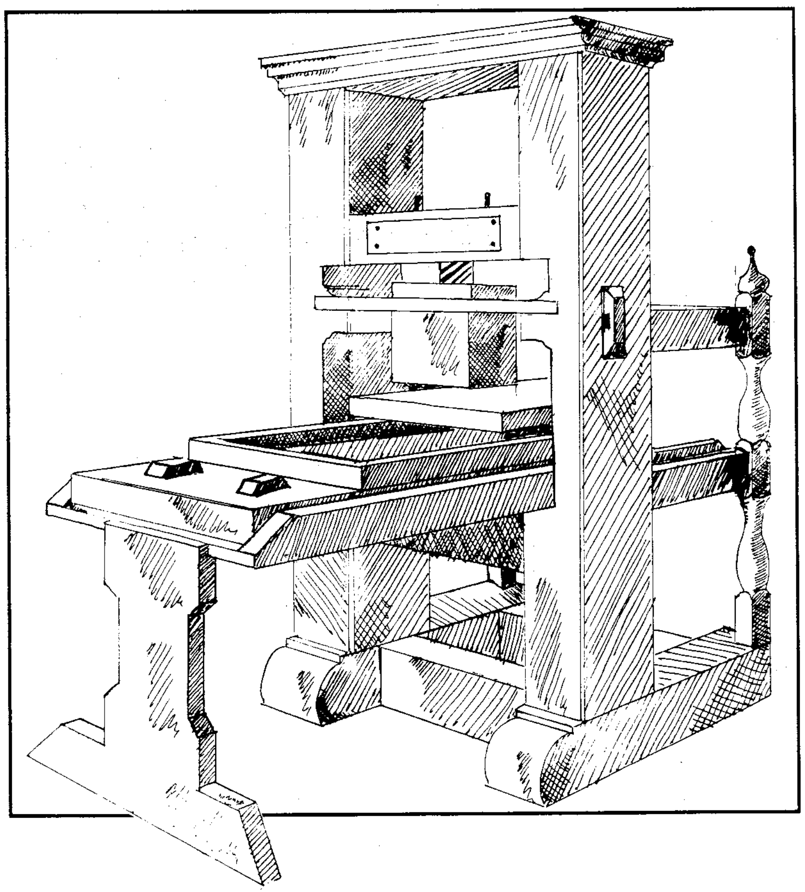
TUGBOAT

The Communications of the TEX Users Group



Volume 14, Number 2, July 1993

TEX Users Group

Memberships and Subscriptions

TUGboat (ISSN 0896-3207) is published quarterly by the TEX Users Group, Balboa Building, Room 307, 735 State Street, Santa Barbara, CA 93101, U.S.A.

1993 dues for individual members are as follows:

- Ordinary members: $60
- Students: $30

Membership in the TEX Users Group is for the calendar year, and includes all issues of *TUGboat* and *TEX and TUG NEWS* for the year in which membership begins or is renewed. Individual membership is open only to named individuals, and carries with it such rights and responsibilities as voting in the annual election. A membership form is provided on page 153.

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

Second-class postage paid at Santa Barbara, CA, and additional mailing offices. Postmaster: Send address changes to *TUGboat*, TEX Users Group, P. O. Box 869, Santa Barbara, CA 93102-0869, U.S.A.

Institutional Membership

Institutional Membership is a means of showing continuing interest in and support for both TEX and the TEX Users Group. For further information, contact the TUG office.

Addresses

General correspondence:
TEX Users Group
P. O. Box 869
Santa Barbara,
CA 93102-0869 USA

Payments:
TEX Users Group
P. O. Box 21041
Santa Barbara,
CA 93121-1041 USA

Parcel post,
delivery services:
TEX Users Group
Balboa Building
Room 307
735 State Street
Santa Barbara, CA 93101
USA

Telephone

805-963-1338

Fax

805-963-8358

Electronic Mail

(Internet)
General correspondence:
`TUG@tug.org`
Submissions to *TUGboat*:
`TUGboat@Math.AMS.org`

The arrangement of type on a surface communicates through our nonverbal system of perception as well as our verbal sense of what the words "say."

Paul Kahn and Krzysztof Lenk
Typography for the Computer Screen: Applying the Lessons of Print to Electronic Documents, *Seybold Report on Desktop Publishing* (July 1993)

TUGBOAT

COMMUNICATIONS OF THE TEX USERS GROUP

EDITOR BARBARA BEETON

VOLUME 14, NUMBER 2 • JULY 1993
PROVIDENCE • RHODE ISLAND • U.S.A.

TUGboat

During 1993, the communications of the TEX Users Group will be published in four issues. One issue (Vol. 14, No. 3) will contain the Proceedings of the 1993 TUG Annual Meeting.

TUGboat is distributed as a benefit of membership to all members.

Submissions to *TUGboat* are reviewed by volunteers and checked by the Editor before publication. However, the authors are still assumed to be the experts. Questions regarding content or accuracy should therefore be directed to the authors, with an information copy to the Editor.

Submitting Items for Publication

The next regular issue will be Vol. 14, No. 4; deadlines for that issue will have passed by the time this issue is mailed. Deadlines for Vol. 15, No. 1 are November 15, 1993, for technical items, and December 13, 1993, for reports and similar items. Mailing dates for these two issues are scheduled for November and March. Deadlines for future issues are listed in the Calendar, page 147.

Manuscripts should be submitted to a member of the *TUGboat* Editorial Board. Articles of general interest, those not covered by any of the editorial departments listed, and all items submitted on magnetic media or as camera-ready copy should be addressed to the Editor, Barbara Beeton (see address on p. 95).

Contributions in electronic form are encouraged, via electronic mail, on magnetic tape or diskette, or transferred directly to the American Mathematical Society's computer; contributions in the form of camera copy are also accepted. The *TUGboat* "style files", for use with either `plain` TEX or LATEX, are available "on all good archives". For authors who have no access to a network, they will be sent on request; please specify which is preferred. For instructions, write or call the TUG office.

An address has been set up on the AMS computer for receipt of contributions sent via electronic mail: `TUGboat@Math.AMS.org` on the Internet.

Reviewers

Additional reviewers are needed, to assist in checking new articles for completeness, accuracy, and presentation. Volunteers are invited to submit their names and interests for consideration; write to `TUGboat@Math.AMS.org` or to the Editor, Barbara Beeton (see address on p. 95).

Other TUG Publications

TUG publishes the series *TEXniques*, in which have appeared reference materials and user manuals for macro packages and TEX-related software, as well as the Proceedings of the 1987 and 1988 Annual Meetings. Other publications on TEXnical subjects also appear from time to time.

TUG is interested in considering additional manuscripts for publication. These might include manuals, instructional materials, documentation, or works on any other topic that might be useful to the TEX community in general. Provision can be made for including macro packages or software in computer-readable form. If you have any such items or know of any that you would like considered for publication, send the information to the attention of the Publications Committee in care of the TUG office.

TUGboat Advertising and Mailing Lists

For information about advertising rates, publication schedules or the purchase of TUG mailing lists, write or call the TUG office.

