Baskerville

The Annals of the UK TEX Users' Group **Editor: Editor: Sebastian Rahtz** **Vol. 3 No. 1**

ISSN 1354–5930 **February 1998**

Articles may be submitted via electronic mail to `baskerville@tex.ac.uk`, or on MSDOS-compatible discs, to Sebastian Rahtz, Elsevier Science Ltd, The Boulevard, Langford Lane, Kidlington, Oxford OX5 1GB, to whom any correspondence concerning *Baskerville* should also be addressed.

This reprint of *Baskerville* is set in Times Roman, with Computer Modern Typewriter for literal text; the source is archived on CTAN in `usergrps/uktug`.

Back issues from the previous 12 months may be ordered from UKTUG for £2 each; earlier issues are archived on CTAN in `usergrps/uktug`.

Please send UKTUG subscriptions, and book or software orders, to Peter Abbott, 1 Eymore Close, Selly Oak, Birmingham B29 4LB. Fax/telephone: 0121 476 2159. Email enquiries about UKTUG to `uktug-enquiries@tex.ac.uk`.

Contents

I Editorial

Sebastian Rahtz
ArchaeoInformatica
York

This issue of *Baskerville* was largely developed by Sue Brooks, but pressure of work forced her to hand over the final production to the current editor. Many thanks to Sue for her work over the last year on this project.

The UKTUG committee is very aware of the fact that *Baskerville* has not been issued as regularly or frequently as the members deserve, and from this autumn it is planned to produce *Baskerville* six times a year on fixed dates. It is also planned to regularly publish articles on common TEX questions in *Baskerville*, and make these available as technical notes to new members in the future. Members are urged to contribute short (one side of A4) 'topical tips' from which we can start to build a library to answer the many queries from those who do not participate in academic electronic network discussions.

This isssue of the journal was created entirely with LATEX and printed on a Linotronic 300 at Aston University. It was set in ITC New Baskerville Roman, with Computer Modern Typewriter for literal text.

II LaTeX2e – A New Version of LaTeX

Leslie Lamport and the LaTeX3 project team

An important announcement was made on July 27th at the Annual meeting of the TeX User's Group (TUG) at Aston University, Birmingham, UK; there will be a new, standardised version of LaTeX (working name: LaTeX2e) to be released before the end of 1993.

Reasons For LaTeX2e

There are two primary reasons for introducing a new version of LaTeX:

- Standardisation: a single format incorporating NFSS2, to replace present multiplicity of incompatible formats (NFSS, lfonts, pslfonts, etc.)
- Maintenance: a standardised system is essential to a reliable maintenance policy.

Note that LaTeX2e is only a "working name"—thus this may change.

Guiding Principles

The following two guiding principles are to be followed:

1. Unmodified version 2.09 input files will produce the same output with LaTeX2e as with version 2.09.
2. All new features of LaTeX2e will conform to the conventions of version 2.09, making it as easy as possible for current users to learn to use them.

Preamble Commands

In order to distinguish old (2.09) documents from those using facilities from the new version, LaTeX2e documents will use a different command on the first line:

```
\documentclass[options]{class}
```

The `documentclass` command specifies what kind of document this is—for example, article, book, letter, slide. There is a second command:

```
\includepackage[options]{package}
```

Including a package adds new commands and/or redefines existing commands to provide additional functionality. The LaTeX2.09 compatibility mode is invoked by

```
\documentstyle[options]{style}
```

Documentation

The new version will be described in a new edition of *LaTeX: A document preparation system* by Leslie Lamport and in *The LaTeX Companion* by Goossens, Mittelbach and Samarin (both to be published by Addison-Wesley). The *Companion* will also contain a complete description of NFSS2.

Distribution Policy

Maintenance of the new system will be undertaken by the LaTeX3 project team.

A complete distribution of all files, incorporating corrections of errors, will be made available twice a year on fixed dates. This will happen even if there were no changes to the files, and hence only the release dates have to be updated.

We are currently looking into the possibility of additionally distributing 'diff' files.

Error Reports

Error reports can be made using a report generating program `latexbug.tex`. This will be part of the main distribution.

Error reports will be accepted only if the version of LaTeX2e that produces the error is not older than one year. Error reports can be sent to the following mail address:

`latex-bugs@rus.uni-stuttgart.de`

or a postal address that is to be announced.

III An Informal Review of TUG '93: July 26th–30th, Aston, Birmingham UK

Kees van der Laan
Dutch TEX Users Group

1 Highlights

- Announcement of LATEX2.9e; the *LATEX Companion* book will be available in the fall, which contains among other items, the NFSS2 documentation
- NTS: organizational and technical issues were surveyed
- Y&Y's scalable outline fonts
- Adobe's Acrobat, and PDF (portable document format), demoed by Doug Henderson (Blue Sky Research, and Adobe β-tester)
- Tutorials: What is LATEX?, Flavours of TEX, Virtual fonts, all for free
- Greenwade's CTAN: Comprehensive TEX Archive Network, released
- The *TUGly Telegraph* spread the news (a mini conference newspaper)

This report[1] contains the main issues as perceived by the author. The idea is to get the flavour and my view of the good items across, at the expense of completeness.

2 Introduction

The meeting of the organized TEX Users of the year was TUG '93 at Aston. It attracted some 165 participants, with a few from financially diasadvantaged countries, thanks to a bursary fund. Aston campus is pleasantly located near the centre of Birmingham, Brum for short. A rich variety of courses was offered before and after the conference. In general the program offered nice presentations, workshops, panels and the like, with the vendor booths fewer in number than usual, Blue Sky Research being sadly absent because of being informed too late. Despite this, to attend the conference was a real thrill. Much attention was paid to details; for example every badge contained a nice proverb, not forgetting the superb logo. In general all was done by creative and playful minds. Next to the lecture room there was a discussion lounge—with among other things some computers with e-mail and FTP facilities, such that participants could read their e-mail and exchange files—and a vendor booth, alias lounge, for display of materials brought along by TUG and the various LUGs.

3 Conference

For the most it was a one-stream schedule. Along with the program we obtained a copy of the pre-proceedings, which be published in *TUGboat* 14.3. Prior to the conference there were the free tutorials: 'What is LATEX all about?', 'Flavours of TEX', and 'Virtual fonts'. During the conference we had workshops about MakeIndex, BIBTEX, and virtual fonts. And there were ample course offerings before and after the conference.

In the next section attention is paid to the highlights neglecting the day-to-day sequence of events as well as the session titles.

3.1 Keynotes

The keynotes were: Jackowski and Ryćko—detailing what happens at the beginning and at the end of a paragraph; Christina Thiele—about the future of TEX and TUG; and Joachim Lammarsch—presenting his view on the historical and organisational issues related to NTS, the New Typesetting System. Later Phil Taylor—the technical director of the project—aptly complemented this with the technical aspects. A nice example of international cooperation.

[1] Cut down for inclusion in *Baskerville*; a fuller version will be published in MAPS, the journal of the Dutch TEX Users Group.

3.2 Presentations

- Michael Doob's paper about the practical use of virtual fonts was very illuminating. Related to this was the work of Alan Jeffrey for manipulating *vpl* files.
- It was rumoured that John Plaice's Ω-TEX will be a nice extension of TEX. It is difficult to overlook its virtues or its impact.
- Derick Wood's theoretical model of tables was very intriguing. The burning question in here is whether we can develop something like normal specification and transformation to the clearest formatted represention (semi)automatically.
- Daniel Taupin surprised us all with a new aspect of his TEX work: TEX and Metafont working together to produce maps. Later, and separately, he also reported about new developments in MusicTEX.
- Martin Bryan's Guide to DSSSL was a bit too theoretical. Upon request he assured us that this extension is there mainly to handle so-called active documents.
- Michel Lavaud preceded his AsTEX lecture by some analogies of the TEX language with natural language—both ambiguous and 'dangerous'—and with the FORTRAN language—both stable, but deficient. The latter has gained a firm place within the scientific work desk, despite its deficiences and other imperfections.
- Laurent Siebenmann in one of his papers proposed a new method for handling the spacing around in-line math.
- Roger Hunter discussed *Scientific Word* and emphasized that the future of TEX has all to do with better user interfaces. It is fun to remember all the various items people have prophesied to be essential for the future of TEX. At the user as well as the language level.
- Włodek Bzyl lectured about the use of literate programming tools to build upon other work: *TUGboat* style customization via change files and pretty printing of TEX code, along with providing an index. By this approach the styles of the GUST bulletin make use of the experience embodied in the *TUGboat* style files, with GUST's modifications easily added. Really a sensible approach and worthy of being studied and followed by other editors.
- Berthold Horn dwelled upon the paucity of math fonts ready for use with TEX, and a checklist to counteract this phenomenon. A very lucid and scholarly paper.
- George Greenwade made the CTAN (Comprehensive TEX Archive Network) go public. A quantum leap!
- At the macro writing level we had Jonathan Fine's "Finite State Automata in TEX" and my "Syntactic Sugar", and "Sorting within TEX". The syntactic sugar paper also touches upon the software engineering aspects of TEX macro writing.
- Very intriguing was Mary Dyson's paper about teaching typography, as a spin-off of the DIDOT project. It has all to do with getting your priorities right; don't sub-optimize! In essence the issues relevant to Electronic Publishing were presented in the right order.
- Richard Southall elaborated on his 'buses and weirdness' effects when using TEX, naively. I'm hoping that this paper will finally appear in print.

4 LUGs

Only CyrTUG and Ukraine TUG (officially to be founded in the autumn) talked in public about their history and revealed their plans. Both will have autumn meetings, open for the TEX community at large to attend.

5 Upcoming, or go where the action is

For next year the working idea is to have EuroTEX '94 organized by GUST in Poland. TUG '94 will be in Santa Barbara, TUG '95 in Florida, and TUG '96 in Europe again. And in the meantime the autumn is crowded with meetings: Ukraine TUG,[2] CyrTuG (early October), the Nordic and NTG's November meetings, next to DANTE's regular and cosy Stammtisch, ..., and GUST in the spring: watch out for the calendar in TUGboat.

6 Conclusion

Meeting people and stimulating each other—that is what conferences are for. This was a particularly good one.

Thanks to you all!

[2] At the moment of writing this report we received the sad news that their leader Yuri Melnichuk has passed away, due to a heart attack.

