$#$\hfil\cr%end template
\times&\times&\times&\times&&\times&
 \times&\times\cr
0     &\times&\times&\times&&\times&
 \times&\times\cr
0     &0      &\times&\times&&\times&
 \times&\times\cr
0     &0      &0      &\times&&\times&
 \times&\times\cr
\noalign{\vglue1ex}
\multispan8\dotfill\cr
0     &0      &0      &0      &&\times&
 \times&\times\cr
0     &0      &0      &0      &&\times&
 \times&\times\cr
0     &0      &0      &0      &&\times&
\times&\times\cr}%end halign (22)
}%end vcenter
\right)\cr %end 22-element
```

```
%Separation between last (border) row
%and previous (second row)
\noalign{\vglue1ex}
{}&\hfil\enspace\mathop{\hbox to.9cm{%
   \upbracefill}}\limits_{\vbox{\kern2pt
                      \icurt{4ex}{2ex}}}
   \enspace\hfil
   \mathop{\hbox to.6cm{%
   \upbracefill}}\limits_{\vbox{\kern2pt
     \icmat{4ex}{1.5ex}}}\enspace\hfil%
\cr  % end last row
}%end halign
}%end vbox
$$
%
%References
%Addison-Wesley. Micro-TeX.
%Wilkinson, J.H. (1963):
%   The Algebraic Eigenvalue problem.
%Swanson, E. (1986):
%   Mathematics into Type. AMS.
%Doob, M. (1989): Gentle TeX.
%Hendrickson, Amy (priv.comm)
%Schwarz, Rutishauser, Stiefel (1972):
%   Matrizen numeriek
%
\ei

}
```

Dialog with TEX

Michael J. Downes
49 Weeks Street
North Smithfield, RI 02895
mjd@math.ams.com

Introduction

On the face of it, of course, 'dialog with TEX' doesn't make much sense, because TEX isn't a person that can carry on a conversation. The truth is that a team of real persons, Knuth and the macro writer (or writers), have tried to anticipate the user's side of the conversation and prepare good answers in advance. It's these packaged answers, relayed to the user through TEX, that form the other side of the conversation.

Yet when we deal with a computer program such as TEX, our human tendency is to translate the pseudo-conversation, carried on by printed messages on the computer screen, into a more familiar framework: natural language conversations with real people. This is done easily enough by pretending that the program is a sort of genie (or lion, in the case of TEX) that happens to live inside the computer. This pretense is particularly convenient in writing about programs, where it helps cut down on awkward circumlocutions.

TEX can be run in batch mode or interactive mode, but the most frequent way of running TEX might best be called *barely interactive*: you start running TEX in interactive mode and give it a file name to process, whereupon TEX typesets the file and quits, without needing any further input from you — but you hang around anyway, in case an error occurs, because if so then you have to type something in response to the question mark prompt, before TEX can finish processing the file. If you don't expect any errors, however, you could go get a cup of coffee while TEX is running, or in the case of a long document maybe even go home and mow the lawn.

By *dialog with TEX*, then, I don't mean errorless typesetting runs where the presence of the user is immaterial; I mean two-way communication with the active participation of the user, not only in responding when TEX prompts for a response, but also in paying attention to any messages TEX may send, whether they require a response or not. More generally, I'll define *dialog with TEX*, for the purposes of this discussion, as *the communication of interesting information, in useful forms, between TEX and the user, while TEX is running.* Thus if you look at a printed document and see that TEX put a certain box in the wrong place, that is useful information, but it doesn't match my definition of dialog because the communication didn't take place while TEX was running.

To give another example, a table macro package written by Ray Cowan, that I encountered under the name `tabls.sty` (an adaptation for LATEX), has the unique feature that you don't have to type a preamble line setting up the format of the columns in a table. The format is determined automatically by the contents of the table. The number of columns are then reported on screen while TEX is running. I classify this as dialog, even though TEX doesn't stop to check for any response, because I believe Cowan primarily envisioned the number-of-columns message as being read by the user while TEX is running, to see if the reported number of columns matches the intended number of columns. In the case of a minor discrepancy the user can just make a mental note to check the input file later for proper syntax; but in the case of a serious discrepancy (*93 columns??! Whoa!*), the user could press the interrupt key to break out of TEX and go fix up the table, before trying again.

On a more practical level, *dialog with TEX* usually involves sending and receiving messages using the `\message`, `\write`, and `\read` commands. In *The TEXbook*, near the end of Chapter 20, Knuth writes "It's easy to have dialogs with the user, by using `\read` together with the `\message` command," and there follows a brief example involving reading the user's name into a macro `\myname`. It's clear from this passage that what Knuth means by "dialog" is the standard sort of programming tasks that involve providing information to the user, displaying menus, asking questions, and handling user responses. It's easy to identify a number of fairly obvious principles that should be followed when writing such dialog into a program:

[0] An extended version of this paper and some example macro files are available on request from the author.

1. When asking a yes/no question, the user should be able to enter `y`, `yes`, or even `ye`, in lowercase or uppercase, and have the answer understood to be "yes".
2. For any menu or question, a default answer should be provided (when this makes sense), and the default answer should be as easy as possible to select.
3. Users' answers should be repeated back to them, so that they can verify that the answer taken in by the program is indeed the answer that was typed.
4. The user should be given a chance to undo mistakes, e.g., by going back to a specified point earlier in the dialog and starting over from there.
5. When practical, users' answers should be checked to make sure they're not nonsense; for example, if the program requests an integer, it should check the response to make sure the user didn't enter something else entirely. An example of this is the LaTeX option file `checknum.sty` that was published by Brian Hamilton Kelly in UKTeX, vol. 1, no. 1 (4 January 1991) in response to a query.
6. When giving information to the user, it should be provided in the *best possible form*, where the meaning of *best possible* should be determined by common sense from the circumstances of a particular application and the targeted user group. For example, a straightforward use of the `\the` command to report the value of a TeX dimension register such as `\vsize` to the user will produce the value in points, down to five decimal places. For an average author it would usually make more sense to convert the value to inches or centimeters, whereas for a typographical designer or compositor it would usually make more sense to convert the value to picas, before it is reported to the user. Depending on the unit chosen, it should also be rounded to the nearest whole unit or tenth of a unit or something sensible that will avoid burdening the user with irrelevant precision.

It appears that Knuth's words "it's easy" weren't intended entirely literally, since the whole section where they appear is marked off with double dangerous bend signs; furthermore, the very next thing after the example mentioned above is Exercise 20.18 — marked with a double dangerous bend sign — which reads,

> The `\myname` example just given doesn't work quite right, because the ⟨return⟩ at the end of the line gets translated into a space. Figure out how to fix that glitch.

The line-ending space is only one of a number of complications that can hamper the efforts of macro writers to write dialog into their macros. The aim of this article is to provide solutions for some of the complications.

Basic capabilities of TeX for sending and receiving messages

Tables 1 and 2 list the various means in TeX for sending messages to the user, and for the user to reply.

Table 1

Sending
`\message`
`\errmessage`
`\write`
`\show`
`\showthe`
`\showbox`
`\showlists`

Table 2

Receiving	prompt displayed by TeX
`\read`	`\controlseq=`
error message interaction	?
show message interaction	?
input file not found	`Please type another input file name:`
interrupt key	none

Notice that there are a few related features of TeX, e.g., `\errhelp`, that have their place in communicating something to the user, but that are dependent on the commands listed in the table: for instance, the user won't normally see `\errhelp` except by way of `\errmessage`. It is merely a temporary storage area for help messages, rather than a function that can be used to make something happen.

The `\message` primitive The `\message` command is a TeX primitive that prints its argument on screen. If the current screen position is not at the beginning of a line, TeX will add a blank space at the beginning of the message text to separate it from the preceding material. If there isn't enough room on the current line to fit the entire message text, then TeX will go to the next line before starting to print the message. If a message is more than one line long, and the macro writer does nothing to break it up into shorter pieces, TeX will break it up without regard to the contents of the message, even splitting words, using the maximum number of characters allowed per line (*max_print_line*, compiled into TeX) as the only line-breaking criterion. To obtain better line breaks, the macro writer can use the current newline character, determined by the value of `\newlinechar`, provided that it is a printable character. It is an idiosyncrasy of TeX that control characters, such as the `^^J` that is the default newline character in AMS-TeX and LaTeX, cannot be used to produce new lines in the argument of a `\message` command, whereas in the argument of a `\write` command they work fine.

The `\errmessage` primitive The `\errmessage` command prints its argument on screen, starting on a new line, with an exclamation point and a space added at the beginning, and a period added at the end. In other words, the command `\errmessage{Surprise}` produces

```
! Surprise.
```

on screen. Actually it produces more than that — it also shows the current context, which means the current line from the current input file, along with the line number, and additional information if there is any (such as the surrounding parts of current macro expansions).[1] Newline characters in the argument of `\errmessage` operate the same as for `\message`.

`\errrmessage` is also noteworthy for the error recovery choices offered by TeX at the ? prompt. Among other things, choosing the 'h' option at the ? prompt will cause TeX to print on screen the current contents of the token register `\errhelp`.

The `\write` primitive The `\write` command, like `\message`, basically just prints a message on screen. But communication with the user is not the primary purpose for which `\write` was designed. Its primary purpose is saving index or table of contents information, with the associated page numbers, in a separate file for later processing. Because this kind of use is closely linked to page numbering, `\write` commands on the current page are normally saved up and only executed when the page is actually shipped out, i.e., after the actual page break has been determined. To avoid such postponement, `\write` must be used with the `\immediate` prefix. But don't forget completely the link between `\write` and `\shipout`, because sometimes it's useful to leave off the `\immediate`. For instance, if you are working on page breaks in a long document and want to find out, without previewing or printing, if a nonforcing pagebreak command had the effect that you wanted, you could insert a non-immediate `\write16` just before and just after the intended page break:

```
\write16{Before the attempted pagebreak.}
\penalty-9999
```

(or, in LaTeX, `\pagebreak[3]`)

```
\write16{After the attempted pagebreak.}
```

The message from a non-immediate `\write16` will appear just before the closing `]` of the `[]` pair that enclose the relevant page number. So if all went well, one message will appear with one page number and the next message with the next page number, like this:

```
[4] [5
Before the attempted pagebreak.
] [6
After the attempted pagebreak.
] [7] [8] [9] ...
```

The `\show` and `\showthe` primitives The `\show` command, used for showing the current meaning of a control sequence (or indeed of any valid token), is rather similar to the `\errmessage` command in what it produces on screen. The prefix is a greater-than character instead of an exclamation point. Here's the result of `\newcount\C \show\C`:

```
> \C=\count78.
l.1 \newcount\C \show\C

?
```

As with `\errmessage`, TeX displays the surrounding context of a `\show` command; it also offers the same error recovery opportunities, except that you can't access `\errhelp` through the 'h' option, after a `\show` command.

The `\showthe` command is like `\show`, but is applied to certain kinds of things such as the names of count registers and token registers, that have not only a meaning but also a current value. For instance, here's the result of `\C=5 \showthe\C` (using the counter we defined earlier):

```
> 5.
l.3 \C=5 \showthe\C

?
```

The `\showbox` and `\showlists` primitives The commands `\showbox` and `\showlists` are similar to `\show` in what they produce on screen. Because of their specialized nature they don't have much relevance to the main theme of this paper, but once

[1] If the parameter `\errorcontextlines` is set high enough.

again don't forget about them entirely, because in certain narrow applications they might be just the ticket.

Error recovery

After an error message or a \show command, the user is presented with a question mark prompt. Typing a second question mark in reply to the prompt will cause TEX to list the options that are available:

```
   ...
? ?
Type <return> to proceed, S to scroll
                future error messages,
R to run without stopping, Q to run
                               quietly,
I to insert something, E to edit your
                                 file,
1 or ... or 9 to ignore the next 1 to
                     9 tokens of input,
H for help, X to quit.
?
```

Because their main use is in recovering from errors, it is convenient to call these *error recovery options.* The insertion and token-skipping options, however, are potentially useful for other things besides error recovery.

An example of a typical use of these error interaction possibilities is given in Example 1.

Example 1: Example of using error interaction possibilities to get past a potentially bad error: a missing \\ before an \hline in a LATEX tabular environment.

```
! Misplaced \noalign.
\hline ->\noalign
                  {\ifnum 0=`}\fi \hru...
l.120 \hline

?
```

Let's see what the help information is.

```
? h
I expect to see \noalign only after the
\cr of an alignment. Proceed, and I'll
ignore this case.

?
```

Let's try skipping one token to verify what LATEX is going to process next:

```
? 1

\hline ->\noalign {
                   \ifnum 0=`}\fi \hru...
l.120 \hline

?
```

All right, the opening curly brace has just gone by, so we are indeed at the beginning of the definition of \hline. We need to insert the \\ that was forgotten, and also replace the two tokens \noalign and { that have slipped by.

```
? i\\ \noalign{
```

Notable examples of dialog with TEX

To flesh out my definition of *dialog with TEX* let me give some examples from widely available macro files. This will also help to illustrate some of the typical difficulties.

Comment in hyphen.tex

The standard **hyphen.tex** containing U.S. English hyphenation patterns has a comment after the \patterns command:

```
\patterns{ % just type <return> if
           % you're not using INITEX
```

(Here I have split the comment into two lines because of the narrow column width, but in the original, the comment is all on the same line.) Ordinarily the macro writer can't use comments to communicate with the user, because comments within the text of a macro disappear as the macro is defined. The beauty of the comment in **hyphen.tex** is that it appears precisely when needed, because of the way TEX displays context with error messages: if you **\input hyphen.tex** when not using INITEX, TEX will give an error message when it encounters the \patterns command, and as usual, will show the context around the point of the error, like this:

```
! Patterns can be loaded only by INITEX.
l.2 \patterns
             { % just type <return> ...
?
```

This idea could be useful in other applications. For example, the file **lfonts.new** of the Mittelbach/Schöpf font selection scheme (LATEX version) has a statement of the form **\input fontdef.tex**, where the file **fontdef.tex** is often missing (intentionally) and the user is supposed to substitute another file name such as **fontdef.ori** or **fontdef.max**. A comment on the same line as the **\input** statement, listing the alternate possibilities, could save users a certain amount of flipping pages in the documentation:

```
\input fontdef.tex % Other possibilities:
                  % fontdef.ori, fontdef.max
```

Amstex.tex: \printoptions Example 2 shows the implementation in AMS-TeX of a `\printoptions` command. This allows the user to choose a syntax check or 'galleys' run, instead of a full typesetting run, the advantage being an increase in processing speed. But our primary interest at the moment lies in the techniques used to present three choices to the user and read the reply.

This example shows one way of dealing with the extra space at the end of a macro created using `\read`: define some macros consisting of the expected answers, with the extra space included, and then use `\ifx` to compare them to the user's response. It also shows how to uppercase the user's response so that lower- and uppercase responses will both be treated the same. The method used is the second method given in the answer to *The TeXbook*'s Exercise 20.19. One more noteworthy feature of `\printoptions`: it runs a loop that doesn't quit until the user gives an acceptable answer.

Other examples

- `latex.tex`: `\typeout` and `\typein` are examples of the kind of basic dialog tools that can be built into a macro package.
- `docstrip.tex`: The `docstrip.tex` utility by Frank Mittelbach is used to strip out comment lines from a documented macro file. It provides 'progress reports' in the form of a message containing a single `%` or `.` character, for each line as it is processed. This produces rows of percent signs and periods in a random pattern across the screen, as the documentation stripping process chugs along, which helps to alleviate the monotony of processing a large file.
- `checknum.sty`: It is often useful to check replies from the user to make sure they're valid. Brian Hamilton Kelly's checknum.sty (posted to UKTeX vol. 91 no. 1 (4 January 1991)) illustrates a technique for reading an integer from the user and making sure they did indeed enter an integer and nothing but an integer.
- `animals.tex`, `basix.tex`: These two files by Andrew Marc Greene (the former published in *TUGboat* 10, no. 4 as part of *TeXreation — Playing games with TeX's mind*, the latter in *TUGboat* 11, no. 3 as *BaSiX: An interpreter written in TeX*) are examples of macro files whose whole purpose is carrying on dialog.
- `Testfont.tex`: This file, written by Knuth for his own use in testing new fonts produced by METAFONT, contains a `\help` command — something that would probably be a good idea for every macro package.

Use of `\message` and `\immediate\write`

Any expandable control sequences in the argument text of a `\message` or `\write` command will be expanded. Consider Table 3. This expansion is usually useful, but occasionally it can be a hindrance, as for example if you want to include a ~ in the text. And generally speaking, if you want to mention any control sequence in the argument text, you'll have to use `\string` before the control sequence (and frequently `\space` after it, as well). Table 3 also illustrates the utility of `\noexpand` for this purpose (something which was pointed out to me by Michael Spivak).

Prompting and reading input

Let's return to the AMS-TeX `\printoptions` example now. Since `\W@` is defined to be `\immediate\write16`, and we know from the earlier discussion that the `\write` command always starts a new line after its message text, we can see that the reply typed by the user will appear on the new line instead of immediately after the colon. This brings to mind the question: what can we do if we want the user's reply to appear on the same line?

The way to do this, in general, is by using `\write` to send all but the last line of a prompt message, and use `\message` to send the last line. (Brian Hamilton Kelly's `checknum.sty` uses this idea.)

```
\W@{Do you want S(yntax check), G(alleys)
  or P(ages)?}%
\message{Type S, G or P, follow by
  <return>: }%
```

Dealing with the Control-M/space character at the end of a \read macro In `\printoptions` a separate macro `\S@`, `\G@`, or `\P@` is defined for each legitimate response. If the menu becomes more extensive, this technique is rather wasteful of hash size, main memory, and other useful commodities. The whole problem here is that `\read` includes the Control-M character at the end of the user's response in the macro being read. Under normal conditions Control-M is converted to a space, of course, but another possibility — if the user just presses Return without typing any response — is that the Control-M will produce a `\par` token (following the general rule that an empty line is equivalent to `\par`). The best approach (IMHO) is to prevent the Control-M character from getting into the read macro in the first place. This can be done in two

Example 2: \W@ *is the* $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX *abbreviation for* \immediate\write16

```
\def\S@{S } \def\G@{G } \def\P@{P }
\newif\ifbadans@
\def\printoptions{\W@{Do you want S(yntax check),
  G(alleys) or P(ages)?^^JType S, G or P, follow by <return>: }\loop
 \read\m@ne to\ans@
 \xdef\next@{\def\noexpand\Ans@{\ans@}}\uppercase\expandafter{\next@}%
 \ifx\Ans@\S@\badans@false\syntax\else
 \ifx\Ans@\G@\badans@false\galleys\else
 \ifx\Ans@\P@\badans@false\else
 \badans@true\fi\fi\fi
 \ifbadans@\W@{Type S, G or P, follow by <return>: }%
 \repeat}
```

ways: setting the catcode of `^^M` to 9 ('ignore'), or setting `\endlinechar` to −1.

But that immediately raises another difficulty: we want to keep the catcode change or `\endlinechar` change local so that it will affect only the `\read`. This could be accomplished by saving the current catcode or `\endlinechar` (just in case) and restoring it after the `\read` is done, but it's simpler to enclose the `\read` in a group:

```
\begingroup
\endlinechar=-1
\global\read16 to\answer
\endgroup
```

Here the `\global` prefix is necessary in order for `\answer` to be properly defined when the group ends.

The tests in `\printoptions` can, with this modification, be simplified to:

```
\if\Ans@ S ...\else
\if\Ans@ G ...\else
\if\Ans@ P ...\else
...
```

and the macros `\S@`, `\G@`, `\P@` are now totally unnecessary. On the other hand, we have advanced to some splendid new complications: `\Ans@` might now be completely empty, if the user just pressed the Return key, and an empty `\Ans@` would bollix up the `\if` tests. This case is easy to handle, though: add an extra branch `\ifx\Ans@\empty...` at the appropriate spot. We have the opposite problem if the user typed more than one letter: on the true branch the extra characters will most likely cause spurious typesetting activity. As it happens, we can kill two birds with one stone, as we'll see shortly when we discuss default responses.

Uppercasing input Next let's look at the procedure used by `\printoptions` for uppercasing the user's reply: after reading `\ans@`, `\xdef` and `\uppercase` are applied to it as follows:

```
\xdef\next@{\def\noexpand\Ans@{\ans@}}%
\uppercase\expandafter{\next@}%
```

I prefer a slightly more economical version of the same technique:

```
\xdef\ans@{\uppercase{%
  \def\noexpand\ans@{\ans@}}}%
\ans@
```

This may be a bit confusing at first sight. If `\ans@` contains 's' to begin with, then after the `\xdef` has been completed, the definition of `\ans@` is `\uppercase{\def\ans@{s}}`. Then calling `\ans@` causes it to redefine itself, but not before the tokens in the argument of `\uppercase` are suitably uppercased. (Only the 's' is affected because the other tokens are control sequences or nonletters.)

Notice that the auxiliary macros `\Ans@` are no longer needed. To simplify the structure of macros using this uppercasing process, it could be embodied in its own macro:

```
\def\uppermac#1{\xdef#1{%
  \uppercase{\def\noexpand#1{#1}}}%
  #1}
```

Default responses One last refinement in `\printoptions` would be to provide a default response if the user's response is empty (i.e., they just pressed the Return key). The method I like involves an auxiliary macro like the LATEX macro `\@car`:

```
\def\@car#1#2\@nil{#1}
```

But since most of us are probably not particularly well acquainted with the arcane terminology of Lisp, let's call this macro `\firsttoken` instead:

```
\def\firsttoken#1#2@{#1}
```

(Using `@` as the ending delimiter is pretty safe if we make sure that it has catcode 11 at the time

Form Letters with 3-Across Labels Capability

Jackie Damrau
Superconducting Super Collider Laboratory
Dallas, Texas, 75237 USA
214-708-6048; FAX: 214-708-5143
Internet: damrau@sscvx1.ssc.gov

Michael Wester
Department of Mathematics and Statistics
University of New Mexico
Albuquerque, New Mexico, 87131 USA
505-277-4613
Internet: wester@spectre.unm.edu

Abstract

This article discusses a general-purpose program for generating form letters, using either TeX or LaTeX. Given three inputs: a preamble file for initializations, a list of blank separated addresses, and a letter template, this program can be used to generate a letter per address and provide personalizations as directed by the template. Sample applications are presented, including one which constructs 3-across mailing labels. Thus, both form letters and mailing labels can be generated from the same list of addresses by simply changing two inputs to the program.

Introduction

With TeX or LaTeX, it is not hard to produce a letter to be sent to a single addressee. Nor is it difficult to create multiple letters that follow a similar format by setting up a form in which changeable parameters, such as the name and address, are specified by macros. The form can then be input a fixed number of times, each time preceded by a redefinition of the parameters. However, there are problems with this approach. Modifying the list of addresses or adding new parameters to the form can be cumbersome. Also, serious reformatting may be required to use the individual pieces of information (such as the names and addresses) in other contexts.

An easy-to-use, general-purpose program to generate form letters has been developed that overcomes the above problems. This program, address, is written using TeX constructs and macros and can be executed by TeXing or LaTeXing it.

The address program requires three user-supplied files: a preamble for performing initializations (which is optional), a list of addresses separated by blank lines, and a template. On execution, address asks for three file names, and then reads in the addresses one by one. For each address, the individual components are assigned to various macros, after which the template file is `\input`. The template can, therefore, refer to these macros.

The address list need not contain any formatting instructions as individual lines of text within a given address are retained by address. This allows the address file to be pure text and usable by any other program. Nor is it required that an address be simply that; any other data, such as telephone numbers, test scores, or whatever, may be included. Macros are provided to extract both individual lines and individual blocks of data.

In the following sections, instructions on how to set up the necessary files and details on running the program are furnished. In addition, a template for producing 3-across labels is provided which demonstrates a simple, but useful application. Finally, some discussion of the construction of the macros in address is given, explaining some of the difficulties encountered and how they were overcome.

Setting Up the Files

The easiest file to set up is the address list. It consists of blocks of text separated by blank lines. Commonly, the first line of a block will be the name of an addressee, while the rest of the lines form the address. The address program therefore assigns these segments of text to the macros \Name and \Address, respectively.

Unlike normal TeX input, individual lines of text within an address (which can be of arbitrary length) retain their identity. This is accomplished by concatenating the lines, using \\ as a separator. For LaTeX, this is quite convenient since a reference to \Address will result in the expansion of each \\ as a new line. In TeX, a similar effect can be obtained by

```
\def\\{\hfil\break}
```

In the template file, individual lines within \Address can be selected by \AddrLine{n}, where n is a positive integer. In addition, \Laddr will count the number of lines in \Address and \Naddr will refer to the current position in the address list.

It is often desirable to include other information along with the address. The easiest way to do this with address is to divide the lines of text in the address segment into subblocks, each of which can be of variable length. The macro call, \AddrBlock{k}, is provided to select the k^{th} subblock, where a line consisting of --- acts as a separator between subblocks. (Two consecutive lines of --- will not produce an empty subblock, but this effect can be achieved by inserting a line consisting of {} between the lines.)

To extract individual lines from a subblock of \Address, a two-step process is required. \StoreAddrBlock{k}\in\Block will define the contents of the macro \Block to be the k^{th} subblock of \Address. \GetLine{n}\of\Block will then select the n^{th} line of the \Block. This procedure is necessary to circumvent some peculiarities of TeX macro expansion.

One of the more salient features of address is the ability to intelligently parse the \Name and break it up into its components. For example, suppose the name is The Honorable and Mrs. Henry \& Matilda Edward Bo van Frothingham III, Royals. The following macro assignments will then be made when this name is parsed:

\SocialTitle	→ The Honorable and Mrs.
\FirstName	→ Henry \& Matilda
\MiddleName	→ Edward Bo
\LastName	→ van Frothingham
\Suffix	→ III
\OtherTitle	→ Royals

Simpler names will result in some of the above macros having null definitions.

The macros associated with the address program are summarized in the following table.

\DEFAULTtolist	default address list (initially, tolist)
\DEFAULTletter	default letter template (initially, letter)
\Name	first line of the address
\Address	subsequent lines of the address
\SocialTitle	e.g., Dr., Mr., Ms.
\FirstName	first name
\MiddleName	middle names
\LastName	last name
\Suffix	e.g., Sr., Jr., III
\OtherTitle	academic and professional titles
\Naddr	position of the address in the tolist
\Laddr	number of lines in \Address
\AddrLine{n}	n^{th} line of \Address
\AddrBlock{n}	n^{th} subblock of \Address
\StoreAddrBlock{n}\in\B	store the n^{th} subblock of \Address in \B
\GetLine{n}\of\B	n^{th} line of \B
\addspace\A	adds a space after \A if it is not null
\topbox{H}{W}{text}	top aligned box of height H and width W

An example of an address list appears below. Notice that following the address in each instance is a separate block of two lines.

```
Mrs. Apple Thesaurus
Apt. Z
234 Gestalt Lane
Cockermouth, Umbria, U.K.
---
artichokes
jalape\~nos

Harry K. Banana
P.O. Box 29Z46
Kahului, Maui, Hawaii
---
mustard greens
okra
```

A typical (LaTeX) letter template that might be used with the above list of addresses looks like:

```
\Name\\
\AddrBlock{1}

\StoreAddrBlock{2}\in\veggies
Dear \addspace\SocialTitle\LastName:

Welcome to the vegetable of the month
club!  Your introductory offer of
two selections this first month are
\GetLine{1}\of\veggies\ and
\GetLine{2}\of\veggies.  ...
\newpage
```

The macro `\addspace` is used to produce correct spacing in the salutation by adding a proper space after `\SocialTitle` if it contains text and doing nothing if `\SocialTitle` is empty (as would be the case for the second addressee).

The last file needed by address is the preamble. This file is `\input` once and is used to perform initializations, such as setting margins and defining macros. If address is being LaTeXed, the preamble is input before the `\begin{document}` that is executed automatically before processing the addresses. A sample preamble that can be used under TeX or LaTeX is given below. (The definition for `\ifundefined` can be found in Knuth [page 40] and is present in address.)

```
%
% Generic TeX/LaTeX preamble for
%    address.tex .
%
\ifundefined{LaTeX}                 % TeX
   \magnification=\magstep1
   \voffset=0in
   \hoffset=0in
   \vsize=9in
   \hsize=6.5in
   \nopagenumbers
   %
   \def\\{\hfil\break}
   \def\newpage{\vfil\eject}
\else                               % LaTeX
   \documentstyle[12pt]{letter}
   \topmargin 0in
   \headheight 0in
   \headsep 0in
   \oddsidemargin 0in
   \textheight 9in
   \textwidth 6.5in
   \pagestyle{empty}
\fi
```

Running the Program

The address program is executed simply by typing *tex address* or *latex address.* This action produces the following set of requests:

```
Enter the filename of the preamble
   [preamble.tex]:
Enter filename of recipients' addresses
   [tolist.tex]:
Enter the filename of the letter
   template [letter.tex]:
```

The filenames in square brackets (`[ ]`) are default values and are accepted by pressing a RETURN. (The default names for the second and third files can be changed by redefining the macros `\DEFAULTtolist` and `\DEFAULTletter` in the preamble.) Of course, the various files should contain commands appropriate to the package actually being used.

address is designed to be reasonably robust. This and TeX's rules for reading input allows some sloppiness in setting up the address list. For example, leading and trailing white space and extra blank lines are all ignored. Also, a `%` can be used to comment-out text. This last item implies that any line with a `%` as its first nonblank character will be treated as a blank line.

Making 3-Across Mailing Labels

By taking advantage of the macros defined in address, it is not difficult to design a template that can produce 3-across labels. The following template will create a three-column, 33 labels-per-page format for a standard sheet of 2.75"×1" labels. The resulting output can be sent directly to sheets of labels or onto regular paper, which can then be photocopied onto label sheets.

```
\ifcase\the\Naddr
  \or\topbox{1in}{2.75in}{\Name\\\AddrBlock1}%
  \or\topbox{1in}{2.75in}{\Name\\\AddrBlock1}%
  \or\topbox{1in}{2.75in}{\Name\\\AddrBlock1}\\
     \Naddr=0
\fi
```

The methodology used here is to produce differing output depending on the value of `\Naddr`. The `\ifcase` construct will result in three horizontally aligned boxes, each containing the current contents of `\Name` and the first subblock of `\Address`, as `\Naddr` takes on the values 1, 2 and 3. At the end of the third case, a new line is started and `\Naddr` is reset to 0. Thus, the next time around, address will have incremented `\Naddr` back to 1 and the process will start over. The macro `\topbox{`*H*`}{`*W*`}{text}`, defined in address by

```
\def\topbox#1#2#3{\leavevmode
  \vtop to #1{\hsize=#2 #3\vfil\eject}}
```

will produce a top-aligned box containing the text of height *H* and width *W*.

The above strategy is easily modified to handle any number of columns and any spacing requirements for a regular gridlike pattern of labels. Depending on the printer settings, the top and left margins established in the preamble (not shown) may need to be changed as well. Using 12-point fonts, it is possible to put 6 lines of text in a one-inch-high label. This can be increased by decreasing the fontsize set in the preamble.

Comments on other labeling schemes. A label-making capability is already available in LaTeX, as well as in other programs; but these do not possess the flexibility nor the ease of use exhibited by address. According to Lamport [page 67], the \makelabels command prints a "list of mailing labels, one for each letter environment, in a format suitable for xerographic copying onto 'peel-off' labels." However, the two-column format produced does not correspond nicely to 3-across and other common mailing label arrangements nor can it be changed easily.

In printing 3-across mailing labels in Microsoft Word, it is necessary to type three sets of field names, a NEXT instruction telling Microsoft Word to place information from more than one record onto a single copy of a form document, and various other commands and formatting statements. Typically, in programs of this type, a number of steps will be required; and again, the choice of output formats will be quite limited. Also, the data typically cannot be a simple ASCII file but must be converted into the program's possibly multifield internal format.

Comments on the Code

In developing address, certain difficulties had to be overcome. The solutions found may be of benefit to other users. One problem encountered was creating a box of text that had a definite height, as well as a definite width. The \topbox macro mentioned above has these features. It is adapted from the definition of LaTeX's \parbox command. With respect to TeX's viewpoint, the *H* in the definition of \topbox is really a depth with the height of the box being zero, but these details can normally be ignored.

A second obstacle was obtaining sequential space delimited strings (e.g., words) from a line of text in a robust manner. Macros to do this were needed to build the name-parsing macro (\breakup). For example, suppose \List is defined by \def\List{ a1 b2 c3 }. The first 'word' of \List is a1, the second is b2, and the third is c3. One way to select elements in this fashion is to construct macros, \wcar and \wcdr, that are analogous to the Lisp functions, CAR and CDR. \wcar\List\nil should select a1 and \wcdr\List\nil should return {b2 c3 }. Moreover, \wcar and \wcdr of { } and {} should be null, and any sequence of multiple spaces should be treated like a single space.

If \List is a simple list of tokens (for example, { a b c }), then a token CAR and CDR can be defined as follows:

```
%
% Test for {}.
%
\def\ifnull#1{\ifx#1\empty}
%
% \tcar\List\nil picks off the first non-
%    blank token (which is typically a
%    character or a control sequence) in
%    the \List.  If the \List is blank or
%    empty, then a null string is returned.
%
\def\tcar#1\nil{\ifnull#1
                  \empty
                \else
                  \tCar#1\nil
                \fi}
\def\tCar#1#2\nil{#1}
%
% \tcdr\List\nil removes the first nonblank
%    token in the \List and any preceding
%    blanks.  If the \List is blank or
%    empty, then a null string is returned.
%
\def\tcdr#1\nil{\ifnull#1
                  \empty
                \else
                  \tCdr#1\nil
                \fi}
\def\tCdr#1#2\nil{#2}
```

The general case is trickier and requires auxiliary macros.