Trademarks

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

APS $\mu5$ is a trademark of Autologic, Inc.
DOS and MS/DOS are trademarks of MicroSoft Corporation
METAFONT is a trademark of Addison-Wesley Inc.
PC TEX is a registered trademark of Personal TEX, Inc.
PostScript is a trademark of Adobe Systems, Inc.
TEX and $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX are trademarks of the American Mathematical Society.
Textures is a trademark of Blue Sky Research.
UNIX is a registered trademark of UNIX Systems Laboratories, Inc.

General Delivery

Opening words

Christina Thiele

It's summer. Beginning of July. What could I write about for this issue of *TUGboat*? You've been anxiously awaiting your copy of 14 #**1**, and it has finally arrived. Shortly after that, TTN 2,3 turned up — and now this issue of *TUGboat*. Information glut!

Should I suggest you take your copies with you on vacation? No ... that would be pushing our mission too far! True, we want to spread information about TEX, but ... at the beach? We're like everyone else — we want a vacation away from our favourite program too! So what can I tell you today?

Well, I've begun having some fun again with TEX. "Well," you say, "don't you work with it all the time? Isn't that enough?!" No ... what I do is *work* with TEX. But *play*? Not often. And usually, it's with a specific project/problem in mind.

Nevertheless, I've been having fun playing with fonts: one was to address a need for some Hebrew text (complete with vowel diacritics) in an article on Roger Bacon's knowledge and use of other languages; the other was to try and bring "an Irish flavour" to a collection of articles.

The Hebrew fonts I used are by Joel Hoffman and come complete with a little user's guide.[1] Having just that little bit of documentation, which itself then serves as a model of how to use the macros, made it so easy to introduce something entirely new into what was otherwise quite a mundane file. And it reminded me that no matter how beautiful or exciting or clever a macro or font or style file is, without documentation it's just not a lot of fun to try to use it. So for all of you who are putting out style files or macros or designing fonts, do give serious thought to writing up some simple little samples and explanations of what you're offering — you'll have some extremely grateful users sending you nice notes of appreciation and thanks.

The other bit of fun was to try using a font which had been suggested as a way of adding some "Irish-ness" to a collection of conference papers: using Irish knotwork as accent pieces — some call them dingbats. Jo Jaquina has designed 7 fonts, variations of one another, and provided a few samples of their use.[2] The results? Quite spectacular, or so it seems to me. This is the first time I've used a font which isn't letters and numbers and symbols, but rather, each "character" is a design element which are then combined, with fairly seamless results, into complex patterns (rather like taking some of those ASCII symbols on a DOS machine, the ones which define corners and lines and shaded boxes and all). Manipulating the font elements hasn't been all that easy, mainly because I may be missing some basic TEX concepts (I do have some interesting gaps ... well, ok, chasms ... in my knowledge of the beast). But I am getting the results I wanted, and hoped for. So tip of the hat to his efforts, and for generously placing his fonts onto the TEX archives.

[1] Available as `hclassic` at a variety of servers; I picked up my copies from `noa.huji.ac.il`.

[2] I acquired the fonts from the Stuttgart server in `/soft/tex/fonts/knot`.

1 TUG '93

Many of us are preparing for the Annual Meeting, writing papers, re-writing papers (!), working on overhead slides, thinking of all the questions we have for this or that person; maybe some vacation time *away* from all this TEX stuff! If you aren't able to make it to this meeting, remember that the *Proceedings* issue of *TUGboat* will bring you all the details. And while I'm talking about the meeting: the Conference Planning Committee has put out a call for proposals for future meeting sites. See TTN 2,3:26 for more information.

Speaking of conferences ...

2 Another society

... let me tell you about one I attended a few weeks ago, in June. It was the 15th Annual Meeting of the Society for Scholarly Publishing, held this year in Washington, D.C. They count some 1,000 members, and over 400 were registered for one, two or all three days of the meeting this year! That's an astonishing figure — and one which I certainly envy! Of course, being on the east coast, where a large proportion of the publishing and printing industry resides, made such a high turnout possible. We ourselves experienced this at the TUG'91 annual meeting, "Inroads into Publishing", where some 240 attendees came for the first day, and overall attendance hovered around 200. I believe the potential exists within our own user group to see attendance figures still grow. But back to the SSP meeting ...

I was initially drawn to the program because I saw, for the first time that I can recall, mention of "TEX"; it was part of a session entitled "Alphabet Soup: SGML, TEX, PS, DTD, FOSI, DSSSL ..." After contacting the organiser, saying how delighted

I was, as TUG's president, to see TeX on the program, I ended up offering to put together a poster display on the use of TeX for humanities journals — a counterpoint to the usual stereotype of "TeX is for math and all that technical stuff."

I've managed to locate a total of 16 publications, but surely there are more! I went through our membership directory of last year, and there were several addresses there which seemed to carry a hint, a whiff of a humanities involvement ...

I would like to see this type of information collected and made available to all users, current and potential, of TeX for the non-math/technical side of things. So I'm turning this into what one might term "a personal on-going project" — sounds rather grandiose!

I would therefore like to enlist your assistance in identifying more humanities journals which are currently using TeX or which once used it: contact me at `cthiele@ccs.carleton.ca`, and I'll send you a form which asks for the basic information. This is for journals which use TeX either as a purely in-house tool, or which accept TeX-encoded files from authors.

There is a large bibliography maintained by Nelson Beebe on publications produced with TeX, or about TeX, and I'll be passing along the information I collect to Nelson so that the collection will be as up-to-date as possible (Nelson's heard this promise before, but now that it's been done publicly ...). As for books produced with TeX: do try to send in your particulars to Nelson Beebe and add to the bibliography.

To step back a bit from the details ... the SSP meeting was well worth the time and money spent. For me, TeX is a terrific tool — but I'm using it in the larger context of scholarly/academic publishing. And that's where I need additional information and guidance. I urge everyone who is in the same situation — using TeX as a tool within a larger context — to look around and consider joining a user group or association or society which is mainly about that context. It's not your TeX knowledge which will improve so much as your knowledge in applying TeX well.

And with that final urging, it's time to bring the column to a close. Enjoy reading this issue. Have a wonderful summer. Take a break from your keyboards. And we'll see you in September.

3 Elections coming up!

Well ... we didn't get this issue off to the printer before the Aston meeting, so I have a chance to insert one final note.

This fall we'll be having elections for the board. There are 15 positions open, and it's going to be a bit tricky as we move the entire board onto staggered terms; that is, this is the beginning of the move to have 5 board positions come up for election at any one time, rather than all 15 positions at once.

The terms of the present board members expire December 31, and the terms of the new board members begin January 1. The ending dates of the terms will be shifted ahead to the annual meeting, and the slate is being split into three groups of 5 (after ballots are counted and it's clear which 15 have been elected) so the shortest term will be $1\frac{1}{2}$ years and the longest, $3\frac{1}{2}$ years. The president's term had already been shifted to coincide with the annual meeting. What this will mean is that eventually we will have elections in the spring, before the annual meeting. First spring election will be in 1995, at which time 5 board positions, along with the position of president, will be decided.