IV Book review — 'How to run a paper mill'

Allan Reese
University of Hull

'How to run a paper mill', **John Woodwark**, Information Geometers, Winchester (1992), ISBN 1-874728-00-3, xv + 111 pages.[3]

Not a manual on processing lumber into landfill. The sub-title is explanatory: *Writing technical papers and getting them published*. This slim volume will interest all members as it describes scientific progress in the modern context. Papers are currency; they are the tickets to attend TEX conferences; they earn kudos, preferment and promotion. The approach is realistic and entertainingly cynical — thought-provoking whether you have "played the system" and published many papers, or are about to embark on your first.

Woodwark considers the reasons and methods for research, writing and publishing — and all the permutations for ordering those three phrases! He gives pithy practical advice to the aspiring author; this covers the text, and the use of graphics, algebraic notations and modern conventions like pseudo-code.

Woodwark himself used TEX for his work, and gives credits on pages xiv, 77 and 99.

An unreserved recommendation to read.

[3] This review is reproduced from the TUG'93 *TUGly Telegraph*

V Book review – 'Handbook on Writing for the Mathematical Sciences'

Malcolm Clark
University of Warwick

SIAM, the Society for Industrial and Applied Mathematics recently published the *'Handbook on Writing for the Mathematical Sciences'* by ***Nicholas J Higham***.[4] The first thing which the reader notices is the delightful use of the very fine Computer Modern typeface. There can be no doubt that for mathematical compuscripts, Computer Modern has no rival. In fact, as the author notes, the book was typeset using LaTeX with the *book* document style and the *jeep* option. I think it looks pretty good.

It contains a lot of good advice for those who want to typeset maths. It firmly acknowledges a debt to a large number of other writers on mathematical writing, and more generally on a wide variety of others who have contributed to editing, writing, language usage and all the other things we tend to take for granted. Having said that, it is clear that Don Knuth is among those acknowledged.

Much of the book is directed at those mathematicians who would write for SIAM, and emphasizes their own publications, stressing their particular style (available from your friendly Aston Archive), but much else is applicable widely. In fact, I would tend to see the whole book as 'A handbook of writing' which just uses mathematics, and sciences, for illustration. There is so much good sense here that it would be a real pity if non-mathematicians ignored it because they thought it in some way irrelevant. So many of his examples are not only thoughtful, but also clear and apposite.

There is also much useful trivia. I was unaware that Euler had 'invented' the notation for e, although I did have a clue that Kronecker had introduced Kronecker's delta, δ_{ij}! There are lots of other little gems hidden away in the text.

But perhaps the main reason for recommending this book is the stress which Higham lays on the use of TeX and its tools. Chapter 10 'Computer aids for writing and research' is the second longest chapter in the book (after 'Writing a paper'), and discusses the use of TeX and its variants, as well as spelling checkers, citation services, and the Internet. It is an extremely useful resource just for this chapter. Obviously it cannot be an in-depth treatment, but it says enough to whet the appetite and send the curious in the right direction. Excellent! A warning however: Higham's world is the world of Unix and PostScript. In my view, the correct world to be in if you don't happen to have a Mac. He does point to the use of Ghostscript for printing and previewing PostScript files on non-PostScript output devices (available for Unix, MS DOS and Macintosh). With that one small proviso — recommended.

[4] This review is reproduced from the TUG'93 *TUGly Telegraph*

VI What should we teach TEX?

Jonathan Fine
Cambridge

> A 'virgin' TEX system that has no macros is like a newborn baby that has an immense amount to learn about the real world; but it is capable of learning fast.
> *The TEXbook* p.342

1 Introduction

I would like to use this article as an opportunity to explain to the UK TEX community what it is that am doing, and have been trying to do, for the last two years. This will not be easy – the subject matter can be difficult and unfamiliar, even to myself – and so the reader's indulgence is asked for.

I am working in what the cosmologists might call the first seconds of the universe. More exactly, I am writing macros that are to be loaded even before the like of `plain` or LATEX are to be loaded. Most users are concerned with later time, when TEX is generating a `dvi` file from the author's typescript.

2 Macro packages

Before explaining further, a few words about `plain` and LATEX. In the beginning was `plain.tex`, which consists of 1229 lines. LATEX 2.09 consists of 1318 lines in `lplain.tex`, the LATEX version of plain, together with 8565 lines in `latex.tex` and 995 lines in `lfonts.tex`. Although the new LATEX 3 is not yet available, two of its intended support macro files, `doc.doc` and `docstrip.doc`, weigh in at 4554 and 2918 lines respectively. (In fairness, it should be added that the relative quantity of documentation is greater in the later files).

A file such as `plain.tex` has two purposes. One is to place particular macros etc. into the memory of TEX– to teach it something. The other is for us to read, so that we might know what is being done. From the beginning there has been a tension between these two functions. In the TEXbook Knuth does *not* list `plain.tex` verbatim. On p.342 he explains:

> "Un-optimized versions of the macros are easier for humans to understand, so we shall deal with those; `plain.tex` contains equivalent constructions that work better on a machine."

and further explanation is given at the top of p.348.

This tension between human comprehension and machine efficiency has grown with time, and is commonly expressed by the complaint, that TEX is difficult to program, or inadequate as a programming language. (I personally like the TEX macro language, and believe its basic conception to be sound. However the complaints are real, for they have a basis in experience).

3 A new point of view

If TEX were a new microprocessor with its own instruction set, then specialist developers would produce language tools. These would compile higher level commands into TEX's instruction set. Such tools would relieve much of the tension and difficulty in programming TEX.

Two years ago I began such a process of development for TEX. If I had known then the work involved, I may not have started. It has now born its first fruit, the `\noname` package, which is a valuable tool for the author of TEX macros.

Briefly, `\noname` removes many of the irritations of TEX by implementing an extension of the TEX programming language. Files written in the `\noname` dialect of TEX can be loaded directly into TEX's memory, or compiled into a form where they can be loaded at high speed and with minimal overhead. The situation is similar to that of a language interpreter and compiler. The `\noname` package, and macros written using `\noname` are completely compatible with, and independent from, the standard packages such as `plain` and LATEX.

4 About noname

Here are some of the features of `\noname` which ease the programmer's task.

- It is easy to write long comments – every line not beginning with a space character or a hash # is ignored.
- Lines beginning with a # are *hash commands*, which control conditional compilation, much as in *C*.
- All white space in macro code is ignored. However, the escape character ~ will *always* produce an ordinary space character, even after a control word or a previous space producing ~.
- As well as letters and `@` the digits `0..9`, the underscore _ and the colon `:` can be used to form a control word. This does not conflict with the normal use of these characters.
- Ready access is provided to active characters, and other characters with unusual catcodes.
- Ready access is provided to control sequences which store numeric constants such as −1, 0, 16 and 255. One simply types `[-1]` in place of `\m@ne`, and `[16]` instead of `\sixt@@n`.
- Support is provided for the powerful control macros `\break`, `\continue`, `\return`, `\CASE` and `\FIND`.

This is only the beginning. Intended, planned, and under development enhancements include the following.

- Indexing and cross-reference utilities.
- Interactive tutorial material for teaching how macros and primitive commands work. This will use the next item.
- A *single-step debugger* which will execute TEX code one step at a time. For example, the expansion of an expandable token such as a macro is a single step, as is the execution of a primitive command. TEX acts on a stream of tokens. After each step the (changed) stream of tokens is displayed and the user regains control.
- A powerful *pretty printer* for typesetting source files written in the `\noname` dialect of TEX. This will require zero or minimal additional markup.
- Named parameters for macros. This means one can write

  ```
  \def\mymacro #\text ...
  ```

  instead of

  ```
  \def\mymacro #1 ...
  ```

  when coding a macro. This makes it easier for a human to read the macro.
- Support for modules, with private and public identifiers. This will remove much of the conflict that arises from different packages using the same identifier to hold different meanings.
- Support for various tricks to help conserve TEX's hash table (maximum number of control words), registers (`\count`, `\skip`, etc.) and other resources.

5 Conclusions

There is currently international discussion within the TEX community – under the title NTS (new typesetting system) – of what if anything should replace TEX the program. The `\noname` package demonstrates that much more can be done with TEX than is commonly realized. In particular, it removes some (not all) of the difficulties in programming TEX. It is important to understand what can and cannot be done with TEX as it is, and how or why not, if one is set on creating a successor.

I began by posing the question, what should we teach TEX? My response is this. First of all we should teach TEX how to read and understand macros written in a well-designed high(er) level extension or dialect of TEX. It will then be much easier to write the many tens of thousands of lines that make up LATEX 2.09 and 3, and countless other macro packages across the world.

If you are interested in `\noname` and would like a demonstration copy, please contact me. The price, for a site license, will be two or three figures. For approved public domain developers, the price will be zero.

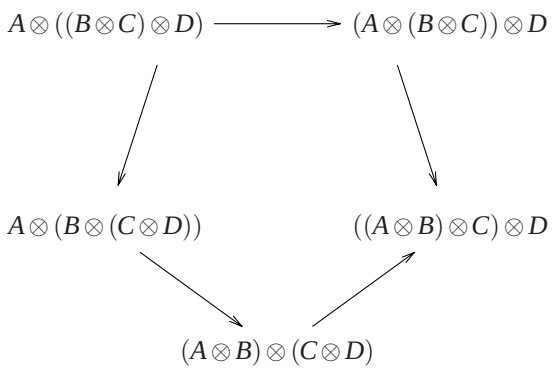


Figure 1. Coherence for monoidal categories

VII On specialist typesetting packages

David Murphy
University of Birmingham

Over the last two years I have been involved in the development of a commuting diagrams package for LaTeX. This article is based on my experience of that work. It is meant neither to be read as distilled wisdom, nor as a provocation, but rather as a basis for discussion of the kinds of packages users 'should' be provided with.

Introduction

Some of my work uses category theory: this is a branch of mathematics which makes extensive use of commuting diagrams, Figure 1 being a typical example.

There are various commuting diagrams packages available to TeX and LaTeX users; both Michael Barr (McGill Unversity) and Paul Taylor (Imperial College) distribute popular ones. However, all popular current packages use TeX's `line` founts, so it is impossible to achieve a diagram with pentagonal symmetry such as Figure 1. Furthermore, both Barr's and Taylor's packages share what is, I find, a common problem amongst TeX shareware, namely that easy things are easy to achieve, but difficult things, such as Figure 2, require an intimate knowledge of the package's workings. Indeed, I was spurred to work on my own diagram macros by my inability to understand Taylor's code or to obtain the diagram I wanted from his package.