```
%
% \ABDReverseExpand{D}{C}{B}{A}
%    first expands A, then expands B, then
%    expands D.
%
\def\ABDReverseExpand#1#2#3#4{%
   \expandafter\expandafter
      \expandafter#1%
         \expandafter\expandafter
            \expandafter#2\expandafter#3#4}
%
% Used to remove leading spaces.
```

```
%
\def\pretrim.#1{#1}
%
% \wcar\List\nil picks off the first word
%    (string of nonblank characters) in the
%    \List.  If the \List is blank or
%    empty, then a null string is returned.
%
\def\wcar#1\nil{%
   \ifnull#1
      \empty
   \else
      \expandafter\wCar\pretrim.#1 \nil
   \fi}
\def\wCar#1 #2\nil{#1}
%
% \wcdr\List\nil removes the first word and
%    any preceding blanks from the \List.
%    If the \List is blank or empty, then a
%    null string is returned.
%
\def\wcdr#1\nil{%
   \ifnull#1
      \empty
   \else
      \ABDReverseExpand
            \ifx\empty\wCdr\pretrim.#1 \nil
         \empty
      \else
         \expandafter\wCdr\pretrim.#1\nil
      \fi
   \fi}
\def\wCdr#1 #2\nil{#2}
```

\ABDReverseExpand is a simplification of the example found in Knuth [page 374] in the "Dirty Tricks" appendix.[1]

One more pair of useful Lisplike macros are \setq and \gsetq, which are defined by

```
\def\setq#1#2{\edef#1{\expandafter#2}}
\def\gsetq#1#2{\xdef#1{\expandafter#2}}
```

These macros allow statements such as

```
\setq\List{\wcdr\List\nil}
```

to function correctly by performing an immediate expansion on the second argument. This particular example results in \List being redefined to be {b2 c3 }.

Using the preceding as building blocks, it is easy to devise more complex macros. address defines \wmember\Element\of\List\nil, which causes \ifwmember to be true if the \Element is found in the \List (false otherwise); and \wendcarcdr\List\nil\A\B, which assigns \A and \B to the CAR and CDR of the \List, starting at the *right*. The assignment to macros provided as arguments in the latter case is done for two reasons. It is more efficient in this situation to make both assignments at once. More importantly, since \wendcarcdr uses recursion via the \loop construct, as well as defining temporary variables, TeX will complain if an attempt to force an immediate expansion of the result is made with \edef. Thus, \wendcarcdr uses \xdef internally to define \A and \B.

The problem of being unable to store and further manipulate the results of certain macro expansions can be solved in a second way. The \StoreAddrBlock macro mentioned earlier is defined by

```
\def\StoreAddrBlock#1\in#2{%
   {\setbox0=\hbox{\AddrBlock#1}%
    \toks0=\expandafter{\Current}%
    \xdef#2{\the\toks0}}}
```

In the first line, \AddrBlock completely expands within the \hbox and the result is assigned to a box which is subsequently ignored. As a side effect of the expansion, the global macro \Current is set to be the value of the reference to \AddrBlock. The last two lines of the macro then store this result in the second argument, operating with a token list to prevent premature expansion of any \\'s that may be present. The extra set of braces ensure that the assignments to \box0 and \toks0 are local to the macro.

This section concludes with a small, but important point. TeX will append an end-of-line character to any line of input text unless \endlinechar=-1 is performed. This character will be treated like a space unless the line is blank, in which case it will be converted into a \par. Since \ifx performs one level of macro expansion on its arguments, one way to read a line from a file and test if it is blank is:

```
\read\file to \Line
\ifx\Line\blank
```

where \def\blank{\par} must be done somewhere previous.

Summary

We have designed a general-purpose form-letter generator that runs directly under TeX or LaTeX. This program requires three files, supplied by the user: a (optional) preamble, a simple format address list, and a letter template. These files are prompted for interactively, and they default to certain names if none are specified. We have also developed various sample files, including a preamble and template

[1] The comment there incorrectly predicts that such a construction is "probably too lengthy to be of any use."

to produce 3-across labels. Thus, form letters and mailing labels can both be generated from the same address list by just changing two inputs to the program. Of course, there is nothing magical about names and addresses; this program can certainly be adapted to other uses.

This form-letter program has been designed to be both robust and easy to work with. The former means, for example, that excess white space and lines commented out with a % in the address list will not cause havoc. To make the program easy to work with, the files have been designed to be easily modified for various types of results. Thus, the files themselves have been made READABLE so that anyone who wishes to alter them may do so easily. These files will be made publicly available from the TEX repositories at sun.soe.clarkson.edu and ymir.claremont.edu.

Acknowledgments

Thanks are due to Ellen Golden and Sam Matthews for providing certain crucial help during the development of this program. Also, Tom Stickels is due thanks for the encouragement and sound editing advice provided throughout the development of this paper. Finally, hats off to the reading club for instigating this endeavor.

Bibliography

Knuth, Donald E. *The TEXbook*. Reading, Mass.: Addison-Wesley, 1984.

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

Microsoft Corporation. *Microsoft Word Processing Program Version 3.0 for IBM Personal Computers and Compatibles*. Redmond, Wash.: Microsoft Corporation, 1986.

Appendix: Selected Macro Definitions

```
\newif\ifnotdone
\newif\ifwmember
% =============================================
%
% Allow \par's within \loop constructs.
%
\long\def\loop#1\repeat{\def\body{#1}\iterate}
% ---------------------------------------------
\def\AddrLine#1{\GetLine#1\of\Address}
%
\def\AddrBlock#1{%
   \GetBlock#1\of\Address\by{\\---\\}}
%
\def\GetLine#1\of#2{\GetBlock#1\of#2\by\\}
%
% \GetBlock{N}\of\List\by\Delim gets block N of
%    the \List.  The blocks in the \List are
%    assumed to be separated by \Delim's.
%    \Current will hold the last block
%    selected.
%
\def\Current{}
\def\GetBlock#1\of#2\by#3{%
   {\count0=#1
    \toks0=\expandafter{#2#3}%
    \edef\List{\the\toks0}%
    \def\lcar##1#3##2\nil{##1}%
    \def\lcdr##1#3##2\nil{##2}%
    \notdonetrue
    \loop
       \ifx\List\empty
          \notdonefalse
          \gdef\Current{}%
       \else
          \advance\count0 by -1
          \ifnum\count0=0
             \notdonefalse
             \toks0=\expandafter\expandafter
                      \expandafter{%
                         \expandafter
                            \lcar\List\nil}%
             \xdef\Current{\the\toks0}%
             \Current
          \fi
       \fi
       \ifnotdone
          \toks0=\expandafter\expandafter
                   \expandafter{%
                      \expandafter
                         \lcdr\List\nil}%
          \edef\List{\the\toks0}%
    \repeat}}
%
% \wendcarcdr\List\nil\A\B picks off the last
%    word in the \List and places it in \A.
%    The rest of the list (stripped of leading
%    blanks) is placed in \B.
%
\def\wendcarcdr#1\nil#2#3{%
   {\edef\List{#1}%
    \def\carList{}%
    \def\newList{}%
    %
    \notdonetrue
    \loop
       \ifnull\List
          \notdonefalse
          \xdef#2{\carList}%
          \xdef#3{\newList}%
       \else
          \ifx\List\space
             \notdonefalse
             \xdef#2{}%
             \xdef#3{}%
          \fi
       \fi
       \ifnotdone
          \ifx\newList\empty
             \edef\newList{\carList}%
          \else
             \edef\newList{%
                \newList\space\carList}%
          \fi
          \setq\carList{\wcar\List\nil}%
          \setq\List{\wcdr\List\nil}%
    \repeat}}
%
% \wmember\Element\of\List\nil causes
%    \ifwmember to be true if the \Element is a
%    member of the \List and false otherwise.
%
\def\wmember#1\of#2\nil{
   {\global\wmemberfalse
    \edef\Element{#1}%
    \edef\List{#2}%
    \setq\carList{\wcar\List\nil}%
    %
    \ifnull\Element
       \notdonefalse
    \else
       \notdonetrue
    \fi
    \loop
       \ifnull\carList
          \notdonefalse
       \fi
       \ifnotdone
          \ifx\Element\carList
             \notdonefalse
             \global\wmembertrue
          \else
             \setq\List{\wcdr\List\nil}%
             \setq\carList{\wcar\List\nil}%
          \fi
    \repeat}}
```

Typesetting Forms with LaTeX

Mark A. Roth, Maj, USAF
Department of Electrical and Computer Engineering, Air Force Institute of Technology, AFIT/ENG,
Wright-Patterson AFB OH, 45433-6583 USA
513-255-9263; FAX: 513-476-4055
Internet: mroth@afit.af.mil

Abstract

The Air Force Forms System (AFFORMS) is combination of a user-friendly fill-in-the-blank front end and a LaTeX-based forms typesetting system. The overall system is described and the procedure to develop a LaTeX style for a form is presented.

Introduction

The United States Air Force (USAF), like any large corporation or government agency, utilizes hundreds of different forms in its day to day business. Some forms are simple to fill out and if a mistake is made the form is changed via erasure or cross out, or is simply reaccomplished. Other forms are either more complex or are such that even the simplest error cannot be tolerated. For example, the USAF Officer Evaluation System (OES) requires that an officer performance report (OPR) be rendered annually for each officer on active duty. Furthermore, before each promotion board, a promotion recommendation form (PRF) must be completed by the officer's commanding general. Although these forms are not exceedingly complex, they must be typographically perfect, and since each completed form undergoes several layers of review and revision, a single form may be accomplished and reaccomplished from ten to twenty times.

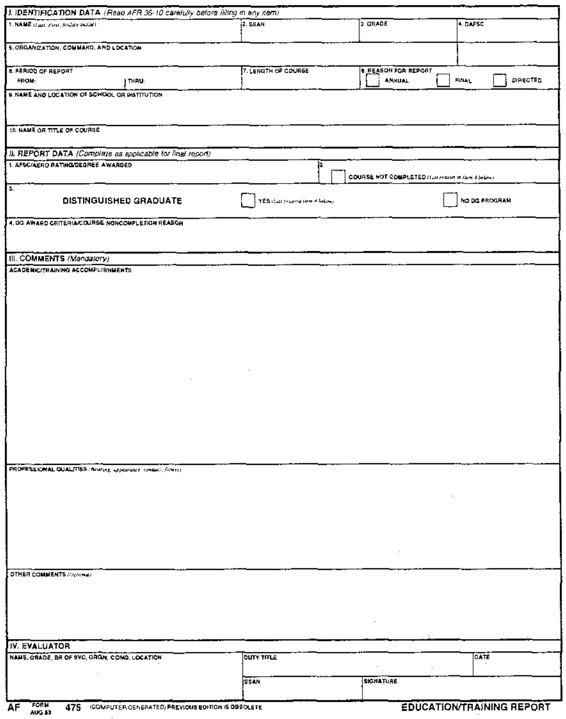
Only a couple of years ago all of this work was accomplished via preprinted forms and an electric (or manual!) typewriter. Now with the widespread availability of computers and laser printers, many offices use word processing programs to lay out the blocks of information for the form and then print directly onto the preprinted forms. This is not entirely satisfactory since the typist needs to maintain the critical spacing requirements within the word processing file—the addition of a line of text means the deletion of a line of space. Furthermore, the Government Printing Office (GPO) rarely supplies consistently printed forms. Each batch is printed on a slightly different position on the page, and, even worse, the forms are sometimes not horizontal. In the days of typewriters these slanted forms could be fed into the typewriter in the same slanted manner so that the typed text would line up with the form boxes. This is quite impossible with laser printers and thus many forms are made useless.

In 1989, several colleagues and myself developed a competely automated forms preparation system for the purpose of preparing and printing OPRs and PRFs. This system avoids all of the above problems by printing the entire form and the user's text onto a plain piece of paper. This system provides a friendly window/menu-oriented interface for the user to compose or edit form entries, and a LaTeX-based typesetting system to produce, preview, and print the complete form.

In this paper, I describe the Air Force Forms Systems (AFFORMS) which evolved from these initial requirements, with specific emphasis on the how the forms were produced using the LaTeX picture environment, and how we standardized the form input parameters. Using these techniques other forms can easily be generated and put into use in your organization. The form I will be presenting is the AF Form 475, Education/Training Report (Figure 1). This form is used to document the progress of Air Force officers in a variety of long term education or training programs. I will present a brief overview of how the current system works from a user viewpoint, and then present the technical issues of designing the LaTeX style for producing the form.

The AFFORMS Package

The AFFORMS package can be functionally broken into two main areas: the user interface and the LaTeX processing files. The user interface provides fill-in-the-blank screens for each of the forms, and a pull-down menu system to perform various functions required to edit, view, or print the forms. The user does not need to know anything about LaTeX, except for the usual rules about the different dashes and



I. IDENTIFICATION DATA *(Read AFR 36-10 carefully before filling in any item)*

1. NAME *(Last, First, Middle Initial)*	2. SSAN	3. GRADE	4. DAFSC
5. ORGANIZATION, COMMAND, AND LOCATION			
6. PERIOD OF REPORT FROM: THRU:	7. LENGTH OF COURSE	8. REASON FOR REPORT □ ANNUAL □ FINAL □ DIRECTED	
9. NAME AND LOCATION OF SCHOOL OR INSTITUTION			
10. NAME OR TITLE OF COURSE			

II. REPORT DATA *(Complete as applicable for final report)*

1. AFSC/AERO RATING/DEGREE AWARDED	2. □ COURSE NOT COMPLETED *(List reason in Item 4 below)*
3. DISTINGUISHED GRADUATE	□ YES *(List criteria item 4 below)* □ NO DG PROGRAM
4. DG AWARD CRITERIA/COURSE NONCOMPLETION REASON	

III. COMMENTS *(Mandatory)*

ACADEMIC/TRAINING ACCOMPLISHMENTS

PROFESSIONAL QUALITIES *(Bearing, appearance, conduct, fitness)*

OTHER COMMENTS *(Optional)*

IV. EVALUATOR

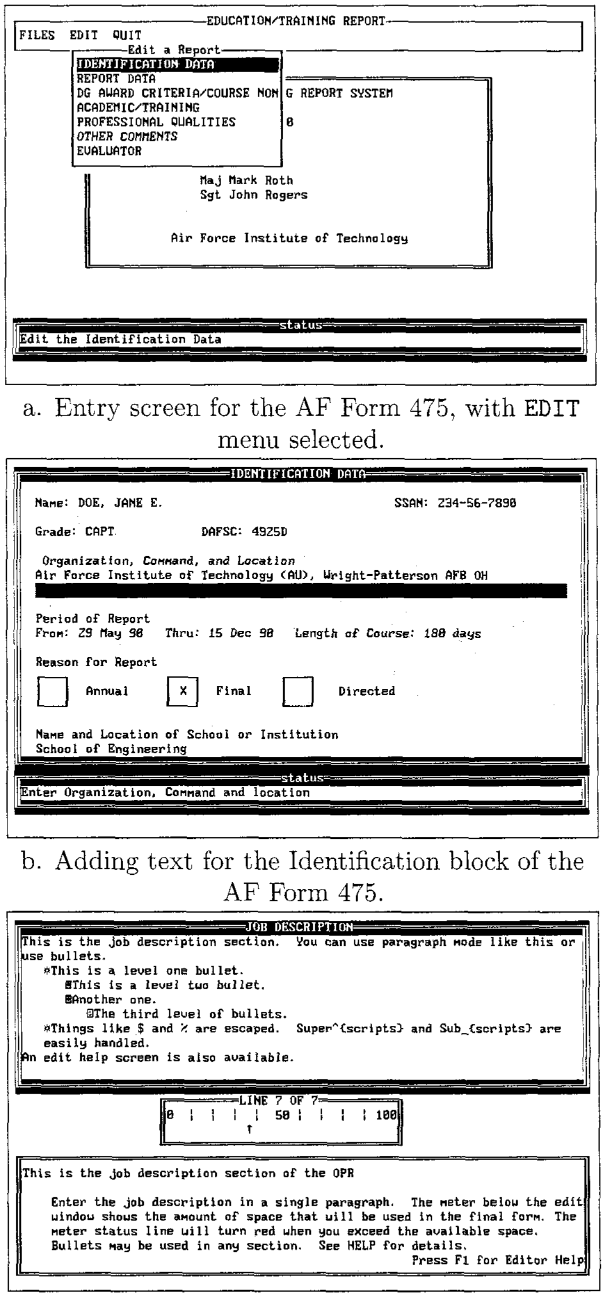
NAME, GRADE, BR OF SVC, ORGN, COMD, LOCATION	DUTY TITLE	DATE
	SSAN	SIGNATURE

AF FORM AUG 83 475 (COMPUTER GENERATED) PREVIOUS EDITION IS OBSOLETE EDUCATION/TRAINING REPORT

Figure 1: Computer generated AF Form 475.

how to get the quote symbols right. Special symbols and actions like dollar signs, percent signs and superscripts and subscripts are automatically captured by the software and translated to the proper codes when the LaTeX input file is prepared. Special keys are used to indicate that bullets, sub-bullets, or sub-sub-bullets are required. Some forms use a visual meter to indicate, for large text blocks, approximately how much actual space LaTeX will need to typeset the input. Figure 2 shows some sample screens from the interface.

The Vitamin C graphics/window library of C functions was used to write the user interface. This turned out to be a fortuitous decision as versions of the Vitamin C package are available for many systems including MSDOS, UNIX, and VMS based computers. We were thus able to port our software to many different architectures, although it is primarily used on MSDOS machines. Any implementation of TeX can be used to run the software. On the MSDOS computers, we used public domain software including SBTeX, DVIVGA and DVIEW screen previewers, and Nelson Beebe's printer driver family. Each interface contains about 1500 lines of C code, most of which is duplicated in each interface. Once



```
EDUCATION/TRAINING REPORT
FILES  EDIT  QUIT
        Edit a Report
IDENTIFICATION DATA
REPORT DATA
DG AWARD CRITERIA/COURSE NON G REPORT SYSTEM
ACADEMIC/TRAINING
PROFESSIONAL QUALITIES       0
OTHER COMMENTS
EVALUATOR

                  Maj Mark Roth
                  Sgt John Rogers

               Air Force Institute of Technology

status
Edit the Identification Data
```

a. Entry screen for the AF Form 475, with EDIT menu selected.

```
IDENTIFICATION DATA

Name: DOE, JANE E.                               SSAN: 234-56-7890

Grade: CAPT            DAFSC: 4925D

 Organization, Command, and Location
Air Force Institute of Technology (AU), Wright-Patterson AFB OH

Period of Report
From: 29 May 90   Thru: 15 Dec 90   Length of Course: 180 days

Reason for Report
[ ]  Annual      [X]  Final      [ ]  Directed

Name and Location of School or Institution
School of Engineering

status
Enter Organization, Command and location
```

b. Adding text for the Identification block of the AF Form 475.

```
JOB DESCRIPTION
This is the job description section.  You can use paragraph mode like this or
use bullets.
   *This is a level one bullet.
      ■This is a level two bullet.
      ■Another one.
         □The third level of bullets.
   *Things like $ and % are escaped.  Super^{scripts} and Sub_{scripts} are
   easily handled.
An edit help screen is also available.

                LINE 7 OF 7
            0 | | | | 50 | | | | 100
                    ↑

This is the job description section of the OPR

    Enter the job description in a single paragraph.  The meter below the edit
    window shows the amount of space that will be used in the final form. The
    meter status line will turn red when you exceed the available space.
    Bullets may be used in any section.  See HELP for details.
                                                     Press F1 for Editor Help
```

c. Entering text using bullets in the Job Description block of an OPR.

Figure 2: Samples screens from the AF Form 475 and OPR user interface.

a developer is familiar with the system it takes only 5-10 days to create and debug a new interface.

The AFFORMS package can currently typeset eight[1] forms which have been approved by the Secretary of the Air Force for use by Air Force agencies, and the Education/Training Report described in this paper which has been submitted for approval.

Creating a Form in LaTeX

Designing a form style with LaTeX is relatively straightforward. Most forms can be done with simple application of `picture` environment commands. Slanted lines that do not conform to the available slopes of the `line` fonts that are available require a more sophisticated package to be added such as the `epic` macros. In our applications slanted lines were only needed to "check" boxes. In forms where it was permitted to allow the computer to check the form, square boxes avoided the slanted line problem. Let me take you through the development of the AF Form 475 style (hereafter referred to as the 475).

Analyze the Fields. Each box of the form which can be filled in needs to be identified and given a macro name. In addition fields which can be checked need a macro flag to indicate whether or not the field should be checked.

First, I created default internal names for each of the entries. I used the name of field with the second character an @ to avoid conflicts with user defined macros. Some of the text fields for the 475 are:

```
% Defaults for entries
\def\N@ME{}   %student name
\def\S@AN{}   %student ssan
\def\G@ADE{}  %student rank/grade
\def\D@FSC{}  %student duty specialty code
\def\O@GANIZATIONONE{}  %1st line of org
\def\O@GANIZATIONTWO{}  %2nd line of org
etc.
```

The REASON FOR REPORT block on the 475 is a set of three boxes, one of which is checked. The macro \R@ASONFLAG will be compared to one of three constant value macros and depending on which one is matched the appropriate box will be checked.

```
\def\R@ASONFLAG{}   %will have one of the
\def\A@NUAL{ANNUAL} % following values
\def\F@NAL{FINAL}
\def\D@RECTED{DIRECTED}
```

Each of the fields then has a macro which will receive the user's input for the field. Alternate names can be specified (e.g., `ssn` and `ssan` in the following) although this is not really necessary in our system, as the user interface generates the LaTeX input file. The system can be used without the user interface so this capability could be useful. We compare each user macro with an empty field, in case non-empty defaults are desired for the entries. Then, if a field is not specified or is specified by the user but with a blank entry, then the default is used. Since all users of our system go through the user interface, all of defaults are blank.

```
\def\@e{}  % empty field for comparison
\def\name#1{\ifx\@e#1\else\def\N@ME{#1}\fi}
\def\ssn#1{\ifx\@e#1\else\def\S@AN{#1}\fi}
\def\ssan#1{\ifx\@e#1\else\def\S@AN{#1}\fi}
\def\grade#1{\ifx\@e#1\else\def\G@ADE{#1}\fi}
\def\dafsc#1{\ifx\@e#1\else\def\D@FSC{#1}\fi}
\def\organizationone#1{
  \ifx\@e#1\else\def\O@GANIZATIONONE{#1}\fi}
\def\organizationtwo#1{
  \ifx\@e#1\else\def\O@GANIZATIONTWO{#1}\fi}
\def\reasonflag#1{
  \ifx\@e#1\else\def\R@ASONFLAG{#1}\fi}
etc.
```

Determine Fonts. The next decision was to identify what fonts were required. We use 12pt roman[2] (cmr12) for all text unless a smaller font is required to let the required text fit in the field. The closest computer modern font is used to match the actual form text. Usually, the forms use sans serif and bold and italic sans serif fonts. I always list all fonts directly used in the style even if some are preloaded, so that a maintenance programmer knows what the font macros mean. For the 475 the fonts are:

```
%commented fonts are preloaded
%\font\tenrm=cmr10                %10pt roman
\font\elhrm=cmr10 scaled1150      %11.5pt roman
%\font\twlrm=cmr12                %12pt roman
\font\sixsf=cmss8 scaled750       %6pt sans serif
\font\sixsfb=cmssbx10 scaled600   %6pt bold ss
\font\egtsfb=cmssbx10 scaled800   %8pt bold ss
\font\ninsfb=cmssbx10 scaled900   %9pt bold ss
%\font\tensfb=cmssbx10            %10pt bold ss
\font\egtsfi=cmssi8               %8pt ss italic
\font\ninsl=cmsl9                 %9pt slant
\font\sixit=cmti8 scaled750       %6pt italic
%\font\egtit=cmti8                %8pt italic
```

We also preload the standard 12, 10, and 8pt fonts for typesetting text and math in a paragraph.

It would be better if we had fonts created at the design size rather than scaled, but usually it doesn't make much difference since we can control the exact placement of individual words and letters, if necessary, in the `picture` environment.

[1] OPRs, EPRs, Travel Orders, and Staff Summary Sheets

[2] To be totally honest, I used PostScript fonts for the form examples shown in this document so I could get better reductions.

Layout of the Form. The picture environment is used to layout the form. Here is where a ruler and little patience pays off. Most boxes and line separations on a form are usually of a standard size, so once you measure a few distances, simple addition gets you the others. We usually like to do our measurements from the top down and left to right, so we set the lower left corner coordinates of the picture to (0,-z), where z is the height of the form. Then for each measurement (x, y) from the top left, the appropriate \put command is \put(x,-y){...}. We used 1cm as the unit length. Here is the first part of the commands to typeset the 475 form:

```
\newcommand{\front}{
\clearpage
\begin{picture}(20.1,25.1)(0,-25.1)
   %second pair is the low. left corner coord.
\linethickness{.06cm}
\put(0,-25.1){\framebox(20.1,25.1)}%outer frame
\linethickness{.03cm}
\put(0.1,-.25){\egtsfb I. IDENTIFICATION DATA
  \egtsfi (Read AFR 36-10 carefully before
  filling in any item)}
\put(0,-.40){\line(1,0){20.10}} %top of box I
\put(0.1,-0.65){\sixsfb 1. NAME\ \sixit
  (Last, First, Middle Initial)}
\put(8.60,-.40){\line(0,-1){.84}} %vert line
\put(8.65,-0.65){\sixsfb 2. SSAN}
\put(12.90,-.40){\line(0,-1){.84}}
\put(12.95,-0.65){\sixsfb 3. GRADE}
\put(16.45,-.40){\line(0,-1){1.68}}
\put(16.50,-0.65){\sixsfb 4. DAFSC}
\put(0,-1.25){\line(1,0){20.10}} %bot of box I
\put(0.1,-1.50){\sixsfb 5. ORGANIZATION,
  COMMAND, AND LOCATION}
etc.
\end{picture}
} %end definition of front
```

Next, within the above picture environment, the user input text entries are positioned at the appropriate places within the form. This is better done after a new blank form is printed, since the computer modern fonts used for the form text usually take up less room than the text on the original form. A \parbox is used to set paragraphs, using \centering to center items as appropriate. We encountered a couple of interesting problems in trying to fit entries into the blocks. The first problem was a size constraint. Since we didn't want the user to have to specify a size function to put in the text, we had to automatically determine which size font to print certain critical items. Also, some entries could be entered as a single line of text, or multiple lines. The following code shows how we handled a two versus one line organization field with multiple sizes for the 475 form:

```
\ifx\O@GANIZATIONTWO\@e %then 1 line org
  \newbox\org
  \setbox\org=\hbox{\twlrm\O@GANIZATIONONE}
  \ifdim\wd\org>19.9cm %then too big at 12pt
    \setbox\org=\hbox{\elhrm\O@GANIZATIONONE}
    \ifdim\wd\org>19.9cm
       %then too big at 11.5pt set at 10pt
       \put(0.17,-1.95){\tenrm\O@GANIZATIONONE}
    \else %OK at 11.5pt
       \put(0.17,-1.95){\elhrm\O@GANIZATIONONE}
    \fi
  \else %OK at 12pt
    \put(0.17,-1.95){\twlrm\O@GANIZATIONONE}
  \fi
\else  %two lines output at 12pt
  \put(5.4,-1.60){\twlrm\O@GANIZATIONONE}
  \put(0.17,-2.00){\twlrm\O@GANIZATIONTWO}
\fi
```

By using a box, we can test the width of the entered text at various sizes and choose the maximum size that allows the entries to fit.

An example of checking boxes is shown below:

```
\ifx\R@ASONFLAG\A@NUAL %check the ANNUAL box
  \put(13.15,-2.85){\line(1,1){.42}}
  \put(13.15,-2.43){\line(1,-1){.42}}
\else\ifx\R@ASONFLAG\F@NAL %check FINAL box
  \put(15.72,-2.85){\line(1,1){.42}}
  \put(15.72,-2.43){\line(1,-1){.42}}
\else\ifx\R@ASONFLAG\D@RECTED %check DIRECTED
  \put(17.78,-2.85){\line(1,1){.42}}
  \put(17.78,-2.43){\line(1,-1){.42}}
\fi\fi\fi
```

The last typesetting problem we had was in the evaluators duty title block. The duty title needs to be centered in this block. When a two line duty title is used, it is possible that the first line, if centered, will overlap with the form text. However, if the first line is shifted right to avoid this overlap, then it looks strange if the second line is not shifted correspondingly. Of course, we can't shift the second line so far that it doesn't stay within the block. Thus some interesting calculations are needed to make this block format aesthetically. The code for this is shown next:

```
\ifx\E@ALUATORDUTYONE\@e\else
 \ifx\E@ALUATORDUTYTWO\@e% then 1 line dutytitle
  \put(8.5,-24.00){\makebox[8.35cm]
    {\centering\twlrm\E@ALUATORDUTYONE}}
 \else% two line dutytitle
  \put(8.5,-24.15){\parbox[b]{8.35cm}{
   \twlrm\twlbase
   %tighten up the baselines a little
   \addtolength{\baselineskip}{-.12cm}
   \newbox\dutyone\newbox\dutytwo\newbox\duty
   \setbox\dutyone=\hbox{\E@ALUATORDUTYONE}
   \setbox\dutytwo=\hbox{\E@ALUATORDUTYTWO}
   \setbox\duty=\vbox{
    \hbox to 8.20cm
     {\hfill\E@ALUATORDUTYONE\hfill}
    \parskip=0pt\par
    \hbox to 8.2cm
     {\hfill\E@ALUATORDUTYTWO\hfill}
    }
   \ifdim\wd\dutyone>5.2cm %5.2=8.2-1.5*2
     %then first line of duty to big to center
     %figure out how much whole thing needs to
     %be moved over to avoid conflict with
     %DUTY TITLE on form
     \newdimen\dutyin \dutyin=\wd\dutyone
     \advance\dutyin by -8.20cm
     \divide\dutyin by2 \advance\dutyin by1.5cm
     \ifdim\wd\dutyone<\wd\dutytwo
      %then see if 2nd line will overflow right
       \newdimen\dutyline \dutyline=\wd\dutytwo
       \advance\dutyline by\dutyin
       \ifdim\dutyline>8.20cm
        %then split lines,right justify 2nd line
         \hspace*{1.5cm}\box\dutyone\newline
         \hbox to 8.20cm{\hfill\box\dutytwo}
       \else%center both lines of duty title
        %left justifying first line
         \hspace*{\dutyin}\box\duty
       \fi
      \else
       %indent to spot right after DUTY TITLE
       \hspace*{\dutyin}\box\duty
      \fi
   \else% center duty title for both lines
    \box\duty
   \fi \vskip-\lastskip}}
 \fi
\fi
```

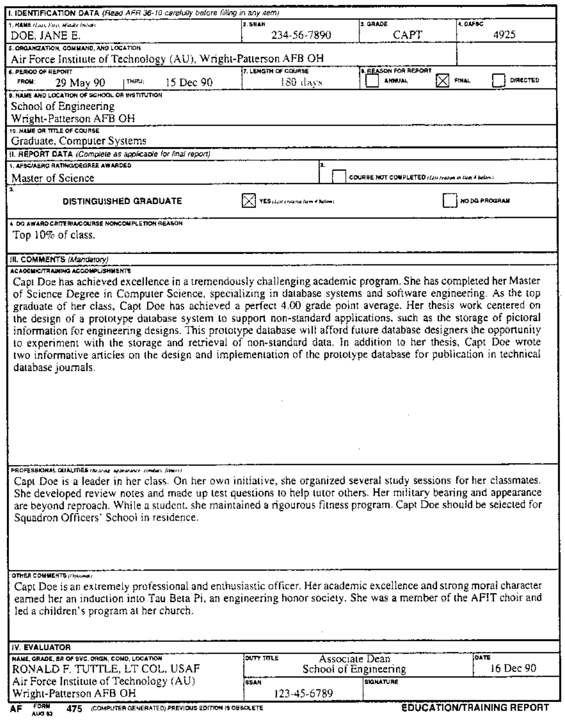
The LaTeX Input

The LaTeX input file simply consists of the definition of each of fields followed by the `\front` command. To get a blank form the commands are:

```
\documentstyle{af475}
\begin{document}
\front
\end{document}
```

Figure 1 showed an example of this. The input for a filled in form, automatically generated by our user interface, looks like the following:

```
\documentstyle{af475}
\begin{document}
\name{DOE, JANE E.} \ssan{234-56-7890}
\grade{CAPT} \dafsc{4925}
\organizationone{Air Force Institute of
Technology (AU), Wright-Patterson AFB OH}
\organizationtwo{}
\from{29 May 90} \thru{15 Dec 90}
\length{180 days} \reasonflag{FINAL}
\schoolone{School of Engineering}
\schooltwo{Wright-Patterson AFB OH}
\course{Graduate, Computer Systems}
\awarded{Master of Science}
\notcompflag{} \distgradflag{YES}
\dgnoncomp{Top 10\% of class.}
\accomplishments{
Capt Doe has achieved excellence ...  }
\qualities{
Capt Doe is a leader ...  }
\comments{
Capt Doe is an extremely ...  }
\evaluatorsigone{RONALD F. TUTTLE, LT COL,
USAF}
\evaluatorsigtwo{Air Force Institute of
Technology (AU)}
\evaluatorsigthree{Wright-Patterson AFB OH}
\evaluatorsigfour{}
\evaluatordutyone{Associate Dean}
\evaluatordutytwo{School of Engineering}
\date{16 Dec 90}
\evaluatorssan{123-45-6789}
\front
\end{document}
```



I. IDENTIFICATION DATA *(Read AFR 36-10 carefully before filling in any item)*

1. NAME *(Last, First, Middle Initial)* DOE, JANE E.	2. SSAN 234-56-7890	3. GRADE CAPT	4. DAFSC 4925
5. ORGANIZATION, COMMAND, AND LOCATION Air Force Institute of Technology (AU), Wright-Patterson AFB OH			
6. PERIOD OF REPORT FROM: 29 May 90 THRU: 15 Dec 90	7. LENGTH OF COURSE 180 days	8. REASON FOR REPORT □ ANNUAL ☒ FINAL □ DIRECTED	
9. NAME AND LOCATION OF SCHOOL OR INSTITUTION School of Engineering Wright-Patterson AFB OH			
10. NAME OR TITLE OF COURSE Graduate, Computer Systems			

II. REPORT DATA *(Complete as applicable for final report)*

1. AFSC/AERO RATING/DEGREE AWARDED
Master of Science

2. □ COURSE NOT COMPLETED *(List reason in item 4 below)*

3. DISTINGUISHED GRADUATE ☒ YES *(List criteria item 4 below)* □ NO DG PROGRAM

4. DG AWARD CRITERIA/COURSE NONCOMPLETION REASON
Top 10% of class.

III. COMMENTS *(Mandatory)*

ACADEMIC/TRAINING ACCOMPLISHMENTS

Capt Doe has achieved excellence in a tremendously challenging academic program. She has completed her Master of Science Degree in Computer Science, specializing in database systems and software engineering. As the top graduate of her class, Capt Doe has achieved a perfect 4.00 grade point average. Her thesis work centered on the design of a prototype database system to support non-standard applications, such as the storage of pictoral information for engineering designs. This prototype database will afford future database designers the opportunity to experiment with the storage and retrieval of non-standard data. In addition to her thesis, Capt Doe wrote two informative articles on the design and implementation of the prototype database for publication in technical database journals.

PROFESSIONAL QUALITIES *(Bearing, appearance, conduct, fitness)*

Capt Doe is a leader in her class. On her own initiative, she organized several study sessions for her classmates. She developed review notes and made up test questions to help tutor others. Her military bearing and appearance are beyond reproach. While a student, she maintained a rigourous fitness program. Capt Doe should be selected for Squadron Officers' School in residence.

OTHER COMMENTS *(Optional)*

Capt Doe is an extremely professional and enthusiastic officer. Her academic excellence and strong moral character earned her an induction into Tau Beta Pi, an engineering honor society. She was a member of the AFIT choir and led a children's program at her church.

IV. EVALUATOR

NAME, GRADE, BR OF SVC, ORGN, COMD, LOCATION RONALD F. TUTTLE, LT COL, USAF Air Force Institute of Technology (AU) Wright-Patterson AFB OH	DUTY TITLE Associate Dean School of Engineering	DATE 16 Dec 90
	SSAN 123-45-6789	SIGNATURE

AF FORM AUG 83 475 (COMPUTER GENERATED) PREVIOUS EDITION IS OBSOLETE EDUCATION/TRAINING REPORT

Figure 3: Computer generated AF Form 475 with entries.

This completed form is shown in Figure 3.

Problems Along the Way

Our initial hurdle was political in nature. Getting Air Force approval for using this system was a long and involved process. Publishing is so little understood that the initial approval letter required us to use a particular computer system (the Zenith 248), but made no restrictions on the laser printer!

That vast variety of computers, versions of MS-DOS, and laser printers (especially those that were "100% compatible", but really weren't) caused us a lot of headaches. One of major physical problems is that the forms use most of an 8 1/2 × 11 page, stretching the limits of most laser printer engines. Also, LaTeX takes up a lot of memory to run. In order to get the most complex forms to run, I created a stripped down version of `latex.tex` and `lfonts.tex` to reduce the memory requirements. Many things such as sectioning, table of contents, etc. will never be used in a form, and so could be eliminated.

Conclusion

The Air Force has been very happy with the system. Although faster commercial forms packages are available, they are not free, not portable, and don't do nearly as nice a job as LaTeX on formatting the text for the blocks.

Acknowledgements

Many thanks go to all of the Air Force personnel who contributed to the AFFORMS package: Col Stan Lewantowicz and Lt Col Charlie Bisbee who with myself created the initial package, and Maj Bob Rebo, SrA Dave Weissfeld and Sgt John Rogers who added many features and distributed the program worldwide.

I also thank all of the people who create public domain TeX software. Without this we could not have been so successful, or produced such beautiful documents.

Developing a Pop-Up Help Facility for TEX on PCs

Peter Flynn†
University College of Cork, Cork, Republic of Ireland
BITnet: cbts8001@iruccvax

Abstract

The manuals, tutorials, and descriptive books about TEX and its allied products are fairly comprehensive, but their very technical completeness means that the novice user is often unable to locate quickly the information needed from among the quantity provided. Many new users also ask, "where is the HELP file?" and too often receive an unsatisfactory response. This paper describes the construction of a memory-resident pop-up help facility for TEX on PCs, developed using publicly-available software. The problems discussed include determining the level of complexity that a help system needs; the balance between fine and coarse detail; cross-referencing and circular references; context-sensitivity; visual presentation; the selection of topics; the regard for later conversion to other systems with structured help facilities (e.g., VMS); and the development of tools for assisting the help file development process. The help file itself, when completed, will be made available with the distributable software in the public archives.

Introduction

At the 5th European TEX Conference in Cork, a paper was presented which summarised the approaches taken by the various authors of a selection of documentation on TEX.[1] The author's conclusions were that the quality was uneven: some documents contained errors of logic, some were difficult for the novice to understand or find a path through (and these were documents *intended* for novices!), and some manifested æsthetic problems in the design and layout which made using them less easy.

The problem facing the present author, in his user-support function, was to provide a faster startup path for novice users of TEX and LATEX.[1] Training courses were out, as the numbers involved were too great for the resources of the Computer Centre, so the use of machine-based assistance was investigated.

Several Computer-Based Training (CBT) packages were examined, but all were too expensive or required too great an investment in time to get running, even at a low level. A simple help facility was therefore identified as a mechanism for providing the user with the information necessary to solve the most common low-level queries. This has now been expanded to handle some more complex material as well, and contains additional matter on more general topics such as design, layout, and typography.

Existing HELP Facilities

Identifying existing HELP files was not difficult as there are relatively few of them. E-mail messages were sent to many contacts asking for copies, where available, of help texts.

The most numerous responses were for the VMS `latex.hlp` file, summarised below. In addition, the help texts supplied with various versions were retrieved and examined (not a long task!).

There is, of course, comprehensive assistance available on TEX through the electronic mail networks for those fortunate enough to be connected; but this is of an asynchronous nature, and something more immediate was required.

† The author would like to express his thanks to all those who generously gave their time and expertise by electronic discussions during the development of this system.

[1] From here on, references to TEX imply also LATEX, for brevity, unless explicitly designated 'Plain' or by mention of some other macro package.

Table 1: Content of `latex.hlp`

Level	Sections	Content
1	1	Description of TEX and LATEX
2	3	Commands, Parameters and Qualifiers
3	22	Things you can do in LATEX
4	106	Command and environment names
5	50	Math and more specialist commands

VMS (LATEX). A DEC VAX/VMS help file is a structured multi-level hierarchy. Sections labelled for level 1 appear as headings when you type `help latex`, and selecting one of these first gives you the associated help text and then displays the relevant level 2 headings associated with it. Selecting one of these repeats the mechanism and takes you to level 3, and so on. Pressing the Return key takes you back one level and eventually quits from `HELP`. The source text of help files is compiled into a help database (`.hlb` file) for speed of access.[3]

The `latex.hlp` file is fairly comprehensive and some 2,100 lines long: its structure is described in Table 1. No information was available about the decision-making process which was used as to what to include and what to omit.

VMS (LSE). The author has reports of a on-line help system for use with the VAX/VMS Language-Sensitive Editor (LSE), but has been unable to locate the text. There is documentation on the use of editor command keys for using the templating facilities of the LSE, but no evidence of help on TEX commands for doing the formatting.

UNIX `man` pages. The `man` pages consist of a brief description of the TEX system with some notes on directory storage and environment variables.

PC and Macintosh versions. No existing help files were located on how to use TEX commands. All the known PC and Mac versions of TEX have some form of help documentation, but this is restricted almost entirely to technical details (usually very good) of how to get the program running and how to organise logical names and subdirectories.

Some commercial versions supply a manual (e.g., PC-TEX, TEXtures); others rely on the *TEXbook*. None supply any of the public-domain workbooks which are available.[2]

Other environments. In the time available no other environments were examined, although there are some help facilities available on Atari and IBM mainframes. It is hoped to include some of this material (if it can be located) at the β test stage.

Selection of Topics

The overriding need to provide some structure within which to develop a help system meant that a categorisation of topics was required. The structure used would need to apply equally to Plain TEX and to LATEX as well as to other macro packages for which help might be needed at a later date.

The commands and facilities of TEX fall into several discrete types:

- Operating system commands needed to run the programs, and the switches, setups and arguments;
- Commands which affect the positioning of text;
- Commands which affect the appearance of text;
- Values such as dimensions and counters;
- Commands to operate logical structures like sectioning, index, contents, tables and mathematics modes;

but as the objective was to create a system which would answer the "how do I..." questions, it was necessary to marry these abstract categories with the requirements of the users. There is of course a considerable degree of overlap between the coarse categories described above.

User requirements. Working out which commands are most useful to a user was the most difficult of all the tasks. Commands used daily by some users are never used at all by others. The only guideline available was experience, and this is naturally site-dependent.

One of the prime expectations of this paper is that other user-support workers will be able to contribute their expertise in order to improve the composition of the system.

From logs kept of user queries, some common requirements were identified:

- How to process a file through TEX and display or print it;
- How to change type size and style, and find out what is available;
- How to change spacing and layout;
- How to insert graphics;
- How to handle logical structures;
- How to do mathematics; and
- How to interpret error messages.

These categories match fairly closely those described in the previous section pertaining to TeX commands themselves, with the addition of graphics and error messages. It was felt that the interests of the users would be best served by a system which attempted to model these needs, rather than mirror the traditional sequential presentation (*à la TeXbook*), even though some of this would mean extensive cross-references to items which the user may not be interested in.

Levels of foreknowledge. The system as it currently stands assumes no foreknowledge whatever about TeX itself. It does however assume that the user knows how to switch on the machine, type a command and use a text editor, and follow simple single-key instructions to run the help system itself.

Although this may be considered a valid approach for novice users, it also means that a substantial amount of explanatory text is required at a low level. Some basic concepts about typesetting therefore need to be covered, which are wholly external to TeX or any other form of desktop publishing. This forms an additional category to those described above.

Depth of detail. No attempt is made to explain *why* things are done às they are in TeX unless this knowledge is required to understand *how* a command or facility works as it does. Thus, for example, no details of the internals of TeX or LaTeX are included.

There is perhaps scope for a more advanced version of this help system, to act as an *aide-mémoire* to experienced users.

Structure

Although the structure of a help text is to some extent conditioned by the software used, it is possible to approach the problem in different ways. Before making a final decision on the software, several options were considered:

Flat file. This is the simplest format. Topics appear as headings, and the file can be searched by system utilities to locate the text required. However, unless more sophisticated software is used, display of search 'hits' is usually restricted to a line or so either side of the target keywords.

Multi-level. Multi-level (hierarchical) structures (like VMS HELP) require much more sophisticated software but provide highly tailorable context-sensitive displays, although locating the help required is usually restricted by the need to navigate down, up, or across the hierarchy, rather than by searching the whole text for matches.

Hypertext. This very powerful information concept is available on a variety of platforms other than the Apple Macintosh, where it is popularised in the Hypercard$^{\text{TM}}$ program. The software is highly complex, and developing 'stacks' (help texts) can be fairly lengthy business if the extensive cross-referencing and multimedia management facilities are used. If such a system could be provided as a memory-resident option under MS-DOS, complete with graphics, serious consideration would be given to using this method.

Hybrid structures. These offer some combination of the facilities of those structures mentioned. Specifically, they use flat files, but with entries tagged in such a way as to facilitate compiling (like VMS HELP); and there is usually some element of hypertext, in that the user can wander *ad libitum* from one cross-reference to another without the restrictions of a strictly hierarchical system.

Compatibility with other systems. In developing the help text, it has been borne in mind that it should be possible to port the system to another file structure without too much rewriting. Although the file-marking itself is specific to the software used, the one selected nevertheless uses a plain text file, which can be edited or stripped apart by data management software for reworking into other systems.

Implementation

Six PC-based systems were evaluated: HELP, PAINTHLP, HELP_SYS, QUIKHELP, QHELP and HELPSB. They provided a mix of file structures and access methods. They were evaluated by checklist and examination against the following criteria:

- Ability to run as TSR (memory-resident pop-up) over TeXware and editors;
- Size of memory taken;
- Ease of use by novice;
- Ease of writing/formatting files for help input;
- Need or otherwise for special configuration of PC; and
- Support and documentation.

Choice of delivery system. The software chosen was QHELP, written by Mark van Kekerix. This product was chosen because it is the simplest and fastest to work with, requires no special facilities, takes a plain input file, and sits as a TSR over the TeX executables tested without interference.

Disk storage. The help file in its present state is around 50kb of text, and a similar size when compiled. The QHELP executable is only 21Kb, and no further disk space is required by users. It is envisaged that the help file should probably contain additional explanatory material, and there will doubtless be additions to the commands covered as a result of β-test users' comments. It is suggested that user-support people develop and keep their own site-specific help file, and append this to the main file for recompilation and distribution locally. Thus the resultant size of the completed text in compiled form is difficult to predict.

Memory constraints. QHELP requires approximately 35kb memory. Future versions may require more or less, depending on added facilities or reprogramming to minimise usage.

Availability and Cost. The software is freely available in the public domain. The author specifically states in his introduction:

> [...] you are free to use QHELP and any help files you have created yourself or obtained from bulletin boards, etc., but if you want to package QHELP with another program you are selling for profit, contact me for written permission to do so (I will ask for a small fee for such a license).
>
> The limitation on "bundling" QHELP with other programs applies only if the program is being sold for profit. If you have developed a public domain program and would like to use QHELP and a help file you have written as the program's help system, please feel free to do so.[4]

Physical file structure. A QHELP source (text) file is plain ASCII, using the colon in column 1 to flag keywords, and the tilde to mark references to them in the text. During use, references in the text are highlighted, and selecting one with the cursor and pressing the Enter key causes the help text for that keyword to be displayed.

Each section (description of a keyword) is formatted to fit the selected size of the help window, and can extend to multiple pages. From user experience it is more effective to have more subdivisions of keywords, with one screen each, than to use multiple screens for a single keyword. Within each description, text can be freely formatted as desired.

Referencing. Because of the undiscriminating nature of the reference-to-keyword mechanism of QHELP, the marking of occurrences of keywords in the text as references must be done with care. Too many, and the user may be confused by which one to choose next; too few, and the user may not be able to find the help required. Marking a word (*i.e.*, a command) as a reference means that that word must occur as a keyword (QHELP will refuse to compile otherwise), but otherwise there are no logical restrictions on what can or cannot be thus marked. The tendency was noted early on to mark too many words; but it is not certain until the system is used more extensively by its target audience whether the level chosen is adequate. The average number of references marked in a section is seven. Care must also be taken to avoid purely circular references which will confuse the user, although an Alt-key function enables the user to jump back to the opening screen if needed.

Development tools. QHELP normally recognises a space character (ASCII 32) or any non-alphabetic character as the end of a keyword or reference, but has the ability to allow the user to specify additional non-alphabetics which are to be taken as valid characters in a keyword string. The present file uses backslash, underscore, and curly braces as additional characters, for obvious reasons. This specification is only possible interactively at compile time, and cannot be stored as a compiler configuration. The author has used the public-domain STACKEY utility to assist compilation, as this enables the passing of keystrokes to an application, in this case to drive file and parameter selection and initiate compilation from an MS-DOS batch file. The STACKEY software is included in the present distribution.

The QHELP compiler reports unmatched references and aborts on the first such occurrence. There is no facility for listing or otherwise reporting on the file structure, so a routine was developed in the P-Stat data management language to read the help file and do its own parsing for keywords and references, and report in listed form on matched, unmatched and unapplied references. This routine is supplied in the distribution, but it should be possible to rewrite it for other data manipulation languages where P-Stat is not available.

Presentation. The colours of the help display can be changed by the user or distributor, as can the hot-key for activating the system. Upon activation, the system will search the keyword list for the word under the cursor (delimited by spaces and punctuation), so the system is to a small extent context-sensitive. If this fails, the default (opening) screen is displayed, and the user can search through the highlighted topics using the arrow keys.

Compatibility with editors, `tex.exe`, etc. QHELP has been tested with PC-Write, PC-TEX (both 8086 and 80386 versions), emTEX (both 8086 and 80286 versions) and the associated `.DVI` drivers on a variety of IBM and compatible PCs. No errors have been reported, and there was no noticeable impact on system performance.

Text Generation

As discussed above, the source of the present text has been generated by the author, in the case of Plain TEX commands, and taken from the VMS help file, in the case of LATEX, with authors' additions and substantial modifications to the language.

Complexity of wording. The level used in writing is that normally associated with instructional material for novice usage. A conscious effort was made to avoid computing jargon and the use of typographical terminology beyond the experience of the normal wordprocessing user. It was assumed that the user has reasonable fluency in the use of English, but the author would be delighted to see versions of the text made available in other languages.

Conclusions

The system provides a first attempt at making TEX formatting information available to the user without the need for tuition or a manual. Although this may be viewed as 'wrong' by educationalists and educators alike, in the situation where demand for help far exceeds supply, it is virtually the only solution.

Work outstanding. As noted above, more work is needed to add some of the less frequently used commands which may come to the users' notice and require explanation.

In particular, the many add-ons for TEX beyond LATEX, such as PICTEX, musTEX, CIRTEX and so on, should be documented in outline at least, so that the user is aware of their existence, and is not left trying to reinvent some of the existing wheels unnecessarily.

Availability. The software is available from the servers at Aston, YMIR and Heidelberg as file `texhelp.zip` (MS-DOS format). Commercial suppliers would need to adhere to the requirements of the author of QHELP if they wish to distribute the system, but no charge is made for the use of the help text file itself, provided it is distributed unmodified (it may be appended to but not changed without arrangement with the present author).

References

[1] Barden, Angela. *Purchasing Pain with all that Joy.* TUGboat **12**:1, 1990 Conference Proceedings.

[2] Doob, Michael. *A Gentle Introduction to TEX.* In: `ymir.claremont.edu::[tex.documentation] gentle.tex`, 1991.

[3] [Author unknown] `LATEX.HLP` (Help file for the DEC VAX/VMS version of LATEX). In: `tex.ac.uk::[tex-archive.utils.editors.lse.help]latex.hlp`, 1990.

[4] Van Kekerix, Mark. `QHELP` *Pop-Up Help Program Users Guide.* In: `simtel20.army.mil::pd1:/msdos/info/qhelp.arc`, 1990.

7 Bits Good, 8 Bits Bad *or* "The Eight-Bit Blight"

Malcolm Clark
Polytechnic of Central London
`malcolmc@mole.pcl.ac.uk`

Brian Hamilton Kelly
Royal Military College of Science
Shrivenham
`tex@rmcs.cranfield.ac.uk`

Niel Kempson
25 Whitethorn Drive
Cheltenham
`tex@rmcs.cranfield.ac.uk`

Abstract

Inter-networking and e-mail systems can usually be relied upon to permit faithful exchange of seven-bit ASCII data. Transfer of eight-bit binary data is not so reliable, especially when the data must traverse gateways or be exchanged between different system types.

Until now, it has been possible to exchange TEX sources of papers by electronic mail without much difficulty. Now that TEX and its relations support eight-bit input their source files will now suffer the same problems as binary data.

The proliferation of electronic archive services has highlighted the need to be able to exchange binary data between disparate systems, often connected via gateways. The authors introduce a new file-encoding standard that meets this need and far surpasses existing schemes.

Reliable and faithful exchange of binary files between computers over networks is a well-known problem, especially if the computers use different operating systems and are connected to different networks via a gateway. Unfortunately internetworking and electronic mail are very much children of the '60s: they might have had to wait until the '70s for their naissance, but their progenitors were mentally locked-in to the concept of the 7-bit ASCII code for conveying textual information. The TEX community has long been aware of this problem when trying to exchange "machine-independent" `.dvi` files and font-related data such as `.tfm` and `.pk` files. It has sometimes been possible to exchange this binary data by using encoding schemes that allow the data to be represented using a subset of the seven-bit ASCII character set.

Academics and authors in many fields have hitherto been able to pass `.tex` files back and forth by electronic mail — apart from a few minor quirks and blemishes, such TEX source files pass unharmed across the planet's networks. Problems are encountered when mail passes through certain gateway machines that introduce irreversible character corruptions. Particularly notorious is the Janet/Bitnet gateway, which has the unfortunate habit of converting '`^`' to '`~`' and '`~`' to '`%`'. Since it leaves '`%`' itself unaffected, this makes recovery of the original file a non-trivial exercise. It sometimes also changes the brace characters '`{}`' into odd characters above 128; this is particularly embarrassing, of course, for `.tex` files!

For some years, many TEX users, particularly those working in languages other than English, and thus familiar with character set encodings containing other than the basic ASCII set, have been agitating for TEX to be able to handle input in their mother tongues, using their own languages' character sets. In 1989, Knuth [1] announced TeX v.3, and implementors world-wide beavered away to bring each implementation up to date. TeX v.3 now supports eight-bit character sets and so `.tex` source files are now effectively 'binary' files and will therefore suffer from the same exchange problems experienced with `.dvi` files.

All those authors who had previously been able to cooperate, despite being separated by hundreds or thousands of miles, might once again be forced to entrust floppy disks to the vagaries of the world's postal systems (although one shouldn't underestimate the bandwidth of the Royal [or other] Mail system).

Unless or until the various e-mail protocols, networks and software are converted to support uncorrupted transmission of characters codes *'040..'176* **and** *'241..'376*, it will have to become the norm for `.tex` sources to be encoded for transmission by e-mail.

The Aston Archive

All three authors are volunteer assistants to Peter Abbott in running the world's principal repository of TeX-related material at Aston University [2] in Birmingham. The archive holds several hundred megabytes of text and binary files including

- program sources for TeX, METAFONT, DVI drivers and many other utilities;
- binary executables for a variety of popular operating systems (e.g., Atari, Macintosh, MS-DOS, Unix, VAX/VMS and VM/CMS);
- METAFONT sources for Computer Modern and other fonts;
- binary font files (mainly `.tfm` and `.pk`) for a number of different output devices;
- text, macro and style files.

The archive provides access to these files via the following services:

- NIFTP[1] from Janet hosts. Typically 300 megabytes of data are transferred every month; this would probably be much greater if we were not limited by the bandwidth of our 9600-baud connection to Janet.
- FTP from Internet hosts. At the time of writing, the Internet connection has been approved and should be available by the third quarter of 1991.
- Interactive browsing service via Janet PAD, including the facility to send files out using NIFTP (and later FTP).
- Interactive browsing service via dial-up modem lines, including the facility to download files using Kermit and similar protocols.
- An e-mail file server that typically sends 150 megabytes of data per month to sites all over the world (though predominantly to EARN/Bitnet sites).
- A magnetic-media distribution service via surface carriers. Copies of the entire archive have been sent to embryonic TeX communities in Czechoslovakia, Hungary and Poland.

We have experienced many problems trying to support all of these file types, operating systems and access methods. The e-mail file server clearly needs a reliable method of encoding files if its many customers are not to be denied access to the non-text files in the archive.

Binary files such as `.pk` font files are stored in different ways to accommodate the requirements of the different operating systems supported. Currently we maintain multiple font directory trees for the Macintosh, MS-DOS, Unix and VAX/VMS, with all the attendant problems of synchronization, disk space and archivists' time. We need a single storage format that allows export to all of our supported operating systems.

Specification for a Coding Scheme

In mid-1990, the archivists came to the conclusion that a universal encoding scheme was required to accommodate the many different kinds of file and file organizations that needed to be supported by the archive.

Niel Kempson formulated the first draft of this specification in mid-1990; the requirements of the encoding scheme may be summarized as follows.

Preserving File Structure. It is insufficient, especially for an archive holding binary files for a variety of machine types, merely to encode data simply as a stream of bytes:

[1] Network Independent File Transfer Protocol — in the UK, one does not perform the pseudo-login that Internet users are accustomed to using with the FTP protocol. Instead, one issues a 'transfer request' for a file to be sent to or from the remote machine — the transfer itself takes place asynchronously. One nice consequence is that such transfers can be queued for overnight execution, leaving daytime bandwidth free for e-mail and true remote interactive logins.

- Virtually all operating systems[2] make a distinction between binary and text files, so the coding system should recognize and maintain this distinction.
- UNIX and most PC-based operating systems treat files as streams of bytes with no further structure imposed. On the other hand, certain widely-used operating systems (e.g., VAX/VMS and VM/CMS) have record-oriented file systems where different types of file are stored in a format appropriate to the type of file.[3]

  For these operating systems, we consider it essential that the encoding scheme identify, preserve and record the most commonly used file organizations. The decoding program should be able to use this information to create the output file using the organization appropriate to the operating system in use. If the information is of no consequence to the receiving system, the default file structure (if any) should be created. If the encoding system does not have structure in its files, the receiving system may provide suitable defaults automatically. In all cases, the programs should permit the user to override or supplement file structure information.
- Whenever possible, these details of structure should be determined automatically by the encoding program; at the very least, an indication of whether the file is text or binary shall be provided (even under an operating system such as UNIX that need make no such distinction for its own use), to allow decoding to an appropriate file organization on those systems that *do* make such a distinction.

Coding Scheme. Whatever method is used for ensuring that encoded data can be e-mailed:

- It should be possible to specify the coding table to be used to encode the data. The coding table used should be recorded with each part of the encoded data.
- If a recorded coding table is found while decoding, it should be used to construct an appropriate decoding table. Simple one-to-one character corruptions should be corrected as long as only one of the input characters is mapped to any one output character.
- The recommended encoding uses only the following characters:

  ```
  +-0123456789
  abcdefghijklmnopqrstuvwxyz
  ABCDEFGHIJKLMNOPQRSTUVWXYZ
  ```

  Such an encoding has been shown to pass successfully through all the gateways that are known to corrupt characters.

Integrity of Encoded Data. We want to ensure that the *whole* encoded file passes through the e-mail network.

- Encoded lines should be prefixed by an appropriate character string to distinguish them from unwanted lines, such as mail headers and trailers. Whilst not essential, this feature does assist the decoding program in ignoring these spurious data.
- Lines should not end with whitespace characters, as some mailers and operating systems strip off trailing whitespace.
- The encoding program should calculate input file parameters, such as the number of bytes and CRC (cyclic redundancy check), and record them at the end of the encoded data.

  The decoding program should calculate the same parameters from the decoded data and compare the values obtained from those recorded at the end of the encoded data.

Making Files Mailable. A mechanism is needed to overcome some gateways' refusal to handle large files.

- The encoding program should be able to split the encoded output into parts, each no larger than a maximum specified size. Splitting the output into smaller parts is useful if the encoded data is to be transmitted using electronic mail or over unreliable network links that do not stay up long enough to transmit a large file. The recommended default maximum part size is 30kBytes.
- The decoding program should be able to decode a multi-part encoded file very flexibly. It should *not* be necessary to:
  1. strip out mail headers and trailers,
  2. combine all of the parts into one file in the correct order, and
  3. process each part of the encoded data as a separate file.

Miscellaneous. Further considerations include:

- Support for character sets other than ASCII is essential if the encoding scheme is to be useful to IBM hosts. The encoding program should label the character set used by the encoded data,

[2] UNIX is a notable exception to this rule.

[3] It is argued that the increase in efficiency more than offsets the increase in complexity.

and both encoder and decoder should enable the conversion between the local character set and another character set. For example, a user on an EBCDIC host should be able to encode text files for transmission to another EBCDIC host, or to convert them to ASCII before encoding and transmission to an ASCII host. Similarly, that user should be able to decode text files from ASCII and EBCDIC machines, creating EBCDIC output files.

- Where possible, the original file's timestamp should be encoded and used by the decoding program when recreating the file; this will permit archives to retain the originator's time of creation for files, and thus permit the users (not to mention the archivists) to identify more clearly when a new version of a file has been made available.
- The encoding and decoding schemes should be able to read and write files compatible with one or more of the well-established coding schemes.
- The source code for the programs should be freely available. It should also be portable and usable with as many computers, operating systems and compilers as possible.

The Search Commences

Naturally, the first step was to examine the existing coding schemes in comparison with the above ideal specification. Such schemes fell into two broad classes: *portable schemes*, which were intended to permit the encoding of files on any computer architecture into a form that could be transmitted electronically, and decoded on the same or a different architecture; and *platform-specific schemes*, which provided rather better support for transferring files between two computers using the same architecture and operating system.

Portable Coding Schemes. The most commonly used coding schemes supported by a variety of platforms are:

- boo
- UUcode
- XXcode

Most implementations of these schemes known to the authors are designed for use with stream file systems. These programs have no means of recording, let alone preserving, record structure and are thus unsuitable for our purposes. This is not surprising since UUcode and its mutation, XXcode, were developed specifically for exchanging files between UNIX systems. In fairness to these schemes, they are well suited to the transmission of text files and certain unstructured binary files.

Standard UUcode encodes files using characters ‘ ’..‘_’ of ASCII. This can result in one or more spaces appearing at the ends of lines; some mailers decide that this is information not worth transmitting, with consequent inability to reconstruct the original file.

Files containing characters such as ‘:’ are often irreversibly corrupted by mail gateways; this problem led to the development of XXcode, which uses a rather more robust character set, namely:

```
+-01234567890
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

The encoding table used is recorded with the encoded data to allow the detection of character corruptions, and the correction of reversible character transpositions. Whilst superficially a step forward, XXcode offered little more than most existing versions of UUcode, which already supported coding tables. Its major contribution was in formalizing the encoding table, and in particular its default table was proof against all the known gateway-induced corruptions.

Platform-Specific Coding Schemes. Encoding schemes have been developed to support transfer of files possessing some structure that therefore cannot be reconstructed correctly when encoded by the portable schemes. When the encoding and decoding programs of such a platform-specific scheme are each used on the same computer and operating system type, files may be encoded and transmitted with a great deal of confidence that the decoded file will reproduce the original's structure and attributes in their entirety.

Examples of such programs are `TELCODE` and `MFTU` for VAX/VMS, `NETDATA` for IBM mainframes, and Stuffit and MacBinary for the Macintosh. But these programs have the major disadvantage that they have each been implemented *only* on the single architecture for which they were designed; thus the only two of these schemes that could be used on the VAX/VMS-based Aston Archive would be of minimal interest elsewhere!

The Archive's content is in some respects artificially inflated by the presence of `.hqx` files for Macintoshes, `.boo` for MS-DOS, etc., which have to be held in pre-encoded form for transfer by those requiring them.

VVcode is Born

Realizing that none of the existing portable schemes were close enough to our ideal, an early version of our specification was circulated on various mailing lists by Niel Kempson towards the end of 1990. When the anticipated 'nil return' was all that resulted, Brian Hamilton Kelly went ahead and created a rudimentary VVencode by modifying an existing VAX-PASCAL implementation of uuencode. After generating the companion VVdecode, he then reimplemented the programs in Turbo C under the MS-DOS operating system on the IBM-PC, and thereby was able to prove that the new scheme was both viable and sufficient.

A Production VVcode. Following the minor feasibility study, Niel Kempson re-engineered the pair of programs from scratch (adding certain features of the evolving specification), paying particular attention to making the code[4] portable across a wide variety of operating systems. Particular care was taken to avoid the use of supposedly standard C functions that experience had shown behaved differently under individual manufacturer's implementations, or were even non-existent in some. Therefore, the code may sometimes appear to be performing certain operations in a very long-winded way; it's very easy to look at it and say, "Why didn't the author use the ... function, which does this much more efficiently?" But this function may not even exist under another implementation of C, or it may behave in a subtly different manner.

The core functions of VVcode are implemented as a collection of routines written in as portable a fashion as possible, with a separate module of a few routines that are operating-system specific.[5] Porting VVcode to a new platform should require only that this latter module be re-implemented, in most cases by adapting an existing one.

VVcode implements all of the features listed in the specification, apart from the ability to generate UUcode- and XXcode-compatible files. However, the decoding program is backwards compatible and can decode files generated by UUcode and XXcode.

Arguments against VVcode. When the advent of the VVcode system was first aired in the various electronic digests, some heated debate followed, along the lines that a new encoding scheme was unnecessary, since UUcode/XXcode sufficed *for them.* However, all these correspondents were UNIX users who had interpreted the 'VV' as meaning 'VAX-to-VAX' (by analogy with 'uu'[6]), and thus felt that such a scheme should be private to VAXen. The authors' response is that the encoding scheme was intended to support the needs of archives like Aston's, and as such, must provide:

1. an automated tool (it would be somewhat difficult to expect our users to be able to tell the encoder what sort of file structure it is handling, when this concept is entirely alien to many of them);
2. facilities to encode binaries for many operating systems;
3. mail server features, such as splitting of large files; and
4. operation across the widest possible combination of platforms.

The overhead of using the VVcode system is at most a couple of hundred bytes over using UUcode, and the extra functionality and *universality* with respect to UUcode or XXcode thereby comes almost for free.

Availability of VVcode

At present, the VVcode system is only available in C, but it has been shown to run successfully on the following combinations of hardware, operating system, and compiler:

Unix

- DEC Mips; Ultrix (BSD 4.2); native C
- HP9000; HPUX 6.5; native C and GNU C
- IBM RS-6000 (BSD 4.3); native C
- ICL DRS6000 (SPARC); System V (Rel 4); AT&T C
- Masscomp 5600; native C
- MIPS M/2000 (MIPS R3000); RiscOS 4.51; native C
- Sun; SUNOS 3.x and 4.0.3; native C and GNU C
- Sun Sparcstation 1; SUNOS 4.0.3; native C and GNU C

VAX/VMS

- All VAXen; VMS 5.2–5.4-1; VAX/C v3.0–v3.1-51 and GNU C

MS-DOS

- IBM PS/2, PC (and clones); MS-DOS 3.3, 4.01; Borland Turbo C 1.5, 2.0 and Turbo C++ 1.0

[4] That written by BHK was, in Niel's words, "PASCAL written as C"!

[5] Such as file I/O, timestamping, command-line or other interface, etc.

[6] 'V' was chosen simply because it followed 'U'; at one time, we had seriously considered calling it YAFES — Yet Another File Encoding Scheme!

- IBM PS/2, PC (and clones); MS-DOS 3.3, 4.01; Microsoft C 5.1 and 6.0

VM/CMS

- VM/CMS; Whitesmith C compiler v1.0 (This implementation was ported by Rainer Schöpf; basing it upon the UNIX implementation, this took him about one day.)

Macintosh

- At the time of writing (May 1991), John Rawnsley of the University of Warwick had commenced development of a Macintosh port. This will encode the resource and data forks in a manner that will permit the former to be ignored by non-Macintosh systems.

Who's Going to Use VVcode?

Obviously, since the whole concept was invented by the archivists at Aston, the Aston Archive will use VVcode when honouring e-mail requests, and the programs will also be available to browsers calling from sites without a binary NIFTP capability.

Rainer Schöpf has indicated that he will support VVcode on the Heidelberg server, as has George Greenwade at Sam Houston State University in Texas. Nelson Beebe intends to provide it as part of the TUGlib archive at Utah.

Naturally, all of these archives will also provide the sources of the programs, and will, wherever possible, provide complete distribution kits for transfer by (NI)FTP; these kits will include "load-and-go" executables for at least MS-DOS, UNIX, VAX/VMS and VM/CMS. The MS-DOS kit will be included on all physical distributions of TEX for the PC from Aston.

References

[1] Knuth, Donald E. "The New Versions of TEX and METAFONT." *TUGboat* **10**#3, pages 325 - 328, 1989.

[2] Abbott, Peter. "The UKTEX Archive at the University of Aston." *TUGboat* **10**#4, pages 675 - 680, 1989.

[3] Abbott, Peter. "A UK-Based TEX Mail Archive Server." *TUGboat* **9**#3, pages 263 - 264, 1988.

Bitmaps and Halftones with BM2FONT

Friedhelm Sowa
Heinrich-Heine-Universität Düsseldorf
Universitätsrechenzentrum
Universitätsstraße 1
D-4000 Düsseldorf
FRG
Bitnet: `tex@dd0rud81`

Abstract

The program BM2FONT converts different kinds of bitmap files to TeX fonts and writes an input file for integration of those graphics into documents. It is the link between a lot of graphic systems and TeX. The main part of BM2FONT is the conversion of colored pictures to halftone output. This paper describes the method of graphics integration done by BM2FONT and the most important aspects of the program.

Integration of graphics into TeX has been done in many ways within the last few years. Most of those methods use the `\special{whatever}` primitive that needs the special features of driver programs. Most of the public domain drivers do not support this kind of graphics integration. That seems to be the reason why the `special` primitive can't become the base for standard graphics integration. Another reason may be the lack of knowledge by TeX about the exact dimensions of the picture or the possible time lag when writing the code for the `special` command and the corresponding text into the `dvi` file.

The best way seems to be to treat graphics as text. In general, drivers do not have any problems printing documents that contain ordinary text. The only difficulty with this is to find the lowest common level of driver programs. In other words: What are their known bugs and what should a font look like to avoid falling into the trap? The answers to this question are:

- The character of a graphic font should be up to half an inch wide and high.

  We will not have overflow errors because of large run counts contained in packed font files.
- The graphic font should not contain more than 65.536 bytes.

  Even on small systems, drivers can load the font by allocating memory dynamically.

Taking into consideration the above, we can be sure of being successful when using most of the existing drivers for printing our documents.

But why not all of the existing drivers? Another lowest common denominator is the correct decoding of the operation codes in a dvi file. Inspite of the fact, that some of them are not used by the current version of TeX, some drivers (especially on small systems) have reduced the ability of using all the fonts, that are allowed by TeX. Some drivers dynamically load the font files into memory and stop the run, when the available memory is exhausted. Those drivers are not correct. There should exist only those limitations, which are concerned to TeX.

The steps to publishing a document should:

1. Make pictures using a suitable graphics system.
2. Translate the pictures by BM2FONT to a language TeX is able to understand.
3. Write the document.
4. TeX it.
5. Print it.

At no point of the process does the author have to be concerned with compatibilities with the output device of the publisher.

Supported Bitmap Formats

Let's start with unsupported vector graphics. This type of graphics information has to be converted to a bitmap format for integration via fonts. There are a lot of conversion programs available, both public domain and commercial, that do that job. So it would have been redundant to write any code concerning that item.

The decision on what formats to support was very easy to make. Only those bitmap formats that have become a quasi standard on the software market could be considered for the choice. The other condition was the availability of documentation. So

BM2FONT now supports the following bitmap formats:

PCX: The PCX format was introduced by ZSOFT. It uses runlength encoding and allows up to 256 colors (up to version 3.0, 16 colors).

TIFF: The Tag Image File Format of Aldus is one of the most used bitmap formats, especially for scanned pictures. The different compression methods of this format are not supported by BM2FONT.

IFF: The IFF standard was introduced by Electronics Arts for storing graphics in files. It was developed for the Amiga, and is now spread all over the PC world.

GIF: The graphics interchange format was developed by CompuServe in 1987. It uses the LZW-algorithm for data compression. If BM2FONT reads a fragmented file, it only uses the first picture; the following ones are ignored.

BMP: The device-independent bitmap format is found in the Windows world. It supports bitmaps as well as RLE compression and allows up to 256 colors. BM2FONT does not support RLE compressed pictures. The reason is that no examples have been available to the author to test the decompression coding.

IMG: The GEM Image File Format uses RLE, bitstreams, patterns, and repetitions. It is used by a lot of graphics systems, and not only on PCs.

CUT: The CUT-format is used, among others, by the ImagePro system, which captures pictures with a video camera. It supports up to 256 colors and uses runlength encoding compression.

Bitmaps: If the used bitmap format is not on this list and conversion to one of those formats is impossible, the picture can be extracted and written as pure bitmap by the user. BM2FONT accepts pure bitmaps, too.