But right now, we have a fall election. Nomination forms had to be at the TUG office by September 1st. Ballots will be mailed to all members about 30 days after that date. Marked ballots must be returned within the following 6 weeks. Results will be published in the next available issues of TTN and *TUGboat*.

Take the time to read the election material when it arrives; this user group of ours is very important to us, and to all TeX users. Contributions you make to the group often are contributions which then travel out further: look at the Technical Council's working groups, look at the archives, think about the ways in which material published by TUG has become the main source of information on such software as PiCTeX and `edmac`. It is very important for you to participate in TUG in whatever way you can. Elections are one direct way to do that; a direct line between you, the member, and those who help set the direction for the user group. What happens to TUG matters a lot to me — show that it matters to you, too.

◇ Christina Thiele
5 Homestead Street
Nepean, Ontario
K2E 7N9 Canada
`cthiele@ccs.carleton.ca`

Editorial Comments

Barbara Beeton

Since this issue didn't get sent to the printer before I left town for the annual meeting, let's take this opportunity to catch up on a few items that were announced in Aston, as well as those of which I was already aware earlier.

1 TeX 3.1415 / METAFONT 2.71 on the way

TeX and METAFONT have both added a digit in their progress toward π and e. Early in June I received a package from Don Knuth containing his responses to the last year's worth of bug reports. The technical details have been sent to the implementors, and system-specific implementations should be making their way to an archive near you, or onto the shelves of your commercial supplier.

These are the features of the new release that are important to system managers and users.

1.1 All volumes of *Computers & Typesetting* have been "frozen"

The final versions of these volumes were printed late in 1992. The numerology is, in part,

- the 22nd printing of *The TeXbook* (softcover) and the corresponding edition of Volume A;
- the 5th printing of Volume B;
- the 4th printing of Volume D.

No more changes will be made to these volumes even if they are reprinted again. Errata will continue to be recorded and posted (no more often than annually) to the file `errata.tex` in the TeX archives. There are now eight numbered errata files, `errata.one` through `errata.eight`, covering various time periods from 1982 to the printing of the final versions. For anyone wishing to obtain this information who doesn't have ftp access, check with the TUG office for information.

1.2 This release is internally consistent

Important changes have been made to nearly all files in the TeX system. In addition to the updated TeX and METAFONT programs, there are new versions of `plain.tex` (3.1415) and `plain.mf` (2.71) as well as of the trip and trap tests. Some of the changes: In `plain.tex` the changes relate to the `\b` accent in slanted fonts and to the `\upbracefill` and `\downbracefill` macros; in `plain.mf` there are two corrections to the `cutdraw` macros.

When upgrading, please upgrade *everything*! Don told me, "Please encourage everybody to install this version ASAP."

2 LaTeX 2e coming in October

It was announced at the meeting that the release of LaTeX version 2e is expected in October, to replace the ever-confusing version 2.09 of various dates.

This version of LaTeX will incorporate the NFSS2 (described in the article by Sebastian Rahtz, page 132) and many features provided for enhanced flexibility in a multilingual environment. It is intended as a full replacement for version 2.09, for which maintenance will cease.

The release is timed to coincide with the publication of a new book, *The LaTeX Companion*, by Frank Mittelbach, Michel Goossens and Alexander Samarin. Proofs of the book in its current state were the reference material for a LaTeX course taught by Frank and Michel just before the meeting, and I understand that the intense scrutiny of the students was most welcome in helping to eliminate bugs.

After the release of LaTeX 2e, and until the appearance of LaTeX 3, there will be regular maintenance, with updates scheduled twice a year. LaTeX users can become used to the principle that they should upgrade, or at least consider upgrading, their system every six months, and not just check for the latest update when something goes wrong. At the very least this should help cut down on questions about old versions that now appear too frequently on `comp.text.tex` and friends.

3 The annual meeting at Aston University

The meeting — TUG's first annual meeting outside North America — was a good one. Although the weather didn't always cooperate, there were plenty of attractions to hold the attention of at least this devoted attendee.

A novel feature was a daily "newspaper", *The TUGly Telegraph*. (Actually, the masthead was set in one of Yannis Haralambous' old-style Fraktur fonts, and there were puzzled expressions on some faces until they figured out what it said.) Congratulations to Steffen Kernstock on making it happen; I understand that he arrived with the idea just before the meeting and put it together on the spot.

The last issue of the *Telegraph* listed the participants, and it was a wide-ranging crowd — 162 names (registrants for both the meeting and the courses before and after) from at least 23 countries! I saw a lot of old friends, and am happy to report that I made some new ones too, in the process attaching faces to quite a few names I knew only from e-mail. I do recommend the meeting to anyone who can manage to attend; it's only at meetings of TUG or the other user groups that one will find such a concentration of serious (well, not *always* serious!)

and enthusiastic TEXers, ready to discuss just about any topic with anyone who is interested.

The conference program had a lot of good material, and the pace was sufficiently relaxed that there was usually plenty of time to ask questions, and to talk to the speakers at coffee and tea breaks afterward. As in previous years the program was "single thread", so it wasn't necessary to decide which presentations to attend or skip. For those who couldn't attend, the editors of the Proceedings (Mimi Burbank and Sebastian Rahtz) have been working hard to get the papers into good shape; the results should arrive in your mailbox sometime in October. And thanks to these two good people for giving me a bit of rest.

Unfortunately, I did miss a number of the presentations. Shortly before the meeting, a Technical Working Group on the 256-character math encoding was constituted, and we spent quite a bit of time putting together a preliminary report for presentation at a workshop late in the week. See below for more details.

A number of vendors had displays and demonstrations set up, and I saw quite a few people asking questions. Along with the "active" displays, a rather large table contained a wide range of books on TEX and related subjects. New publications included the two LATEX books reviewed by Nico Poppelier in this issue (page 127), and, wonder of wonders!, at long last, Stephan von Bechtolsheim's *TEX in Practice*, all four volumes of it. A reviewer has been assigned, and a book review will appear in an upcoming issue.

There was one truly special event: an excursion to Stratford on Avon for a performance of *King Lear* by the Royal Shakespeare Company. Before the performance (which was stunning), we were treated to a private showing at the Shakespeare Centre Library of a copy of the First Folio and a number of other books that were related in some way either to Shakespeare or to *King Lear*. These included a couple of herbals (books about plants and their uses; these described plants mentioned in Shakespeare's works), a compendium of literary personages in the Stratford area contemporaneous with Shakespeare (his entry isn't even particularly long, while the other names there have faded from common memory), several fine editions of the plays, some books that might have been source material for the Bard, and various other rare items. The curator of the collection was both enthusiastic and knowledgeable, and she readily answered our many questions, sometimes leafing through the displayed volumes for other relevant examples. What a wonderful time I had! The idea for this excursion was inspired.