I was very lucky in finding a solid base to build upon: Sean Bechhofer (Manchester University) had already implemented a rudimentary commuting diagrams package based on a `picture`-like environment which compiled arrows directly as PostScript `\specials`. I had merely to extend his work, rewrite the `\specials` to work with `dvips`, ensure that his environment interacted reasonably sensibly with the rest of LaTeX, and generally tidy up the code. The result is a package capable of producing diagrams such as Figure 2 of reasonable quality and complexity.

What's Wrong?

The package I produced, based on Bechhofer's work, has several manifest disadvantages:

- It works exclusively with PostScript printers and `dvips`. There are a substantial number of LaTeX users without access to these facilities.
- The syntax I have adopted, as the code which produces Figure 2 (shown in Figure 3) makes clear, is fairly ad hoc and ill-chosen.
- There is a considerable jump in complexity between simple rectangular diagrams and more complex ones. This is wholly due to the difficulty of doing standard trigonometry in TeX. If I could perform fairly simple numerical

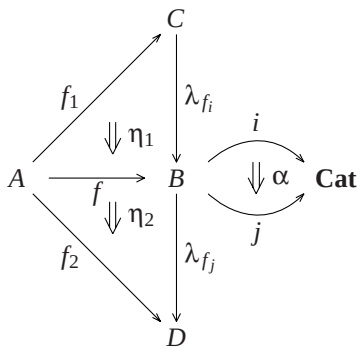


Figure 2. A simple two-categorical diagram

```
\begin{diagram*}(2,2)
\object(0,1){A}
\object(1,1){B}
\object(1,2){C}
\object(1,0){D}
\object(2,1){{\bf Cat}}

\curvearrow(1,1){\east}[i]{\po}
\curvearrow(1,1){\east}[j]{\st}

\twocell(1,1){\east}{\clock}[\alpha][\far]

\twocell(0,1){\eneast}{\clock}[\eta_1][\far]
\twocell(0,1){\eseast}{\clock}[\eta_2][\far]

\plarrow(0,1){\neast}{f_1}[\po]
\plarrow(0,1){\east}{f}[\st]
\plarrow(0,1){\seast}{f_2}[\st]
\plarrow(1,2){\south}[\lambda_{f_i}][\po]
\plarrow(1,1){\south}[\lambda_{f_j}][\po]
\end{diagram*}
```

Figure 3. The code for the two-categorical diagram shown in Figure 2

calculations without descending to the PostScript level, users would have to think rather less hard about the relative positions and dimensions of diagram objects. Some hand-tuning is always going to be necessary in a complex graphical environment, but more automation should be possible than I have provided.

My real concern in this article is how much do these problems matter? I do not doubt that the problems I have outlined above make my macros entirely unacceptable as a standard package; if something like `multicol` suffered from similar problems, no one would use it. However, there is a big difference between carefully-designed packages like `multicol`, which must work in a wide range of circumstances and with a wide range of users, and a specialist package like my own which is designed for a small community of relatively skilled users. For the moment I am willing and able to continue to support the package, but this may not always be the case. Is it reasonable to ask people to use software for which no support guarantee can be given?

My suspicion is that there are rather a lot of packages like mine: medium-sized, badly-written pieces of code with a small but basically satisfied user base. Is this a good thing? These users are not numerous enough to be able to command high-quality software to meet their needs: not enough people are interested in commuting diagrams for a diagrams package to be a sensible addition to standard LaTeX. (Opinions differ on this point: Barr is currently working on a package that may come as standard with LaTeX 3.) Must the category theorist, then, always be content with software such as mine? More generally, what is the best way for the specialist typographical needs of a small community to be met? I hope that these questions will find some answers in the current debate over the future of TeX.

VIII METAFONT for Beginners

Geoffrey Tobin
ecsgrt@luxor.latrobe.edu.au

Scope

This is not a tutorial on METAFONT.[5] It is an attempt to describe how some of the pitfalls in running the program may, hopefully, be avoided.

It is a common experience to have initial (and medial and final) difficulty with running METAFONT, and not all 'TEXnicians' are as familiar with METAFONT as they are with TEX. Still, nothing ventured, nothing gained. So let's be of good cheer, and get down to work.

1 What is METAFONT?

METAFONT is a program for making bitmap fonts for use by TEX, its viewers, printer drivers, and related programs. It interprets a drawing language with a syntax apparently derived in part from the Algol family of programming languages, of which C, C++, Pascal and Modula-2 are members.

The input can be interactive, or from a source file. METAFONT source files are usually suffixed '`.mf`'.

METAFONT sources can utilize scaling, rotation, reflection, skewing and shifting, and other complex transformations in obvious and intuitive ways. But that is another story, told (in part) by *The METAFONTbook*.

METAFONT's bitmap output is a GF (*generic font*) file. This may be compressed to an equivalent PK (*packed*) font by the auxiliary program *GFtoPK*.

Why doesn't METAFONT output PK fonts directly? Firstly, Tomas ROKICKI had not invented PK at the time Donald Knuth was writing METAFONT. Secondly, to change METAFONT now would be too big a change in Knuth's opinion. (Knuth is a very conservative programmer; this fact is a two-sided coin.)

GF and PK files are suffixed '`.*gf`' and '`.*pk`' respectively, where, in a typical UNIX installation, the '`*`' stands for the font resolution. (Resolution will be explained below.) MS-DOS truncates file name suffixes to three characters, so a font suffix '`.1200gf`' becomes '`.120`' — beware of this!

A bitmap is all that's needed for large-scale *proofs*, as produced by the *GFtoDVI* utility, but for TEX to typeset a font it needs a TFM (*TEX Font Metric*) file to describe the dimensions, ligatures and kerns of the font. METAFONT can be told to make a TFM file, by making the internal variable '`fontmaking`' positive. Most output device modes (see below) do this.

Remember that TEX reads only the TFM files. The *glyphs*, or forms of the characters, as stored in GF or PK font files, do not enter the picture (I mean, are not read) until the DVI drivers are run.

TEX can scale TFM files. Unfortunately, bitmaps such as GF and PK are not scalable. However, METAFONT files can be compiled into fonts of arbitrary scale by METAFONT, even by non-programmers.

Incidentally, properly constructed TFM files are device-independent, so running METAFONT with different modes normally produces the identical TFM. Dimensions in TFM files are specified to METAFONT in device independent 'sharped' dimensions (commonly suffixed by #), where a value of 1 corresponds to the dimension of `1pt` (typographical point). Most of METAFONT's calculations are done with (resolution and device dependent) pixels as units. Care must be taken by font designers to *always* calculate unsharped dimensions from sharped ones, and never the other way round, so as to keep roundoff errors or similar effects from influencing the TFM files to depend on resolution or device. Although type quality will be influenced only in minuscule ways, this is one of the more common reasons for checksum errors reported by printer drivers. Note that the only way to be sure that a TFM file is device-independent is to create the font in different modes and compare the resulting TFM's, perhaps using *tftopl*.

More detailed descriptions of TFM and GF files, and of *proof* mode, are found in Appendices F, G, and H, respectively of *The METAFONTbook*.

[5] This article is a trimmed version of the tutorial file made available on TEX archives; a current full version can be obtained from the *documentation* directory on any CTAN archive (see elsewhere in this issue of *Baskerville* for more details of CTAN).

2 Getting METAFONT's Attention

*2.1 Typing at METAFONT's '**' prompt*

If you type the name of the METAFONT program alone on the command line:

```
mf
```

then `mf` displays a '`**`' prompt, which 'is METAFONT's way of asking you for an input file name'. (See *The METAFONTbook*, Chapter 5: 'Running METAFONT'.) Thus, to process a METAFONT file named `fred.mf`, you may type:

```
fred
```

A backslash ('`\`') can also be typed here. This causes all subsequent commands at the prompt line to be interpreted as in a METAFONT file. (Concerning the backslash, see *The METAFONTbook*, Chapter 20: 'More About Macros', pages 179 and 180 in the 1986 edition.) Thus we can respond to the `**` prompt with:

```
\ input fred
```

or even:

```
\ ; input fred
```

The backslash is useful because certain commands are often executed before a METAFONT file is input. In particular, quality printing requires the METAFONT command `mode`, and output magnification employs the `mag` command. For example:

```
\mode=localfont; mag=magstep(1); input fred
```

To read MS-DOS pathnames at the `**` prompt, this satisfies METAFONT:

```
\ input \seldom\fred.mf
```

as does:

```
d:\seldom\fred.mf
```

2.2 Typing on the Command Line

Most METAFONT implementations permit you to type METAFONT commands on the command line, instead of at the `**` prompt. (Rather, it is automatically passed to that prompt.)

On MS-DOS, type commands as at the `**` prompt:

```
mf \mode=localfont; input myfont10
```

On UNIX, command shells typically interpret semicolons, backslashes and parentheses specially, unless they are 'quoted'. So, when typing those characters as part of instructions to METAFONT on the UNIX command line, it is wise to accustom yourself to protecting them with *apostrophes*:

```
mf '\mode=localfont; input myfont10'
```

If `localfont` makes fonts for a 300 dots per inch (dpi) device, this should produce a TFM file, '`myfont10.tfm`', and a 300 dpi GF font file, '`myfont10.300gf`'. Almost all of the following will presume a 300 dpi device, and other resolution devices will have appropriately different font file names.

These command lines are a bit long, very often used, and rather intolerant of mistakes so you might type the repetitive parts into a UNIX shell script or an MS-DOS batch file, as appropriate.

In UNIX, the `**` prompt has the advantage that those pesky apostrophes are not needed. (Indeed, those apostrophes are always wrong at the `**` prompt — METAFONT doesn't understand them. It would not understand them on the command line either—it is just that the shell does not hand them over to METAFONT.) However, for shell scripts (and for batch files in MS-DOS), the command line is a boon.

2.3 'Please type another input file name: '

When METAFONT cannot find the main source file, it doesn't quit. For example, when I typed `mf fred`, METAFONT said:

```
This is METAFONT ...
**fred
! I can't find file 'fred.mf'.
<*> fred

Please type another input file name:
```

The usual program interrupts (eg, Control-C) don't work here, and the '`Please type ...`' prompt does not understand METAFONT commands: it will read only the first word, and insist on interpreting this as a file name.

Beginners faced with this often wonder how to avoid an endless loop or a reboot, or try to think of a METAFONT file that they do have in METAFONT's path. In the latter case, the canonical name to use is '`null`', standing for the file '`null.mf`'.