Steps of integration

There is a picture file named `cheeta.gif` that seems to be good enough to illustrate an article. The document will be printed on a 1200-dpi device. Now it's time to use BM2FONT. To make sure that all files will be in the right directories, we set environment variables like

```
set texinputs=\tex\inputs
set texfonts=\tex\fonts\tfm
set dirpxl=\tex\fonts\pk
```

After that we run the program:

```
bm2font cheeta.gif -u5 -h1200 -v1200 -ry
```

and get the following files:

```
\tex\fonts\tfm\acheeta.tfm
        up to
\tex\fonts\tfm\xcheeta.tfm
\tex\fonts\pk1200\acheeta.pk
        up to
\tex\fonts\pk1200\xcheeta.pk
\tex\inputs\cheeta.tex
```

The file `cheeta.tex` contains the TeXnical description of our picture, generated for a rather high-resolution device.

```
\newbox\cheetabox
\newdimen\cheetaw
\font\acheeta=acheeta
\font\bcheeta=bcheeta
⋮
\font\wcheeta=wcheeta
\font\xcheeta=xcheeta
\setbox\cheetabox=\vbox{\hbox{
\acheeta !\bcheeta !\ccheeta !%
\dcheeta !\echeeta !\fcheeta !}}
\cheetaw=\wd\cheetabox
\setbox\cheetabox=\hbox{\vbox{
\hsize=\cheetaw
\parskip=0pt\offinterlineskip\parindent0pt
\acheeta !\bcheeta !\ccheeta !%
\dcheeta !\echeeta !\fcheeta !\vskip0pt%
\gcheeta !\hcheeta !\icheeta !%
\jcheeta !\kcheeta !\lcheeta !\vskip0pt%
\mcheeta !\ncheeta !\ocheeta !%
\pcheeta !\qcheeta !\rcheeta !\vskip0pt%
\scheeta !\tcheeta !\ucheeta !%
\vcheeta !\wcheeta !\xcheeta !}}
\ifx\parbox\undefined
   \def\setcheeta{\box\cheetabox}
\else
   \def\setcheeta{\parbox{\wd\cheetabox}
              {\box\cheetabox}}\fi
```

All we have to do now is to put `cheeta.tex` into our document and then set it where we want to see it.

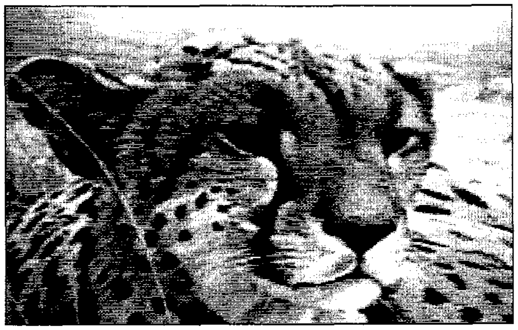


Figure 1: Cheeta, looking for a fat mouse

Dithering

For generating halftone pictures, single pixels must be represented on paper by black dots of different size. So we get the impression of different grey shades. Within a defined grid of bits for every pixel of the original picture, a certain amount of bits are set from white to black. This is called dithering.

If a grid with u pixels in the width and c pixels in the height is used, we will have an amount of G grey shades, calculated by $G = 4uc$. A 3x3 grid will have the following rasters:

	raster			
shade	1	2	3	4
1	● ○ ○ ○ ○ ○ ○ ○ ○			
2	● ○ ○ ○ ○ ○ ○ ○ ○	○ ○ ○ ○ ○ ○ ○ ○ ●		
3	● ○ ○ ○ ○ ○ ○ ○ ○	○ ○ ○ ○ ○ ○ ○ ○ ●	○ ○ ○ ○ ○ ○ ● ○ ○	
4	● ○ ○ ○ ○ ○ ○ ○ ○	○ ○ ○ ○ ○ ○ ○ ○ ●	○ ○ ○ ○ ○ ○ ● ○ ○	○ ○ ● ○ ○ ○ ○ ○ ○
5	● ○ ○ ● ○ ○ ○ ○ ○	○ ○ ○ ○ ○ ○ ○ ○ ●	○ ○ ○ ○ ○ ○ ● ○ ○	○ ○ ● ○ ○ ○ ○ ○ ○
6	● ○ ○ ● ○ ○ ○ ○ ○	○ ○ ○ ○ ○ ● ○ ○ ●	○ ○ ○ ○ ○ ○ ● ○ ○	○ ○ ● ○ ○ ○ ○ ○ ○

⋮

	raster			
shade	1	2	3	4
20	● ● ○ ● ● ○ ● ○ ○	○ ○ ● ○ ● ● ○ ● ●	● ○ ○ ● ● ○ ● ● ○	○ ● ● ○ ● ● ○ ○ ●
21	● ● ● ● ● ○ ● ○ ○	○ ○ ● ○ ● ● ○ ● ●	● ○ ○ ● ● ○ ● ● ○	○ ● ● ○ ● ● ○ ○ ●
22	● ● ● ● ● ○ ● ○ ○	○ ○ ● ○ ● ● ● ● ●	● ○ ○ ● ● ○ ● ● ○	○ ● ● ○ ● ● ○ ○ ●
⋮	⋮	⋮	⋮	⋮
34	● ● ● ● ● ● ● ● ●	● ● ● ● ● ● ● ● ●	● ● ○ ● ● ● ● ● ●	● ● ● ● ● ● ○ ● ●
35	● ● ● ● ● ● ● ● ●	● ● ● ● ● ● ● ● ●	● ● ● ● ● ● ● ● ●	● ● ● ● ● ● ○ ● ●
36	● ● ● ● ● ● ● ● ●	● ● ● ● ● ● ● ● ●	● ● ● ● ● ● ● ● ●	● ● ● ● ● ● ● ● ●

For every row of the dithered picture the rasters r_i are used in the order:

r_1	r_4	r_1	r_4	r_1	r_4	...
r_2	r_3	r_2	r_3	r_2	r_3	...
r_4	r_1	r_4	r_1	r_4	r_1	...
r_3	r_2	r_3	r_2	r_3	r_2	...
r_1	r_4	r_1	r_4	r_1	r_4	...
r_1	r_4	r_1	r_4	r_1	r_4	...

The quality of a halftone picture depends on the resolution of the output device. A high-resolution device allows us to choose a greater grid with more grey shades than does a low-resolution device. BM2FONT dithers on the base of the matrix

	column						
row	1	2	3	4	5	6	7
1	01	03	06	10	18	26	38
2	02	04	08	12	20	28	40
3	05	07	09	14	22	30	42
4	11	13	15	16	23	32	44
5	17	19	21	24	25	34	46
6	27	29	31	33	35	36	48
7	37	39	41	43	45	47	49

where the rows and columns define the height and width of the grid BM2FONT was told to work with. Each element of the matrix represents a grey shade. The different grey rasters are filled with black pixels in the corresponding positions up to that grey shade.

On paper a grey dot is composed by four different grey rasters so that we make one grey dot from four original pixels. This means that each grey shade may appear with four different expressions.

Gradation

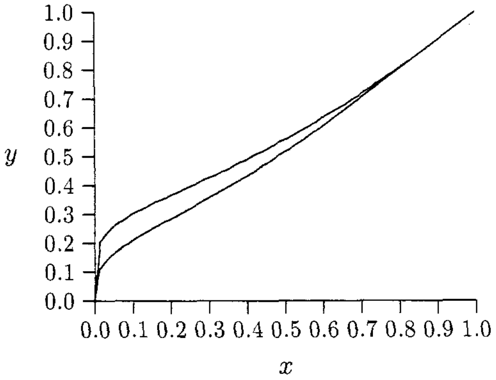
As a lot of printers tend to become dark very quickly in the darker part of the grey scale, it is necessary to compensate for this behavior. This is done by gradation.

The following function describes the development of the output values of halftone pixels, which are changed depending on parameters t and z. In the darker area ($x = 0$ represents black), an intensive lightening is done that decreases in the brighter area until it disappears in point x_0.

$$f(x) = \begin{cases} \frac{1}{2x_0^\alpha} x^{1+\alpha} + \frac{1}{2} x_0^\alpha x^{1-\alpha} & 0 \le x < x_0 \\ x & x \ge x_0 \end{cases}$$

$$\text{with } \alpha = \tfrac{t}{100},\ x_0 = \tfrac{z}{100}$$

With the default value 70 of both t and z BM2FONT generates a picture with low lightening, which works on 70 percent of the grey scale.



The upper curve represents the lightening when using value 80 for parameter t and 90 for parameter z. Because of memory problems — the plot was done by PICTEX — only very few points were used. That's why the curves look rather bad.

Gradation will not occur if BM2FONT was called with -t0. In the case of working on an original picture with only 16 or 32 grey shades or colors, gradation may produce a halftone picture by losing a lot of information. Then lightening should be done by using parameter bx, where x stands for the amount of shades to be subtracted from the available number of shades.

Error Distribution

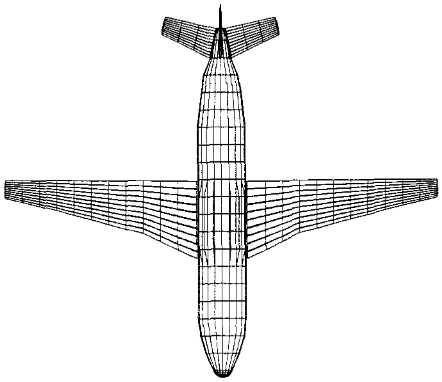
When generating halftones, there is a problem concerning the number of colors in the input file and the number of grey shades that are available depending on the raster size. Usually there is a difference, which means that we will loose some information when reducing the amount of colors. We get rounding errors. A well-known algorithm (Floyd/Steinberg) solves this problem by distributing the rounding errors to the neighboring pixels cumulatively.

BM2FONT uses a similar method by using different expressions of rasters. In spite of that, the loss of information is sometimes visible by very harsh changes of neigboring shades. For smoothing those changes, BM2FONT walks throw the pixel rows, looking for neighboring pixels with different colors. In this phase, the colors are already scaled down for output. Then the grey value becomes the average of the neighboring pixels by considering the rounding error. In most cases, a much better quality picture can be achieved by this method.

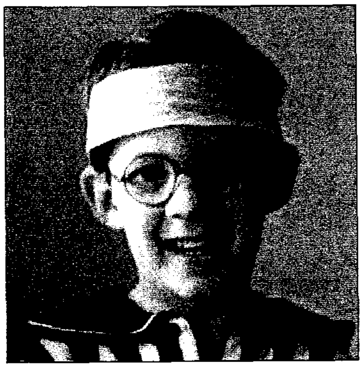
Nevertheless, sometimes it is not useful to distribute rounding errors this way. It usually makes sense when working on photographs. Comics or art work should be generated by switching off error distribution.

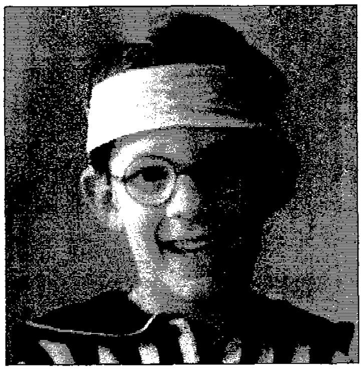
Some Examples

The first figure is converted from a vector graphics file (HPGL) to a PCX file by using the public domain hp2xx, which is written by Heinz W. Werntges, Heinrich-Heine-University Düsseldorf.



The following haltone images show the same picture of the same size. The second image is done by using a greater size of the dot raster.





Bibliography

Born, Günter. *Referenzhandbuch Dateiformate*, Addison Wesley Publishing Co., 1990.

Burger, Peter, and Duncan Gillies. *Interactive Computer Graphics,* Addison Wesley Publishing Co., 1989 .

Childs, Bart, Alan Stolleis, and Don Berryman. "A Portable Graphics Inclusion." *TUGboat* 10(1), pages 44 – 46, 1989.

Clark, Adrian F. "Halftone Output from TEX." *TUGboat* 8(3), pages 270 – 274, 1987.

Heinz, Alois. "Including Pictures in TEX." Pages 141 – 151 in *TEX Applications, Uses, Methods: TEX88 Proceedings*, Malcolm Clark, ed. (Ellis Horwood Series in Computers and their Applications, 1990.)

Knuth, Donald E. *The TEXbook, Computers & Typesetting* — A. Addison Wesley Publishing Co., 1986.

Knuth, Donald E. *The* METAFONT*book, Computers & Typesetting* — C. Addison Wesley Publishing Co., 1986

Knuth, Donald E. "Fonts for Digital Halftones." *TUGboat* 8(2), pages 135 – 160, 1987.

Messinger, Heinz. *Langenscheidts Großwörterbuch, "Der kleine Muret-Sanders", Deutsch-Englisch.* Langenscheidt, 1982.

Pickrell, Lee S. "Combining Graphics with TEX on IBM PC-Compatible Systems and LaserJet Printers." *TUGboat* 11(1), pages 26 – 31, 1990.

Pickrell Lee S. "Combining Graphics with TEX on IBM PC-Compatible Systems and LaserJet Printers, Part II." *TUGboat* 11(2), pages 200 – 206, 1990.

Rogers, David F. "Computer Graphics and TEX, A Challenge." *TUGboat* 10(1), pages 39 – 44, 1989.

Simpson, Richard O. "Nontraditional Uses of METAFONT." In *TEX Applications, Uses, Methods: TEX88 Proceedings*, Malcolm Clark, ed. Pages 259 – 271. (Ellis Horwood Series in Computers and their Applications, 1990.)

Ulichney, Robert. *Digital Halftoning.* The MIT Press Cambridge, Mass., 1987

Wilcox, Patricia. "METAPLOT: Machine-Independent Line Graphics for TEX." In *TUGboat* 10(2), pages 179 – 187, 1989.

TEX and Those Other Languages

Yannis Haralambous
101/11, rue Breughel, 59650 Villeneuve d'Ascq, France
33 20052880; FAX: 33 20910564
Bitnet: yannis@frcitl81

Abstract

This paper relates the author's experiences while creating a TEX package for typesetting in several languages of scholarly interest, such as Arabic, Hebrew, Syriac, Greek, Armenian, and Saxon. First, a combined use of METAFONT and a PostScript font creation program is described and commented; next, the TEXnical problems (and their solutions) with relation to each language are presented; finally, some new ideas for further development and application of TEX in non-Latin alphabet transmission through the electronic communication media are given.

Introduction

TEX's box-oriented approach to typesetting makes it the ideal tool for "exotic" languages which need two-dimensional constructions. The lack of WYSIWYG is compensated by the infinite possibilities of a programming language and the compatibility between different devices and electronic communication medias.

But let's start from the beginning. After developing the Arabic–Persian–Ottoman TEX system presented in the summer of 1990 ([2]), I was so delighted by TEX and METAFONT's possibilities in this area that I decided to continue making more alphabets for scholars. I found that this domain was underdeveloped and that many scholars were still adding non-Latin alphabet text by hand or using primitive low-resolution bitmap fonts which they had to create themselves.

For the first round (programmers call it *version*), I attacked Hebrew, Syriac, Armenian, Greek, epigraphical Greek, and Saxon. It was a beautiful experience (which took all my holidays, week-ends, free hours, and many entire nights). The result is *Scholar*TEX, a package of fonts, preprocessors, macros, documentation, and everything a *non-TEX-expert* scholar could need for his or her typesetting activities.

Since many public domain packages can be extremely interesting to scholars but sometimes difficult to find, or in need of some adaptation, I thought that *Scholar*TEX should be a platform for distribution of related important public domain software (with explicit notification of its status and origin). In this way I was kindly allowed to include EDMAC ([8]), a version of TEX–XET featuring sbTEX (for PCs), and the wsuipa IPA fonts ([1]).

A short presentation of *Scholar*TEX was made at the DANTE meeting, January 1991, in Vienna; a more general one (with an expanded part) will be made at the 6th European TEX Conference, September 1991, in Paris. In this paper I would like to describe some techniques and experiences in making the fonts and present some TEXnical problems I encountered, with their solutions.

How to Make a Font

The aim of this section is to show how METAFONT and a PostScript font creation program (in this case, Letraset Fontstudio v2.0) can be combined in a complementary way to produce an aesthetically satisfying font that would be very cumbersome to produce with either alone.

I will subdivide this method of creating a font for some "exotic" language into eight steps, each technically and emotionally different. Please note that by no means do I pretend this to be the best solution. It is more a kind of *poor man's* method, which can be used at home with the least possible equipment. Much better results could be achieved by high technology, and in far shorter time; but my method is more fun!

First step: choosing the types. This is the "outside world" part of the job. It involves looking in libraries, finding highly specialized grammars and dissertations, and trying to extract the scarce information you need about the letters, punctuation marks, symbols, etc. Sometimes you will feel like

Umberto Eco, finding out many little-known things about the past and collecting interesting information or theory[1] on the history of languages, alphabets, and typography.

But the focus of your search is to find some reliable, preferably large samples of the types you want to reproduce. You have to study them well, to see all the small typographical details, and try to find out which of these belong to tradition and which are just the result of the typographer's relative ignorance of the specific "exotic" language. Once you have found enough sources and discussed details with some specialists, you have to choose one principal type which you will "copy" and two or three others which will give you ideas for modifications.

Second step: taking photos. Now you become a photographer. You have to take pictures of small objects called letters which live on rough and not always flat paper; these pictures should be very clear and identically scaled.

The paper problem can be solved if you press a glass plate over your paper (which is not always easy, as in the case of large, old books). To have clear pictures, use very strong light sources; then you can focus more easily and use smaller lens apertures with more depth of field. As for the scaling problem, you can insert some reference object in each picture. I did all of my pictures with an old and faithful Olympus OM-2, a bellows, and an inverted f:3.5/24mm lens (only recently I bought the special f:2/20mm macro lens).

Don't hesitate to make pictures even of punctuation marks, dashes, and surrounding Latin characters — everything is important.

Third step: first paper draft. Put the developed film in the darkroom projector and copy the outlines of your characters with a black pencil on white letter format paper. At the same time you can also trace the "invisible" extensions of your character's strokes. You can see an example in Fig. 1; it's the well-known Hebrew letter aleph.[2] Since printed characters are not always as smooth as you would expect under high magnification, you may have to make some corrections while copying. I usually make these first corrections in some other colour.

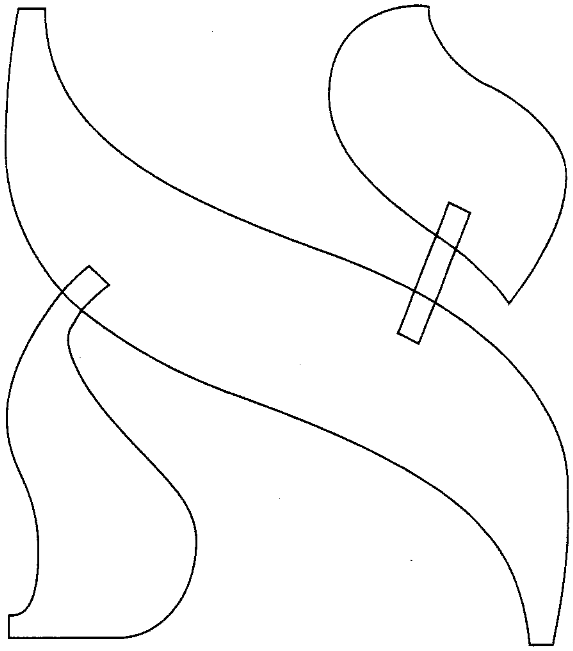


Figure 1: A first paper draft.

Fourth step: setting guidepoints and pen positions. This is the most important and most artistic part of the job. Now you are a designer. You have pens of every possible shape and can make pen strokes with them; you can also fill outlines; the only restriction is that all curves must be Bézier curves. You are entirely free, *but* there are two fundamental rules:

1. use as few Bézier curves as possible; and
2. don't forget METAness.

The first rule is to remember that your curves are so beautiful, not — or *not only* — because you are a great designer, but because they are Bézier curves, obeying very strict mathematical rules (see the METAFONTbook, p. 131, or Yanai and Berry's paper [13]; and if you haven't done it yet, I sincerily advise you to read "Mathematical Typography" [6]). You must be extremely careful when joining such curves — the result may disappoint you.

The second rule is more METAFONT-specific: all pen positions or guidepoints must keep track of the character's ability to transform according to the parameter values you are going to impose on it. For example, in Fig. 2 — where numbered line segments denote penpositions and the arrows indicate orientation — you see that pen positions 3 and 4 (left stem)

[1] For example, did you knew that the Greek letter alpha A, the Arabic alif ا, and the Hebrew aleph א are derived from the same Phœnician letter ∀ called aleph which means "calf" because it looks like a calf's head? (See [5].)

[2] I don't guarantee this to be the exact shape of the aleph א in font `yhbr`; many changes occurred later.

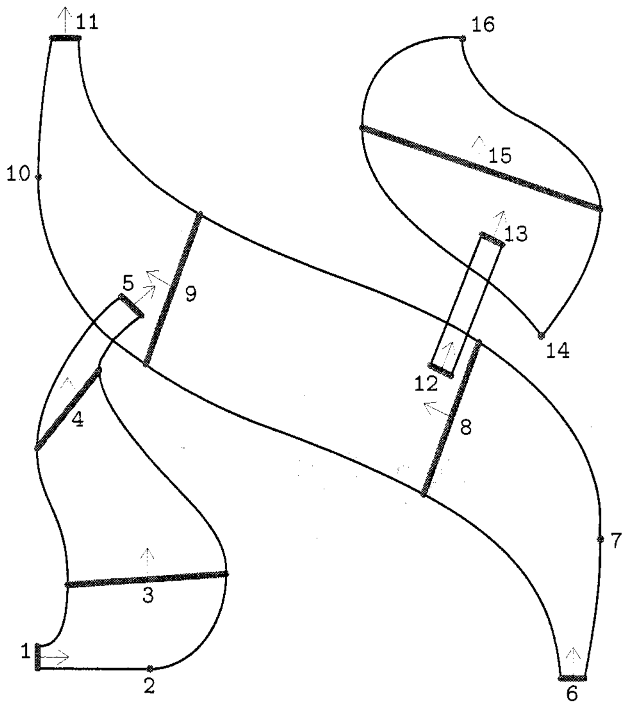


Figure 2: Guidepoints and penpositions.

have been chosen so that the tangents at their left edges are vertical, but the pen angles are different. So if you increase the pen widths, the upper part of the stem will change to a lesser extent than the lower part. It is a good habit to put pen positions and guidepoints at places where tangents to curves are vertical or horizontal. But sometimes this is not enough, as in the case of pen positions 8 and 9 (diagonal strokes).

This step requires the most attention and, at the same time, the most intuitive handling (*if I were a pen stroke leaving here in this direction, where would I go?*). I usually use a second colour for pen positions and guidepoints.

Fifth step: writing and running the METAFONT source. Once you have the outline, you become a METAFONTer. The goal is not only to write all this in METAFONT language, but also to define the necessary parameters to make it possible for your picture to provide other styles (at least **boldface**, since *slanted* is easy, while `typewriter` and sans serif often need a completely different code). You should start with the minimum number of parameters (often `hair` and `stem` are enough for lowercase). If there is a need for other, more special parameters, you can always add them later.[3] If the fourth step has been well done, this one should be straightforward.

Try not to use the `...tension` *xxx*`...` command too often. Every time I tried to use it there was a more natural way which gave a far better result. (Bézier curves are like humans; their best state is the natural one — too much tension spoils them.)

Some points will have to be defined by coordinates. Be careful when defining pen positions; the pen width may change later on, and you must take this into consideration now. In our example, for pen position 3, if you fix the coordinates of point `z3` then by increasing the width, the path `z3l{up}..z4l{up}` would become more and more flat. In this case, you should fix the coordinates of `z3l`. In the same way, pen position 15 was defined by fixing the coordinates of `z15r`, etc.

While you are running METAFONT and visualizing your character on screen, you will already discover many weak points in your draft; you can then go back to step 4 and make the necessary changes. On the other hand, I am always amazed to see how easily one can obtain *exactly* the same curve as in a good sample, which means that the old masters of the past may have used Bézier curves, without knowing it.

Sixth step: going to Fontstudio. By using the following `mode_def` (which changes the definition of `endchar`, and instead of a grid, inserts cropmarks), enter:

```
mode_def fontstudio =
def nothing(text r) = enddef;
pixels_per_inch :=2200; blacker :=0;
o_correction :=.4; fillin :=0;
proofing :=1; fontmaking :=0;
tracingtitles:=1; mag:=2.4;
screen_rows:=1200; screen_cols:=2000;
let makebox=nothing;
enddef;

def endchar = scantokens extra_endchar;
if (proofing>0) and not (mode=fontstudio):
makebox(proofrule); fi
if (mode=fontstudio):
pickup pencircle scaled 1;      %really 1 pixel!
draw (l-10,0)--(l+10,0);draw (r-10,0)--(r+10,0);
draw (l,-10)--(l,10);   draw (l,h-10)--(l,h+10);
draw (r,-10)--(r,10);   draw (r,h-10)--(r,h+10);
draw (l-10,h)--(l+10,h);draw (r-10,h)--(r+10,h);
```

[3] For example, in Arabic, besides `hair` and `stem`, a parameter was needed for the width of the basestroke, which can be completely independent from the widths of other strokes.

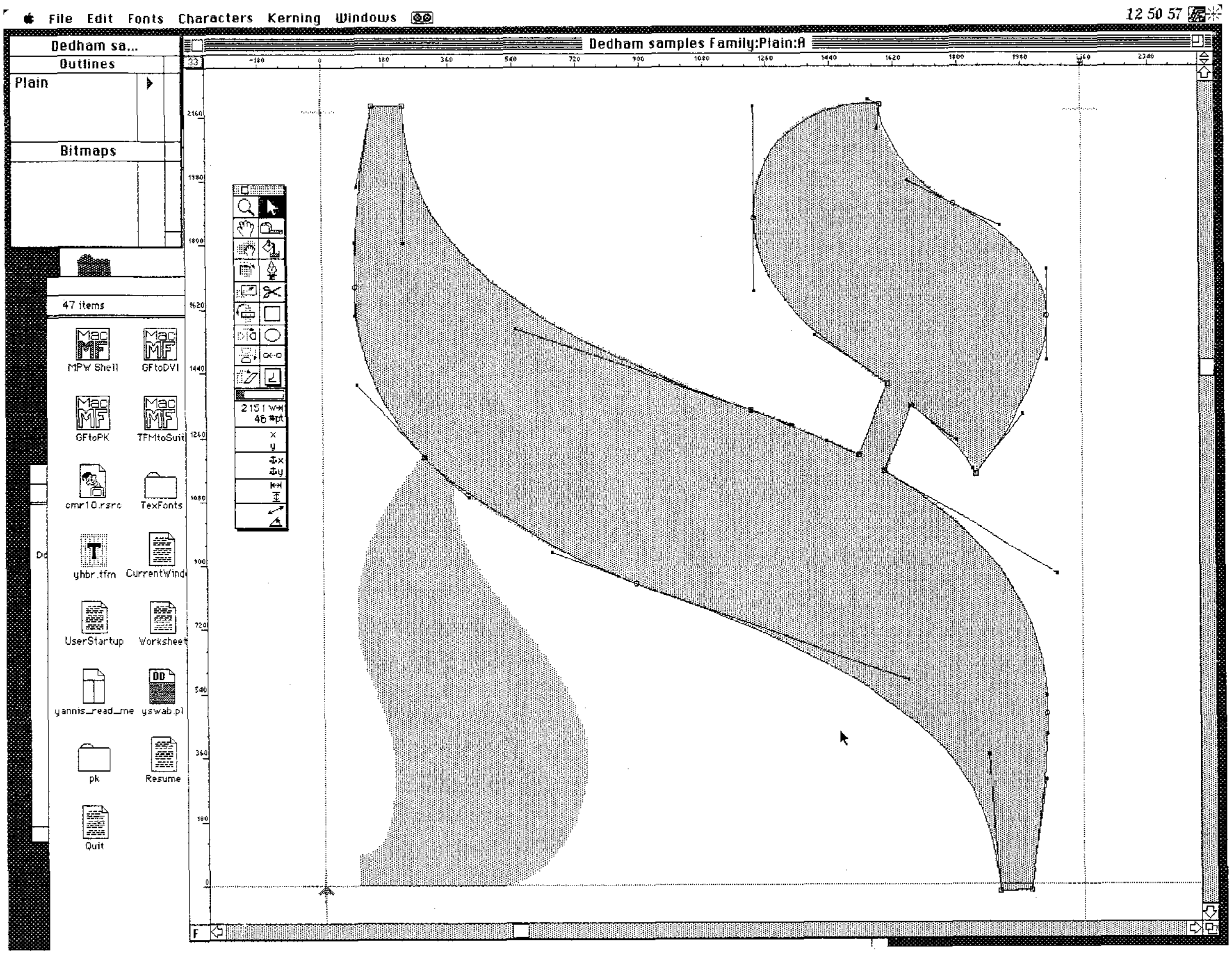
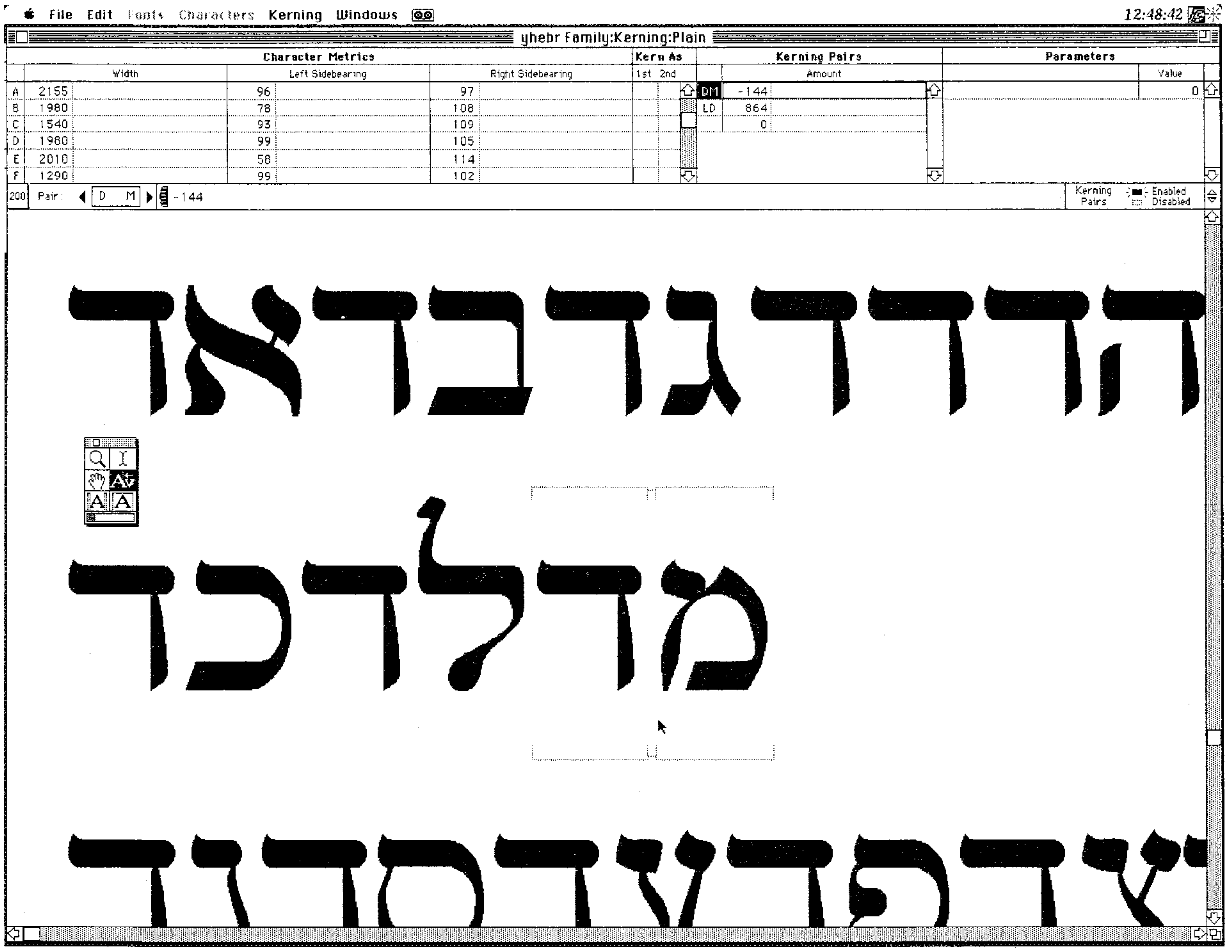


Figure 3: Outline drawing in Fontstudio.

```
fi
chardx:=w; shipit;
if displaying>0: makebox(screenrule); showit; fi
endgroup enddef;
```

I make screen copies of all characters (at the same scale, approximately 10 to 12 inches high for uppercase characters). These screencopies are visualized in Fontstudio's workbench as grey masks. First, you have to place the mask in the right position by identifying the left lower cropmark as point (0,0) and setting the right cropmark to be the character's width (finer corrections can be done later). Then, using Illustrator-like techniques you copy the outline of the character, as in Fig. 3. Don't use the "Autotrace" feature! The point is that by drawing the character yourself, you can follow the guidepoint scheme you have established in step 4. You might argue that this way of doing it is unprofessional and that there are METAFONT→PostScript translation programs (for example, see Yanai and Berry [13]). This may be the viewpoint of a programmer, but not that of a designer. The best work is still done by hand. Seeing the character in front of you, and at this size, enables you to learn it better. Playing with Bézier curves on the screen will give you ideas, and you may even go back to step 5 and change your METAFONT source (this happened to me several times).

Both programs contribute in a complementary way: METAFONT keeps the uniformity (all thin lines will be exactly of `hair` width, all stems will be of `stem` width, and so on), while Fontstudio brings new ideas and a deeper understanding of the character's shape because of its WYSIWYG features. This is true for the new fonts you are just creating. If you just want to make outlines from existing fonts, you can achieve an acceptable degree of precision by this method, but a translation program will do it more quickly.


File Edit Fonts Characters Kerning Windows
12:48:42
yhebr Family:Kerning:Plain
Kerning Pairs ■ Enabled ▨ Disabled

Character Metrics				Kern As	
	Width	Left Sidebearing	Right Sidebearing	1st	2nd
A	2155	96	97		
B	1980	78	108		
C	1540	93	109		
D	1980	99	105		
E	2010	58	114		
F	1290	99	102		

Kerning Pairs	
	Amount
DM	-144
LD	864
	0

Parameters	
	Value
	0

200 Pair: D M -144

Figure 4: Kerning in Fontstudio.

Seventh step: kerning and going back to METAFONT. A big advantage of Fontstudio is visual kerning. As you see in Fig. 4, you can display any text on your screen, select a character, and move it to the correct position. All kerning pairs are written in alphabetical order, and you just have to copy their list to make a METAFONT `ligtable`. If you see a character occurring in pairs with more than half of the others, you should change its margins; in Fontstudio this is done visually and in METAFONT just by changing the value of `adjust_fit`.

Eighth step: making `tfm` files. Fontstudio delivers an AFM metric file which you can easily convert to a PL one. I advise you to compare this PL file with the one originally created by METAFONT; this will prevent many errors. Of course, the PL file coming from Fontstudio is useless for TEX; take, for example, characters such as the Computer Modern "large math operators" from file `bigop.mf` (see Knuth 1986, p. 103 - 121]); they all have considerable depth, but their boxes are of depth zero, and this is not accepted by.Fontstudio. You will of course use the `tfm` and PL files created by METAFONT. Some systems (such as *Textures*) may need small changes in the PL file; for example:

```
(FAMILY YARBA)
(CODINGSCHEME PostScript YarbaNaskhi)
```

where `YARBA` is the font name for QuickDraw and `YarbaNaskhi` for the printer. Other systems, e.g., OzTEX, keep a list of all PostScript font names in a special file (for OzTEX version 1.3, any file in the `Configs` folder).