Finally, I'd like to extend my thanks to the Program and Arrangements Committees, to the TUG office staff, who were also on hand to help "person" the registration desk, and to the staff of the hosting institution, Aston University. Good show!

3.1 And next year's meeting ...

A proposal regarding meeting sites was accepted by the TUG Board of Directors, recommending that locations be varied in a regular cycle of western North America, eastern North America, and Europe. Since this year's meeting was in the U.K., next year's should be in the west. 1994 will be TUG's 15th year, and Santa Barbara, new home of the TUG office, was chosen as a tentative site. A committee is being formed, and a call for papers will be distributed when information about dates is available.

4 CTAN — the Comprehensive TEX Archive Network

A new wrinkle has been added to our friendly TEX archives, as announced at the meeting. Three major sites, with the possibility of expansion, are now fully synchronized, their contents mirrored within 24 hours. George Greenwade, archivist extraordinaire and Chair of the Technical Working Group on TEX Archive Guidelines, in cooperation with Rainer Schöpf, Sebastian Rahtz, Karl Berry, Joachim Schrod and others, has not only established workable guidelines, but also put them into effect.

The three sites that now comprise CTAN are
`ftp.shsu.edu`
`ftp.tex.ac.uk` and
`ftp.uni-stuttgart.de`
I understand that some site changes may be in the works, to get access to faster, more reliable network connections (the traffic to these sites is truly overwhelming), so keep a lookout for announcements.

A preliminary note can be found in TTN 2,3:15, and a full report from George will appear in the Proceedings of the meeting.

4.1 Part of the `labrea` archive superseded

With the regularization of these reliably maintained archives, the authoritative sites for some classes of files will move away from **`labrea.stanford.edu`**, the last connection of the TEX archives to their original home. For a long time, **`labrea`** has been without an on-site archive manager, and updating of files there is irregular at best. **`labrea`** will remain the canonical site for the core of TEX — Knuth's files — but LATEX maintenance is now in the care of the LATEX 3 project group in Germany, and George has given me an account on the SHSU machine to maintain the *TUGboat* files that have been released for public access.

5 *TUGboat* tables of contents on-line

The *TUGboat* area in the SHSU archive is

`tex-archive/digests/tugboat`

It contains, among other things, the plain and LaTeX styles for *TUGboat* and the TUG meeting proceedings, complete tables of contents for all issues, and Nelson Beebe's explosion of this information into a BibTeX file and permuted index.

The tables of contents are stored by year in files named `tb`*vvyy*`.cnt` (*vv* is the volume number and *yy* the year); they can be run through TeX in 5-volume chunks with "driver" files named `tbcv`*vv*`.tex` where *vv* is a multiple of 5 giving the number of the last volume in the group. To process the tables of contents, the file `tbcont.def` is also needed.

The file of *TUGboat* contents in BibTeX form is `tugboat.bib`; `tugboat.kwic` and `tugboat.ptx` present the same information in keyword-in-context and permuted index form respectively.

I've done some work toward creating a "real" index for *TUGboat* but it is still quite a way from completion. Until that is available, anyone looking for a particular *TUGboat* article should be able to find it with the help of one of these files if they know either the author or some word(s) from the title.

6 Encoding of 256-character math fonts

A project is underway to define an encoding for math symbols in the form of 256-character fonts to accompany the 256-character text fonts in the Cork encoding. This is a joint effort of the TUG Technical Working Group on TeX Extended Mathematics Font Encoding (of which I am chair) and members of the LaTeX 3 project. Active participants include

- for the TWG: Alan Jeffrey, Jan Michael Rynning, and myself;
- for the LaTeX 3 project: Justin Ziegler (a French student whose services have been provided for the summer with support from GUTenberg), Frank Mittelbach, and Chris Rowley.

On the basis of e-mail discussions before the meeting and a lot of working sessions at Aston, a preliminary proposal was presented at a workshop. Since then, an electronic discussion list has been set up to continue to gather ideas, and the traffic has been heavy. A completed proposal is expected by early September, when Justin must return to school. The report of the group will be published in an upcoming issue of *TUGboat*.

7 Gimme an A!

Earlier this year, a contest was announced (TTN 2,2:28), soliciting contributions of macro definitions that would produce a robust, well-behaved A in the LaTeX logo. The winners were to be announced at the Aston meeting. Sadly, perhaps because insufficient time was available between the time the announcement reached its audience and the time of the meeting, there were almost no entries and no winners. Instead, the contest has been extended until December 31. Read the new announcement (page 103) and start writing.

8 A comment on font naming

Although it should be well known by now that the CM prefix should be applied only to fonts that use Don Knuth's Computer Modern shape programs without change, an occasional announcement appears in the electronic discussion lists to the effect that some new font, with new shapes in it, is being added to the repertory available to TeX, and a name proposed with CM at the beginning. The last such instance, early in April, was the subject of this message from Pierre MacKay to the author of the new shapes and to the TeX archive managers:

> [...] I am extremely uneasy about appropriating the CM prefix without a reference to Don Knuth. When AMS adds to the CM repertory, it is by changing parameter files, not by supplying new designs. [...]
>
> My own practice, even for fonts that are only of interest to a limited clientele, and even for fonts that do not seriously alter the character programs of Computer Modern is to change the first letter. The relation to Modern, which is a well-known family of designs, should remain part of the name, but unless Don officially adopts a new set of character designs into the Computer Modern typeface, I think we should be cautious about adding an entire set of new glyphs to the known set that we now call Computer Modern.
>
> At very least, I should like to be assured that Don is aware of, and consents explicitly to the addition of these glyphs to a set that is otherwise almost entirely his own creation. It may be a rendering of Monotype Modern No. 8, but it is **his** rendering.
>
> Don't change the font — just the name.

Now, let's get this off to the printer.

◇ Barbara Beeton
American Mathematical Society
P. O. Box 6248
Providence, RI 02940 USA
`bnb@Math.AMS.com`

The Donald E. Knuth Scholarship: 1993 Scholar and 1994 Announcement

The Donald E. Knuth Scholarship was unfortunately not awarded this year, due to lack of eligible candidates. The amount of money allocated for the award was transferred to the bursary fund of this year's TUG Annual Meeting, which enabled several TeX users from currency-poor countries to attend the Meeting.

At the 1993 TUG Annual Meeting the Board of Directors of TUG has decided that, starting with the 1994 Scholarship, the Knuth Scholarship will be open to all TeX users, not just to TUG members. Otherwise the same rules for the Scholarship competition will apply, and the current committee will serve again for the 1994 Scholarship.

Announcement of the 1994 competition

One Knuth Scholarship will be available for award next year. The competition will be open to all TeX users holding support positions that are secretarial, clerical or editorial in nature. It is therefore not intended for those with a substantial training in technical, scientific or mathematical subjects and, in particular, it is not open to anyone holding, or studying for, a degree with a major or concentration in these areas.