In fact, the solution is much easier: on the systems that I have tried, a simple end of file marker ('control-Z' in MS-DOS, 'control-D' in UNIX) stops METAFONT in its tracks:

```
! Emergency stop.
<*> fred

End of file on the terminal!
```

3 Base files

In versions 2.7 and 2.71, the METAFONT language contains 224 (previous versions had fewer) primitives, which are the commands preceded by an asterisk in the Index (Appendix I) to *The METAFONTbook*. From these we can build more complex operations, using macros. In METAFONT macros have some of the desirable characteristics of functions in other languages. Collections of macros can be stored in METAFONT source files.

Base files are *precompiled internal tables* that METAFONT loads faster than it loads the original METAFONT source files. Thus, they are closely analogous to TEX's *format* files.

3.1 The `plain` *base*

The `plain` base provides the commands that *The METAFONTbook* describes. (See Appendix B of *The METAFONTbook*, if you have it around — maybe a library has it — I'm learning from a copy borrowed from the local university's library.)

When it starts, METAFONT automatically loads[6] the `plain` base. This is usually called `plain.base`, or sometimes only `mf.base`, although for those systems concerned (such as UNIX), both file names should really be present.

EmTEX for MS-DOS calls the plain base `plain.bas`, due to filename truncation.

3.2 Loading a Different Base

Suppose that you have a base named `joe.base`. Typing

```
mf &joe
```

or (on unix, where we must either quote or escape the ampersand)

```
mf \&joe
```

or responding

```
&joe
```

to the `**` prompt, omits loading `plain` base, and loads the `joe` base instead. Typically, however, the `joe.mf` file which originally produced the `joe` base will have included `plain.mf`, because working without the `plain` base macros would be too cumbersome. (Refer to *The METAFONTbook* (1986), Chapter 5: 'Running METAFONT', page 35, 'dangerous bend' number two.)

The '`cm`' base, for making the Computer Modern fonts, can be loaded in that way:

```
mf &cm
```

Remember to quote the ampersand under UNIX!

3.3 The Linkage Trick

On systems such as UNIX where programs can read their own command line name, and where files may be linked to two or more names, then programs can modify their behavior according to the name by which they are called. Many UNIX TEX and METAFONT installations exploit this in order to load different *format* and *base* files, one for each of the various names to which TEX and METAFONT are linked. Such installations can often be recognized by the presence of the executable '`virmf`' in one of the directories in the `PATH`.

For example, if a base file called '`third.base`' resides where METAFONT can find it, then `virmf` can be linked to `third`. In UNIX, a *hard link* is formed by

[6] There are releases of METAFONT that contain the `plain` base, and so don't have to load it. However, on most computers, including personal computers, reading bases is so fast that such a *preloaded* base is unnecessary.

```
ln virmf third
```

On systems supporting *symbolic links*, you should make all of these links symbolic, rather than hard, or else you will have to redo them every time you install a new copy of `virmf`; see below. In UNIX, this is done by

```
ln -s virmf third
```

Normally one wants `mf` to load the `plain` base, so in such installations one links `plain.base` to `mf.base`:

```
ln plain.base mf.base
```

Again, you'd best make that link symbolic. This comment applies for the rest of this section as well.

As another example, take the '`cm`' base. In *web2c*:

```
ln virmf cmmf
ln cm.base cmmf.base
```

so that '`cmmf`' automatically loads '`cm.base`'.

This applies equally to TEX, which is why `tex` and `latex` are then links to `virtex`, `tex.fmt` is a link to `plain.fmt`, and `latex.fmt` is a link to lplain.fmt:

```
ln virtex tex
ln plain.fmt tex.fmt

ln virtex latex
ln lplain.fmt latex.fmt
```

Karl Berry's *web2c* distribution for UNIX uses this '*linkage trick*'.

If you used symbolic links, you can laugh off the following

WARNING: This linkage is convenient, but watch out during updates! If `mf.base` is a *hard link* to `plain.base`, then replacing `plain.base` with its new version severs the link: `mf` will still load `mf.base`, but it will be the old version! The proper procedure is to remove the old `mf.base`, and relink. On UNIX:

```
rm mf.base
ln plain.base mf.base
```

On most UNIX systems, `ln -f` will automatically remove the second file (if present) — in this case, mf.base — before linking.

Alternatively, *web2c* will update '`plain.base`' (and '`plain.fmt`', and so on) for you, if you tell *web2c*'s `Makefile` to

```
make install
```

Symbolic links, on systems that have them, are probably the best method of handling updates, at least when doing them manually. (Consult your system administrator for details.)

3.4 Making a Base; the Local Modes file

The `plain` base is made from a METAFONT file named `plain.mf` and, commonly, from some other file, often called `local.mf` or `modes.mf`.

The `local`/`modes` file lists printers (and monitors), giving each output device a font-making *mode*, containing a description of some refinements that must be made in order to produce good-looking output. For instance, how to make the characters just dark enough, and how to make diagonal lines come out sharply.

If you want to make a base, you need a variant of the METAFONT program called '`inimf`'. (See *The METAFONTbook*, p 279.) For example, `plain.base` can be made in UNIX by typing:

```
inimf 'plain; input local; dump'
```

If using the emTEX version of METAFONT for a PC, type:

```
mf/i plain; input local; dump
```

4 Fonts

4.1 Proof Mode

The purpose of METAFONT is to make fonts. For æsthetically pleasing PK bitmaps, the correct device mode must be selected.

An obstacle to beware of is that `plain` METAFONT uses *proof* mode by default. (*The METAFONTbook*, page 270, defines this mode.) That means writing unmagnified font files with a resolution of 2601.72 dots per inch (dpi); that's 36 pixels per point. (One point is 1/72.27 of an inch.) Proof mode does **not** produce a TFM file.

What good is proof mode, and why is it the default? *Proofs* are blown up copies of characters used by font designers to judge whether they like the results of their work. Naturally, proofs come first, and normal sized character production later — if you're a font designer.

So there are two clues that proof mode is on: font files with extensions like '`.2602gf`' (or on MS-DOS, '`.260`'), and the 'failure' to produce any TFM file.

On some systems, such as X11, a third clue is that the proof font may be drawn on the screen — it is so large, you can't miss it!

4.2 Localfont Mode

When using a stable font, or when testing the output of a new font, we *don't* want proof mode, we want our local output device's mode. Usually, METAFONT is installed with a '`localfont`' assigned in the `local/modes` file. On our department's Sun Network, we have assigned

```
localfont:=CanonCX
```

We use Karl Berry's '`modes.mf`'[7], which contains modes for many, many devices. We chose the `CanonCX` mode because '`modes.mf`' recommends it for Apple Laserwriters and HP Laserjet II printers, which we use.

To process a METAFONT source file named '`myfont10.mf`' for the most usual local device, specify the local mode to `mf` before inputting the font name:

```
\mode=localfont; input myfont10
```

This should produce a GF font file, '`myfont10.300gf`' ('`myfont10.300`' in MS-DOS), and a TFM file, '`myfont10.tfm`'.

4.3 Font Naming

By the way, if you modify an existing, say a Computer Modern (cm), font, you must give it a new name. This is an honest practice, and will avoid confusion.

4.4 Magnification (and Resolution)

Now suppose that you want `myfont10` to be magnified, say to magstep 1 (magnified by 1.2), for a 'jumbo' printer. Assuming that the `local/modes` file has a mode for the jumbo printer, you may then run METAFONT with the following three commands:

```
\mode=jumbo; mag=magstep(1); input myfont10
```

to produce '`myfile10.tfm`' (again!) and a GF font, '`myfile10.360gf`'. On MS-DOS, the file names will be truncated; for example, '`myfile10.360`'.

The '`360`' is '300 * 1.2', indicating the magnification. A 360 dpi font can be used either as a magnification 1.2 font on a 300 dpi printer or as a normal sized font on a 360 dpi printer.

Note, however, that the METAFONT language includes special hints for each output device which clue METAFONT as to the reactions of the output device to pixel-sized minuscule changes.

So for highest quality, you would not even want to mix the fonts for two 300 dpi printers, unless they share the same mode and most probably the same print engine.

4.5 GFtoPK

TEX uses only the TFM file, which METAFONT will produce if it is in a font-making mode. (*The METAFONTbook*, Appendix F.) Most DVI drivers read the PK font format, but METAFONT makes a GF (Generic Font) file. So we need also to apply the *GFtoPK* utility:

```
gftopk myfile10.300gf
```

to produce the wanted '`myfile.300pk`' (or, on MS-DOS, '`myfile.pk`') PK font.

4.6 Storing the Fonts

Now we have the fonts, where do we store them? TEX, METAFONT and the various driver programs are compiled with default locations written in. These can be overridden by certain environment variables. The names of these variables differ between systems, but on UNIX they might, for example, be 'TEXFONTS' for the TFM files, and either 'PKFONTS' or 'TEXPKS' (or both of those) — before searching 'TEXFONTS' — for PK fonts. You can find out what environment variables you now have by typing '`set`' in MS-DOS and '`env`' in the Bourne shell, sh, in UNIX. In the UNIX C shell, `csh`, type '`setenv`'.

If you want TEX and METAFONT to find files in the current directory (as you almost certainly do!), then one

[7] Available at `ftp.cs.umb.edu` in the `pub/tex` directory.

way is to put '`.`' into their search paths. (Both UNIX and MS-DOS accept the `.` notation for the current directory.) Default search paths are compiled into TEX and METAFONT, but users can customise the environment variables that the programs read, to override the defaults.

METAFONT as well as the DVI drivers, can also be given full path specifications for input files.

On the other hand, you may be content with your new font, and you may have write access to the place where most of the fonts are stored. In that case, copy your font to there. There will be a place for the TFM files, and another for the PK files. It is up to you or your local system administrator(s) to know where these directories are, because their names are very locale dependent.

5 Some Limitations of METAFONT

METAFONT contains some builtin limitations, some obvious, others less so.

Parts of the following list are most useful to budding programmers, though casual users may wish to read it to learn whether an error message produced by somebody else's METAFONT file is very serious or not.