We have seen how the combined use of METAFONT and Fontstudio allows easy and efficient font creation and solves many problems such as kerning or better understanding of Bézier curves in the case of METAFONT, and drawing pen strokes and keeping homogeneity of all characters in the case of Fontstudio. I would conclude by calling METAFONT's approach a rationalistic one, while Fontstudio's approach is empirical; and you know that both are important.

Problems with Languages

In the previous section we saw how to make a font for an exotic alphabet. I specify that this approach is valid only if one or two styles are required (plain and **boldface**, for example). If you are planning to use a complete library of styles (such as `CM`), you will have to spend infinitely more time in organizing and checking METAness parameters.

Now we have to use these fonts. Several problems arise, and I propose to examine each alphabet separately.

Arabic

صحيح اننا نستفيذ من هذه الوضعية بتفتحنا
على العالم الخارجي ولكن الاستعمال
المكثف للغة الفرنسية ينتج عنه حيف في
المجتمع المغربي لايسمح بتنمية متوازنة ۞

Most of the details about the Arabic alphabet have already been set forth in Haralambous, 1990. Since the first `YARB` version, I have added new characters to cover also Pashto, old and modern Urdu and Malay; I have also entirely changed the preprocessor (now called `ysemtex`): the characters are taken from 3 *real* fonts and about 14 *virtual* ones (the virtual fonts are used for the precise placement of diacritical marks). The input encoding, as well as all escape characters, are now user-definable; this data is stored in a text file which is loaded while running. You have the choice between plain TEX and TEX-XET output, for the same input.[4]

Syriac

ܘܠܡܫܡܫܘ ܐܬܐ ܗܘܐ ܒܪܐ ܚܕ
[illegible]
ܒܢܝܢܗ ׃ ܒܪܐ [illegible] [illegible]
[illegible] ׃

From the typographical point of view, Syriac is structured like Arabic, so I just had to define a new escape character and tell `ysemtex` which input and output data to load.

The problem with Syriac is the lack of typographical evolution in the last few centuries. There are at least two kinds of Syriac alphabet: *Estrangello* and *Serto*. I began with Estrangello (Serto will follow). The Estrangello type I encountered in most books is just an imitation of handwriting. I tried to make some aesthetic improvements, which I had to withdraw *immediately* when I showed the font to specialists.

Hebrew

שִׁמְעוּ הַדָּבָר הַזֶּה פָּרוֹת הַבָּשָׁן אֲשֶׁר
בְּהַר שֹׁמְרוֹן הָעֹשְׁקוֹת בַּלִּים הָרֹצְצוֹת
אֶבְיוֹנִם הָאֹמְרוֹת לַאֲדֹנֵיהֶם הָבִיאָה
וְנִשְׁתֶּה׃

Since it was strictly forbidden to change the Holy Texts and the Jewish people saw the oral tradition disappearing (because of changes in the pronunciation of the language), they decided to add diacritical marks to the Text, starting with vowels and going to more and more specialized and rare symbols. Today, one can find up to four of these symbols which, for the sake of briefness, we will call *accents*, on each letter (plus eventually the *dagesh* point inside the character). TEX can handle this situation very well by using box constructions. The output provided by `ysemtex` contains the information on accents in the following way: for each letter which contains at least one accent, a macro

$$\texttt{"}n_0o_0\texttt{-}n_1o_1\texttt{-}n_2o_2\texttt{-}n_3o_3\texttt{-}n_4o_4\texttt{!}$$

is used, where o_0 is the octal code of the character which is in font n_0, and o_1 to o_4 the octal codes of the four possible accents (in fonts n_1 to n_4), starting from upper left and finishing with lower right. Of course, some combinations of accents and characters deserve special accent positioning and are contained as separate characters in the font (e.g. ךּ, ךְ, ךָ, תּ, לֹ, etc.).

Contrary to Syriac, Hebrew has a very rich typographic tradition (Tamari, 1989). I chose a rather simple type which better suits a 10-point text than big titles (a real calligraphic type for head titles is planned). The accents were taken from the TABULA ACCENTUUM of the BIBLIA HEBRAICA STUTTGARTENSIA.

[4] Even if you use TEX-XET, the preprocessor is unavoidable because of the Arabic character forms.

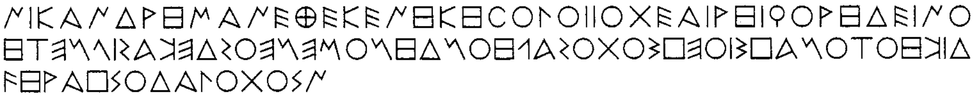


Figure 5: Sample of Greek epigraphical text.

Greek

> ῾Ο Βασιλεὺς Δουριοδάνας, ἰδὼν παρατεταγμένον τὸν στρατὸν τῶν Πανδοϊδῶν προσελθὼν τῷ διδασκάλῳ Δρόνᾳ ἔλεξε τάδε· «Σκόπει διδάσκαλε τόν μέγα τοῦτον στρατὸν τῶν υἱῶν τοῦ Πανδοῦ, τὸν παρατεταγμένον ὑπὸ τοῦ εἰδήμονος μαθητοῦ σου Δρυσταδεούμνα, υἱοῦ τοῦ Δρουπάδα».

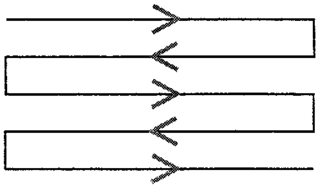
Some Greek fonts have already been designed by Silvio Levy ([9]). For several reasons — one is that I found a really beautiful type in a Greek book — I decided to make some new fonts from scratch, except of course for uppercase letters, which remain those of the CM family. The circumflex accent is encoded as = and *all* accented characters are included as ligatures; thus, to start typesetting in Greek, you just have to select the Greek font.

I included the symbols |, ‖, ⟦, ⟧, {, }, ⟨, ανδ ⟩ , used in epigraphical texts by Oxford editions (see [11]). All possible ⟨*character* + *accent*⟩ combinations are separate characters in the font and can be reached either by 8-bit input, or by 7-bit ligatures. You will find a discussion on encoding and transliteration problems concerning both ancient and modern Greek in [4].

Epigraphical Greek. See Fig. 5 for a sample of the fonts. Designing this font was straightforward; it is meant to be simple and most of the lines are straight. The problems which arose were more of a TEXnical nature:

1. there are no blank spaces between words;
2. lines are imposed; and
3. all lines (except the last) should be of equal length.

And adding the fact that most of the inscriptions are written βουστροφηδὸν (which means that text direction *as well as character shapes* alternates at each line), this is enough to cause a typographer's headache.



Another — more TEX-related — problem is encoding: there are 14 different alphas, 10 different betas, etc. The user would like a readable text and not a sequence of `\char'`*xxx*`\char'`*yyy*`\char'`*zzz*. Even a sequence such as `\A12\B07\G03\A04` would not be very readable.

Here are the solutions I propose:

1. The uppercase ASCII characters `A..Z` are locally set to be active. They receive definitions of the form `\def␣A{\char'xxx\␣}` where *xxx* is set by the user as the octal font position of the required alpha, depending on the epoque and idiom in which the inscription is written (one can always use ordinary macros for exceptions, as long as their names do not contain uppercase characters).
2. You may be wondering what the `\␣` stands for. Well, the second idea is to set the blank space of length 0, expandable to `10u#`:

   ```
   font_normal_space 0;
   font_normal_stretch 10u#;
   font_normal_shrink 0;
   ```

   (where, at 10pt size, `u#` is as usual `20/36pt#`).

   You proceed in the following way: choose the potentially longest line (I should write an algorithm to make that automatic) and write it first inside a macro

   ```
   \longestline{...}
   ```

   Then the contents of all lines will be placed in centered boxes with this length, and by the expanding feature of ␣, letters will be equally spread inside each box.[5]
3. The βουστροφηδὸν problem is solved by having a second font which is the mirror image of the first. You can choose between writing your text from left to right, or from right to left (here the well-known `\reflect` macro is applied).

[5] Actually, the definitions of the active uppercase characters are slightly more complicated because of the last character of the line, which should not be followed by a `\␣`.

Armenian

ԳԱՅԼԸ ՊԱՅՏԱՐ

Գայլը երբ մի օր շրջում էր լեռներում, արօտների մէջ կապած մի էշ տեսաւ: Էշը հասկացաւ, որ իր վերջն եկել է, ուստի դիմեց գայլին և ասաց. «Գոհութի՛ն Աստուծոյ, որ քեզ ինձ մօտ բերաւ, ո՛վ գայլ: Ուրախ եմ, որ ուտես ինձ և ազատես այս սուտ կեանքից.»

When I started working on Armenian, I thought it would be a straightforward job. Most of the lowercase characters are made of straight lines; uppercase characters exist in two forms: plain and calligraphic. Since slanted as well as upright characters are used (as a matter of fact, their rôles have been exchanged), this makes four fonts. Text is written from left to right, hyphenation is allowed — Dikran Karagueuzian offered me his hyphenation table for Armenian, which I adapted to my 8-bit and/or 7-bit ligature-based font encodings — so there should not have been any particular problem.

There was: *the kerning!* Armenian has many combinations such as լ + ո = լո, ա + յ = այ where kerning is unavoidable. Armenian printers have solved the worst cases by creating the following beautiful ligatures:

ե + ւ	մ + է	մ + ի	մ + խ
և	մէ	ﬕ	ﬗ

մ + կ	մ + ե	մ + ն	վ + ն
մկ	ﬔ	ﬓ	ﬖ

After a night of Fontstudio kerning — Armenian has 38 characters in uppercase (U) and lowercase (L) form, the number of UU, UL and LL combinations is 4332... — I had a minimum of 450 kerning and ligature pairs. In the METAFONTbook Prof. Knuth asserts that "Novices often go overboard on kerning. Things usually work out best if you kern by at most the half of what looks right to you at first, since kerning should not be noticeable by its presence (only by its absence)." But surely he was not thinking of Armenian.

Saxon

Æꝼꞇeꞃ

uꞃe Dꞃıhꞇneꞅ Hælanꝺeꞅ Cꞃıꞅꞇeꞅ ᵹebẏꞃꞇıꝺe an þuꞅenꝺ ƿınꞇꞃa· ⁊ ꞅeoꝼan ⁊ hunꝺeahꞇaꞇıᵹ ƿınꞇꞃa· on þam an ⁊ ꞇƿenꞇıᵹann ᵹeaꞃe þæꞃ þe Pıllelm ƿeolꝺe ⁊ ꞅꞇıhꞇe Enᵹlelanꝺ· ꞅƿa hım Goꝺ uðe· ᵹeƿeaꞃð ꞅƿıðe heꝼelıc ⁊ ꞅƿıðe ƿolꝺbeꞃenꝺlıc ᵹeaꞃ on þıꞅꞅum lanꝺe. Sþẏlc coðe com on mannum· ꝥ ꝼullneah æꝼꞃe þe oðeꞃ man ƿeaꞃð on þam ƿẏꞃꞃeꞅꞇan ẏꝼele· þeꞇ ıꞅ on þam ꝺꞃıꝼe· ⁊ þeꞇ ꞅƿa ꞅꞇꞃanᵹlıce ꝥ mænıᵹe menn ꞅƿulꞇon on þam ẏꝼele.

In the absence of an Old English font, scholars often use characters from the International Phonetic Alphabet to represent ʒ, þ, ð, etc. As a matter of fact, a cmr-like font (with IPA characters taken from wsuipa) for a "modern" output of the same input text is provided below:

> Æfter ure Drihtnes Hælandes Cristes ʒebẏrtide an þusend vintra; and seofan and hundeahtatiʒ vintra; on þam an and tventiʒann ʒeare þær þe Pillelm veolde and stihte Enʒleland; sva him God uðe; ʒevearð sviðe hefelic and sviðe voldberendlic ʒear on þissum lande. Sþẏlc coðe com on mannum; þæt fullneah æfre þe oðer man vearð on þam vẏrrestan ẏfele; þet is on þam drife; and þet sva stranʒlice þæt mæniʒe menn svulton on þam ẏfele.

The symbol ꝥ is an abbreviation for þæꞇ, and ⁊ is a runic symbol for "and". There were no problems with this font: the j J encoding is used for the thorn character þ Þ because of the TEX input transliteration of the Greek θ which has the same sound (in modern Greek!). The pointed ẏ Ẏ are separate characters.

Old German

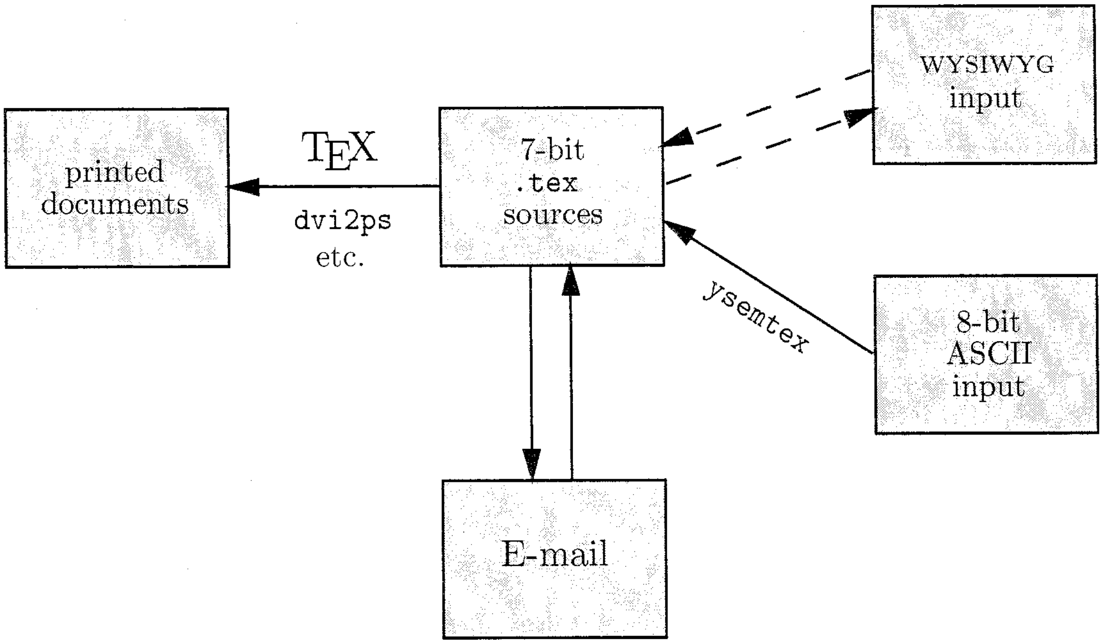
Het verdriet van België

Toen kwam Zuſter Adam achter de doornhaag. Louis was zeker dat hij net vóór zij verſcheen het geritſel van haar kleed gehoord had, toen het langs de doornen ſtreek. Zij bleef ſtaan, niet lang, met gevouwen armen, zodat de wijde mouwen voor haar middenrif een zwart altaartje vormden. Dondeyne zag haar ook.

Old German fonts have been described in [3]. I included two of them (Fraktur and Schwabacher) in *Scholar*TEX, for scholars who want to distinguish old German text, or want to keep track of the original orthography (concerning long and short s, ligatures, etc.) in study editions (see also [12]). Now there is an end-of-word ligature for the "short" s, but the ligature `s:` must still be used inside words such as `Aus:gang` for Ausgang.

Further Ideas

The alphabets which are next on my schedule are Glagolitic, Old Church Cyrillic, Byzantine Greek (the uppercase letters used today by the Greek Orthodox Church), Coptic, Irish (calligraphic), Uiguric Mongolian (written from top to bottom), and a second Syriac font (Serto).



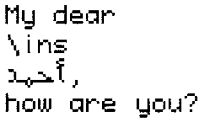
Another project is to combine the preprocessors with elementary WYSIWYG text editors providing screen fonts for all *Scholar*TEX fonts. The ordinary 7-bit TEX sources written by ysemtex would then still be read as source files, but each language would be displayed in the proper font.

For example, the .tex source file

```
My dear
\ins\arbon
\arbword{\yarb{}{\char'170}{\char'327}%
{\char'160}{\char'024}}
\arboff,
how are you?
```

which, if printed would produce

My dear أحمد, how are you?
would be visualized (here, in 9pt Monaco) as



```
My dear
\ins
أحمد,
how are you?
```

The text editor would only need to suppress all brackets, backslashes, and unnecessary macros, and display the characters in an Arabic screen font.

This would solve input encoding problems as well as problems concerning communication by electronic media (which allow only 7-bit ASCII text). The TEX sources produced by ysemtex could serve as an intermediary between e-mail, screen visualization and input, and printed output (see Fig. 6). By automating these procedures, one would have real e-mail in any possible alphabet.

Conclusion

TEX can easily and efficiently handle "those other languages", reaching the same quality level as with the more usual languages. METAFONT is essential for the creation of fonts of professional quality; the tools it provides are so powerful that you can make such fonts even at home, during your free hours, provided you invest the necessary care and feeling. But I think they deserve it, don't they?

References

[1] Guenther, Dean. "TEXT1 Goes Public Domain." *TUGboat* 11(1), pages 54–56, 1990.

[2] Haralambous, Yannis. "Arabic, Persian and Ottoman TEX for Mac and PC." *TUGboat* 11(4), pages 520–524, 1990.

[3] Haralambous, Yannis. "Typesetting Old German with TEX: Fraktur, Schwabacher, Gotisch and Initials." *TUGboat* 12(1), pages 129–138, 1991.

[4] Haralambous, Yannis. "On TEX and Greek..." *TUGboat* 12(2), pages 224–226, 1991.

[5] Jensen, H. *Die Schrift in Vergangenheit und Gegenwart*. Glückstadt/Hamburg, 1935.

[6] Knuth, Donald E. "Mathematical Typography." *Bulletin of the American Mathematical Society*, 1979.

[7] Knuth, Donald E. *Computers & Typesetting: Vol. E, Computer Modern Typefaces.* Reading, Mass.: Addison-Wesley, 1986.

[8] Lavagnino, John and Dominik Wujastyk. "An Overview of EDMAC: A plain TEX Format for Critical Editions." *TUGboat* 11(4), pages 623–643, 1990.

[9] Levy, Silvio. "Using Greek Fonts with TEX." *TUGboat* 9(1), pages 20–24, 1988.

[10] Tamari, Ittai Joseph. "Digitization of Hebrew fonts." In *Raster Imaging and Digital Typography*, Jacques André and Roger Hersch, eds. The Cambridge series on Electronic Publishing. Cambridge: Cambridge University Press, 1989.

[11] Tod, Marcus N., ed. *A Selection of Greek Historical Inscriptions.* Oxford: Clarendon Press, 1948.

[12] Wonneberger, Richard. "'Verheißung und Versprechen'. A Third Generation Approach to Theological Yypesetting." Pages 180–198 in *TEX for Scientific Documentation*, Jacques Désarménien, ed. *Lecture Notes in Computer Science* 236. Heidelberg: Springer, 1986.

[13] Yanai, Shimon and Daniel M. Berry. "Environment for Translating METAFONT to PostScript." *TUGboat* 11(4), pages 525–541, 1990.

Siamese TEX: Joining dvi Files at the Hip and Other Novel Applications of VF Files

Don Hosek
Quixote Digital Typography
349 Springfield #24
Claremont, CA 91711
714-621-1291
Internet: dhosek@ymir.claremont.edu

Abstract

When the utility of XPL files was revealed at the 1989 TUG meeting at Stanford University, and Donald Knuth announced that he would be working on an updated format for use with TEX, it was expected that this new format, VF, would become quickly and widely accepted in the TEX community. As it turns out, nearly two years after the creation of the format, the use of VF files is still fairly rare. This is due partly to lack of understanding of what can be done with VF files and partly to a lack of tools for implementing these capabilities. This paper will seek to fill both gaps: by presenting an introduction to what can be done with virtual fonts and also by describing some recently created utilities to facilitate the implementation of their potential.

What are VF Files?

First off, let's open up the acronym and point out that VF stands for "Virtual Fonts." There are some who would claim that this term is a little misleading in the context of other computer science technology and prefer the term "Composite Fonts." As a non-computer scientist, I prefer to stick with the term "Virtual Fonts" myself, mostly because it matches the acronym better.

Now that we have that formality out of the way, perhaps it is time to ask what it means for a font to be "Virtual" or "Composite." It means that what TEX thinks is a single character is not necessarily that same single character to the printer. Some dvi drivers have had a limited version of this capability built into them by necessity: for example, many dvi-to-HP LaserJet converters will map character codes to different positions (to allow for restrictions on permissible character codes on fonts) and will send larger characters as bitmapped graphics or "tiled" pieces of character. However, this capability is rarely within the control of the user.[1]

With the VF format, we have an opportunity to handle many useful features in a way that *could* be device-independent.[2] In fact, with VF files, we find that not only can we handle remapping of fonts rather easily, but we can also build composite characters: by combining characters from the same or different fonts, and also by mixing any elements which may appear in a dvi file such as rules or \special commands as well. There are also device-

[1] Tom Reid's TEXROX drivers came close to implementing some version of VF support in its ROXDEX files which at least allowed manual control over the mapping from TEX character code/font combinations to Xerox character code/font combinations. However, this facility was not terribly robust and an early experiment in creating a Times Caps/Small Caps font revealed that the TEXROX was expecting better behaved ROXDEX files than I was creating. In particular, a given Xerox font could only be referenced from a single TEX font in any dvi file causing problems when the 10pt Times Roman was accessed in both the Roman and Caps/Small Caps fonts. Tom Reid later adapted the program to support this sort of tinkering, but by then I had left my beloved Xerox 8790.

[2] The current state of matters requires the "could" in the preceding sentence; Appendix A has details on precisely what is supported by the current software.

dependent applications for VF files which we will discuss in more detail later.

TeX 3.0 and VF Files

Since TeX generally doesn't know much more about a font than the amount of space that each character takes up, it isn't even aware that VF files are being used in a TeX run. Support of VF is left entirely up to the dvi-to-output converter.

Incidentally, I think that this is a big design flaw. Had VF support been built into TeX itself, TeX would have become much more versatile. For example, the problem of hyphenating accented characters could have been eliminated once and for all since one would have been able to build accented characters "on the fly." In addition, the ability to have arbitrary remapping of fonts when TeX is run would have solved the notorious "code page" problem. However, since Knuth was in a hurry to finish TeX 3.0, we'll forgive him this oversight.

Code Pages and Remapping

We'll ignore the problem of mapping the input character code to the output code (perhaps I'll write a brief *TUGboat* article on my experiences on the topic). Instead, we'll focus on a rather specific problem: a physical font may not have the mapping of character codes in it which we would want to modify to conform to a scheme of our own. For example, if we use Personal TeX's PTI Fontware Interface to generate `.pk` files from Bitstream fonts, we get three 128-character fonts which are not quite in an arrangement suitable for our desires. Most likely, we would want a 256-character font corresponding to the proposed standard developed at the 1990 TeX Users Group meeting at Cork [1].[3] To effect this we can employ VF files to handle the remapping of character codes, as seen by TeX, to the character codes that are actually used in the font.

One way to accomplish this would be to create a VPL file by hand (a VPL file is a VF file converted to a mnemonic form readable by humans). As it turns out, this isn't too bad; Figure 2 shows how a remapping of this sort might look. However, in practice, this can get rather tedious since we need to give metric information for every character.[4] Even without this hindrance, it would still be overly tedious for a font where the remapping involves fairly direct remappings, e.g., for a small caps font, where almost the entire font would be mapped directly except for the lowercase letters which would be mapped from the uppercase in a smaller font.[5]

```
(MAPFONT D 0
   (FONTNAME beckman)
   (FONTCHECKSUM O 10537600616)
)
(CHARACTER O 15 (comment quotesingle)
   (CHARWD R 366)
   (CHARHT R 816)
   (MAP
      (SETCHAR O 47)
   )
)
```

Figure 2: An extract from a VPL file showing how remapping of characters could be accomplished. The above sample was generated by Tom Rokicki's AFM2TFM.

The REMAP **utility.** To simplify this task, I wrote REMAP, which provides a far simpler format for specifying this most common application for virtual fonts. A REMAP input file begins with a series of lines indicating the fonts that are used in the format:

`FONT` *font_number* *font_name*
[*optional_scaling_factor*]

The *font_number* is any number between 0 and 15.[6] The most commonly used font should be assigned number 0 as in a VPL file; we will see that this helps cut down the coding. The *font_name* is the font as it is known to TeX, e.g., `cmr10`; this could be the name

[3] Or not. Speaking from the font designer's standpoint, I find that the Cork standard has an inadequate number of vacant font spaces for face-specific characteristics. For example, five "f-ligatures" are provided, but not all are necessary in all fonts, but no space is reserved for an f-j ligature which is useful for typesetting Scandinavian languages. For some classical designs, other characters are necessary as well, e.g., ligatures for c-t and s-t, long s, and others.

[4] A fact not adequately documented, incidentally.

[5] This is actually a rather simplistic view. In a high-quality small caps font the weights of the small caps are adjusted to correspond to the weights of the lowercase letters of that size. With most modern digital type technologies, where a single outline is linearly scaled for all sizes, using 8pt caps for the lowercase in a 10pt caps small caps font ends up with small caps that are too light for the surrounding text.

[6] The upper limit of 15 is arbitrary and was intended to keep memory requirements low

```
FONT 0 BS0011
FONT 1 BS0011 800
RANGE 0 127
DATA DECIMAL
0:0 1-96 1:65-90 123-127
END
```

Figure 3: Sample REMAP input file. The specification 0:0 at the beginning is intended to show the format of a single character mapping. In practice, the data would begin 0-96.

of another virtual font, although such nesting can be dangerous. Finally the *optional_scaling_factor* is an integer which gives the scaling factor in the same terms as the scaled keyword in TEX: e.g., 1000 refers to no scaling, 500 gives a 50% scaled face, and so on.

After all the FONT statements have been declared, a line must appear which gives the range of character codes which define the extent of the font. This serves as a simple checksum against typographical errors in the last segment of the REMAP file. Its format is:

RANGE *first last*

Finally, the remap data is provided. All numbers must be in the same radix but a choice of hexadecimal, octal or decimal is provided. The remap data consists of a font number – character code pair for each character to be remapped. If the font number is 0, it may be omitted. Each pair is joined with a colon, and pairs can be separated by one or more spaces or a new line. A contiguous range can be specified by listing the first and last character code separated by a hyphen. An unused character position is indicated with XX. The data begins with DATA followed by one of HEX, OCTAL or DECIMAL (the default, if none of the choices is listed, is HEX). At the end of the data and the file, END should appear on a line by itself. Figure 3 shows a sample of how this might appear for a small caps font.

One feature of REMAP which is not readily apparent from the above discussion is that kerning and ligatures from the fonts being remapped are preserved as far as possible. For example, a kern between the A and V of font 0 would be preserved as would the corresponding kern between the A and V of font 1. However, no kern would automatically be inserted between, say, the A of font 0 and the V of font 1. Also, kernings and ligatures for characters not included in the remapped font would be ignored, so the user need not worry about getting “DIffICULT” instead of “DIFFICULT.” If the user feels the need to add or delete ligatures or kerns explicitly, this can be accomplished through the keywords LIG, NOLIG, KERN and NOKERN.

VF Files, POSTSCRIPT Fonts, and My Previewer

As was mentioned earlier, VF files can be a useful tool for dealing with *any* device-dependent typefaces. Tom Rokicki's AFM2TFM converter, for example, uses VF files for remapping character codes and creating small caps versions of the fonts. However, this technique still relies on there being a font on the printer somehow associated with a set of tfm dimensions on your computer.

If we want to preview a dvi file which refers to these fonts, we have two options: the first would be to have .pk files which match the POSTSCRIPT fonts. This is certainly a possibility and there exist from various sources many options for creating these files.[7] Any necessary remapping of character codes can be handled using the REMAP features described above. This, however, is not interesting.

A more interesting approach would be the case where an exact preview is not necessary and it would be adequate to use, say, Computer Modern to give an approximate the appearance of the printed document in screen preview or possibly even using POSTSCRIPT fonts to create proofs of a document which is to be ultimately printed using native fonts on a Linotronic typesetter. Mapping the characters to appropriate places, as noted above, is trivial; however, problems will be encountered in proper spacing of the letters if the metrics for the font do not match. VF fonts can be used to remedy this situation by adding or subtracting appropriate amounts of space from the sidebearings to get the character widths to match. This is done with the SHADOW command in REMAP which causes all characters in the virtual font to have the same metrics as the characters with the corresponding codes in the font mentioned in the SHADOW command. This feature can also be used to create “invisible” fonts like those used by SLITEX. Invisible characters can be created by referring to characters in an unspecified font number: an example of this appears in Figure 4. This approach gives a slight storage advantage over the tfm/.pk strategy,

[7] Perhaps the best option in the MS-DOS world is to use Adobe Type Manager to create CHR files which can be converted into TEX-style fonts with the CHtoPX and PXtoPK utilities supplied with the public domain emTEX. This will allow any POSTSCRIPT font to be printed or previewed just like a METAFONT-generated font.

```
SHADOW ptmr
RANGE 0 255
DATA DECIMAL
0-255
```

Figure 4: A REMAP input file to create an "invisible" version of PostScript Times.

which is standard for the SLiTeX fonts, and there already exist VF fonts distributed with some versions of Tom Rokicki's DVIPS for this sort of use.[8]

Device-dependent virtual font files. At this point, it would be worthwhile to point out that VF files fall into two categories. A simple remapping of the characters in a font such as that described in the introduction to REMAP would fall into the category of non-device-dependent VF files since they would be used with any output device. On the other hand, VF files created using the SHADOW feature of REMAP for proofing purposes would be device-dependent.

The classification into the two categories is not always self-evident. For example, the VF files created by Tom Rokicki's AFM2TFM would *not* be device-dependent! As it turns out, the remapping of characters performed by these VF files is still needed for VF files which will be used in the proofing stages.

Accented Characters

Another useful application of VF technology is the ability to create pre-accented letters. This is the only way to have TeX automatically accent words containing accents like "Explosionsgefährlich".[9] REMAP provides this capability with the ACCENT command which allows one to define accenting capabilities. The algorithim used for constructing accented characters is identical to that used for \accent by TeX. At the time of this writing, facilities for diacritics below letters (e.g., ç) are under development.

"Joined at the Hip" Explained

One other program was written as a sort of prolog to the work done to REMAP as described above. SIAMDVI is a program which takes a dvi file and creates a VF file in which each page in the dvi file is represented by a single character in the VF file. By default, the character dimensions are determined by the locations of print on the output from the dvi file, but this can be altered through the use of \special commands. The program had originally been conceived as a clumsy way of handling some of the features of REMAP, but use and discussion of its potential have revealed the following possible applications:

- Simple dvi inclusion. Actually, this approach is somewhat more sophisticated than that provided by Michael Spivak's DVIPASTE [3] or Stephan v. Bechtolsheim's DVI2DVI [4] since it is possible to load the virtual font scaled by some factor to include reduced views of output pages.
- The previous item can be expanded on with some simple macros to allow TeX to handle composing signatures. The current version of SIAMDVI limits this to books of 256 or fewer pages (because of the limitation on the number of characters which can appear in a single TeX font) but a future release will allow longer documents to be remapped. This has the advantage over many processors for signatures in that the positioning of pages can be adjusted slightly to compensate for the thickness of the sheets of paper in the final printing.

  If the output driver supports 180° rotation of characters, full signature pages could be composed in this manner.
- Using Alan Hoenig's METAFONT and TeX code described elsewhere in this proceedings issue, university seals can be typeset as single characters.

These are just a sampling of the possible uses of SIAMDVI. I had originally viewed it as more a novelty than a genuinely useful product—that evaluation has changed.

A Device Drivers Supporting VF Files

At the time of this writing, the following drivers were the only ones which I was aware of which supported VF features:[10]

[8] The files I have were in the MS-DOS distribution and are of unknown origin.

[9] This is because of a design decision in the TeX program. Knuth has justified TeX's deficiency in this regard as an encouragement for font designers to provide pre-accented characters. METAFONT-based fonts designed on this basis are just becoming available.

[10] Please be aware that any subjective comments in the list below are exclusively my opinion and are based on direct experience except where noted. I apologize for any inaccuracies.

- ArborText drivers updated since 1990. Some drivers in the ArborText collection undergo infrequent revision (e.g., dvixer) and so may not yet have this feature. However, since the VF format is based on Arbortext's XPL format for the DVIAPS driver, they have had a head start on implementing the features in their drivers. I have not used versions of these drivers containing VF support.
- DVIPS by Tom Rokicki. This public domain dvi-to-POSTSCRIPT converter is the first public domain driver supporting VF to include source code. Also included with DVIPS is a program, AFM2TFM, written by Donald Knuth [2] and modified by Tom Rokicki, which uses VF files in creating remapped versions of POSTSCRIPT fonts with support for ligatures and other convenient features. DVIPS runs on Unix, VMS and MS-DOS.
- The emTEX drivers. These drivers, usually bundled with the public domain emTEX for MS-DOS, provide excellent functionality for the user, although I have never bothered with the font library support which seems cumbersome to me.
- Radical Eye drivers for AmigaTEX. All programs have support for POSTSCRIPT programs (even on non-POSTSCRIPT devices!) and so come with an AFM2TFM program based on that with DVIPS (see above).

There are also two programs available for converting a dvi file which contains references to VF files into a dvi file with the references expanded into "clean" code which could be translated by any dvi processor. These are:

- DVIcopy by Peter Breitenlohner. MS-DOS, UNIX, VM/CMS.
- DVIvfDVI by Wayne Sullivan. MS-DOS only.

B Status of the Programs

At present, the programs exist only in ugly VAX C code. Before release, the code will be translated into CWEB with change files for Turbo C and VMS available on release.

References

[1] Ferguson, Michael J. "Report on Multilingual Activities." *TUGboat*, 11(4), pages 514–516, 1990. [Page 516 of the report is a full-page table depicting the character set.]

[2] Knuth, Donald E. "Virtual Fonts: More Fun for Grand Wizards." *TUGboat*, 10(1), pages 13–23, 1990.

[3] Spivak, Michael, Michael Ballantyne, and Yoke Lee. "Hi-TEX Cutting & Pasting." *TUGboat*, 10(2), pages 164–166, 1989.

[4] Bechtolsheim, Stephan v. "A .dvi File Processing Program." *TUGboat*, 10(3), pages 329–332, 1989.

[5] Youngen, Ralph, William B. Woolf, and Dan C. Latterner. "Migration from Computer Modern Fonts to Times Fonts." *TUGboat*, 10(4), pages 513–519, 1989.

When TeX and METAFONT Talk: Typesetting on Curved Paths and Other Special Effects

Alan Hoenig
Department of Mathematics, John Jay College
17 Bay Avenue, Huntington, NY 11743 US
(516) 385–0736
Bitnet: `ajhjj@cunyvm`

Abstract

It is possible to successfully ask TeX to typeset text on arbitrarily curved paths provided one enables TeX and METAFONT to communicate with one another in an appropriate manner. In this paper, we describe one method for setting text on convex paths. One possible application of this work may be toward setting text along the circular rims of institutional seals so that TeX can include such images in letterheads. We discuss this particular example in some depth, and also present some examples of fanciful typesetting made possible when TeX and METAFONT communicate with one another.

Old Work

A few years ago, I thought of a way to get TeX to typeset around a circle, and I spent some time teaching this trick to TeX [1]. Here's the basic idea. I imagined inscribing a regular n-gon inside the circle. The generality of METAFONT makes it easy to generate n rotated fonts, so that characters from the i^{th} font would sit properly on the i^{th} face of my polygon. For purposes of testing, I used `cmbx12` as the font to rotate, and I let $n = 32$.

Many people to whom I showed the end products were kind enough to applaud my feeble efforts. But the kindest of all was one individual who scolded me in no uncertain terms. I had arranged things so that each letter was centered on its polygon face. This might have been acceptable had I used a monospace font (such as `cmtt10`), but with a variable spaced font like `cmbx12`, it looked just like someone had used a *computer* to set type around a circle. This critic closed his review with a scathing remark: PostScript could do better!