The award will consist of an expense-paid trip to the 1994 TUG Annual Meeting at Santa Barbara, USA, and to the Scholar's choice from the short courses offered in conjunction with that meeting; and TUG membership for 1993, if the Scholar is not a TUG member, or for 1994, if the Scholar is already a TUG member. A cap of $2000 has been set for the award; however, this does not include the meeting or course registration fee, which will be waived.

To enter the competition, applicants should submit to the Scholarship Committee, by the deadline specified below, the input file and final TeX output of a project that displays originality, knowledge of TeX, and good TeXnique.

The project as submitted should be compact in size. If it involves a large document or a large number of documents then only a representative part should be submitted, together with a description of the whole project. For example, from a book just one or two chapters would be appropriate.

The project may make use of a macro package, either a public one such as LaTeX or one that has been developed locally; such a macro package should be identified clearly. Such features as sophisticated use of math mode, of macros that require more than "filling in the blanks", or creation and use of new macros will be taken as illustrations of the applicant's knowledge.

All macros created by the candidate should be well documented with clear descriptions of how they should be used and an indication of how they work internally.

All associated style files, macro-package files, etc., should be supplied, or a clear indication given of any widely available ones used (including version numbers, dates, etc.); clear information should be provided concerning the version of TeX used and about any other software (e.g. particular printer drivers) required. Any nonstandard fonts should be identified and provided in the form of `.tfm` and `.pk` files suitable for use on a 300dpi laser printer.

While the quality of the typographic design will not be an important criterion of the judges, candidates are advised to ensure that their printed output adheres to sound typographic standards; the reasons for any unusual typographic features should be clearly explained.

All files and documents comprising the project must be submitted on paper; the input files should be provided in electronic form as well. Suitable electronic media are IBM PC-compatible or Macintosh diskettes, or a file sent by electronic mail.

A brochure with additional information is available from the TUG office. To obtain a copy, or to request instructions on e-mail submission, write to the address at the end of this announcement, or send a message by e-mail to `TUG@Math.AMS.org` with the subject "Knuth Scholarship request".

Along with the project, each applicant should submit a letter stating the following:

1. affirmation that he/she will be available to attend the 1994 TUG Annual Meeting;
2. affirmation of willingness to participate on the committee to select the next Scholar.

Each applicant should also submit a *curriculum vitae* summarizing relevant personal information, including:

1. statement of job title, with a brief description of duties and responsibilities;
2. description of general post-secondary school education, TeX education, identifying courses attended, manuals studied, personal instruction from experienced TeX users, etc.;
3. description of TeX resources and support used by the candidate in the preparation of the project.

Neither the project nor the *curriculum vitae* should contain the applicant's name or identify the applicant. These materials will be reviewed by the committee without knowledge of applicants' identities.

If, despite these precautions, a candidate is identifiable to any judge, then that judge will be required to make this fact known to the others and to the TUG board members responsible for the conduct of the judging.

The covering letter, *curriculum vitae*, and all macro documentation that is part of the project input should be in English. (English is not required for the output of the project.) However, if English is not the applicant's native language, that will not influence the decision of the committee.

Selection of the Scholarship recipient will be based on the project submitted.

Schedule

The following schedule will apply (all dates are in 1994):

March 7	Deadline for receipt of submissions
March 21–May 16	Judging period
May 23	Notification of winner
July or August (to be announced)	1994 Annual Meeting, Santa Barbara, USA

The 1994 Scholarship Committee consists of

- Chris Rowley, Open University, UK (Chair);
- David Salomon, California State University, Northridge, USA;
- Jenny Smith, John Wiley and Sons, Ltd. Chichester, UK.

Where to write

All applications should be submitted to the Committee in care of the TUG office:

TEX Users Group
Attn: Knuth Scholarship Competition
P.O. Box 869
Santa Barbara, CA 93102 USA
e-mail: `TUG@Math.AMS.org`

Nico Poppelier
Liaison to the 1993 Committee

The A-in-LATEX Contest: Deadline Extended

The A-in-LATEX Contest deadline has been **extended to December 31, 1993**. We felt that the official contest notice did not reach the LATEX users with enough time to work on contest solutions. Take a few minutes, days, or weeks and develop your entry; submit it for judging! You may just have the winning entry! And if you haven't heard about it ...

Currently there are several macros that let you set the LATEX logo. Some print the 'A' as a small-cap 'a' from a caps-and-small-caps font, while others use the `\scriptsize` capital 'A' of the current math family. Some work fine only with 10pt roman and larger; some work fine with several font faces; and some are robust, while others are fragile.

All current TUG members are invited to participate in "The A-in-LATEX Contest." Participants may enter any of the following contest categories:

- **Basic Solution:** The macro(s) should be simple, easy-to-understand, not necessarily robust, although robustness is a favorable feature;
- **Elegant and Sophisticated Solution:** The macro(s) should exhibit good coding manners, and guarantee complete robustness and reliability;
- **Guru Solution:** The macro(s) can resort to the weirdest and most arcane internal TEX commands and need not be understandable except by real gurus.

All solutions should be plain-TEX compatible; it must be specified if the New Font Selection Scheme is required. Additional requirements are that the solution should also work with any other form of (LA)TEX that makes use of the extended 256-glyph fonts.

The contest entries must be submitted electronically to the contest judges shown below:

North America:	Jackie Damrau `damrau@sscux1.ssc.gov`
Elsewhere:	Claudio Beccari `beccari@polito.it`

Submissions should be in the form of a complete LATEX source file with the definition for the new logo located in the preamble and accompanied by detailed comments.

The contest winners will be announced and prizes awarded during the 1994 TEX Users Group Annual Meeting.

Judges for the contest are: Claudio Beccari, Jackie Damrau, Michael Downes, and Peter Flynn.

Philology

Fonts for Africa: The fc Fonts

Jörg Knappen Chair of the TWG on African Languages with Latin Writing

Abstract

A font (fc) with 256 characters for the needs of african languages with latin writing is presented. The so-called critical languages with more than 1 mio. speakers are supported. It is implemented in METAFONT using Sauter's tools for the parametrisation.

Introduction

On the continent of Africa, several hundred different languages are spoken, the numbers of speakers of each ranging from more than 100 mio. down to few hundreds. Many languages do not have a written form; other languages have some written form, but it is not standardised yet. Or even worse: there exists more than one writing system for the same language (e.g. protestant and catholic orthography). Clearly, it was impossible to check out all writing systems of all possible languages. This work confines itself to the so-called 'critical languages' according to a definition of the US Department of Education in 1986. All these languages have more than 1 mio. speakers.
Three main writing systems are in use in Africa:

- *æthiopian* writing, used for several languages in Æthiopia and Eritrea.
- *arabic* writing, used in northern Africa and on the east coast for several languages. It is now losing ground against latin writing; languages like Suaheli and Hausa are now mainly written in latin.
- *latin* writing is now being introduced or established in the rest of Africa.

In this overview, I want to mention also some other writing systems. The Tuareg living in the Sahara use the over 2,000 years old *tifinagh* writing system, a right-to-left writing; the Kopts in Ægypt still use *koptic* for religious purposes.