1. All valid numbers are strictly less than 4096.
2. *The METAFONTbook*, in 'Appendix F: Font Metric Information', warns of one limitation that I've met when processing some fonts.
   'At most 15 different nonzero heights, 15 different nonzero depths, and 63 different nonzero italic corrections[8] may appear in a single font. if these limits are exceeded, METAFONT will change one or more values, by as little as possible, until the restriction holds. A warning message is issued if such changes are necessary; for example
   `(some charht values had to be adjusted by as much as 0.12pt)`
   means that you had too many different nonzero heights, but METAFONT found a way to reduce the number to at most 15 by changing some of them; none of them had to be changed by more than 0.12 points. No warning is actually given unless the maximum amount of perturbation exceeds $\frac{1}{16}$ pt.'
   Every correct implementation of METAFONT will adjust character box dimensions by the same amount, giving the same TFM files, so we ignore small perturbations in other people's fonts. When designing your own fonts, however, I think it is courteous to keep within the limits, so as not to worry inexperienced users.
3. In the `addto` picture command, `withweight` only accepts values that round to `-3`, `-2`, `-1`, `+1`, `+2`, or `+3`. To obtain other pixel weights, you can apply further `addto` commands.
4. The memory size of the version of METAFONT you use is an evident, implementation dependent restriction, but it may be, as in TEX, that memory is not enough simply because, if you'll pardon my saying so, some of your coding may be seriously inefficient or logically invalid.

6 What Went Wrong?

The complexity of wrong things far exceeds that of things intended. *The METAFONTbook*, chapter 5, 'Running METAFONT', contains instructive examples, and supposedly 'dangerous', but actually basic and useful, notes.

In that chapter, and in chapter 27, 'Recovery from Errors', Knuth discusses the diagnosis of METAFONT's error messages. I find this perhaps the hardest part of the book — if not of using METAFONT.

Incidentally, METAFONT's error messages are contained in an ASCII file called '`mf.pool`'. Reading the `pool` file can be entertaining.

6.1 Big fonts, but Unwanted

Recently, I found myself accidentally producing fonts with extensions like '`3122gf`'. How?

METAFONT will take **anything** *as an excuse to revert to* **proof mode**.

The '`3122`' is a magstep 1 proof mode. It is

```
(1.2)^1 * 2601.72  = 3122.164  dots per inch.
```

My intention was for METAFONT on a PC to use an HP Laserjet mode in place of proof mode. However, METAFONT's command line resembles the law: *every stroke of the pen is significant*. What I had forgotten was that on my setup, '`localfont`' must be explicitly requested.

EmTEX's METAFONT, with `plain.mf`, defaults to proof mode. However, I usually want a local printer's font-making mode. So to process `pics.mf` correctly, I need to say:

```
mf '\mode=localfont; input pics'
```

[8] Respectively, `charht`, `chardp` and `charic` values.

6.2 Consequences of Some Typing Errors on METAFONT's command line

Small typing errors are so common, and yet undocumented (why are common mistakes not documented?), that I thought I'd list several that have tripped me up on innumerable occasions. After all, why reinvent the car crash?

Consider a source file '`pics.mf`' that contains '`mag=1200/1000;`', so it is automatically scaled by 1.2 (ie, by magstep 1). If the target printer has 300 dpi, then a 360 dpi GF font is wanted.

Here is the gist of what happens for various typing errors, when using emTEX's '`mf186`' on a 286 PC to process '`pics.mf`'.

1. `mf186` ⟹ METAFONT will keep prompting for arguments:

```
**
```

   We can type the contents of the command line here; for example, I can now type '`pics`'. In fact, even if you use the command line, the `.log` ('transcript') file shows METAFONT echoing its interpretation of the command line to a ** prompt.

2. `mf186 pics` ⟹ proof mode:

```
! Value is too large (5184)
```

   No TFM is produced, and the GF file has resolution 3122 dpi. (3121.72 dpi, to be precise.)

3. `mf186 mode=localfont; input pics` ⟹ misinterpretation:

```
! I can't find file 'modes=localfont.mf'.
```

   So, '`modes`' needs that backslash, otherwise METAFONT thinks it is the start of a source font's filename. Backslash ('\') and ampersand ('&') are escapes from this standard interpretation by METAFONT of the first argument. (Ampersand is in fact only a temporary escape, as METAFONT resumes the `mf` filename prompting attitude as soon as a base is read.)

4. `mf186 \mode=localfont input pics` ⟹ weird effect:

```
>> unknown string mode_name1.2
! Not a string
<to be read again>
                   ;
mode_setup-> ...ode)else:mode_name[mode]fi;
l.6 mode_setup
                  ;
```

   Wow! What a difference a semicolon can make!

5. `mf186 \mode=localfont pics` ⟹ almost nothing happens:

```
** \mode=localfont pics

*
```

   There's the echo I mentioned. From the lack of activity, `pics` evidently needs to be '`input`'.

6. `mf186 \mode=localfont; pics` ⟹
   Same as 5.
   So, yes, when the mode is specified, we need '`input`' before '`pics`'.

7. `mf186 &plain \mode=localfont; input pics` ⟹
   Works.
   Just as without the '`&plain`', it writes a GF file, '`pics.360gf`', which is correct. (MS-DOS truncates the name to '`pics.360`'.) So, redundancy seems okay. Does it waste time, though?

6.3 Finding the Fonts

Finding the fonts (`*.mf`, `*.tfm`, `*.gf`, and `*.pk`) trips up TEX, METAFONT, *GFtoPK* and the output drivers continually. '`pics.tfm`' needs to be put where TEX will look for TFMs, so I needed to ensure that '`.`' was in the appropriate path environment variable. Similarly for the METAFONT, GF and PK font files.

Environment variables can be tricky. For instance, emTEX's font-making automation program '*MFjob*' cannot make fonts in the current directory unless both '`.`' and '`..`' are added to `MFINPUT`. This was not documented.

Also, some popular TEX output drivers, such as the emTEX drivers on MS-DOS and OS/2, and Tomas Rokicki's '`dvips`' which has been ported to many systems, make missing fonts automatically — provided that they can find the necessary METAFONT source files. Again, making fonts in the current directory can require some tweaking.

6.4 Strange Paths

METAFONT satisfactorily fills simple closed curves, like '`O`' and '`D`', but filling a figure eight, '`8`', causes a complaint:

```
Strange path (turning number is zero)
```

because METAFONT's rules for distinguishing inside from outside might or might not give what you want for an '`8`', as there is more than one conceivable answer. You can use the 'positive turning rule' for all cases, and also turn off complaints, by setting

```
turningcheck := 0;
```

Chapter 13: 'Drawing, Filling, and Erasing', and Chapter 27: 'Recovery from Errors', discuss `strange paths` in greater depth.

Sometimes, when making a perfectly valid font, but in *low* resolutions, as for previewers (eg, VGA has 96 dpi), one may get flak about a '`Strange path`' or '`Not a cycle`' or something similar. Don't be alarmed. Fonts for previewing will still be OK even if not perfect.

Consequently, it is an idea to make low resolution fonts in METAFONT's `nonstopmode`.

Examples of fonts that give messages of this nature are the pleasant Pandora, and — from memory — the commendable Ralf Smith's Formal Script (`rsfs`). Everything is fine at higher resolutions.

Mind you, some fonts provoke sporadic (that is, design size dependent) strange path messages at 300 dpi (phototypesetter users would consider that low resolution), yet the printed appearance showed no visible defect.

Why do strange paths occur? One cause is rounding error on relatively coarse grids.

To summarize, if your viewed or printed bitmaps are fine, then you are OK.

7 METAFONT Mail List

Since 10 December 1992, there has been an e-mail discussion list for METAFONT, created:

1. as a means of communication between hooked METAFONTers;
2. as a way to bring the "rest of us" closer to them;
3. as a means to get quick and efficient answers to questions such as:
   - why do I always get a ".2602gf" file?
   - what is a "strange path", and what can I do to avoid it?
   - is there a way to go from METAFONT to PostScript and vice-versa?
   - where can I find a Stempel Garamond font written in METAFONT?
   - what is metaness?
4. and finally, as a first step to encourage people to undertake METAFONTing, and start a new post-Computer Modern era of METAFONT!

To subscribe to this list, send the following two lines to `listserv@ens.fr` on the Internet:

```
SUBSCRIBE METAFONT <Your name>
SET METAFONT MAIL ACK
```

The address of the list is `metafont@ens.fr` (at the notorious Ecole Normale Superieure de Paris). Owner of the list is Jacques Beigbeder (`beig@ens.fr`), coordinator is Yannis Haralambous (`yannis@gat.citilille.fr`). Language of the list is English; intelligent mottos are encouraged.

8 Conclusion

METAFONT, like TEX and many another 'portable' program of any complexity, merits the warning: '*Watch out for the first step*'.

I hope that a document like this may help to prevent domestic accidents involving METAFONT, and so contribute to making the task of using and designing meta-fonts an enjoyable one. My brief experience with METAFONT suggests that it can be so.

All the Best!

IX Typesetting paragraphs of a specified shape

Donald Arseneau
asnd@reg.triumf.ca

(*Editor's note*: This description of an unusual macro file, `shapepar.sty`, is taken from the documentation; the full style file can be found in CTAN archives.)

`\shapepar` is a macro to typeset paragraphs of a specified shape. The total size is adjusted automatically so that the entire shape is filled with text. This is distinct from the normal `\parshape` command which specifies a shape *and* a size, which may be only partially filled, or over-filled, from top to bottom. In a `\shapepar` there can be no displayed math, and no `\vadjust` material, (including `\vspace`). `\Shapepar` (capital S) is just like `\shapepar` except the paragraph is boxed so it cannot be split over two pages. Shaping paragraphs this way is a slow process, so this style is mainly intended for cards, invitations etc., not for whole books! Although short paragraphs process much faster, only long paragraphs accurately fill complex shapes.

These macros work for both LaTeX and plain TeX. For LaTeX, specify `\documentstyle[...shapepar...]`, or for either, `\input shapepar.sty`.

The command `\shapepar` should be used at the beginning of a paragraph, and it applies to the entire paragraph. There is one parameter: a description of the shape, <*shape_spec*>.

```
\shapepar {<shape_spec>} Text of para...
```

The syntax rules for <*shape_spec*> are very specific, and must be followed closely. (In these rules, `{ }` mean explicit braces, `[ ]` denote optional parts, `< >` surround a keyword that is defined (perhaps loosely), and `|` means "or"; do not type `[ ] < >` or `|`, only `{}`.)

```
<shape_spec}> = {<h_center>} <lines>
<lines> = <line_spec> [\\<lines>]
```

That is, the shape is specified as a single number in braces, followed by the specifications for the lines, with the lines separated by `\\`. The final paragraph will have its <*h_center*> position centered on the page. <*h_center*> is a number (like 10.5) of arbitrary units; whatever units are used for lengths and positions in the <*lines*>, they just need to be consistent.