A new and substantially more acceptable but different approach has since occurred to me, and it's this set of techniques that I will discuss today.

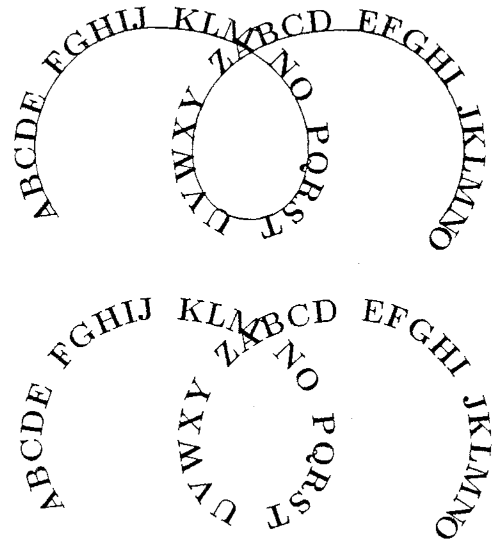


FIGURE 1. Curved typesetting.

Communication between TeX and METAFONT. Here's the main problem. TeX would be better at typesetting in nonlinear baselines if it were able to do more advanced mathematics. METAFONT *does* do that kind of mathematics, and therefore one immediately envisions some kind of dialog between the two programs as they generate and exchange information with one another. But METAFONT's file handling abilities are greatly crippled when compared to TeX. Other than font pixel files, font metric files, and log files, METAFONT cannot write files. Furthermore, although METAFONT can read files, it cannot read records individually—it's the whole file or nothing. Therefore, we have to design an inter-program dialog with some care.

In an earlier presentation [2], I had suggested that METAFONT might embed useful geometric information for later use by TeX in the `\fontdimen` parameters which accompany any font. This approach works, but more extensive testing revealed a problem. The syntax given in both *The TeXbook* and the METAFONT*book* suggests that there is no upper limit on the number of font dimens in any font, but the METAFONT program has a hard-coded upper limit of 50 such parameters per font. It looks like a simple change to the `WEB` listing could augment this value, but few users, not including myself, have ready access to `WEB` source (or to `WEB` expertise) which readily compiles in their operating system.

A better solution appeals to METAFONT's ability to store character kerning information in the `tfm` file. With old METAFONT, we were limited to 256 kern pairs, but the new limit with METAFONT2.7 is something like 32k or 64k—a much greater number of pairs.

Here's one way to pass numeric information from METAFONT to TeX using kern information. Suppose, for example, we need to tell TeX that the result of some important calculation is −14.2 pt. (There's nothing significant about this value; it was chosen purely for illustrative purposes.) We ask METAFONT to record that the kern between character 0 and character 1 (say) of a font be that value (−14.2 pt, in this example). The METAFONT code to do that is something like

```
ligtable 0: 1 kern -14.2pt#;
```

which should appear somewhere in the METAFONT driver program for this font.

How can TeX read that information? Let's suppose that the files `special.tfm` and `special.pk` store the information on this font. We can say something like

```
\font\specfont=special
```

in the TeX source document. To access this value, we say something like

```
\setbox0=\hbox{specfont\char0 \char1}
\setbox2=\hbox{\specfont
 \hbox{char0 }\hbox{\char1 }}
```

in the TeX file. The difference in the widths `\wd0` `\wd2` of these two boxes will be the number TeX needs. In practice, it is straightforward to create batch files which perform the necessary METAFONTing and which then invoke TeX, and to embed the details of the computation into a macro so this cumbersome routine is workable.

With these observations in hand, let's return to the main problem—how to typeset along any convex path, not just a circle.

Convex Paths

Here's just a quick word on what we mean by a *convex path*. Imagine that a tiny bug drives along the path in a tiny car, and that the bug has started at the beginning of the path and proceeds towards the endpoint without backing up at all. We say the path is *convex* wherever the bug turns the steering wheel to the right to stay on the path.

For typesetting purposes, convex paths are easier to treat then concave paths. The bottoms of adjacent letters butt against each other on convex paths. (On concave paths, the letters butt together at the top, and there are thorny problems in deciding where the bottoms of the letters will sit. That's why this paper only considers convex paths.)

A Three-Pass Method

I have been able to adapt a three-pass method to accomplish curvilinear typesetting. The end product of the three passes will be a new special purpose font, created just for the purpose of printing the curvaceous message. The characters in the font will not be those of the standard font layout, but will rather be the individual characters of the message, each one rotated or transformed by an amount appropriate for its position along the curved path.

Step One. The first pass belongs to TeX. In this step, TeX creates two files for later use by METAFONT.

TeX first examines the text of curvilinear material but does not typeset it. Rather, it examines each character, places it in an `\hbox` to measure its width, and writes this information into a file which METAFONT will use in the second step. TeX has adequate file handling abilities, so it's a straightforward task to create a file whose lines and records conform to METAFONT syntax. This file will be `widths.mf`.

By the way, the code to examine individual characters in a list is identical to the answer to exercise 11.5 in *The TeXbook* [3, page 67]. (The only difference lies in the definition of the macro `\\`, which I used to write the width information to an auxiliary file.)

The second file is `letters.mf` and contains essentially the individual characters of the message

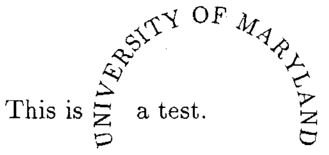


FIGURE 2. Text for a university seal.

gussied up with additional information that METAFONT will soon use to create the letter with its special rotation.

Step Two. In the second step, we invoke METAFONT. We make METAFONT use the information in the TEX output file `widths.mf` to create new information for the actual typesetting that TEX will do in the third (and final) step.

In our new font, I use `\char0` to store the representation of the actual curved path. (In this way, we can typeset the path as well as the curved text if we so choose.) Since I don't expect there to be any normal kerning between adjacent characters on a curved path, I am free to use all kerning information to transmit information back to TEX for the next step.

Now, for each character in your message, METAFONT performs a sequence of steps which I describe below. The purpose of these steps is to determine the position of this letter on the path, and to record this information for subsequent retrieval by TEX.

First, suppose the point z_0 marks our current position on the curve. Then we use METAFONT's extraordinary `solve` macro to find the point z_1 such that the length of the chord $z_1 - z_0$ is the same as the width of the current character. (We need to decrease the `tolerance` when using `solve`. Plain METAFONT sets `tolerance=0.1`; we need a smaller value, such as `tolerance=0.0001`.) Using other METAFONT commands, we can easily determine the angular orientation of this chord, and therefore the amount by which the letter should be rotated. (The chord is really an imaginary construct. We never draw it.) Really, we are *approximating* our path by a series of straight chords which "inscribe" the convex path such that each face of this approximate path will be the exact width of each letter or character.

TEX will eventually need two pieces of information about each letter in order to typeset it properly—the x- and y-offsets of that letter from the previous letter. We can pass this information to TEX using kerning pairs.

But there are other modifications we need to make to some standard METAFONT files such as `romanu.mf` (and other program files). `romanu.mf` contains the actual programs which METAFONT uses to construct the uppercase characters. This file is organized as a series of programs for each letter, one after the other. Here's how the program for A begins and ends.

```
cmchar "The letter A";
beginchar("A", 13u#, cap_height#,0);
...
penlabels(0,1,2,3,4,5,6); endchar;
```

The ellipsis denotes the details of the construction which are not important here. We add some lines to each such program as follows.

```
def A_(expr rotation_angle)=
currenttransform:=identity rotated
 rotation_angle;
def t_=transformed currenttransform
 enddef;
cmchar "The letter A";
beginchar("A", 13u#, cap_height#,0);
...
penlabels(0,1,2,3,4,5,6); endchar;
enddef;
```

That is, we embed the program for each letter within a METAFONT subroutine. (As you can see, METAFONT macro syntax differs from that of TEX.) The argument for each subroutine is the angle by which the letter needs to be rotated.

METAFONT finishes the second stage by using these subroutines together with the letter information passed to it in `\letters.mf` (first step using TEX) to generate the special purpose font.

Step Three. Finally, it's TEX's turn again. TEX takes your message text, and, character by character, it typesets it on the page. It extracts the information from the kerning pairs in the way I suggested earlier.

A frivolous example of curved typesetting appears in figure 1. The typesetting appears twice—with and without its path. If you look closely, the curved typesetting here looks a bit ragged. The reason is that I used an inferior method, in which it was only possible to get letters to match rotations to the nearest "quantum" of rotation, which was 360/32. In the improvement to that method, which is the method I just discussed, the fit would be better.

Getting to the Point. There is one application which may be of interest to curvilinear typesetters.

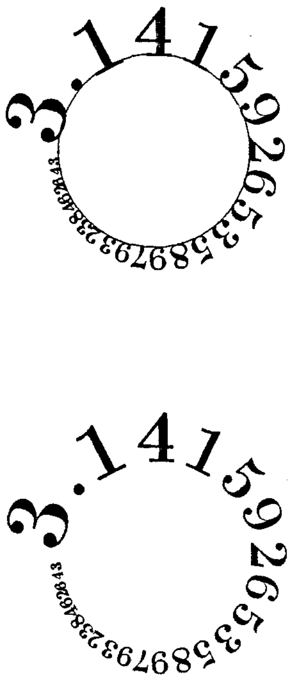


FIGURE 3. An approximation to π. We generate this figure by simultaneously rotating and magnifying numerals. We see the figure with and without its path.

Many university and institutional seals employ some text in a circle, such as that of the University of Maryland (shown in this article). It's a reasonably straightforward matter to use METAFONT to create the pictorial elements of any seal, but we would still need a method of setting the text, and of centering this text along the circle. (It's reasonably easy to perform this centering since we can take advantage of the circle's having constant curvature everywhere.)

Figure 2 displays a circular inscription suitable for a TEX letterhead. Unfortunately, the rest of the pictorial components for the seal have not yet been METAFONTed!

Breakthrough in Thinking

It was hard for me to get used to the idea that a single font could be created for one-shot uses. I am used to thinking of fonts as sacred collections that can serve long and honorably in many contexts. Nevertheless, if special-effect typesetting is needed, these special-purpose, one-shot fonts may be quite versatile.

It is clear, for example, that we can vary other of METAFONT's parameters at the time we do the rotation. In figure 3, we see the effect of simultaneously shrinking each numeral at the same time as we rotate it.

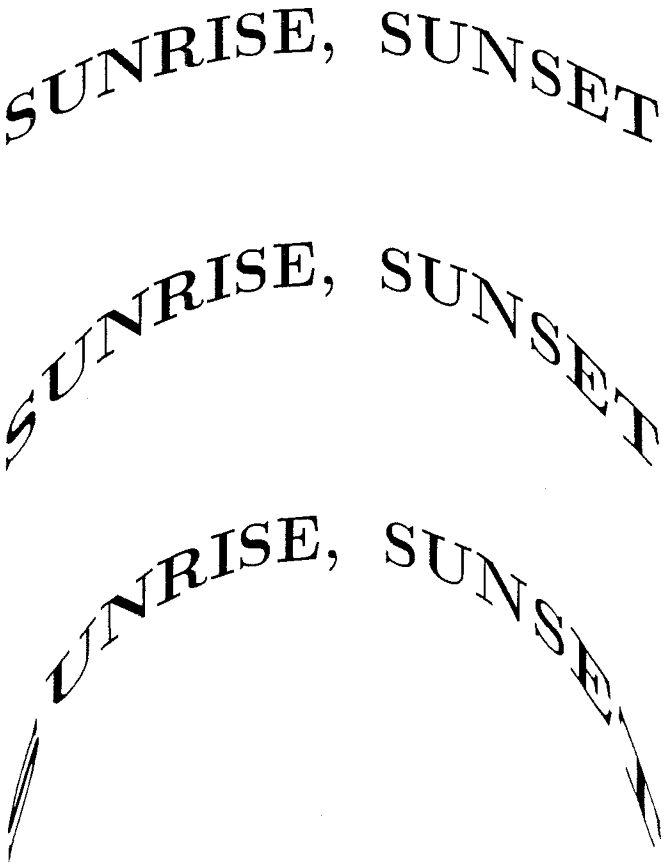


FIGURE 4. With the proper instructions, METAFONT will gladly generate these curve letters. We can adjust the rise of the "sunrise" as we see here.

But of course, we need not rotate the characters at all. We can use METAFONT to apply whatever special effect we want, character by character. Figure 4 is one such example.

Bibliography

1. Hoenig, Alan. "Circular reasoning: typesetting on a circle, and related issues" *TUGboat*, **11**#2 (June 1990), pages 183–190.
2. Hoenig, Alan. "Labelling Figures in TEX Documents" *TUGboat*, **12**#1 (March 1991), pages 125–129 (TEX90 Conference Proceedings).
3. Knuth, Donald E., *The TEXbook*. Reading, MA: Addison-Wesley, 1984.

Copyright 1991 Alan Hoenig

Participants at the 12th Annual TUG Meeting

July 15–18, 1991 Dedham, Massachusetts

* indicates an exhibitor

Ted Adamczyk
New York, New York

Robert A. Adams
University of British Columbia
Vancouver, British Columbia, Canada

Clifford Alper
TeX Users Group
Providence, Rhode Island

Robine Andrau
PWS-KENT Publishing Company
Boston, Massachusetts

Geraldine Aramanda
Menil Foundation
Houston, Texas

Dennis Arnon
Xerox Palo Alto Research Center
Palo Alto, California

Jennifer B. Bagdigian
Addison-Wesley Publishing Company
Reading, Massachusetts

Neil G. Bartholomew
American Mathematical Society
Providence, Rhode Island

Frederick H. Bartlett
Bartlett Press Incorporated
Somerset, New Jersey

Charles W. Beardsley
New York, New York

Nelson H. F. Beebe
University of Utah
Salt Lake City, Utah

Barbara Beeton
American Mathematical Society
Providence, Rhode Island

Diane M. Berezowski
Carleton University
Ottawa, Ontario, Canada

Jerry T. Borges
Lawrence Berkeley Laboratory
Berkeley, California

Harriet B. Borton
Rensselaer Polytechnic Institute
Troy, New York

Colleen Brosnan
Prentice Hall
Englewood Cliffs, New Jersey

Mimi Burbank
Florida State University
Tallahassee, Florida

Karen Butler
TeX Users Group
Providence, Rhode Island

William Butler
TeX Users Group
Providence, Rhode Island

Katherine Butterfield
University of California, Berkeley
Berkeley, California

Keith G. Calkins
Andrews University
Berrien Springs, Michigan

John F. Carleo
McGraw-Hill Incorporated
New York, New York

*__Lance Carnes__
Personal TeX Incorporated
Mill Valley, California

Paula E. Carroll
Jones and Bartlett
Boston, Massachusetts

Christopher Carruthers
University of Ottawa
Ottawa, Ontario, Canada

Katherine S. Carter
Princeton University
Princeton, New Jersey

Sheryl D. Chapman
Cogni Seis Development
Houston, Texas

Ling Ling Chen
Academia Sinica
Taipei, Taiwan, Republic of China

Ian J. Chin
Xenergy, Inc.
Burlington, Massachusetts

Malcolm W. Clark
Polytechnic of Central London
London, England

David M. Cobb
SAIC
Oak Ridge, Tennessee

Donald Cohn
Boston, Massachusetts

Daniel Comenetz
Belmont, Massachusetts

Jacquie Commanday
Houghton-Mifflin Company
Boston, Massachusetts

A.C. Conrad
Menil Foundation
Houston, Texas

Raylene Cooper
Lawrence Livermore National Laboratory
Livermore, California

Ray Cowan
Stanford Linear Accelerator Center
Stanford, California

*__Betsy J. Dale__
ArborText Incorporated
Ann Arbor, Michigan

Laura L. Dale
ArborText Incorporated
Ann Arbor, Michigan

Jackie Damrau
Superconducting Super Collider Laboratory
Dallas, Texas

Paul Davis
Worcester Polytechnic Institute
Worcester, Massachusetts

Donald W. DeLand
Integre Technical Publishing Co., Inc.
Albuquerque, New Mexico

Dian DeSha
California Institute of Technology
Pasadena, California

Norman Dobbs
Houghton-Mifflin Company
Boston, Massachusetts

Andrew E. Dobrowolski
ArborText Incorporated
Ann Arbor, Michigan

Paula Donovan
TeX Users Group
Providence, Rhode Island

Michael Doob
University of Manitoba
Winnipeg, Manitoba, Canada

Michael J. Downes
American Mathematical Society
Providence, Rhode Island

Ken Dreyhaupt
Springer-Verlag New York, Inc.
New York, New York

Allen R. Dyer
Computer Law Laboratory
Ellicott City, Maryland

Teresa A. Ehling
MIT Press
Cambridge, Massachusetts

Victor L. Eijkhout
University of Illinois
Urbana, Illinois

Bennet Fauber
Impressions - A Division of Edwards Brothers
Ann Arbor, Michigan

Michael J. Ferguson
Université du Québec
Verdun, Québec, Canada

Anita Flanzbaum
MIT Press Journal
Cambridge, Massachusetts

Frank Flynn
University of British Columbia
Vancouver, British Columbia, Canada

Peter Flynn
University College of Cork
Cork, Republic of Ireland

Jim Fox
University of Washington
Seattle, Washington

Elena Fraboschi
Indiana University
Bloomington, Indiana

Jac A. Fried
New York University
New York, New York

David R. Fuchs
Palo Alto, California

Harumi Fujiura
ASCII Corporation
Kawasaki, Japan

Edward A. Garay
University of Illinois at Chicago
Chicago, Illinois

Bernard Gaulle
CIRCE/CNRS
Orsay, France

Christopher A. Gibson
Harcourt Brace Jovanovich
London, England

Helen M. Gibson
Wellcome Institute for History of Medicine
London, England

Hans Th.J.E. Gieskes
Elsevier Science Publishers BV
Amsterdam, Netherlands

James L. Giles
Syntax International PTE
York, Pennsylvania

Regina Girouard
American Mathematical Society
Providence, Rhode Island

Richard E. Glass
American Institute of Physics
Woodbury, New York

Christina M. Gorecki
Aware, Inc.
Cambridge, Massachusetts

Roswitha Graham
K.T.H. Royal Institute of Technology
Stockholm, Sweden

Geeti Granger
John Wiley & Sons Ltd
West Sussex, England

Alfred Gray
University of Maryland
College Park, Maryland

William L. Haberman
Rockville, Maryland

Ian W. Hall
Oxford University Press
Oxford, England

Hope Hamilton
National Center for Atmospheric Research
Boulder, Colorado

Chris Hamlin
American Institute of Physics
Woodbury, New York

Nancy Kruse Hannigan
MIT Lincoln Laboratory
Lexington, Massachusetts

Yannis Haralambous
Villeneuve d'Ascq, France

Donna L. Harmon
American Mathematical Society
Providence, Rhode Island

Robert L. Harris
Micro Programs Incorporated
Syosset, New York

Roger H. Hauck
Smithsonian Astrophysical Observatory
Cambridge, Massachusetts

*****Richard N. Hayes**
ETP Services Co.
Portland, Oregon

*****Doug Henderson**
Blue Sky Research
Portland, Oregon

Amy Hendrickson
TEXnology Incorporated
Brookline, Massachusetts

Matthew N. Hendryx
Brooklyn, New York

Robert H. Hilbert
John Wiley & Sons Incorporated
New York, New York

John D. Hobby
AT&T Bell Laboratories
Murray Hill, New Jersey

Alan Hoenig
John Jay College (CUNY)
New York, New York

Anita Z. Hoover
University of Delaware
Newark, Delaware

Berthold Horn
MIT Lincoln Laboratory
Lexington, Massachusetts

Blenda Horn
Y & Y
Carlisle, Massachusetts

Cay S. Horstmann
San Jose State University
San Jose, California

Don Hosek
Quixote Digital Typography
Claremont, California

*****Roger Hunter**
TCI Software Research, Inc.
Las Cruces, New Mexico

Elizabeth Hyman
Brookline, Massachusetts

Patrick Ion
Mathematical Reviews
Ann Arbor, Michigan

Calvin W. Jackson
California Institute of Technology
Pasadena Angeles, California

Roger B. Jagoda
Cornell University
Ithaca, New York

Philip H. Jensen
Hypersoft Corporation
Cambridge, Massachusetts

Peter H. John
Cranston, Rhode Island

Gordon C. Johnson
Interactive Composition Corporation
Pleasant Hill, California

Carl H. Jones
Parsippany, New Jersey

Cheryl A. Jones
Massachusetts Institute of Technology
Cambridge, Massachusetts

*****Claire Kahan**
Personal TEX Incorporated
Mill Valley, California

Elise Kaiser
PWS-KENT Publishing Company
Boston, Massachusetts

Takashi Kakiuchi
Matsushita Electric Ind Company Ltd
Osaka, Japan

Toru Kawate
York Graphic Services
York, Pennsylvania

Victoria Keirnan
Houghton-Mifflin Company
Boston, Massachusetts

*****David Kellerman**
Northlake Software
Portland, Oregon

Charlene Kellner
Los Alamos National Laboratory
Los Alamos, New Mexico

Niel Kempson
Gloucestershire, England

John T. Kesich
New York University
New York, New York

Jimmie W. Killian
Niantic, Connecticut

Ann Kostant
Newton, Massachusetts

Shoshanna Kostant
Auburndale, Massachusetts

Brett Kotch
Monsey, New York

David H. Kratzer
Los Alamos National Laboratory
Los Alamos, New Mexico

Albert Kuo
Yale University
New Haven, Connecticut

Anna Kurica
American Society of Mechanical Engineers
New York, New York

Kees van der Laan
Groningen, Netherlands

*****Anthony B. Lafrenz**
ETP Services Co.
Portland, Oregon

Mimi Lafrenz
ETP Services Co.
Portland, Oregon

Joachim Lammarsch
Universität Heidelberg
Heidelberg, Federal Republic of Germany

Lauren F. Landsburg
Technique Typsetting
Rochester, New York

Timothy R. Larson
Behrend College
Erie, Pennsylvania

Peggy Lashway
Rensselaer Polytechnic Institute
Troy, New York

Dan C. Latterner
Mathematical Reviews
Ann Arbor, Michigan

Charlotte V. Laurendeau
TEX Users Group
Providence, Rhode Island

John Lavagnino
Brandeis University
Waltham, Massachusetts

John S. Lee
Los Angeles, California

Eve Lehmann
PWS-KENT Publishing Company
Boston, Massachusetts

John L. Lincoln
Pawtucket, Rhode Island

Susan London
PWS-KENT Publishing Company
Boston, Massachusetts

Pierre MacKay
University of Washington
Seattle, Washington

John Mancia
Elsevier Science Publishing Co., Inc.
New York, New York

John W. Manly
Amherst College
Amherst, Massachusetts

Adam Mann
Long Island City, New York

Jeffrey McArthur
Atlis Publishing Services, Inc.
Beltsville, Maryland

Betty McCarthy
IBM T. J. Watson Research Center
Yorktown Heights, New York

Sam K. McCollum
William Byrd Press, Inc.
Richmond, Virginia

Marret McCorkle
D.C. Heath
Lexington, Massachusetts

Sarah McCracken
Addison-Wesley Publishing Company
Reading, Massachusetts

Joni H. McDonald
Jones and Bartlett
Boston, Massachusetts

Robert W. McGaffey
Martin Marietta Energy Systems Incorporated
Oak Ridge, Tennessee

Wendy McKay
University of Montréal
Montréal, Québec, Canada

Carol A. Meyer
Association for Computing Machinery, Inc.
New York, New York

Lothar Meyer-Lerbs
Bremen, Federal Republic of Germany

Frank Mittelbach
Electronic Data Systems
Rüsselsheim, Federal Republic of Germany

Cornelia M. Monahan
American Society of Mechanical Engineers
New York, New York

Patricia Monohon
Santa Barbara, California

Mary Jean Moore
University of California
Oakland, California

Carol L. Moura
American Mathematical Society
Providence, Rhode Island

Gillian S. Murray
Carleton University
Ottawa, Ontario, Canada

Jane Muse
Houghton-Mifflin Company
Boston, Massachusetts

Norman Naugle
Texas A&M University
College Station, Texas

Marion U. Neubauer
Universität Heidelberg
Heidelberg, Federal Republic of Germany

Herb Niemirow
Springer-Verlag New York, Inc.
New York, New York

Pamela B. O'Connor
MIT Lincoln Laboratory
Lexington, Massachusetts

Jose Luis Olivares
Sociedad Mexicana de Física
Coyoacan, Mexico

*__Daniel D. Olson__
ETP Services Co.
Portland, Oregon

Eileen Olszewski
Princeton University
Princeton, New Jersey

Yoko Ozawa
NEC Research Institute, Inc.
Princeton, New Jersey

Janet F. Pecorelli
American Mathematical Society
Providence, Rhode Island

Beth Perry
Addison-Wesley Publishing Company
Reading, Massachusetts

Noel C. Peterson
Library of Congress
Washington, District of Columbia

Laurie Petrycki
Addison-Wesley Publishing Company
Reading, Massachusetts

Teresa Pires
TeX Users Group
Providence, Rhode Island

Craig R. Platt
University of Manitoba
Winnipeg, Manitoba, Canada

Nico A.F.M. Poppelier
Elsevier Science Publishers BV
Amsterdam, Netherlands

Gary L. Price
David Systems
Sunnyvale, California

Lynne A. Price
Frame Technology Corporation
San Jose, California

Martin Rabinowitz
Academic Press Incorporated
Cambridge, Massachusetts

Stanley Rabinowitz
Mathpro Press
Westford, Massachusetts

Jon Radel
Reston, Virginia

Howard Ratner
Springer-Verlag New York, Inc.
New York, New York

Tom Renfrow
Jet Propulsion Laboratory
Pasadena, California

Samuel E. Rhoads
Honolulu Community College
Honolulu, Hawaii

*__Guy Rivers__
TCI Software Research, Inc.
Las Cruces, New Mexico

Pam Rockwell
PWS-KENT Publishing Company
Boston, Massachusetts

Cynthia Rodriguez
University of Illinois at Chicago
Chicago, Illinois

David F. Rogers
Annapolis, Maryland

Nancy Rogers
Annapolis, Maryland

Shoshana Rosenthal
Smithsonian Astrophysical Observatory
Cambridge, Massachusetts

Mark A. Roth
Wright-Patterson AFB, Ohio

Bernard Rous
ACM
New York, New York

Chris Rowley
Open University
London, England

Beverly J. Ruedi
Mathematical Association of America
Washington, D.C.

Beardsley Ruml
Legal Support Systems Incorporated
Cambridge, Massachusetts

Jan Michael Rynning
Royal Institute of Technology
Stockholm, Sweden

David Salomon
California State University, Northridge
Northridge, California

Tina Samaha
PWS-KENT Publishing Company
Boston, Massachusetts

Ed Sarkel
Beacon Graphics Corporation
Ashland, Ohio

Shashi Sathaye
University of Kentucky
Lexington, Kentucky

Aaron Sawdey
Publication Services
Champaign, Illinois

Antoinette T. Schleyer
American Mathematical Society
Providence, Rhode Island

Fred Schulte
McGraw-Hill Incorporated
New York, New York

Ronald Scott
American Geophysical Union
Washington, District of Columbia

Luigi Semenzato
University of California, Berkeley
Berkeley, California

John T. Sheridan
Sheridan Printing Company, Inc.
Alpha, New Jersey

Joseph H. Silverman
Brown University
Providence, Rhode Island

Sally Simpson
Addison-Wesley Publishing Company
Reading, Massachusetts

*__Barry Smith__
Blue Sky Research
Portland, Oregon

Lowell Smith
Salt Lake City, Utah

*__Scobie Smith__
Kinch Computer Company
Ithaca, New York

Joe Snowdon
Cambridge, Massachusetts

Michael Sofka
Publication Services
Champaign, Illinois

Annie Soltys
CEBAF
Newport News, Virginia

Friedhelm Sowa
Heinrich Heine University
Düsseldorf, Federal Republic of Germany

C.M. Sperberg-McQueen
University of Illinois at Chicago
Chicago, Illinois

*__Michael D. Spivak__
TeXplorators Corporation
Houston, Texas

David K. Steiner
Rutgers University
Piscataway, New Jersey

Caroline B. Stewart
CEBAF
Newport News, Virginia

Martin Stock
Cambridge, Massachusetts

Carol K. Sullivan
United States Geological Survey
Menlo Park, California

Pam Suwinsky
Addison-Wesley Publishing Company
Redwood City, California

Christina Thiele
Carleton University
Ottawa, Ontario, Canada

Lee F. Thompson
University of Wisconsin, Madison
Madison, Wisconsin

Brian E. Travis
Teleprint
Englewood, Colorado

Velma M. Tyler
United States Geological Survey
Menlo Park, California

Frank H. Ulmer
Grumman Melbourne Systems Division
Melbourne, Florida

Jeri Uzzo
IEEE
New York, New York

Craig W. Van Dyck
Springer-Verlag New York, Inc.
New York, New York

Irene Vankanan
Digital Equipment Corporation
Nashua, New Hampshire

Jiři Veselý
Charles University
Prague, Czechoslovakia

*__Michael Vulis__
Micro Press, Inc.
Forest Hills, New York

Helen Walden
PWS-KENT Publishing Company
Boston, Massachusetts

Edward Wang
University of California, Berkeley
Berkeley, California

Margaret L. Ward
Massachusetts Institute of Technology
Cambridge, Massachusetts

Leslie C. Watson
SFA, Inc.
Temple Hills, Maryland

Esther K. Weil
Albion, Michigan

Carol A. Weiss
Sun Lakes, Arizona

Neil A. Weiss
Arizona State University
Tempe, Arizona

Michael J. Wester
Albuquerque, New Mexico

Alan Wetmore
White Sands Missle Range, New Mexico

Samuel B. Whidden
Cumberland, Rhode Island

Ron Whitney
TeX Users Group
Providence, Rhode Island

Julie A. Wilczek
American Mathematical Society
Providence, Rhode Island

William Willey
McGraw-Hill Incorporated
New York, New York

Linda Williams
University of Tennessee Space Institute
Tullahoma, Tennessee

Cheryl W. Winstead
NASA Langley Research Center
Hampton, Virginia

David B. Witonsky
Philadelphia, Pennsylvania

Richard Wong
Princeton University
Princeton, New Jersey

Derick Wood
University of Waterloo
Waterloo, Ontario, Canada

William B. Woolf
American Mathematical Society
Providence, Rhode Island

Cheryl Wurzbacher
Addison-Wesley Publishing Company
Reading, Massachusetts

Helen M. Wythe
Addison-Wesley Publishing Company
Reading, Massachusetts

Ralph E. Youngen
American Mathematical Society
Providence, Rhode Island

I-Pen Yuan
Taipei, Taiwan, Republic of China

Eva A. Ziem
Texas Instruments
Dallas, Texas

Calendar

1991 (For the record)

Nov 18 Meeting of the Nordic TeX Group, Royal Institute of Technology, Stockholm, Sweden. For information, contact Roswitha Graham (roswitha@admin.kth.se).

Nov 20 ukTeXug: "Macro Packages", Oxford University, England. For information, contact Chris Rowley (ca_rowley@vax.acs.open.ac.uk).

Nov 21 NTG Fall Meeting, "Fun with TeX", Technische Universiteit te Eindhoven, The Netherlands. For information, contact Piet Tutelaers (rcpt@URC.TUE.NL).

1992

San Diego, California

Jan 20 – 24 Intensive Beginning/Intermed. TeX

Jan 27 – 31 Intensive LaTeX

Feb 11 ukTeXug: "TeX for Book and Journal Production", School of Oriental and African Studies, London, England. For information, contact Chris Rowley (ca_rowley@vax.acs.open.ac.uk).

Providence, Rhode Island

Feb 10 – 14 Intensive LaTeX

Feb 17 – 21 Intensive Beginning/Intermed. TeX

Feb 18 – 21 Seybold '92 Seminars, Hynes Convention Center, Boston, Massachusetts. For information, contact Seybold, P. O. Box 578, Malibu, CA 90265-0578 (213-457-5850).

Mar 10 ***TUGboat* Volume 13, 2nd regular issue:** Deadline for receipt of *technical* manuscripts (tentative).

Mar 16 **Donald E. Knuth Scholarship:** Deadline for receipt of applications. (See page 565.)

Mar 24 – 27 TeX-Tagung DANTE '92, Universität Hamburg, Hamburg, Federal Republic of Germany. For information, contact Reinhard Zierke or Gerhard Friesland-Köpke (dante92@informatik.uni-hamburg.de).

Apr 7 ***TUGboat* Volume 13, 2nd regular issue:** Deadline for receipt of news items, reports (tentative).

Apr 7 – 10 EP'92 Swiss Federal Institute of Technology, Lausanne, Switzerland. For information, contact ep92@eldi.epfl.ch.

Easter ukTeXug: [subject to be announced], Scotland [location to be announced]. For information, contact Chris Rowley (ca_rowley@vax.acs.open.ac.uk).

May 23 CyrTUG: First Annual Meeting, Institute of High Energy Physics, Protvino (suburb of Moscow), Russia. For information, contact Irina Makhovaya (irina@mir.msk.su).

May/Jun NTG Spring Meeting, "Science with TeX", CWI, Amsterdam, The Netherlands. For information, contact Gerard van Nes (vannes@ECN.NL).

Jun 16 – 18 GUTenberg'92, "The dark side of TeX", Les Diablerets, Switzerland. For information, contact Denis Megevand (megevand@scsun.unige.ch or megevand@cgeuge54.bitnet). (See page 566.)

Jun/Jul ukTeXug: "Design Issues", [location to be announced]. For information, contact Chris Rowley (ca_rowley@vax.acs.open.ac.uk).

Jul 27 – 30 **TUG'92: "TeX in Context"**, Portland, Oregon. For information, contact the TUG office. (See page 570.)

Status as of 25 October 1991

Aug 18	***TUGboat* Volume 13, 3rd regular issue:** Deadline for receipt of *technical* manuscripts (tentative).
Sep	EuroTeX92, Prague, Czechoslovakia. For information, contact Jiří Veselý (`ummjv@csearn.bitnet`)
Sep 15	***TUGboat* Volume 13, 3rd regular issue:** Deadline for receipt of news items, reports (tentative).

For additional information on the events listed above, contact the TUG office (401-751-7760, email: `tug@math.ams.com`) unless otherwise noted.

Announcement: The Donald E. Knuth Scholarship for 1992

The intent of the Donald E. Knuth Scholarship is to encourage the increase of knowledge about TeX and to sharpen the TeX skills of non-technical users.

Owing to an administrative foulup, no Knuth Scholarship was awarded for 1991. The TUG Board and the Scholarship Committee regret this, and invite participation from eligible TUG members for next year.

One Knuth Scholarship will be awarded in 1992. The competition will be open to all 1992 TUG members holding support positions that are secretarial, clerical or editorial in nature, as determined by job title and duties, and not holding a degree with a major in a technical, scientific or mathematical subject area. The award will consist of an expense-paid trip to the TUG annual meeting and to the Scholar's choice from the short courses offered in conjunction with that meeting. A cap of $2,000 has been set for the award; however, registration fees for the meeting and short course will be waived, and not counted in the limit.

To enter the competition, applicants should submit to the Scholarship Committee, by the deadline specified below, the input file and final TeX output of a project that displays originality, knowledge of TeX, and good TeXnique. The project may make use of a macro package, either a public one such as LaTeX or one that has been developed locally; such a macro package should be identified clearly. Such features as sophisticated use of math mode, of macros that require more than "filling in the blanks", or creation and use of new macros will be taken as illustrations of the applicant's knowledge.

All macros created by the candidate should be well documented with clear descriptions of how they should be used and an indication of how they work internally.

All associated style files, macro-package files, etc., should be supplied, or a clear indication given of any widely available ones used (including version numbers, dates, etc.); clear information should be provided concerning the version of TeX used and about any other software (e.g. particular printer drivers) required. Any nonstandard fonts should be identified and provided in the form of `.tfm` and `.pk` files suitable for use on a 300dpi laser printer.