An interesting development was the invention of a somali alphabet by Osman Yusuf, which combined influences from the three major writing systems, but this system did not achieve official status in Somalia.

The introduction of latin writing was first done by missionaries, who invented orthographies quite arbitrarily. But since the first half of this century, there have been attempts to standardise the newly introduced alphabets.

As a result of these attempts, an 'african reference alphabet' emerged, based on the principle of *one sound, one sign* which is also the principle of the international phonetic alphabet. There are many borrowings from the phonetic alphabet in african writing. These efforts are now coordinated by the UNESCO.

The fc Fonts

The fc font encoding scheme encodes characters necessary to typeset the major african languages with latin writing. In selecting these languages, I followed the list of so-called critical languages. Unfortunately, it proved impossible to put really all characters into one 256-character font. Therefore, I applied the following selection rules:

- The standard TEX character set is still available.
- Characters with entirely new shapes are included with highest priority.
- Characters which need another accent in some languages have the second highest priority.
- Characters which are difficult to handle by TEX macros are preferred over those which are easier to handle (e.g. characters with diacritics below are preferred to those with diacritics above).
- Characters which occur only in tonal languages and carry a tonal mark, are most likely discarded.

The fc fonts have several other interesting features some of which I list here:

- The lower half of the fc encoding scheme is identical to the Cork (or ec) encoding scheme for european languages.
- If a character occurs both in the fc scheme and in the Cork scheme, it has the same encoding.
- The difference between uppercase and the corresponding lowercase character is a constant.
- It is possible to create virtual fonts to obtain all those characters of tonal languages which needed to be discarded.
- It is possible to rebuild the cm fonts and the ec fonts as virtual fonts from the fc fonts. In the latter case three letters are missing (edh, thorn and Thorn) and cannot be done without larger loss in quality. Of course, there are smaller quality losses in building such a virtual font, e.g. for î, where the ^-accent comes out too wide.

The following languages are supported (with the proviso about tones made above): Akan, Bamileke, Basa (Kru), Bemba, Ciokwe, Dinka, Dholuo (Luo), Efik, Ewe-Fon, Fulani (Fulful), Gã, Gbaya, Hausa, Ịgbọ, Kanuri, Kikuyu, Kikongo, Kpelle, Krio, Luba, Mandekan (Bambara), Mende, More, Ngala, Nyanja, Oromo, Rundi, Kinya Rwanda, Sango, Serer, Shona, Somali, Songhai, Sotho (two different writing systems), Suaheli, Tiv, Yao, Yoruba, Xhosa, and Zulu.

I decided to support two european languages, which are not covered by the ec-scheme, namely Maltese and Sami[1].

There are some writing systems which are not supported by the fc fonts. First to mention is the writing of Khoi-San languages with their characteristic click sounds. Missing is the letter ≠ which can be constructed as a macro. The obsolete orthographies of Xhosa and Zulu are not supported (they used a cyrillic B (Б) as uppercase of ɓ).

Also not supported is the obsolete orthography of Ịgbọ which contains an o with horizontal bar (ɵ). A proposed alphabet for Tamasheq (a berber language) is not supported because there is no evidence that it ever caught on.

Design of the letters

The design of the letters closely follows the *computer modern* fonts by Donald E. Knuth. Much of the METAFONT code made its way unchanged to the fc fonts. However, several modifications were necessary in the case of italics. Since Eʋe has both 'f' and 'ƒ', the latter looking like the italic letter f, the italic letter f was redesigned to have a straight, uncurved descender (*f*, *ƒ*). Similarly, the italic letter v was redesigned to have a sharp edge (*v*). In consequence of the last change, the italic letters w and y were changed, too (*w*, *y*).

Deliberately, I changed the appearance of the roman letter *scharfes s*. Its new shape exhibits the ligature of long and short s from which it originally derives (ß).

How to use the fc fonts

At the present time, I support only LaTeX and plain TeX with the new font selection scheme (NFSS) by R. Schöpf and F. Mittelbach. There are the files **`fontdef.fc`** and **`fclfont.sty`** which load the fonts and set the standard TeX commands for accents correctly. But, at the moment there are no standard control sequences to get the newly added characters, except that `\|` shall produce the universal accent (ʼ).

Name	Shape	Input
Hooktop b	ɓ, Ɓ	`+b, +B`
Hooktop c	ƈ, Ƈ	`+c, +C`
Hooktop d	ɗ, Ɗ	`+d, +D`
Open e	ɛ, Ɛ	`+e, +E`
Long f	ƒ, Ƒ	`+f, +F`
Ipa gamma	ɣ, Ɣ	`+g, +G`
Latin iota	ɩ, Ɩ	`+i, +I`
Enj	ɲ, Ɲ	`+j, +J`
Hooktop k	ƙ, Ƙ	`+k, +K`
Eng	ŋ, Ŋ	`+n, +N`
Open o	ɔ, Ɔ	`+o, +O`
Hooktop p	ƥ, Ƥ	`+p, +P`
Esh	ʃ, Ʃ	`+s, +S`
Hooktop t	ƭ, Ƭ	`+t, +T`
Variant u	ʋ, Ʋ	`+u, +U`
Ezh	ʒ, Ʒ	`+z, +Z`
Crossed d	đ, Đ	`/d, /D`
Crossed h	ħ, Ħ	`/h, /H`
Crossed t	ŧ, Ŧ	`/t, /T`
Tailed d	ɖ, Ɖ	`=d, =D`
Inverted e	ə, Ǝ	`=e, =E`
Long t	ʈ, Ʈ	`=t, =T`

Table 1: Overview of the special letters in the fc fonts and their access via `fcuse.sty`

There is another style option (**`fcuse.sty`**) which makes the special characters accessible via active characters. Three characters needed to be activated, since the letters d and t carry the load of three new variants (ɗ, ɖ, đ; ƭ, ʈ, ŧ). An overview of the assignments is given in Table 1.

Implementation

The implementation follows closely to the one of the cm fonts. The distinction between parameter files, driver files and programme files is kept. The parameters are computed from the design size of the fonts with the help of the Sauter tools. This allows an easy nonlinear scaling. I added only one new parameter, **`univ_acc_breath`**, which governs the shape of the universal accent. To adjust the height of accented capitals the parameter **`comma_depth`** was employed. The result is excellent with serif fonts, but not that good with sans serif ones. I made up one

[1] The Cork scheme includes the letter eng (ŋ) especially for Sami, but it is missing the letter t with bar (ŧ). However, with the letter eng several african languages can be written using the Cork scheme as well.