The lines in the spec are not lines of text; nor are they the lines that you would use to draw the shape itself. They are horizontal scans across the shape at irregular intervals. Curved shapes need many scan lines for accurate rendering while simple shapes need few. Draw a shape on paper, then draw a series of horizontal lines across the shape, including lines that just touch the top and the bottom of the figure. Each line crosses over pieces of the figure in some region. These intersections of line and figure define a <*line_spec*>.

```
<line_spec> = {<v_pos>} <segment> [ other <segment>s ]
```

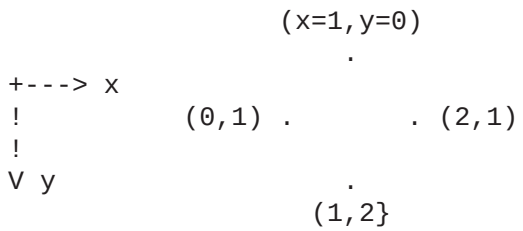
The <*v_pos*> is the vertical position of the line. Each <*line_spec*> must have a position greater than or equal to that of the previous line, and with all <*v_pos*> > -1000. Position is measured from top to bottom, and always moving down. Each <*segment*> represents a region where text will go in the final paragraph; it is the segment of the horizontal scan line that overlaps the body of the figure. There are five types of segment:

```
<segment> = t{pos}{len} | b{pos} | e{pos} | s | j
```

`b{pos}`	begin text at a point at horizontal position pos
`e{pos}`	end text at a point at horizontal position pos
`t{pos}{len}`	make a block of text at position pos with length len
`s`	split text (begin whitespace)
`j`	join two text blocks (end a gap)

The most common type of segment is *t* (text). The other types are degenerate in that they are single points rather than finite segments. Types *s* and *j* have no explicit position, but they must appear between text segments, and those texts should abut; e.g., `t{3}{2}st{5}{4}` (text from 3 to 5 and text from 5 to 9).

Let's jump right into a simple example, and the meanings will be clearer. A "diamond" shape can have the four vertices:



This shape can be exactly specified by just three scan lines passing through the vertices. The specification is:

`{1}%`	*h_center*: $x = 1$
`{0}b{1}\\%`	text block begins at point y=0, x=1
`{1}t{0}{2}\\%`	this scan (at y=1) crosses text (len=2) starting at x=0
`{2}e{1}`	text block ends at point y=2, x=1

Other specification lines, like `{1.5}t{0.5}{1}\\%` could be inserted, but would make no difference–the shape is interpolated linearly between scan lines.

Every block of text must start with a *b* specifier and end with an *e* spec. on some line below. Every segment specified by *t* must have a length greater than zero. If two blocks of text merge to form one (like at the top of a heart shape) there should be a *j* spec at the point of junction. If one block bifurcates (like at the top of a hole in a doughnut) there should be an *s* spec.

Thus, the first line for any valid shape description must consist of only *b* segment descriptors; the last line can only have *e* type descriptors. Although the definition of the units is arbitrary, the numbers should range in magnitude from ~.1 – 100 to avoid numeric overflows and underflows.

If there are errors in the format of the specification, `\shapepar` might complain with the error message “Shaped Paragraph Error: Error in specification. Check carefully!” At this point you may as well type `x` or `e`, as there is very little chance that TEX will continue successfully. You might also get one of TEX’s regular error messages, like “Illegal unit of measure (pt inserted).” or “ Missing number, treated as zero.” or you might get no error message at all, just ridiculous formatting. Check shape syntax carefully against the rules and the examples before running them through TEX.

What to do if the figure does not start at a point–if it has a flat top? It can start at a single point, but have the next scan line at the same vertical position! A square paragraph is specified by:

`{1} %`	centerline at x=1
`%`	(x=1 is horizontally centered on page)
`{0}b{0}\\%`	begin at (0,0)
`{0}t{0}{2}\\%`	text at y=0, width=2
`{2}t{0}{2}\\%`	text at y=2, width=2
`{2}e{1}%`	end at (1,2)

Both `\diamondpar` and `\squarepar` are defined as paragraphs with these shapes.

Sit
at word
processor for
two hours com-
posing title for new
book. Head for tea room at
11 0'clock where six colleagues are
sitting round walls in silence. Reminds
me of mental hospital day room but no strait
jackets, except perhaps intellectual. Make coffee with
back to them in order to surreptitiously use someone else's
milk from fridge. Wonder why no one ever asks
me what I have been doing in the department
for the last three years. Listen to sudden
burst of animated conversation about
new Mac software, and realise
that my social isolation
is due to the fact
that I have
an Am-
strad.

Now let's get more ambitious. A heart shape must have two simultaneous beginnings, a short stretch of separate text, ending with a join, thereafter there is just one stretch of text leading to the final bottom point. This shape has many scan lines so that the smooth flowing curves are preserved.

```
\def\heartshape{%
{20}{0}b{13.32}b{26.68}%
\\{.14}t{10.12}{4.42}t{25.46}{4.42}%
\\{.7}t{9.14}{7.16}t{23.7}{7.16}%
\\{1.4}t{8.4}{9.02}t{22.58}{9.02}%
\\{2.1}t{7.82}{10.42}t{21.76}{10.42}%
\\{2.8}t{7.36}{11.58}t{21.06}{11.58}%
\\{3.5}t{6.98}{12.56}t{20.46}{12.56}%
\\{4.2}t{6.68}{13.32}jt{20}{13.32}%
\\{4.9}t{6.48}{27.04}%
\\{5.6}t{6.34}{27.32}%
\\{6.3}t{6.28}{27.44}%
\\{7}t{6.26}{27.48}%
\\{7.7}t{6.27}{27.46}%
\\{8.4}t{6.32}{27.36}%
\\{9.1}t{6.4}{27.2}%
\\{9.8}t{6.52}{26.96}%
\\{10.5}t{6.68}{26.64}%
\\{11.9}t{7.12}{25.76}%
\\{13.3}t{7.72}{24.56}%
\\{14.7}t{8.51}{22.98}%
\\{16.1}t{9.5}{21}%
\\{17.5}t{10.69}{18.62}%
\\{18.9}t{12.08}{15.84}%
\\{20.3}t{13.7}{12.6}%
\\{21.7}t{15.62}{8.76}%
\\{22.4}t{16.7}{6.6}%
\\{23.1}t{17.87}{4.26}%
\\{24.6}e{20}%
}
```

Sit at word processor for
two hours compos- ing title for new
book. Head for tea room at 11 0'clock where six
colleagues are sitting round walls in silence. Re-
minds me of mental hospital day room but no strait
jackets, except perhaps intellectual. Make coffee
with back to them in order to surreptitiously use
someone else's milk from fridge. Wonder why no
one ever asks me what I have been doing in the
department for the last three years. Listen to
sudden burst of animated conversation
about new Mac software, and re-
alise that my social isolation
is due to the fact that
I have an Am-
strad.

Look at `\heartshape` and find the two *b* specifiers at the beginning; find the *j* a few lines below. Notice that above the *j* there are two segments per line, but only one below it; the text to the left and right of the join meet at the join point: 20. I drew this heart freehand, and measured lengths from the sketch, so you should be able to do better!

Text can have holes. For example, a doughnut-shape would have a *b* on the first line, followed by some lines with a single *t*, then a line with *t s t* at the start of the hole. The hole is represented by lines with two *t* specs–the gap between them is the hole. A line with *t j t* ends the hole. There are more lines with single *t*, and then an *e* line to end with. Our final example is a nut. Not a doughnut, but a hex-nut (for a machine screw) — a regular hexagon with a circular hole in the center. The hexagon is flat on top and bottom so the specification begins and ends like the square shape. The circle is rendered as a 24-gon, beginning with a split (*s*) of the surrounding text and ending with a join (*j*). If the spacing of the scan lines looks odd, it is because the hexagon alone would need few scans, but the circle needs many; the points on the circle are at 15 degree intervals.

```
\def\nutshape{%
{0}%
{0}b{0}\\%
{0}t{-12.5}{25}\\%
{11.65}t{-19.23}{19.23}st{0}{19.23}\\%
{11.99}t{-19.42}{16.835}t{2.59}{16.835}\\%
{12.99}t{-20}{15}t{5}{15}\\%
{14.58}t{-20.92}{13.85}t{7.07}{13.85}\\%
{16.65}t{-22.11}{13.45}t{8.66}{13.45}\\%
{19.06}t{-23.51}{13.85}t{9.66}{13.85}\\%
{21.65}t{-25}{15}t{10}{15}\\%
{24.24}t{-23.51}{13.85}t{9.66}{13.85}\\%
{26.65}t{-22.11}{13.45}t{8.66}{13.45}\\%
{28.72}t{-20.92}{13.85}t{7.07}{13.85}\\%
{30.31}t{-20}{15}t{5}{15}\\%
{31.31}t{-19.42}{16.835}t{2.59}{16.835}\\%
{31.65}t{-19.23}{19.23}jt{0}{19.23}\\%
{43.3}t{-12.5}{25}\\%
{43.3}e{0}%
}
```

Sit at word processor for two
hours composing title for new
book. Head for tea room at 11
0'clock where six colleagues are sit-
ting round walls in silence. Reminds me
of mental hos- pital day room
but no strait jackets, except
perhaps intel- lectual. Make
coffee with back to them in order
to surreptitiously use someone
else's milk from fridge. Wonder
why no one ever asks me what I
have been doing in the department
for the last three years. Listen to sudden
burst of animated conversation about
new Mac software, and realise that
my social isolation is due to the
fact that I have an Amstrad.

`\shapepar` cheats a bit when the horizontal gap between two bits of text is small (like down in the notch of `\heartpar`). When the gap is less than an inter-word space it is eliminated, and the texts are joined; when it is somewhat larger it is expanded to give it more visibility.

Since the processing is slow, there are some messages to say how things are going. These can be eliminated to save space (Put a % at the start of every `\message` line.) Or you can get even more verbose messages by *removing* the % that precedes many other `\message` commands.

There are also a number of parameters which can be changed to affect the size-optimization procedure. Search for the word *optimize* in the source.

X Report on 'Book and Journal Production'

Carol Hewlett
London School of Economics

The first meeting of the UK TEX Users' Group of 1992 was held at the School of Oriental and African Studies, London, on Tuesday 11 February 1992. The subject for the day was 'Book and Journal Production' which attracted a large audience including overseas visitors from as far away as Belfast!

Rod Mulvey

Printing House, Cambridge University Press

Rod Mulvey assumed that the audience was familiar with TEX and LATEX, and aimed his talk at publishers, with the following topics:

1. What type of TEX files should the authors submit
2. How should the TEX files be submitted
3. What agreements should there be with authors
4. Checks to judge if TEX files will work
5. How to convert files to *your* design
6. Sub-editing
7. Artwork.