While the quality of the typographic design will not be an important criterion of the judges, candidates are advised to ensure that their printed output adheres to sound typographic standards; the reasons for any unusual typographic features should be clearly explained.

The input files should be provided in electronic form as well as on paper. Suitable electronic media are IBM PC-compatible or Macintosh diskettes, or a file sent by electronic mail.

A brochure with additional information is available from the TUG office. To obtain a copy, or to request instructions on e-mail submission, write to the address at the end of this announcement, or send a message by e-mail to `TUG@Math.AMS.com` with the subject "Knuth Scholarship request".

Along with the project, each applicant should submit a letter stating the following:

1. affirmation that he/she will be available to attend the 1992 annual meeting;
2. affirmation of willingness to participate on the committee to select the next Scholar.

Each applicant should also submit a *curriculum vitae* summarizing relevant personal information, including:

1. statement of job title, with a brief description of duties and responsibilities;
2. description of general post-secondary school education, TeX education, identifying courses attended, manuals studied, personal instruction from experienced TeX users, etc.;
3. description of TeX resources and support used by the candidate in the preparation of the project.

Neither the project nor the *curriculum vitae* should contain the applicant's name or identify the applicant. These materials will be reviewed by the committee without knowledge of applicants' identities. If, despite these precautions, a candidate is identifiable to any judge, then that judge will be required to make this fact known to the others and to the TUG board members responsible for the conduct of the judging.

The covering letter, *curriculum vitae*, and all macro documentation that is part of the project input should be in English. (English is not required for the output of the project.) However, if English is not the applicant's native language, that will not influence the decision of the committee.

Selection of the Scholarship recipient will be based on the project submitted.

Schedule

The following schedule will apply; all dates are in 1992:

March 23	Deadline for receipt of submissions
April 7–May 25	Judging period
June 1	Notification of winner
July 27–30	Annual meeting, Portland, Oregon

Committee

The Scholarship Committee consists of

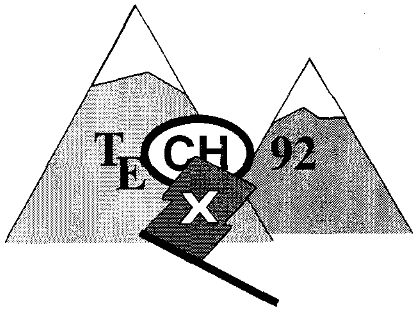
- Chris Rowley, Open University, U.K. (Chair)
- Nico Poppelier, Elsevier Science Publishers
- David Salomon, California State University, Northridge
- Linda Williams, University of Tennessee Space Institute

Where to write

All applications should be submitted to the Committee in care of the TUG office:

TEX Users Group
Attn: Knuth Scholarship Competition
658 North Main Street
P. O Box 9506
Providence, RI 02940-9506
U.S.A.
email: `TUG@math.ams.com`

Conférence GUTenberg'92
Les Diablerets, Switzerland
16–18 June 1992



Theme: ***The dark side of TEX***

TECH'92, the 1992 GUTenberg conference, will be held in Les Diablerets, Switzerland, June, 16–18th, and will be organized by the Observatory of Geneva.

TEX is mostly known as a mathematical and scientific typesetting system. While scientists are very happy using it for formatting and submitting their papers, they are often not aware of the installation, support and teaching problems related to TEX and its fellow tools.

Non-scientific applications make TEX in many ways a very universal tool, but TEX developers are not always aware of the humanistic applications and of the social sciences, in a general sense, who represent a potentially large base of users.

The 1992 conference of the TEX French speaking group (GUTenberg) will address the problems of TEX's social environment at the support and training level, and the use of TEX in non-scientific applications.

The conference will focus on these two main topics, but, as usual, other TEX related topics will also be treated.

Both main themes will be covered by invited review papers and workshops. Contributed papers and posters are welcome, addressing either the main themes or more generally TeX and DTP related subjects.

Suggested topics

- Installation, development and maintenance of a TeX site.
- TeX local macro writing and user support.
- Local font developments.
- Teaching TeX.
- TeX and METAFONT usage in the non-scientific world.

Language: The official language of the conference is French but presentations in English will be welcome.

Call for papers

The following deadlines apply to TeCH'92:

January 15th, 1992: A one page abstract (about two 80 × 25 screens) should be submitted by mail, e-mail or fax to the program coordinator.

January 31st, 1992: Authors of accepted papers will be notified, by mail, e-mail or fax. Typesetting instructions for the author will be sent at the same time.

March 15th, 1992: Complete texts must be submitted, preferably in LaTeX source format conforming to the instructions; if necessary, camera-ready form will be accepted. The papers received after this date will be accepted for the conference, but they will not be included in the proceedings, as these will be published before and distributed at the conference.

June 16–18th, 1992: Papers are presented at the conference.

Posters

We would like to see a lot of posters presenting old and new "non-standard" macro packages, with both internal structure and applications, as a way to promote a wider use of TeX.

Tutorials

Tutorials on TeX and its fellow tools will be organized on June, 14–15th, 1992.

People who wish to be in charge of such a tutorial are requested to get in touch with the organizing committee. The required equipment will be made available to the teachers so that the tutorials meet the highest standards.

Tutorials will be taught in French.

Exhibits

An exposition will be organized jointly with the conference; prospective exhibitors should contact the organizing committee.

GUTenberg, TeCH'92
BP 21
78354 Jouy en Josas cedex, France
Telephone: + 33 1 34 65 22 32
Fax: + 33 1 34 65 20 51

Organizing Committee

Denis MÉGEVAND (Geneva Observatory), chairman

Anne-Marie CNOPS (Geneva University)

Philippe LOUARN (IRISA, Rennes), proceedings editor

Suzanne MÉGEVAND (Commugny)

Program Committee

Paul BARTHOLDI (Geneva Observatory), chairman

Jacques ANDRÉ (IRISA, Rennes)
Marie-Louise CHAIX (Éd. de physique, Les Ulis)
Eric CORNELIS (Namur University)
André DESNOYERS (Blaise Pascal Institute, Paris)
Yannis HARALAMBOUS (Lille University)
Roger D. HERSCH (EPFL, Lausanne)
Eric VAN HERWIJNEN (CERN, Geneva)
Maurice LAUGIER (Imprimerie Louis-Jean, Gap)
Philipp TAYLOR (RHBNC, Londres)
Eric WEHRLI (Geneva University)

Geneva Observatory
51, ch. des Maillettes
CH-1290 Sauverny, Switzerland
Telephone: +41 22 755 2611
Fax: +41 22 755 3983

Denis MÉGEVAND
E-mail: `megevand@scsun.unige.ch`
`megevand@cgeuge54.bitnet`
`20579::ugobs::megevand`
`(0228) 4682161350::megevand`
Paul BARTHOLDI
E-mail: `bartho@scsun.unige.ch`
`bartho@cgeuge54.bitnet`
`20579::ugobs::bartho`
`(0228) 4682161350::bartho`

Institutional Members

The Aerospace Corporation,
El Segundo, California

Air Force Institute of Technology,
Wright-Patterson AFB, Ohio

American Mathematical Society,
Providence, Rhode Island

ArborText, Inc.,
Ann Arbor, Michigan

ASCII Corporation,
Tokyo, Japan

Belgrade University,
Faculty of Mathematics,
Belgrade, Yugoslavia

Brookhaven National Laboratory,
Upton, New York

CERN, *Geneva, Switzerland*

Brown University,
Providence, Rhode Island

California Institute of Technology,
Pasadena, California

Calvin College,
Grand Rapids, Michigan

Carleton University,
Ottawa, Ontario, Canada

Centre Inter-Régional de
Calcul Électronique, CNRS,
Orsay, France

College of William & Mary,
Department of Computer Science,
Williamsburg, Virginia

Communications
Security Establishment,
Department of National Defence,
Ottawa, Ontario, Canada

Construcciones Aeronauticas, S.A.,
CAE-Division de Proyectos,
Madrid, Spain

DECUS, Electronic Publishing
Special Interest Group,
Marlboro, Massachusetts

Department of National Defence,
Ottawa, Ontario, Canada

E. S. Ingenieres Industriales,
Sevilla, Spain

Edinboro University
of Pennsylvania,
Edinboro, Pennsylvania

Elsevier Science Publishers B.V.,
Amsterdam, The Netherlands

European Southern Observatory,
Garching bei München,
Federal Republic of Germany

Fermi National Accelerator
Laboratory, *Batavia, Illinois*

Florida State University,
Supercomputer Computations
Research, *Tallahassee, Florida*

Fordham University,
Bronx, New York

General Motors
Research Laboratories,
Warren, Michigan

Grinnell College,
Computer Services,
Grinnell, Iowa

GTE Laboratories,
Waltham, Massachusetts

Hatfield Polytechnic,
Computer Centre,
Herts, England

Hughes Aircraft Company,
Space Communications Division,
Los Angeles, California

Hungarian Academy of Sciences,
Computer and Automation
Institute, *Budapest, Hungary*

IBM Corporation,
Scientific Center,
Palo Alto, California

Institute for Advanced Study,
Princeton, New Jersey

Institute for Defense Analyses,
Communications Research
Division, *Princeton, New Jersey*

Intevep S. A., *Caracas, Venezuela*

Iowa State University,
Ames, Iowa

The Library of Congress,
Washington D.C.

Los Alamos National Laboratory,
University of California,
Los Alamos, New Mexico

Louisiana State University,
Baton Rouge, Louisiana

MacroSoft, *Warsaw, Poland*

Marquette University,
Department of Mathematics,
Statistics and Computer Science,
Milwaukee, Wiscon

Masaryk University,
Brno, Czechoslovakia

Mathematical Reviews,
American Mathematical Society,
Ann Arbor, Michigan

Max Planck Institut
für Mathematik,
Bonn, Federal Republic of Germany

McGill University,
Montréal, Québec, Canada

Michigan State University,
Mathematics Department,
East Lansing, Michigan

NASA Goddard
Space Flight Center,
Greenbelt, Maryland

National Institutes of Health,
Bethesda, Maryland

National Research Council
Canada, Computation Centre,
Ottawa, Ontario, Canada

Naval Postgraduate School,
Monterey, California

New York University,
Academic Computing Facility,
New York, New York

Nippon Telegraph &
Telephone Corporation,
Software Laboratories,
Tokyo, Japan

Northrop Corporation,
Palos Verdes, California

The Open University,
Academic Computing Services,
Milton Keynes, England

Pennsylvania State University,
Computation Center,
University Park, Pennsylvania

Personal TEX, Incorporated,
Mill Valley, California

Princeton University,
Princeton, New Jersey

Purdue University,
West Lafayette, Indiana

Queens College,
Flushing, New York

Rice University,
Department of Computer Science,
Houston, Texas

Roanoke College,
Salem, VA

Rogaland University,
Stavanger, Norway

Ruhr Universität Bochum,
Rechenzentrum,
Bochum, Federal Republic of Germany

Rutgers University, Hill Center,
Piscataway, New Jersey

St. Albans School,
Mount St. Alban, Washington, D.C.

Sandia National Laboratories,
Albuquerque, New Mexico

Smithsonian Astrophysical Observatory, Computation Facility,
Cambridge, Massachusetts

Software Research Associates,
Tokyo, Japan

Space Telescope Science Institute,
Baltimore, Maryland

Springer-Verlag,
Heidelberg, Federal Republic of Germany

Springer-Verlag New York, Inc.,
New York, New York

Stanford Linear Accelerator Center (SLAC),
Stanford, California

Stanford University,
Computer Science Department,
Stanford, California

Talaris Systems, Inc.,
San Diego, California

Texas A & M University,
Department of Computer Science,
College Station, Texas

UNI-C, *Aarhus, Denmark*

United States Military Academy,
West Point, New York

University of Alabama,
Tuscaloosa, Alabama

University of British Columbia,
Computing Centre,
Vancouver, British Columbia, Canada

University of British Columbia,
Mathematics Department,
Vancouver, British Columbia, Canada

University of Calgary,
Calgary, Alberta, Canada

University of California, Berkeley,
Space Astrophysics Group,
Berkeley, California

University of California, Irvine,
Information & Computer Science,
Irvine, California

University of California,
Los Angeles, Computer Science Department Archives,
Los Angeles, California

University of Canterbury,
Christchurch, New Zealand

Universidade de Coimbra,
Coimbra, Portugal

University College,
Cork, Ireland

University of Crete,
Institute of Computer Science,
Heraklio, Crete, Greece

University of Delaware,
Newark, Delaware

University of Exeter,
Computer Unit,
Exeter, Devon, England

University of Glasgow,
Department of Computing Science,
Glasgow, Scotland

University of Groningen,
Groningen, The Netherlands

University of Heidelberg,
Computing Center Heidelberg,
Germany

University of Illinois at Chicago,
Computer Center,
Chicago, Illinois

University of Kansas,
Academic Computing Services,
Lawrence, Kansas

Universität Koblenz-Landau,
Koblenz, Federal Republic of Germany

University of Maryland,
Department of Computer Science,
College Park, Maryland

University of Maryland
at College Park,
Computer Science Center,
College Park, Maryland

University of Massachusetts,
Amherst, Massachusetts

University of Oslo,
Institute of Informatics,
Blindern, Oslo, Norway

University of Oslo,
Institute of Mathematics,
Blindern, Oslo, Norway

University of Ottawa,
Ottawa, Ontario, Canada

University of Salford,
Salford, England

University of Southern California,
Information Sciences Institute,
Marina del Rey, California

University of Stockholm,
Department of Mathematics,
Stockholm, Sweden

University of Texas at Austin,
Austin, Texas

University of Washington,
Department of Computer Science,
Seattle, Washington

University of Western Australia,
Regional Computing Centre,
Nedlands, Australia

Uppsala University,
Uppsala, Sweden

Vereinigte Aluminium-Werke AG,
Bonn, Federal Republic of Germany

Villanova University,
Villanova, Pennsylvania

Vrije Universiteit,
Amsterdam, The Netherlands

Washington State University,
Pullman, Washington

Widener University,
Computing Services,
Chester, Pennsylvania

Worcester Polytechnic Institute,
Worcester, Massachusetts

Yale University,
Department of Computer Science,
New Haven, Connecticut

Tame That TEX Lion!

The TEX Tamer Hits the Road

Each year, TEX Users Group offers essential TEX courses in cities convenient to you. Catch these courses in '92!

Beginning/ Intermediate TEX

(No prior knowledge of TEX required)

January 20–24	**San Diego** Hanalei Hotel
February 17–21	**Providence** Biltmore Hotel
July 20–24	**Portland, Oregon** Benson Hotel
October 19–23	**Chicago** TBA
November 2–6	**San Diego** Hanalei Hotel

Intensive Course in LATEX

(No prior knowledge of TEX required)

January 27–31	**San Diego** Hanalei Hotel
February 10–14	**Providence** Biltmore Hotel
May 18–22	**Boston** TBA
August 3–7	**Portland, Oregon** Benson Hotel
October 26–30	**San Diego** Hanalei Hotel
November 9–13	**Providence** Biltmore Hotel

Lodging: Special TUG room rates available to course participants at host hotels

...and makes house calls

On-site courses in TEX and LATEX from TEX Users Group

"I can't get away from the office," you say? No problem. We'll come to you!

- ▼ Courses in TEX and LATEX tailored to the needs of your group
- ▼ Courses at every level from beginning to advanced
- ▼ Five full days of instruction at your site
- ▼ One-week course fee includes all instructor fees and expenses PLUS textbooks and other materials for up to 15 students.
- ▼ Save time and money!
- ▼ If a properly equipped training facility is not available, TUG will arrange computer rentals (not included in course fee.)

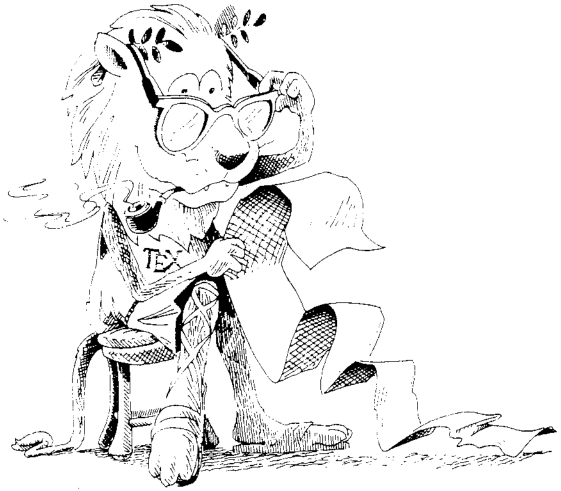
Take a course or two with TUG in '92.

Annual Conference Courses

Before and after the
TEX Users Group 13th Annual Meeting at
The Benson Hotel
Portland, Oregon July 27–30, 1992

July 20–24	Beginning/Intermediate TEX
July 26	TEX for Publishers
July 31–August 1	Practical SGML and TEX Introduction to Typography LAMS-TEX
August 3–7	Advanced TEX and Macro Writing Intensive Course in LATEX

Site/Lodging: Courses will be held at the Benson Hotel, Portland's premier establishment recently restored to its handsome stature of decades ago. The Benson is also the site of TEX Users Group 13th Annual Meeting, **TEX in Context**. Special TUG room rates are available.



©Lions, *The TEXbook*, 1986; used by permission of Addison-Wesley Publishing Co.

Here's what TEXers say about TUG courses:

"Excellent presentation"

"Excellent documentation"

"Flow of the course led to desired learning path; buildup of material to more complicated projects was good."

"No weak points"

"Instructor had a wealth of knowledge regarding the subject and had a strong desire to give us as much as possible."

TEX
USERS
GROUP

For fee information, detailed course descriptions, a registration form and information on the 13th Annual Meeting, contact:

TEX Users Group
P.O. Box 9506
Providence, Rhode Island 02940 U.S.A.

Phone: (401) 751-7760
Fax: (401) 751-1071
Email: tug@math.ams.com

Most courses are conducted lab-style; each participant has his/her own computer.

All TUG members receive discounts on TUG classes.

July 27 to 30, 1992

13th Annual TeX Users Group Meeting

PORTLAND, OREGON

▲ Mark your calendars and join us in Portland, the home of 20-pound salmon and 20-story buildings. Ride light rail trains over cobblestone streets, ski Mt. Hood and attend the symphony in the same day—even in July. A friendly city, Portland charms its visitors with a variety of attractions including:

Windsurfing

A trip up the Columbia River on a sternwheeler

Tours of the wine region

The Metro Washington Park Zoo

Portland Center for the Performing Arts

Oaks Amusement Park

Oregon Art Institute

Scenic Washington County

Oregon Museum of Science and Industry

World Forestry Center

Mt. Hood

Portland Saturday Market for arts and crafts

Of special interest to TUG Meeting attendees may be the 11th Annual Mt. Hood Festival of Jazz to be held August 1st and 2nd in Gresham, Oregon, a suburb of Portland.

For a complete visitors' guide, *The Portland Book*, call the Portland Visitors' Center at (800) 345-3214.

TeX in Context

Resources, Support Tools, and Comparative Studies

▲ During four information-packed days, we'll delve into front-ends for TeX, inclusion of graphics within TeX documents as well as exportation of TeX output to other graphics programs, comparisons of implementations of TeX on microcomputers, network access and resources, educational issues, and translation between TeX and word-processors.

Presentations

Workshops

Networking Luncheons

Exhibits

Panel discussions

Classes

TeX USERS GROUP

▲ We'll meet and stay at the Benson Hotel, Portland's premier hotel recently restored to its grand stature of the early 1900s. A registered historic landmark, the Benson was built by Oregon lumberman, Simon Benson using elaborate craftsmanship and imported wood interiors. Special TUG rates: $89/night (available until June 26 only.)

▲ Program coordinator:

Mimi Lafrenz
ETP Services Co.

Program committee:

Helen Gibson
Wellcome Institute

Doug Henderson
Blue Sky Research

Ron Whitney
TeX Users Group

▲ Watch your mail and future issues of *TUGboat* and *TeX & TUG News* for more details. In the meantime, if you have questions, contact:

TeX Users Group
Phone: (401) 751-7760
Fax: (401) 751-1071
e-mail: `tug@math.ams.com`
P. O. Box 9506
Providence, RI 02940

TeXniques

Publications for the TeX Community

Available now:

1. **VAX Language-Sensitive Editor (LSEDIT) Quick Reference Guide for Use with the LaTeX Environment and LaTeX Style Templates** by Kent McPherson
2. **Table Making – the INRSTeX Method** by Michael J. Ferguson
3. **User's Guide to the IdxTeX Program** by R. L. Aurbach
4. **User's Guide to the GloTeX Program** by R. L. Aurbach
5. **Conference Proceedings**, TeX Users Group Eighth Annual Meeting, Seattle, August 24–26, 1987, Dean Guenther, Editor
6. **The PiCTeX Manual** by Michael J. Wichura
7. **Conference Proceedings**, TeX Users Group Ninth Annual Meeting, Montréal, August 22–24, 1988, Christina Thiele, Editor
8. **A Users' Guide for TeX** by Frances Huth
9. **An Introduction to LaTeX** by Michael Urban
10. **LaTeX Command Summary** by L. Botway and C. Biemesderfer
11. **First Grade TeX** by Arthur Samuel
12. **A Gentle Introduction to TeX** by Michael Doob
13. **METAFONTware** by Donald E. Knuth, Tomas G. Rokicki, and Arthur Samuel
14. **A Permuted Index for TeX and LaTeX** by Bill Cheswick

Coming soon:

15. **EDMAC: A Plain TeX Format for Critical Editions** by John Lavagnino and Dominik Wujastyk

TeX Users Group
P.O. Box 9506
Providence, R.I. 02940, U.S.A.

Individual Membership Application

TEX USERS GROUP

Complete and return this form with payment to:

TEX Users Group
Membership Department
P. O. Box 594
Providence, RI 02901 USA

Telephone: (401) 751-7760
FAX: (401) 751-1071
Email: `tug@Math.AMS.com`

Membership is effective from January 1 to December 31 and includes subscriptions to *TUGboat, The Communications of the TEX Users Group* and the TUG newsletter, *TEX and TUG News*. Members who join after January 1 will receive all issues published that calendar year.

For more information ...

Whether or not you join TUG now, feel free to return this form to request more information. Be sure to include your name and address in the spaces provided to the right.

Check all items you wish to receive below:

□ Institutional membership information
□ Course and meeting information
□ Advertising rates
□ Products/publications catalogue
□ Public domain software catalogue
□ More information on TEX

Name ______________________________

Institutional affiliation, if any ______________________________

Position ______________________________

Address (business or home (circle one)) ______________________________

____

City ______________________________

State or Country ____________________ Zip ____________

Daytime telephone ____________________ FAX ____________

Email addresses (*please specify networks, as well*) ______________________________

____

I am also a member of the following other TEX organizations:

Specific applications or reasons for interest in TEX:

Hardware on which TEX is used:

Computer and operating system *Output device/printer*

____

____

____

There are two types of TUG members: regular members, who pay annual dues of $60; and full-time student members, whose annual dues are $50. Students must include verification of student status with their applications.

Please indicate the type of membership for which you are applying:

□ Regular @ $60 □ Full-time student @ $50

Amount enclosed for 1992 membership: $ ____________

(*Prepayment in US dollars drawn on a US bank is required*)

□ Check/money order payable to TEX Users Group enclosed
□ Charge to MasterCard/VISA

Card # ____________________ Exp. date ________

Signature ______________________________

TEX
USERS
GROUP

Institutional Membership Application

Institution or Organization ____________________

____

Principal contact ____________________

Address ____________________

____

City ____________________

State or Country ____________ Zip ________

Daytime telephone ____________ FAX ________

Email addresses (*please specify networks, as well*) ____________

____

Each Institutional Member is entitled to:

- designate a number of individuals to have full status as TUG individual members;
- take advantage of reduced rates for TUG meetings and courses for *all* staff members;
- be acknowledged in every issue of *TUGboat* published during the membership year.

Educational institutions receive a $100 discount in the membership fee. The three basic categories of Institutional Membership each include a certain number of individual memberships. Additional individual memberships may be obtained at the rates indicated. Fees are as follows:

Category	Rate (educ./non-educ.)	Add'l mem.
A (includes 7 memberships)	$ 540 / $ 640	$50 ea.
B (includes 12 memberships)	$ 815 / $ 915	$50 ea.
C (includes 30 memberships)	$1710 / $1810	$40 ea.

Please indicate the type of membership for which you are applying:

Category _____ + _____ additional individual memberships

Amount enclosed for 1992 membership: $ ________

□ Check/money order payable to TEX Users Group enclosed

(*payment is required in US dollars drawn on a US bank*)

□ Bank transfer bank ____________________

ref # ____________________

□ Charge to MasterCard/VISA

Card # ____________________ Exp. date ________

Signature ____________________

Please attach a corresponding list of individuals whom you wish to designate as TUG individual members. Minimally, we require names and addresses so that TUG publications may be sent directly to these individuals, but we would also appreciate receiving the supplemental information regarding phone numbers, email addresses, TEX interests, and hardware configurations as requested on the TUG Individual Membership Application form. For this purpose, the latter application form may be photocopied and mailed with this form.

Complete and return this form with payment to:

TEX Users Group
Membership Department
P. O. Box 594
Providence, RI 02901 USA

Bank transfers

TEX Users Group, #002-031375
Hospital Trust National Bank
One Hospital Trust Plaza
Providence, RI 02903
USA

Membership is effective from January 1 to December 31. Members who join after January 1 will receive all issues of *TUGboat* published that calendar year.

For more information ...

Correspondence

TEX Users Group
653 North Main Street
P. O. Box 9506
Providence, RI 02940
USA

Telephone: (401) 751-7760
Fax: (401) 751-1071
Email: tug@math.ams.com

Whether or not you join TUG now, feel free to return this form to request more information.

Check all items you wish to receive below:

□ Course and meeting information

□ Products/publications catalogue

□ Public domain software catalogue

TEX Consulting and Production Services

North America

AMERICAN MATHEMATICAL SOCIETY
P. O. Box 6248, Providence, RI 02940; (401) 455-4060
Typesetting from DVI files on an Autologic APS Micro-5 or an Agfa Compugraphic 9600 (PostScript).
Times Roman and Computer Modern fonts.
Composition services for mathematical and technical books and journal production.

ANAGNOSTOPOULOS, Paul C.
433 Rutland Street, Carlisle, MA 01741; (508) 371-2316
Composition and typesetting of high-quality books and technical documents. Production using Computer Modern or any available PostScript fonts. Assistance with book design. I am a computer consultant with a Computer Science education.

ARBORTEXT, Inc.
535 W. William, Suite 300, Ann Arbor, MI 48103;
(313) 996-3566
Typesetting from DVI files on an Autologic APS-5.
Computer Modern and standard Autologic fonts.
TEX installation and applications support.
TEX-related software products.

ARCHETYPE PUBLISHING, Inc.,
Lori McWilliam Pickert
P. O. Box 6567, Champaign, IL 61821; (217) 359-8178
Experienced in producing and editing technical journals with TEX; complete book production from manuscript to camera-ready copy; TEX macro writing including complete macro packages; consulting.

THE BARTLETT PRESS, Inc.,
Frederick H. Bartlett
Harrison Towers, 6F, 575 Easton Avenue,
Somerset, NJ 08873; (201) 745-9412
Vast experience: 100+ macro packages, over 30,000 pages published with our macros; over a decade's experience in all facets of publishing, both TEX and non-TEX; all services from copyediting and design to final mechanicals.

COWAN, Dr. Ray F.
141 Del Medio Ave. #134, Mountain View, CA 94040;
(415) 949-4911
Ten Years of TEX and Related Software Consulting
Books, Documentation, Journals, and Newsletters
TEX & LATEX macropackages, graphics; PostScript language applications; device drivers; fonts; systems.

DOWNES, Michael
49 Weeks Street, North Smithfield, RI 02895;
(401) 762-3715
Instruction in AMS-TEX, AMS-LATEX, plain TEX, and advanced macro writing. Custom documentstyles.
Consulting: ■ advanced mathematical typesetting topics; ■ tuning mathematics fonts; ■ getting the most out of TEX in a production environment. Troubleshooting.

ELECTRONIC TECHNICAL PUBLISHING SERVICES CO.
2906 Northeast Glisan Street, Portland, Oregon 97232-3295;
(503) 234-5522; FAX: (503) 234-5604
Total concept services include editorial, design, illustration, project management, composition and prepress. Our years of experience with TEX and other electronic tools have brought us the expertise to work effectively with publishers, editors, and authors. ETP supports the efforts of the TEX Users Group and the world-wide TEX community in the advancement of superior technical communications.

HOENIG, Alan
17 Bay Avenue, Huntington, NY 11743; (516) 385-0736
TEX typesetting services including complete book production; macro writing; individual and group TEX instruction.

KUMAR, Romesh
1549 Ceals Court, Naperville, IL 60565; (708) 972-4342
Beginners and intermediate group/individual instruction in TEX. Development of TEX macros for specific purposes. Using TEX with FORTRAN for custom-tailored software. Flexible hours, including evenings and weekends.

MAGUS, Kevin W. Thompson
P. O. Box 390965, Mountain View CA 94039-0965;
(800) 848-8037; (415) 940-1109; magus@cup.portal.com
LATEX consulting from start to finish. Layout design and implementation, macro writing, training, phone support, and publishing. Can take LATEX files and return camera ready copy. Knowledgeable about long document preparation and mathematical formatting.

OGAWA, Arthur
920 Addison, Palo Alto, CA 94301; (415) 323-9624
Experienced in book production, macro packages, programming, and consultation. Complete book production from computer-readable copy to camera-ready copy.

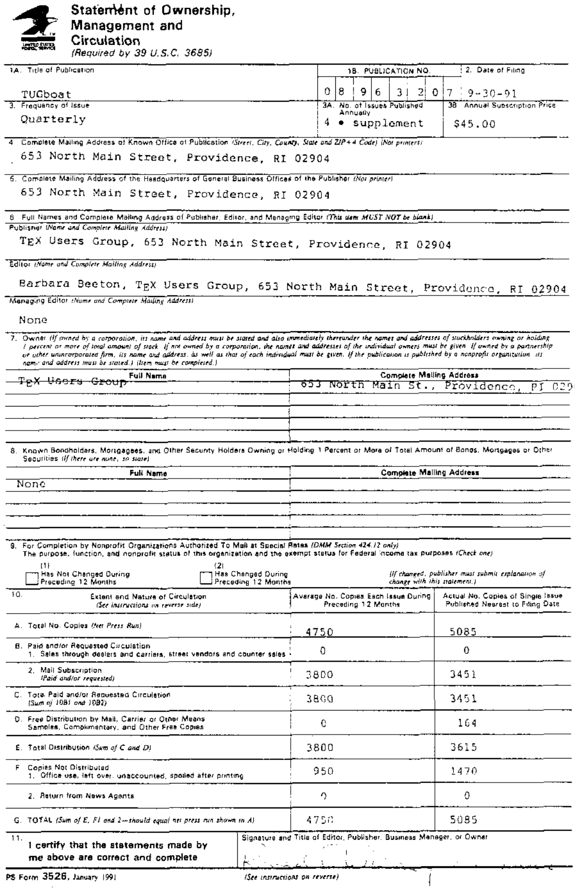
QUIXOTE, Don Hosek
440F Grinnell, Claremont, CA 91711; (714) 625-0147
Complete line of TEX, LATEX, and METAFONT services including custom LATEX style files, complete book production from manuscript to camera-ready copy; custom font and logo design; installation of customized TEX environments; phone consulting service; database applications and more.
Call for a free estimate.

RICHERT, Norman
1614 Loch Lake Drive, El Lago, TX 77586;
(713) 326-2583
TEX macro consulting.

TEXNOLOGY, Inc., Amy Hendrickson
57 Longwood Ave., Brookline, MA 02146;
(617) 738-8029.
TEX macro writing (author of MacroTEX); custom macros written to meet publisher's or designer's specifications; instruction.

Outside North America

TYPOTEX LTD.
Electronical Publishing, Battyány u. 14. Budapest, Hungary
H-1015; (036) 11152 337
Editing and typesetting technical journals and books with TEX from manuscript to camera ready copy. Macro writing, font designing, TEX consulting and teaching.



Statement of Ownership, Management and Circulation
(Required by 39 U.S.C. 3685)

1A. Title of Publication TUGboat	1B. PUBLICATION NO. 0 8 9 6 3 2 0 7	2. Date of Filing 9-30-91
3. Frequency of Issue Quarterly	3A. No. of Issues Published Annually 4 • supplement	3B Annual Subscription Price $45.00

4 Complete Mailing Address of Known Office of Publication *(Street, City, County, State and ZIP+4 Code) (Not printers)*
653 North Main Street, Providence, RI 02904

5. Complete Mailing Address of the Headquarters of General Business Offices of the Publisher *(Not printer)*
653 North Main Street, Providence, RI 02904

6 Full Names and Complete Mailing Address of Publisher, Editor, and Managing Editor *(This item MUST NOT be blank)*
Publisher *(Name and Complete Mailing Address)*
TEX Users Group, 653 North Main Street, Providence, RI 02904

Editor *(Name and Complete Mailing Address)*
Barbara Beeton, TEX Users Group, 653 North Main Street, Providence, RI 02904

Managing Editor *(Name and Complete Mailing Address)*
None

7. Owner *(If owned by a corporation, its name and address must be stated and also immediately thereunder the names and addresses of stockholders owning or holding 1 percent or more of total amount of stock. If not owned by a corporation, the names and addresses of the individual owners must be given. If owned by a partnership or other unincorporated firm, its name and address, as well as that of each individual must be given. If the publication is published by a nonprofit organization, its name and address must be stated.) (Item must be completed.)*

Full Name	Complete Mailing Address
~~TEX Users Group~~	653 North Main St., Providence, RI 02904

8. Known Bondholders, Mortgagees, and Other Security Holders Owning or Holding 1 Percent or More of Total Amount of Bonds, Mortgages or Other Securities *(If there are none, so state)*

Full Name	Complete Mailing Address
None	

9. For Completion by Nonprofit Organizations Authorized To Mail at Special Rates *(DMM Section 424.12 only)*
The purpose, function, and nonprofit status of this organization and the exempt status for Federal income tax purposes *(Check one)*

(1) □ Has Not Changed During Preceding 12 Months (2) □ Has Changed During Preceding 12 Months *(If changed, publisher must submit explanation of change with this statement.)*

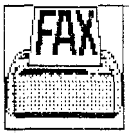
10. Extent and Nature of Circulation *(See instructions on reverse side)*	Average No. Copies Each Issue During Preceding 12 Months	Actual No. Copies of Single Issue Published Nearest to Filing Date
A. Total No. Copies *(Net Press Run)*	4750	5085
B. Paid and/or Requested Circulation 1. Sales through dealers and carriers, street vendors and counter sales	0	0
2. Mail Subscription *(Paid and/or requested)*	3800	3451
C. Total Paid and/or Requested Circulation *(Sum of 10B1 and 10B2)*	3800	3451
D. Free Distribution by Mail, Carrier or Other Means Samples, Complimentary, and Other Free Copies	0	164
E. Total Distribution *(Sum of C and D)*	3800	3615
F Copies Not Distributed 1. Office use, left over, unaccounted, spoiled after printing	950	1470
2. Return from News Agents	0	0
G. TOTAL *(Sum of E, F1 and 2—should equal net press run shown in A)*	4750	5085

11. **I certify that the statements made by me above are correct and complete** Signature and Title of Editor, Publisher, Business Manager, or Owner

PS Form **3526**, January 1991 *(See instructions on reverse)*

Index of Advertisers

587	American Mathematical Society
576	ArborText
Cover 3	Blue Sky Research
584	Computer Composition
579	Electronic Technical Publishing Services
586	Yannis Haralambous
575	job ctl
582	K-Talk Communications
580, 581	Kinch Computer Company
583	MicroPress, Inc.
587	Micro Programs, Inc.
578	Personal TEX Inc.
585	TCI Software
577	Type 2000
588	Y&Y



faxpak provides Group 3 facsimile capabilities to networked SUNs and XENIX 2.3.2 and compatible systems. Supports SIERRA type modems such as WORLDPORT's 2496 and "Class Two" modems.