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0y	`	´	ˆ	˜	¨	˝	˚	ˇ	˘	¯	˙	¸	˛	,	‹	›
1y	“	”	„	«	»	–	—		0	ı	ȷ	ff	fi	fl	ffi	ffl
2y	␣	!	"	#	$	%	&	’	(	)	*	+	,	-	.	/
3y	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4y	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5y	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	ˆ	_
6y	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7y	p	q	r	s	t	u	v	w	x	y	z	{	|	}	˜	-
8y	Ɓ	Ɗ	Ɛ	Ǝ	Ƒ	Ĕ	Ɣ	Ħ	Ƙ	Ɲ	Ɔ	Ń	Ʃ	Ŋ	Ʊ	Ƴ
9y	Ƈ	Ƥ	Š	Ṅ	N̲	Ṣ	Ʒ	Ŧ	Ė	Ẹ	Ƭ	Ʈ	ʧ	fj	đ	˵
Ay	ɓ	ɗ	ɛ	ə	ƒ	ĕ	ɣ	ħ	ƙ	ɲ	ɔ	ń	ʃ	ŋ	ʊ	ƴ
By	ƈ	ƥ	š	ṅ	n̲	ṣ	ʒ	ŧ	ė	ẹ	ƭ	ʈ	"	¡	¿	'
Cy	Ɩ	Ị	Ɛ̃	Ã	Ḿ	Ɔ̃	Æ	Ç	È	É	Ê	Ë	E̲	Ē	Ẽ	Ĩ
Dy	Đ	Ñ	Ò	Ȯ	Ô	Õ	Ö	Œ	Ø	Ọ	O̲	Ō	Ŏ	Ụ	Ũ	‐
Ey	ɩ	ị	ɛ̃	ã	ḿ	ɔ̃	æ	ç	è	é	ê	ë	e̲	ē	ẽ	ĩ
Fy	ɖ	ñ	ò	ȯ	ô	õ	ö	œ	ø	ọ	o̲	ō	ŏ	ụ	ũ	ß

Table 2: The font fcr10

new parameter file, creating a `sans serif typewriter` fontshape. There were only minor modifications needed in the programmes of the letters i, I, and l to make it work.

There is not much to say about the driver files, except that there are additional kerns for kV, kW, mV, mW, and eV. The ligatures from the Cork scheme to get german and french quotes are included, too. The programme files are completely reorganised. The huge files are split into smaller ones each containing only one to three letters of the alphabet. The shape of the letter is saved as a picture and only the diacritic is calculated to get an accented letter. This is done to save cpu time and to make the maintenance of the files easier. The accented letters are generated only on demand, i.e. if their code is known. The codes are given in the file `coding.fc`. It is possible to use the same source to create other real fonts by replacing the coding file. The so-created fonts may not be called fc, since this abbreviation is reserved to fonts which are fc-encoded.

Outlook

The fonts are done now, but there is lot of work left in creating suitable styles for african languages and virtual fonts, if needed. For this work, the speakers and users of the african languages are qualified best, and I hope that there will be some results of this work soon.

◇ Jörg Knappen
Chair of the TWG on African Languages with Latin Writing
Institut für Kernphysik
D 55 099 Mainz
R. F. A.
`knappen@vkpmzd.kph.uni-mainz.de`

Fonts

Implementing the extended TEX layout using PostScript fonts

Sebastian Rahtz

1 Introduction

When Donald Knuth made virtual fonts part of the general TEX repertoire at the end of 1989 [5], at the same time as extending both TEX and METAFONT to support 8-bit input, it immediately became obvious that the TEX world needed a new standard for the layout of fonts. This is not the place to review the discussion of what characters should be in a generalized 256 character text font (see [4, 3]), nor to pass judgement on the proposed solution, finalized at the TEX Users Group meeting in Cork in September 1990, but rather to demonstrate how the layout can be implemented using POSTSCRIPT fonts (Donald Knuth and Tom Rokicki have already shown how to create POSTSCRIPT fonts in the original TEX layout, using virtual fonts, implemented in the latter's distributed `afm2tfm` program). To remind ourselves what we are aiming at, Fig. 1 shows the new layout using the Extended Computer Modern Roman font (`dcr10`, from the `dc` family created by Norbert Schwarz). Some characters needed extra work with METAFONT, but most were created from existing building blocks, and we shall follow this approach with the POSTSCRIPT fonts.

The set of procedures described below has only been made to work in a limited environment, but the principles can be used for most common `dvi`-to-POSTSCRIPT drivers, so long as they support virtual fonts. Our assumption in what follows is that we are using Rokicki's `afm2tfm` program (to convert Adobe font metrics to TEX font metrics) and the same author's `dvips` program, currently the only public domain `dvi` to POSTSCRIPT program to implement virtual fonts. The system has only been tested under Unix and OS/2, but should be easily portable, consisting of a program written in C, and the standard TEXware programs.

Examples are given in a Lucida Bright font.

2 Strategy

It may be necessary to remind readers that normal POSTSCRIPT fonts are essentially arranged as a set of procedures which use a lookup table of *names* for the characters; generally, fonts include a mapping between the names and numbers, but this is easily changed. Thus the character `exclamdown` (¡) is *normally* mapped to decimal 161, but it can always be called by name, or mapped to another number. The standard Adobe mapping is given in [1], and corresponds to no obvious standard; the layout is as shown in Fig. 2.

We can use straightforward methods to do the basic redefinition of characters so that they suit the extended layout. Firstly, we follow Rokicki's lead in `afm2tfm` and build up a virtual font which fools TEX into thinking that character X is at position A in the font, when it is in fact at position B; since we shall need virtual fonts anyway, this is an economic solution. Until version 7.0 of `afm2tfm`, this was made difficult by the fact that `afm2tfm` mapped undefined characters in an Adobe font more or less randomly; these are characters not assigned a font position in the Adobe Font Metric file (such as composite letter/accent combinations) which we want to use at specific positions. This required changing the AFM file (using a simple program to automate the process) so that characters were given the number we actually wanted—this forced `afm2tfm` to allocate them correctly. Thus the AFM entry

```
C -1 ; WX 444 ; N zcaron ; B 25 0 418 674 ;
```

was changed to

```
C 186 ; WX 444 ; N zcaron ; B 25 0 418 674 ;
```

This also needed a slightly revised version of the `afm2tfm` program itself, as it used to have a fixed table listing the first 128 characters in the font (in traditional TEX layout). I implemented this change by having `afm2tfm` read an encoding table from an external file.

With version 7.0 onwards of `afm2tfm`, Rokicki provided a more general mechanism for changing the encoding of a font at successive stages (at the level of the virtual font, and also at the level of the PostScript font itself). This has made the process described here rather easier.

When we have constructed a suitable AFM file, we can run `afm2tfm` with the option to generate a `.vpl` file; but this only rearranges the existing characters—what about the symbols which are not in the original font? These fall into four groups:

- New composite letter/accent combinations like 'S acute' (**Ś**) which can be generated using the available floating accents.
- Composite characters which can be 'fudged' using existing letters, such as the pseudo-character for capital **ß** (**SS**) or 'ij' (**ij**).
- New glyphs, such as the visible space character (which can be created using rules, as in this experiment) or 'dotless j' (**ȷ**).
- New ligatures not provided in POSTSCRIPT fonts, such as ffi (**ffi**).