After the receipt of the author's initial manuscript there needed to be a TEX check and a report and at the same time the work should be checked by the sub-editor who would prepare a marked copy. The next step was to determine the final method of production and the costs. Following that, and assuming TEX was to be used, the TEX manuscript would be edited and the artwork prepared separately. The artwork would be pasted in to the pages output from TEX to make up the pages and this, subject to late corrections, made up the camera-ready copy.

The journal production cycle was similar, except that before the manuscript reached the production stage it would have been refereed and passed by the journal editor. If the journal was using TEX or LATEX then any typescripts would have to be re-keyed.

Rod then enumerated the kinds of TEX input that an author might submit: it could be at the initial stage, before any pre-subediting had been done; it could be at the stage where only re-design and final corrections were needed; it could include the publisher's design and just need final corrections and an index – or it could be a `dvi` file (CRC on disk), or even the camera-ready copy itself.

What the author should submit depended partly on which of these stages of TEX input was involved. If TEX source files were being submitted, than all author-defined macros must be included. If the author were submitting `dvi` files then a paper copy must also be sent, as there could be problems printing from a `dvi` file if non-TEX fonts had been used. So it was essential for the author to send proper documentation of what was being submitted, to include a list of all input files and macros used and full details of any unusual fonts required.

Authors might submit their work on disks or magnetic tapes, or by electronic mail. Some problems with electronic mail were the possibility of the files becoming corrupted and the chance of the printing house mislaying them because the files were not expected or not identified. Even disks and tapes did not always contain what they were supposed to!

Rod pointed out that a lot of work may be needed to convert author's TEX to printer's TEX, and gave some examples.

It was necessary for subeditors to understand TEX– he showed an example of unnecessary subediting for a TEX manuscript. He referred to an article in *Learned Publishing*, volume 4, number 9, 1991 by R J Skaer on subediting for TEX manuscripts.

Authors tended to use too wide a measure for their text; in LATEX, this could be changed very easily although an automatic change could also make problems, particularly with mathematical formulae.

Rod favoured making style files or macros available to intending authors so that the work was in the right format from the start. This does imply the existence of suitable style files and macros. It was worth designing them for journals and standard monographs and for long books — over 300 pages. For shorter, one-off books designing a style file could cost as much as conventional typesetting, although if the style file were then used from the beginning it would be worth it.

One particular problem that Rod had encountered was the use of Times font for setting maths. CUP had licensed a TEX simulation of Times for authors to use. The increased use of PostScript fonts will help get round this kind of problem.

Artwork was often a problem for publishers. Typically, artwork would be done by a drawing office and pasted in. Now authors could include some artwork with a TEX file: if this were the case, then it was again essential that the author provided any macros used.

A major weakness of TEX was that was not an automatic typesetting program. This was particularly true with respect to floats. There was no interactive page make up with current TEX, and this was an area that he would particularly wish to see improved.

Rod then discussed the question of who does the work with TEX manuscripts. It could be some or all of the author, the typesetter, the subeditor the academic editor/institution and the publishing house. It was important to make an agreement with the author covering these issues and to establish a policy with respect to electronic text regarding subediting and correction. He recommended setting up standard designs in LATEX and TEX.

Geeti Granger
John Wiley & Sons Ltd

Books produced by Wiley are generally scientific in subject and that their markets cover UK, Europe, Middle East, Africa and Japan. They produce about 185 new books each year and over 600 journal issues, together amounting to some 130,000 pages. Geeti's section was established in 1984 initially to process disks, but with a set of objectives which included building a digital archive of the books produced and moving towards true 'demand printing'. A new set of objectives had been established in 1989; these were:

1. To develop technical expertise with a view to enhancing John Wiley's competitive position in the long term.
2. To assist in the most competitive market of all, that for the best authors.
3. To offer an increasing level of support to their authors.
4. To prepare for major changes in technology.
5. To open the possibility of genuine on-demand printing.

Geeti's department used Sun systems running Unix, PCs running MS-DOS and Apple Macintosh machines, linked with LocalTalk and ethernet networks. Various peripherals were linked in, including a scanner, cassette tape drive and a number of LaserWriters.

Various items of software were used: all the machines ran TEX of some flavour and this, together with Ventura Publisher occupied 6 members of staff. TEX was used for the books and journals, with Ventura Publisher being used for some in-house material. Other standard software for DTP, drawing and translation was available and this was covered by another member of staff and a technical support person. All the output was PostScript.

In a typical year, Geeti's department was responsible for about 25 new books, six complete journal issues plus several individual papers and about 1000 pages of large indexes made for 'non-disk' books.

The book production cycle at John Wiley was as follows: To begin with, the author submitted a test disk and hard copy. The next stage was a transmittal meeting from the editorial side to the production side. A standard schedule was used for disk based manuscripts. Copy editing and artwork preparation were both done by free-lance people. Then followed page proofs, author's corrections and camera-ready copy. Geeti said that they imposed no restrictions on the kind of disks supplied — provided they were readable and contained what they were supposed to. As far as TEX was concerned, six standard styles had been developed, but the macros still needed to be tweaked. Page balancing in TEX was done by hand at the last stage.

Geeti identified some of the problems with disks: authors still make mistakes. They did not always take enough care to distinguish between 1 and l, O and 0. The hard copy supplied was not always the same as the text on the disk. Authors tended to be inconsistent and didn't follow guidelines. Where complex maths and chemistry and tables were included the work had sometimes to be re-input to conform to the required style. Geeti commented that she found that spell checkers were not particularly useful in scientific work.

To conclude, Geeti commented on the position of her department as an in-house production unit for John Wiley & Sons Ltd. She found that scheduling could be difficult: the work load had great variation. In very slack periods, they would need to get ordinary manuscripts typed to disk (by free-lances) so that they could be treated as disk-based. There was also a lack of flexibility. Being in-house meant there was not a normal commercial relationship between her department and the rest of the company. She felt that decision-making was driven by the technology and that there was an investment cycle or spiral. She further commented that it was difficult to find and then to retain trained staff. Colleagues at John Wiley & Sons needed to recognise the change in working practices.

Peter Robinson and Stephen Miller
Oxford University
Peter Robinson is from the Computer and Manuscripts Project and Stephen Miller is a member of the Computing Service, both at the University of Oxford. They used a Macintosh to demonstrate a program and a set of macros that together can be used to produce critical editions. Stephen illustrated what a critical edition is by showing some lines of Shakespeare's *Hamlet*. The top part of the page contained the text according to a particular edition. In the bottom part of the page were notes on various differences between the chosen edition and other editions. These can include different use of upper or lower case, different forms or spellings, and commentary on the text.

The traditional process of making a critical edition involves visiting a great many libraries and using index cards to note all the variations. Peter has developed a program, COLLATE, which will put all this into computer files. Having sorted the text using COLLATE, it is then possible to include TEX markup commands so that the output can be processed by Dominik Wujastyk's *edmac* macros to produce a typeset critical edition.

The COLLATE program can be obtained from Peter Robinson, email: `peterr@uk.ac.ox.vax` and the *edmac* macros can be obtained from Dominik Wujastyk, Wellcome Institute for the History of Medicine, 183, Euston Road, London, NW1 2BN; email: `d.wujastyk@uk.ac.ucl`

Christina Thiele
Carleton University Press
Christina Thiele said that her work was virtually all in the humanities — and that she used TEX for it all. The Carleton University Press published in various languages, principally English and French. TEX was used in-house, not by their authors. There was about 80% electronic submission of manuscripts. Authors were given a form to complete. Christina always included a log of the file's history at the start of each TEX file. The publishers made any necessary corrections to the text, as they can't fix the errors that the authors introduce. With this particular work, Christina uses some 20–30 basic TEX commands and modifies existing macros. She usually starts by coding the text and writes the macros later. They mainly use IBMs on which to run TEX. She does 12–14% of the University Press's output; previously all the typesetting was farmed out.

She emphasized how important it was to document your own work and reminded us that TEX was for humanities as well as maths.

Malcolm Clark
Polytechnic of Central London
The final speaker of the conference was Malcolm Clark of the Polytechnic of Central London and current President of TUG. His talk described the problems he had faced when producing the proceedings of the TEX88 conference at Exeter, and how he had solved them.

Malcolm started by giving the history of his previous experience of producing books with TEX. He then discussed how he had chosen the papers to appear in the Proceedings. His basic idea was to print the papers that had actually been given, but the editor's decision was final and he did include one paper that had not been given – and had to omit papers that had been given but had not achieved any permanent form. He pointed out the choices facing an editor where not all the authors were writing their native language: he liked to edit the work enough for the meaning to be clear but so as to preserve the author's voice. He said that it is not possible to achieve uniformity of texture over a multi-author work as styles varied too much. He told of his difficulties of finding a publisher, and his determination to do so – if only for the warehousing. He had chosen to use Computer Modern typeface, and pointed that at 1270 dpi resolution it was excellent. Malcolm had used a professional indexer to compile the index for the book but he was not entirely happy with the result.

His conclusions were that this kind of publishing was time consuming. The book needed 'objective' editing and copy editing was also essential. He had discovered that publishing is more than just assembling the papers. He pointed out that it was tempting to keep refining, but that the temptation should be resisted. He reminded us that other amateurs (his authors) had their own priorities and so didn't keep to Malcolm's timetable. And finally he said don't expect thanks, but it is fun.

The Conference ended with a general forum. Three main points were raised. The first one was that there was a need for a common set of tfms for PostScript. (These are the font metrics that determine how much horizontal space each character occupies.)

The second was the availability of publisher's style files. Geeti Granger said that John Wiley's style files were

available only to intending authors. Rod Mulvey said that this was for the publishers to decide; some of the style files that he uses are in the Aston Archive and others are on the Cambridge University computer. The question of out-of-date style files was mentioned, but there is no easy or complete answer.

The third point was to do with the potential archival nature of the electronic manuscript. On the whole, the publishers represented by the speakers did keep the electronic manuscripts, but only Geeti Granger said that as a matter of course she made all late corrections to the electronic manuscript.

Reference was made to the work done by Jane Dorner on the arrangements (or the lack of them) between authors and publishers for dealing with electronic manuscripts. Her report is called *Authors and Information Technology. New Challenges in Publishing*, BNB Research Fund Report 52, published by The British Library 1991 and available from Publications Sales Unit, The British Library, Boston Spa, Wetherby, West Yorkshire, LS23 7BQ. This book was reviewed in the Newsletter of the British Computer Society Electronic Publishing Specialist Group, volume 7, number 1, December 1991, which contains a further article by Ms Dorner.