File Formats: • Plain ASCII • TEX and LATEX[1] • POSTSCRIPT text and graphics • SUN raster and other bitmaps • Easily extended to any bitmap or file format.

Configuable Options: • Multiple phone lines • View received faxes on screen • Aliases, distributions lists, batch jobs • Complex permissions scheme or unrestricted access • Departmental or system wide "cover pages" • Page numbering • Pasting up of bitmaps • Restriction of transmissions to "off peak" rates • "Pickup Mode" to avoid resending confirmed pages after errors • Accounting.

faxpak comes with a site licence and costs $360. plus $25 for shipping by AIR MAIL.. Details from: job ctl, Klaus Schallhorn, 28 Belgravia St., Penzance, Cornwall, TR18 2BJ, UK. FAX +44 736 330083, <faxinfo@cnix.uuxp>.

[1] This ad has been faxed by *faxpak*.

A complete TeX solution that implements all the new TeX 3.0 capabilities, virtual fonts, and the Extended Font standard adopted at the TUG '90 meeting in Cork.

ArborText put it all together. You don't have to!

ArborText's TeX 3.14 provides everything you need in a complete, ready-to-use package:

Utilize the Extended TeX Font Encoding capability with pre-built virtual fonts for Computer Modern and PostScript

Use the conversion utilities we supply to make your own extended fonts from existing TeX 2.0 style fonts

Easily accent characters from your foreign language keyboard

Create multi-language documents

Choose from included hyphenation patterns for English, French, German, Dutch, Spanish, Portuguese, or add your own

Use the extended version of Plain TeX and LaTeX

We've provided access to the New Extended Fonts directly—
macro source included!

TeX 3.14 and support software is available for Sun, IBM RS6000 DEC/Risc-Ultrix, HP 9000, and IBM PCs,

ARBORTEXT INC. 535 West William Street AnnArbor, MI 48103 FAX (313) 996–3573 Phone (313) 996–3566

TYPESETTING: JUST $2.50 PER PAGE!

Send us your TEX 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 PCTEX® 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

Everything You Need At One Low Price...

Announcing the New PCTEX Systems!

You can now receive a new PC TEX System, which includes PC TEX/386 plus a full set of printer drivers, complete with everything you need to create the highest quality typeset documents possible using a PC, all at one low price. We offer a **20% Discount to TUG Members.** Here are your choices:

The PC TEX System for Laser Printers Includes:

- PC TEX
- PC TEX/386
- PTI View
- PTI Laser/HP
- PTI Laser/PS
- PTI Jet
- CM 300dpi Fonts

Retail: $599
TUG Members: $479

The Big PC TEX System for Laser Printers Includes:

- PC TEX
- Big PC TEX/386
- PTI View
- PTI Laser/HP
- PTI Laser/PS
- PTI Jet
- CM 300dpi Fonts

Retail: $699
TUG Members: $559

The PC TEX System for Dot Matrix Printers Includes:

- PC TEX
- PC TEX/386
- PTI View
- PTI Dot/FX
- PTI Dot/LQ
- CM 240dpi & 180dpi Fonts

Retail: $499
TUG Members: $399

Upgrade your Current Products and Get a Full Set of Printer Drivers, plus PCTEX/386, for only $195

For those of you who already own PC TEX, PTI View, and at least one PTI Printer Driver, special System Upgrades* are available to you as follows:

PC TEX Laser System Upgrade - $195
Big PC TEX Laser System Upgrade - $245
PC TEX Dot Matrix System Upgrade - $175

One Stop Shopping from Personal TEX, Inc.

We offer you a full range of TEX products to meet your every need... including graphics programs, fonts, spell-checkers, text editors, and TEX macros. Look for our new LATEX book, *LATEX for Everyone*, coming soon. For our free 1991 Product Catalog, demo diskette, or for further information, **call us today at (415) 388-8853.**

PERSONAL
TEX
INC

12 Madrona Avenue • Mill Valley, CA 94941 • Phone: (415) 388-8853 • Fax: (415) 388-8865

In Europe: (31) 703237241 • (49) 24167001 • (49) 80248011 • (49) 73126932 • (44) 742351489 • (39) 290091773 (33) 169073688 • In Asia: (886) 35335179 • In Australia: (61) 34599671

* You must provide proof of prior purchase of PC TEX, PTI View, and a PTI Printer Driver. Upgrades do not include CM Fonts. PC TEX is a registered TM of Personal TEX, Inc. TEX is an American Mathematical Society TM. Site licenses available to qualified organizations. Inquire about PTI distributorships. This ad was typeset using PC TEX and Bitstream Fonts.

The solution is ETP.

$$
\begin{aligned}
\Delta\mathcal{P} = \sum_{W} \Big[Q_{\mathrm{IPR}} \int_{4\text{-}1\text{-}87}^{\infty} & (D_p + D_m + D_s)^T \\
& + \epsilon(P_m - I_P)\,dt \Big] \\
\equiv \mathrm{ETP} &
\end{aligned}
$$

ETP Services offers solutions to the problems
facing the publishers of technical books
and journals, with a complete array
of composition-related services.

Electronic Technical Publishing Services Company
2906 N.E. Glisan Street
Portland, Oregon 97232
503-234-5522 • FAX: 503-234-5604
mimi@etp.com

AP-TEX Fonts

TEX-compatible Bit-Mapped Fonts Identical to Adobe PostScript Typefaces

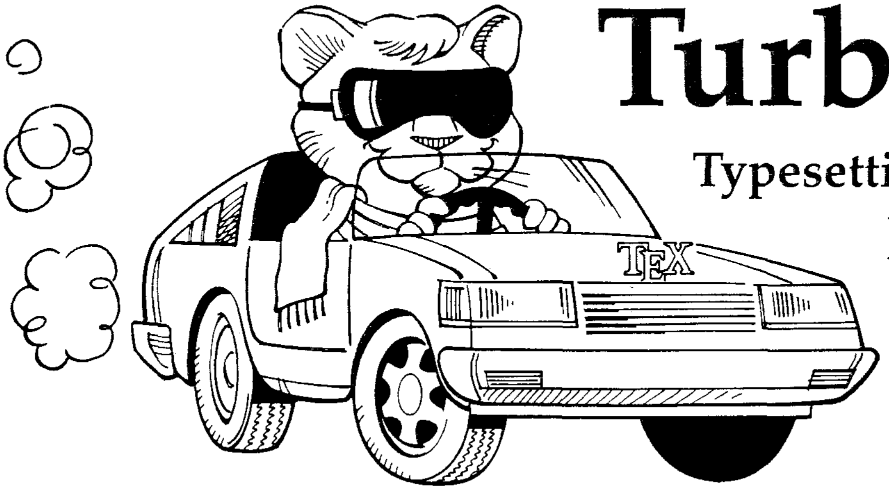
If you are hungry for new TEX fonts, here is a feast guaranteed to satisfy the biggest appetite! The AP-TEX fonts serve you a banquet of gourmet delights: 438 fonts covering 18 sizes of 35 styles, at a total price of $200. The AP-TEX fonts consist of `PK` and `TFM` files which are exact TEX-compatible equivalents (including "hinted" pixels) to the popular PostScript name-brand fonts shown at the right. Since they are directly compatible with any standard TEX implementation (including kerning and ligatures), you don't have to be a TEX expert to install or use them.

When ordering, specify resolution of 300 dpi (for laser printers), 180 dpi (for 24-pin dot matrix printers), or 118 dpi (for previewers). Each set is on ten 360 KB 5-1/4" PC floppy disks. The $200 price applies to the first set you order; order additional sets at other resolutions for $60 each. A 30-page user's guide fully explains how to install and use the fonts. Sizes included are 5, 6, 7, 8, 9, 10, 11, 12, 14.4, 17.3, 20.7, and 24.9 points; headline styles (equivalent to Times Roman, Helvetica, and Palatino, all in bold) also include sizes 29.9, 35.8, 43.0, 51.6, 61.9, and 74.3 points.

The Kinch Computer Company
PUBLISHERS OF TURBOTEX
501 South Meadow Street
Ithaca, New York 14850
Telephone (607) 273-0222
FAX (607) 273-0484

Helvetica, Palatino, Times, and New Century Schoolbook are trademarks of Allied Linotype Co. ITC Avant Garde, ITC Bookman, ITC Zapf Chancery, and ITC Zapf Dingbats are registered trademarks of International Typeface Corporation. PostScript is a registered trademark of Adobe Systems Incorporated. The owners of these trademarks and Adobe Systems, Inc. are not the authors, publishers, or licensors of the AP-TEX fonts. Kinch Computer Company is the sole author of the AP-TEX fonts, and has operated independently of the trademark owners and Adobe Systems, Inc. in publishing this software. Any reference in the AP-TEX font software or in this advertisement to these trademarks is solely for software compatibility or product comparison. LaserJet and DeskJet are trademarks of Hewlett-Packard Corporation. TEX is a trademark of the American Math Society. TurboTEX and AP-TEX are trademarks of Kinch Computer Company. Prices and specifications subject to change without notice. Revised October 9, 1990.

Avant Garde Bold
Avant Garde Bold Oblique
Avant Garde Demibold
Avant Garde Demibold Oblique
Bookman Light
Bookman Light Italic
Bookman Demibold
Bookman Demibold Italic
Courier
Courier Oblique
Courier Bold
Courier Bold Oblique
Helvetica
Helvetica Oblique
Helvetica Bold
Helvetica Bold Oblique
Helvetica Narrow
Helvetica Narrow Oblique
Helvetica Narrow Bold
Helvetica Narrow Bold Oblique
Schoolbook New Century Roman
Schoolbook New Century Italic
Schoolbook New Century Bold
Schoolbook New Century Bold Italic
Palatino Roman
Palatino Italic
Palatino Bold
Palatino Bold Italic
Times Roman
Times Italic
Times Bold
Times Bold Italic
Zapf Chancery Medium Italic
Symbol ΔΦΓϑΛΠΘ
Zapf Dingbats ✂✄☞❑



TurboTeX

Typesetting Software

Executables $150
With Source $300

Now Windows Compatible!

Now you can run the TeX typesetting system in the powerful and convenient graphical environment of Microsoft Windows, with the new Windows-compatible TurboTeX Release 3.1.

TurboTeX brings you the latest TeX 3.1 and METAFONT 2.7 standards and certifications: preloaded plain TeX, LaTeX, AMS-TeX and AMS-LaTeX, METAFONT, preview for EGA/VGA displays, Computer Modern and LaTeX fonts, and printer drivers for HP LaserJet and DeskJet, PostScript, and Epson LQ and FX dot-matrix printers. This wealth of software runs on your IBM PC (MS-DOS, Windows, or OS/2), Unix, or VAX/VMS system.

■ **Best-selling Value:** TurboTeX sets the standard for power and value among TeX implementations: one price buys a complete, commercially-hardened typesetting system. *Computer* magazine recommended it as "the version of TeX to have," *IEEE Software* called it "industrial strength," and thousands of satisfied users worldwide agree.

TurboTeX gets you started quickly, installing itself automatically under MS-DOS or Microsoft Windows, and compiling itself automatically under Unix. The 90-page User's Guide includes generous examples and a full index, and leads you step-by-step through installing and using TeX and METAFONT.

■ **Classic TeX for Windows.** Even if you have never used Windows on your PC, the speed and power of TurboTeX will convince you of the benefits. While the TeX command-line options and *TeXbook* interaction work the same, you also can control TeX using friendly icons, menus, and dialog boxes. Windows protected mode frees you from MS-DOS limitations like DOS extenders, overlay swapping, and scarce memory. You can run long TeX formatting or printing jobs in the background while using other programs in the foreground.

■ **MS-DOS Power, Too:** TurboTeX still includes the plain MS-DOS programs. Even without expanded memory hardware, our virtual memory simulation provides the same sized TeX that runs on multi-megabyte mainframes, with capacity for large documents, complicated formats, and demanding macro packages.

■ **Source Code:** The portable C source to TurboTeX consists of over 100,000 lines of generously commented TeX, TurboTeX, METAFONT, previewer, and printer driver source code, including: our WEB system in C; PASCHAL, our proprietary Pascal-to-C translator; Windows menus and text-mode interface library; and preloading, virtual memory, and graphics code, all meeting C portability standards like ANSI and K&R.

■ **Availability & Requirements:** TurboTeX executables for IBM PC's include the User's Guide and require 640K, hard disk, and MS-DOS 3.0 or later. Windows extensions require Microsoft Windows 3.0. Order source code (includes Programmer's Guide) for other machines. On the PC, source compiles with Microsoft C 5.0 or later (and Windows SDK for Windows extensions), Watcom C 8.0, or Borland C++ 2.0; other operating systems need a 32-bit C compiler supporting Unix standard I/O. Media is 360K 5-1/4" or 720K 3-1/2" PC floppy disks (please specify).

■ **Upgrade at Low Cost.** If you have TurboTeX Release 3.0, upgrade to the latest version for just $40 (executables) or $80 (including source). Or, get either applicable upgrade free when you buy the AP-TeX fonts (see facing page) for $200!

■ **No-risk trial offer:** Examine the documentation and run the PC TurboTeX for 10 days. If you are not satisfied, return it for a 100% refund or credit. (Offer applies to PC executables only.)

■ **Free Buyer's Guide:** Ask for the free, 70-page Buyer's Guide for details on TurboTeX and dozens of TeX-related products: previewers, TeX-to-FAX and TeX-to-Ventura/Pagemaker translators, optional fonts, graphics editors, public domain TeX accessory software, books and reports.

Ordering TurboTeX

Ordering TurboTeX is easy and delivery is fast, 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. International orders gladly expedited via Air or Express Mail.

The Kinch Computer Company
Publishers of TurboTeX
501 South Meadow Street
Ithaca, New York 14850 USA
Telephone (607) 273-0222
FAX (607) 273-0484

Publishing Companion® translates

WordPerfect

to

TeX or LaTeX

IN ONE EASY STEP!

With **Publishing Companion**, you can publish documents using TeX or LaTeX with **little or no TeX knowledge.** Your WordPerfect files are translated into TeX or LaTeX files, so anyone using this simple word processor can immediately begin typesetting their own documents!

Publishing Companion translates EQUATIONS, FOOTNOTES, ENDNOTES, FONT STYLES, and much more!

Retail Price	$249.00
Academic Discount Price	$199.00

For more information or to place an order, call or write:

K-TALK®
COMMUNICATIONS

30 West First Ave, Suite 100
Columbus, Ohio 43201
(614)294-3535
FAX (614)294-3704

TYPESET QUALITY WITH THE EASE OF WORD PROCESSING

9009-9

VECTOR TEX

TEX FOR THE 90'S

Are you still struggling with PXL's, PK's or GF's? Move on to scalable fonts:

- Save megabytes of storage–entire VT$_E$X fits on one floppy.
- Instantly generate any font in any size and in any variation from 5 to 100 points.
- Standard font effects include compression, slant, smallcaps, outline, shading and shadow.
  New: landscape. New: scalable graphics.
- Discover the universe of MicroPress Font Library professional typefaces: not available from any other T$_E$X vender.

List price **$299**

Includes the VT$_E$X typesetter (superset of T$_E$X), 10 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.
S/H add $5. COD add $5.
WordPerfect Interface add $100. Site licenses available.
Dealers' inquiries welcome. Professional typefaces available for older implementations of T$_E$X.

MICRO
PRESS

MicroPress Inc.
68-30 Harrow Street, Forest Hills, NY 11375
Tel: (718) 575-1816 Fax: (718) 575-8038

VT$_E$X is a trademark of MicroPress Inc. Other Products mentioned are trademarks of their respective companies.

For TEX Users....

New Services and Prices from Computer Composition Corporation

We are pleased to announce the installation of several ***new output services*** now available to TEX users:

1. High Resolution Laser Imaging (1200 dpi) from *Postscript diskette files* created on either Mac- or PC-based systems.
2. High Resolution Laser Imaging (960 dpi) from *DVI magnetic tape or diskette files* using a variety of typefaces in addition to the Computer Modern typeface family.
3. High quality laser page proofs at 480 dpi.
4. **NEW PRICING** for high resolution laser imaging:

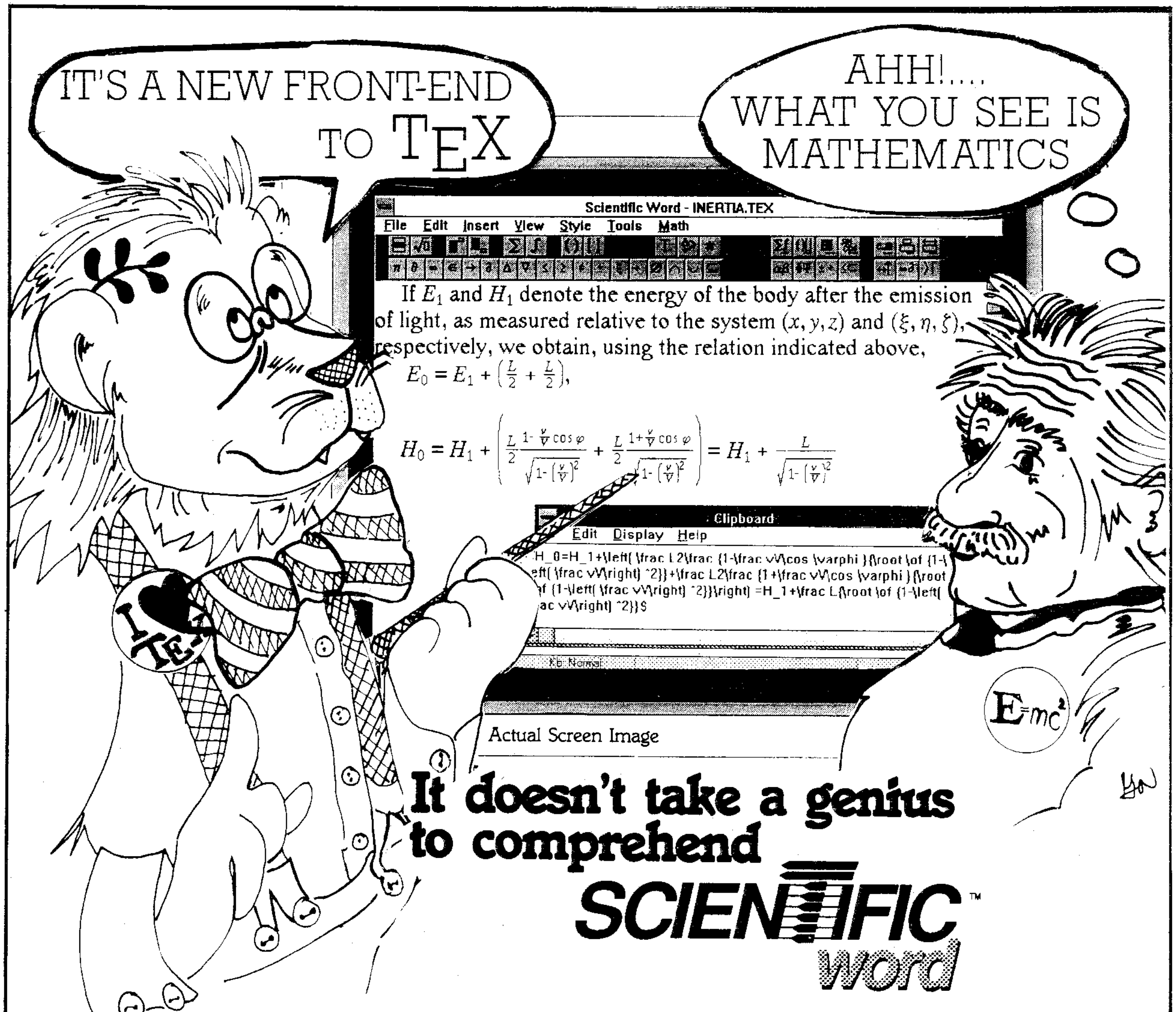
a.	From **Postscript text files** in volumes over 400 pages	**$2.00 per page**
b.	From **Postscript text files** in volumes between 100 & 400 pages	**$2.25 per page**
c.	From **Postscript text files** in volumes below 100 pages	**$2.40 per page**
d.	From **DVI files** in volumes over 400 pages	**$2.15 per page**
e.	From **DVI files** in volumes between 100 & 400 pages	**$2.30 per page**
f.	From **DVI files** in volumes below 100 pages	**$2.45 per page**

NOTE: DEDUCT $1.00 FROM THE ABOVE PRICES FOR HIGH QUALITY LASER PAGE PROOFS.

5. **All jobs shipped within 48 hours.**

Call or write for page samples or send us your file and we will image it on the output unit of your choice.

COMPUTER COMPOSITION CORPORATION
1401 West Girard Avenue • Madison Heights, MI 48071
(313) 545-4330 FAX (313) 544-1611
— *Since 1970* —


IT'S A NEW FRONT-END TO TEX
AHH!.... WHAT YOU SEE IS MATHEMATICS
Scientific Word - INERTIA.TEX
File Edit Insert View Style Tools Math
Clipboard
Edit Display Help
Kb Normal
Actual Screen Image
E=mc²
I ♥ TEX
It doesn't take a genius to comprehend
SCIENTIFIC™ word

If E_1 and H_1 denote the energy of the body after the emission of light, as measured relative to the system (x, y, z) and (ξ, η, ζ), respectively, we obtain, using the relation indicated above,

$$E_0 = E_1 + \left(\frac{L}{2} + \frac{L}{2}\right),$$

$$H_0 = H_1 + \left(\frac{L}{2}\frac{1-\frac{v}{V}\cos\varphi}{\sqrt{1-\left(\frac{v}{V}\right)^2}} + \frac{L}{2}\frac{1+\frac{v}{V}\cos\varphi}{\sqrt{1-\left(\frac{v}{V}\right)^2}}\right) = H_1 + \frac{L}{\sqrt{1-\left(\frac{v}{V}\right)^2}}$$

```
H_0=H_1+\left( \frac L2\frac {1-\frac vV\cos \varphi }{\root \of {1-\
eft( \frac vV\right) ^2}}+\frac L2\frac {1+\frac vV\cos \varphi }{\root
f {1-\left( \frac vV\right) ^2}}\right) =H_1+\frac L{\root \of {1-\left(
ac vV\right) ^2}}$
```

In fact, it's created by the same scientists who brought you T3™, TCI Software Research Inc.

Scientific Word™ is the latest in PC word processing for Windows 3.0.

The file storage format is TEX. It's a full document editor, not a previewer. You compose and edit directly on the screen without being forced to think in TEX.

Your input is mathematics, and your output is TEX.

Discover the genius when you combine the power of TEX with the simplicity of **Scientific Word**™.

To be a part of this exciting new discovery, contact TCI Software Research Inc. Call today, toll free **1-800-874-2383** for more information.

TCI
SOFTWARE RESEARCH, INC.

1190 FOSTER ROAD
LAS CRUCES, NM
88001

1-800-874-2383
TEL: (505) 522-4600
FAX: (505) 522-0116

T3 and Scientific Word are trademarks of TCI Software Research Inc. TEX is a trademark of the American Mathematical Society
Windows is a trademark of Microsoft.

كَٱلْأَنْعَٰمِ بَلْ هُمْ أَضَلُّ أُو۟لَٰٓئِكَ هُمُ ٱلْغَٰفِلُونَ

וַיָּשִׂימוּ עָלָיו שָׂרֵי מִסִּים לְמַעַ

ܘܫܘܠܛܐ ܐ ܒܪܢܫܐ ܠܟܠܗܘܢ ܥܒܕ̈ܘܗܝ ܘܢܘܨܚܘܢ

Եւ ոչ ունէք պարտապանք մնացեալ

Ἦμος δ’ ἠριγένεια φάνη ῥοδοδάκτ

ΕΓΙΤΟΥΤΩΝΑΓΗΝΕΓΚΑΝΟΙΦΩΚΙΓΥΛΑΙΑ
ꟼƎTTΥƎΔΑΤИOꓘΑIꟼTΑTИΑΛΑTƸΑИIꟼHƸ

Ac ꞅƿẏlce þinᵹ ᵹeƿuꞃðað ꝼoꞃ ꝼolceꞃ ꞅẏ

Ihr naht euch wieder, ſchwankende Geſtalten,

Von der Tůnŏwe, als ſi gat und

SCHOLAR TEX®

Use the combined power of TEX, Metafont and PostScript® to create high quality documents providing classical and modern Arabic, Persian, Ottoman Turkish, Pashto, Urdu, Malay, classical Hebrew, Ivrit, Yiddish, Syriac Estrangelo, Armenian, Greek, epigraphical Greek & Latin, Saxon, old German Fraktur and Schwabacher (*forthcoming:* Glagolitic, old Church Cyrillic, Byzantine Greek, Coptic, old Irish, Syriac Serto and Uiguric Mongolian). • All fonts in pk, EPSF, PostScript® Type 1 *and* TrueType™ Format • User-defined transcription for input and output of Semitic languages & virtual fonts used for accented Arabic characters • Continuous support for improvements & additions.

Individuals: $200 (specify Macintosh®-Textures™, Macintosh®-OzTex or PC); for the sources (in Metafont, WEB and PostScript®) additional $100. *Institutions, Publishers:* $500 (sources included). Orders and information from: Yannis Haralambous, 101/11 rue Breughel, 59650 Villeneuve d'Ascq, France, Fax (33) 20.91.05.64.

ScholarTEX is a registered trademark of Yannis Haralambous.

TeX 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 401-455-4060 or 800-321-4AMS in the continental U.S.

VALUABLE ADDITIONS TO YOUR TeX TOOLBOX

CAPTURE

Capture graphics generated by CAD, circuit design, data plotters, and other application programs that support the LaserJet. Make LaserJet images compatible with TeX. Create pk files from pcl or pcx files. $115.00

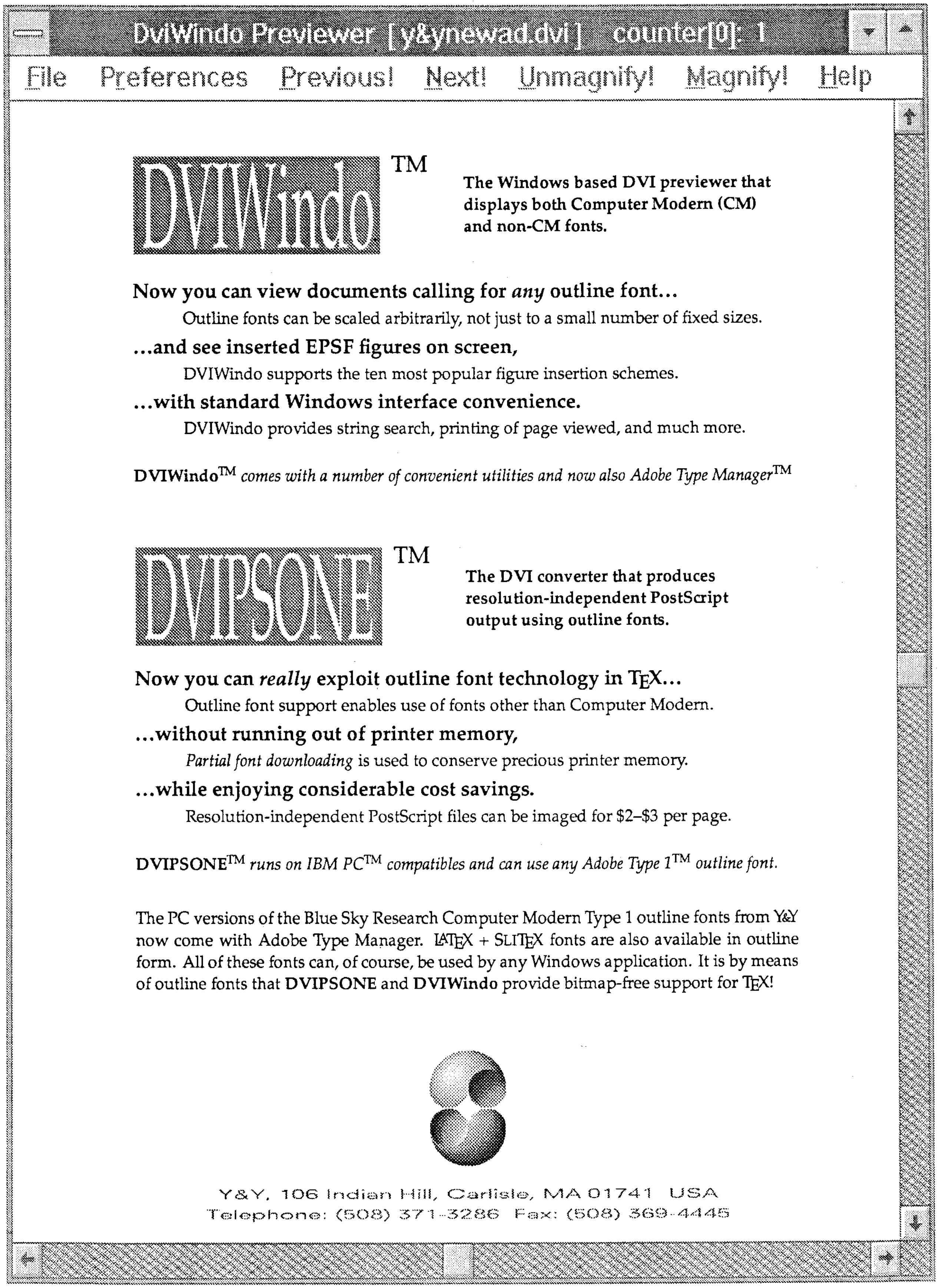
texpic

With texpic graphics package, you have the tools to integrate simple graphics—boxes, circles, ellipses, lines, arrows—into your TeX documents. Maintains output device independence. $79.00

Voyager

Macros to produce viewgraphs quickly and easily using TeX. They provide format, indentation, font, and spacing control. Macros included to produce vertical and horizontal bar charts. $25.00

MP Micro Programs Inc. 251 Jackson Ave. Syosset, NY 11791 (516) 921-1351



DVIWindo™

The Windows based DVI previewer that displays both Computer Modern (CM) and non-CM fonts.

Now you can view documents calling for *any* outline font...

Outline fonts can be scaled arbitrarily, not just to a small number of fixed sizes.

...and see inserted EPSF figures on screen,

DVIWindo supports the ten most popular figure insertion schemes.

...with standard Windows interface convenience.

DVIWindo provides string search, printing of page viewed, and much more.

DVIWindo™ *comes with a number of convenient utilities and now also Adobe Type Manager*™

DVIPSONE™

The DVI converter that produces resolution-independent PostScript output using outline fonts.

Now you can *really* exploit outline font technology in TEX...

Outline font support enables use of fonts other than Computer Modern.

...without running out of printer memory,

Partial font downloading is used to conserve precious printer memory.

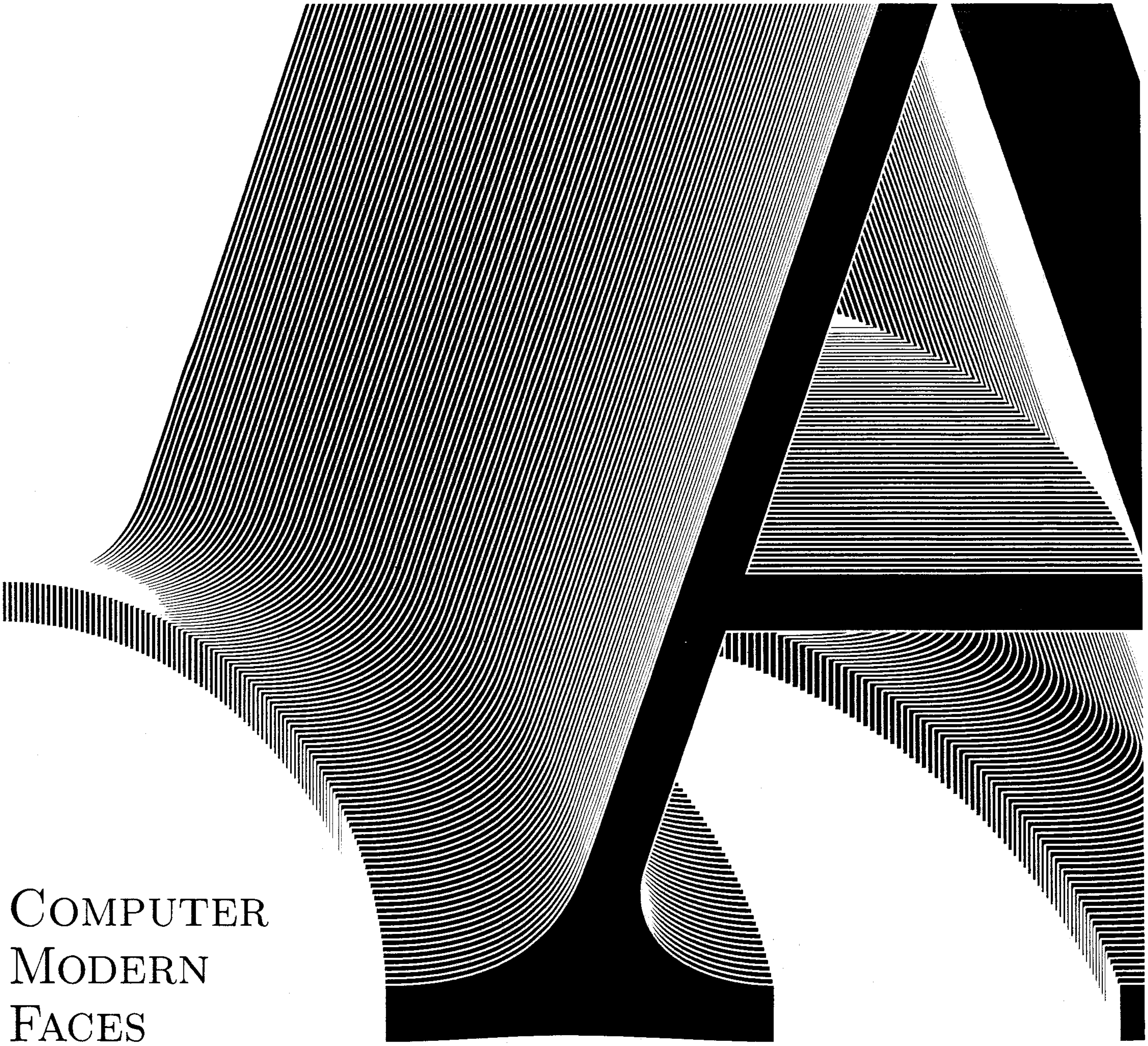
...while enjoying considerable cost savings.

Resolution-independent PostScript files can be imaged for $2–$3 per page.

DVIPSONE™ *runs on IBM PC*™ *compatibles and can use any Adobe Type 1*™ *outline font.*

The PC versions of the Blue Sky Research Computer Modern Type 1 outline fonts from Y&Y now come with Adobe Type Manager. LATEX + SLITEX fonts are also available in outline form. All of these fonts can, of course, be used by any Windows application. It is by means of outline fonts that **DVIPSONE** and **DVIWindo** provide bitmap-free support for TEX!

Y&Y, 106 Indian Hill, Carlisle, MA 01741 USA
Telephone: (508) 371-3286 Fax: (508) 369-4445



COMPUTER
MODERN
FACES

ADOBE
TYPE 1
POSTSCRIPT
FONTS

BLUE
SKY
RESEARCH

Forty faces of Computer Modern
designed by Donald Knuth
published in Adobe Type 1 format
compatible with
Adobe Type Manager
and all PostScript printers

$345.00 Educational $195.00
Macintosh or MS-DOS

Blue Sky Research
534 Southwest Third Avenue
Portland, Oregon 97204 USA
(800) 622-8398, (503) 222-9571
FAX (503) 222-1643

TUGboat

Volume 12, Number 4 / December 1991
1991 TUG Conference Proceedings — Part 2

Part 1

Part 2