XI Report of the 1992 UKTUG AGM

R. A. Bailey
Hon. Secretary

Official report of the AGM of the UK TEX Users Group
held at Aston University (Room 708, Main Building)
on Wednesday 14 October 1992 at 1100 hours

In the absence through illness of the UKTUG chair, P. Abbott, the meeting was chaired by P. Taylor. The group sent its condolences to P. Abbott. There were, at least, twenty-five members present. The following is a brief summary of the business transacted; it is categorized by, roughly, the numbered agenda items.

1. **Chairman's Report** This was read by P. Taylor.
2. **Report of the 1991 AGM** This report had already been published in *Baskerville*, Volume 2, Number 1. Copies were also presented at the meeting. The report was received as correct, and signed by P. Taylor.
3. **Matters Arising** There were no matters arising.
4. **Approval of Accounts** The Treasurer's report was given orally. Copies of the unaudited accounts for 1991–92 were presented. When audited, the accounts will appear in *Baskerville*. It was reported that UKTUG had increased its balance by about £1000 during the year, partly because of an increase in membership to about 170, partly because two of the meetings showed a small profit.
5. **Appointment of Auditor(s)** Colin Smith was reappointed auditor for 1993, and thanked for his work to date.
6. **Election of Committee** Of the previous committee of twelve, one continues as Chair and two had been co-opted. Six members retired, only one of whom was willing and eligible to stand for re-election. There were three further nominations: thus there was no need for an election and the total size of the committee became seven (excluding the Chair).
   Subsequently, at its next meeting, the committee invited P. Taylor to continue as Acting Chair until P. Abbott recovered his health. The committee once again co-opted S. F. Brooks (as *Baskerville* editor) and J. Petts; and also co-opted I. McNeil-Sinclair and D. V. J. Murphy. It appointed the following honorary officers:

R. A. Bailey	Committee Secretary
I. McNeil-Sinclair	Membership Enquiries
I. W. Hall	Treasurer
D. V. J. Murphy	Meetings Secretary
D. W. Penfold	Membership Secretary

   The remaining committee membership consists of D. Eckersley, R. Fairbairns, S. P. Q. Rahtz and G. Toal.
7. **Membership Fees** The Treasurer proposed the following motion, on behalf of the committee:

   The Committee proposes maintaining membership fees at their 1992 levels, that is, £15.00 for full membership or £7.50 for student membership.

   G. Toal proposed an amendment: that the word *full-time* be inserted before the word *student*. After some discussion, I. W. Hall seconded this amendment, on which the voting was

For	17
Against	2
Abstentions	3.

   The voting on the amended proposal was

For	20
Against	0
Abstentions	2.

8. **Meetings Fees** The Treasurer proposed the following motion, on behalf of the committee:

   The Committee proposes an increase in fees for attending ordinary meetings:
   from £15 to £20 for members
   from £20 to £30 for non-members
   and for attending workshops:
   from £25 to £30 for members
   from £30 to £40 for non-members.

   He explained that one motivation for the change was to make membership more attractive, by enabling members to recoup their membership fees by attending only two meetings. After some discussion of the benefits of allowing the committee flexibility in setting fees (for example, to give a discount for booking in advance, or to put on a popular loss-leading meeting), J. Fine proposed an amendment: that the word *customary* be inserted before the word *fees*. I. McNeil-Sinclair seconded the amendment, and the voting was

For	15
Against	3
Abstentions	5.

   The Treasurer then moved the amended motion, seconded by S. F. Brooks: the voting was

For	21
Against	1
Abstentions	2.

9. **Cathy Booth Memorial Fund** On behalf of the committee, C. Hewlett proposed the following motion:

   The UK TEX Users' Group will establish a fund to be called the Cathy Booth Memorial Fund.
   The Fund will be used to support education and research in electronic publishing in general and in the use and development of TEX and its relatives in particular, and for other charitable purposes connected with education.

   The motion was seconded by C. A. Rowley and passed unanimously.

10. **Donations** P. Taylor reported two donations made by the committee during the year: £500 to the LaTeX3 project; and computing equipment to D. Osborne to enable him to maintain the UKTEX Digest and TEXhax from home. Both recipients expressed thanks at the meeting.
11. **Newsletter Editor** One edition of the Group's newsletter, *Baskerville*, had appeared in 1992 under the editorship of P. Taylor. S. Brooks had then taken over the editorship, and will soon produce another edition. In the immediate future, these two people are willing to continue to share the editorship. Contributions from the membership are always gratefully received.
12. **Other Business** There was constructive discussion of several issues, which the committee agreed to look at further.
    - Should the UKTUG organize professional training of TEX, and, if so, how?
    - Could members elect to receive notices from UKTUG primarily by electronic mail or primarily by ordinary mail, at choice?
    - The committee would examine the feasibility of creating, maintaining and circulating a UKTUG membership list. The list would not be for sale or gift to any non-member. On membership renewal forms members would be invited to give such details as they were willing to have published (e.g. telephone number, fax number, electronic mail address, address, TEX platform), and would be given the opportunity to opt out of the list completely.

The meeting concluded by expressing its thanks to M. Campbell and the local catering and technical staff, for making the meeting possible.

XII The Comprehensive TEX Archive Network

Sebastian Rahtz
ArchaeoInformatica, York

1 Introduction

This article briefly describes the UK TEX Archive (Internet 'Daughter' archive) on `ftp.tex.ac.uk` for the benefit of UKTUG members. This archive is part of a collaborating network of archives known as CTAN (Comprehensive TEX Archive Network). This is the creation of a TEX Users Group working party on archives chaired by George Greenwade which discussed the issue by electronic mail during 1992. Following an initial implementation on `ftp.tex.ac.uk` in July 1992, the three main archives now follow the same structure and have identical files (`ftp.tex.ac.uk`, `ftp.shsu.edu` and `ftp.uni-stuttgart.de`).

The preferred access method to the UK TEX Archive is using the *gopher* program which has a set of useful indexes to help you locate what you are looking for, but Internet *ftp* access is also very common. JANET users may only access the machine using the `ftp-relay` site, as it has no X25 connection. The 'Father' archive on `uk.ac.tex` remains open for JANET X25 access, and is regularly updated from the 'Daughter'.[9]

2 FTP access

The CTAN archives all run an enhanced *ftp* server, which may possibly confuse your client. If your *ftp* client crashes or hangs shortly after login please try using a dash (-) as the first character of your password. This *ftp* server supports dynamic compression, uncompression, and archive creation options. Fetch the top-level file `README.archive-features` for information. The server also supports site-defined commands to assist you. Please read `README.site-commands` for a brief overview.

On Aston and SHSU servers, you can go to the top of the archive tree using the alias `CTAN:` — type `cd CTAN:` (*with* the colon).

Please report any problems via e-mail to `ctan-mgr@shsu.edu`.

3 Submitting material to the CTAN archives

To submit a file to CTAN using *ftp* to `ftp.tex.ac.uk` or `ftp.shsu.edu` as your point of entry, please go to the `/incoming` directory at the root level where you entered this host. `cd /incoming` from any *ftp* prompt should get you there.

Once in the `/incoming` directory, please do one of the following:

1. if you are submitting a single file, simply put it in the `/incoming` directory; or
2. if you are submitting a set of files intended to be used together as a package, please create a directory within `/incoming` with: `mkdir your_name` (replace the string your_name with the directory name you wish to use), then: `cd your_name`, then put your files in this directory.

Be sure to use the proper mode (ascii or binary) when you put your files into this directory. Following this, please send an electronic mail message to: `ctan-mgr@shsu.edu` with the suggested subject of: *CTAN Submission*. Your message should (a) state what file(s) you have put where, (b) include a brief overview of what the file(s) is/are intended to do, and (c) [optionally] where you recommend placing your submission within the CTAN directory hierarchy.

Once classified and moved into the CTAN directory hierarchy, your submission will automatically be propagated to the other CTAN hosts.

Please do not abuse your privilege to access the `/incoming` directory by placing unauthorized files in it or for purposes of making your files accessible to another site without the prior knowledge and consent of the host university. All *ftp* transactions to the host are logged. Abuse of this service will very likely result in complete loss of *ftp* access

[9] Access the machine (DTE 000020120091) with username *public* and password *public*.

to the host from your site until your system administrators have been notified and appropriate intervention is made on their behalf.

Please forward any inquiries about this service to `ctan-mgr@shsu.edu`.

4 Archive hierarchy description

We now describe the main directories which make up CTAN; readers are referred to David Jones' *Index of TEX Styles and Macros* for details of macro packages and individual style files.

archive-tools contains the various archiving tools which users may find useful.

bibliography contains bibliography-related files, such as BIBTEX.

digests contains back issues of TEX-related periodicals.

documentation contains files and tutorials which document various aspect of TEX.

dviware contains the various dvi-to-whatever filters and drivers.

fonts contains a collection of fonts, both sources and pre-compiled.

graphics contains utilities and macros related to graphics.

help contains files which provide an overview to the archive and the TEX system.

indexing contains utilities and related files for indexing documents.

languages contains non-English related implementations of TEX.

macros contains macros for TEX and its derivatives in unique subdirectories.

support contains files and programs which can be used in support of TEX.

systems contains complete system setups, organized by operating system.

web contains WEB-related files and utilities.

XIII Disk and tape TEX distributions in the UK

For a complete Unix TEX distribution, a 1/4 inch cartridge, QIC-120 or QIC-150 format (DC600A or DC6150) can sent with envelope *and* stamps for return postage to:

David Osborne
Cripps Computing Centre,
University of Nottingham,
Nottingham NG7 2RD

Due to currency exchange, this service is offered only within the UK.

For copies of emTEX (for OS/2, PC-DOS and MS-DOS), and a free catalogue detailing other disk formats, precompiled fonts and lots of other goodies, you can also contact: Eigen PD Software, P.O. Box 722, Swindon SN2 6YB (tel: 0793-611270) (JANET e-mail address: `kellett@uk.ac.cran.rmcs`)

Enquiries for TEX for the Atari ST etc. can be directed to: The South West Software Library, P.O. Box 562, Wimborne, Dorset BH21 2YD (JANET e-mail address: `mdryden@uk.co.compulink.cix`)

The international TEX Users Group can also supply many TEX materials on disk. Contact:

TEX Users Group
PO Box 869
Santa Barbara, CA 93102
USA
Phone: 805-899-4673 *E-mail*: `tug@tug.org`