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	`	´	ˆ	˜	¨	˝	˚	ˇ	"0x
'01x	˘	¯	˙	¸	˛	‚	‹	›	
'02x	“	”	„	«	»	–	—		"1x
'03x	₀	ı	ȷ	ﬀ	ﬁ	ﬂ	ﬃ	ﬄ	
'04x	␣	!	"	#	$	%	&	'	"2x
'05x	(	)	*	+	,	-	.	/	
'06x	0	1	2	3	4	5	6	7	"3x
'07x	8	9	:	;	<	=	>	?	
'10x	@	A	B	C	D	E	F	G	"4x
'11x	H	I	J	K	L	M	N	O	
'12x	P	Q	R	S	T	U	V	W	"5x
'13x	X	Y	Z	[	\	]	ˆ	_	
'14x	‘	a	b	c	d	e	f	g	"6x
'15x	h	i	j	k	l	m	n	o	
'16x	p	q	r	s	t	u	v	w	"7x
'17x	x	y	z	{	|	}	˜	-	
'20x	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ	"8x
'21x	Ĺ	Ľ	Ł	Ń	Ň	Ŋ	Ő	Ŕ	
'22x	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů	"9x
'23x	Ÿ	Ź	Ž	Ż	Ĳ	İ	đ	§	
'24x	ă	ą	ć	č	ď	ě	ę	ğ	"Ax
'25x	ĺ	ľ	ł	ń	ň	ŋ	ő	ŕ	
'26x	ř	ś	š	ş	ť	ţ	ű	ů	"Bx
'27x	ÿ	ź	ž	ż	ĳ	¡	¿	£	
'30x	À	Á	Â	Ã	Ä	Å	Æ	Ç	"Cx
'31x	È	É	Ê	Ë	Ì	Í	Î	Ï	
'32x	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	Œ	"Dx
'33x	Ø	Ù	Ú	Û	Ü	Ý	Þ	SS	
'34x	à	á	â	ã	ä	å	æ	ç	"Ex
'35x	è	é	ê	ë	ì	í	î	ï	
'36x	ð	ñ	ò	ó	ô	õ	ö	œ	"Fx
'37x	ø	ù	ú	û	ü	ý	þ	ß	
	"8	"9	"A	"B	"C	"D	"E	"F	

Figure 1: Extended TEX layout

	´0	´1	´2	´3	´4	´5	´6	´7	
´04x		!	"	#	$	%	&	’	″2x
´05x	(	)	*	+	,	-	.	/	
´06x	0	1	2	3	4	5	6	7	″3x
´07x	8	9	:	;	<	=	>	?	
´10x	@	A	B	C	D	E	F	G	″4x
´11x	H	I	J	K	L	M	N	O	
´12x	P	Q	R	S	T	U	V	W	″5x
´13x	X	Y	Z	[	\	]	^	_	
´14x	‘	a	b	c	d	e	f	g	″6x
´15x	h	i	j	k	l	m	n	o	
´16x	p	q	r	s	t	u	v	w	″7x
´17x	x	y	z	{	|	}	~		
´24x		¡	¢	£	⁄	¥	ƒ	§	″Ax
´25x	¤	'	“	«	‹	›	ﬁ	ﬂ	
´26x		–	†	‡	·		¶	•	″Bx
´27x	‚	„	”	»	…	‰		¿	
´30x		`	´	ˆ	˜	¯	˘	˙	″Cx
´31x	¨		˚	¸		˝	˛	ˇ	
´32x	—								″Dx
´33x									
´34x		Æ		ª					″Ex
´35x	Ł	Ø	Œ	º					
´36x		æ				ı			″Fx
´37x	ł	ø	œ	ß					
	″8	″9	″A	″B	″C	″D	″E	″F	

Figure 2: Unadulterated Adobe font layout (using AvantGarde)

```
(CHARACTER O 221 (comment Sacute)
   (CHARWD R 525.00)
   (CHARHT R 908.00)
   (CHARDP R 20.00)
   (MAP
      (SETCHAR C S)
      (MOVERIGHT R -429.00)
      (MOVEUP R 231.00)
      (SETCHAR O 1)
      )
   )
```

Figure 3: Virtual font file entry for S acute

```
(CHARACTER O 274 (comment ij)
   (CHARWD R 425.00)
   (CHARHT R 688.00)
   (CHARDP R 283.00)
   (MAP
      (SETCHAR C i)
      (MOVERIGHT R -100.00)
      (SETCHAR C j)
      )
   )
```

Figure 4: Virtual font file entry for ij

All of these need a character added to the virtual font; each virtual font character description consists of metrics for character width, height and depth (and italic correction where appropriate), and some instruction for what to do. These instructions usually involve setting one or more characters from base fonts, and inserting vertical or horizontal movement (which can, of course, be negative). Typical entries are shown in Figs. 3 and 4.

Making accent/letter combinations work involves setting a letter, moving back to the right, and then setting the accent. Assuming that accents go onto the centre of a letter, the movement to the right can be calculated automatically from the width of the letter and the accent. In some cases there must also be an upward movement to place an accent over, for instance, a capital letter. This upward movement would be a problem were it not for the fact that most fonts are consistently designed, and accents are already at the right height to sit over lower-case letters. The upward (or downward for sub-letter accents) movement can be observed in the AFM file which supplies model instructions for the creation of the normal composites; this example gives the right and upward movement of the acute over a capital E:

```
CC Eacute 2 ; PCC E 0 0 ; PCC acute 139 214 ;
```

Some cases will present special problems, such as 'l acute' (ĺ), but an algorithm can be developed for each of these characters which seems to be consistent across fonts; thus 'l acute' needs the accent raising more than normal. For those fonts which lack characters like Thorn (Þ) and eth (ð), these have to be fudged in a way copied from the original Cork demonstration chart.

To create completely new characters, we may need to resort to coding directly in PostScript; for this demonstration, I experimented with creating an extra font which contained just four new characters. Luckily, the code to do this had already been derived, and posted in a Usenet news group, by Amanda Walker. Greater patience would probably permit these characters to be generated entirely by `\special` commands in the virtual font, which would avoid the need for a font metric file for the tiny font. The code for creating the new font is given in Appendix B.

3 Usage

The approach outlined in the previous section resulted in two programs written in C, `vpltovpl` and `afm2afm`, and a slightly amended `afm2tfm` program. The latter two were made redundant by `afm2tfm` 7.0. Once the program `vpltovpl` has been compiled, the procedure for generating the final font is as follows, with examples shown in Unix syntax for a font 'Times-Roman' with an AFM file called `ptmr.afm`; we generate a new font called `ptmrq` to distinguish it from 'normal' fonts, utilising a 'raw' font called `ptmr0`.[1]

1. Run `afm2tfm` on the AFM file to generate a `tfm` file for an unmapped version of the font, and a `vpl` file describing the virtual font. The mapping is taken from an encoding file `ec.enc`, and the `-T` option means that the encoding is to be used both for the virtual font layout, and also as instructions to the PostScript interpreter to change the internal mapping of the font to make the normally inaccessible characters visible. The start of `ec.enc` is listed in Fig. 6.

   ```
   afm2tfm ptmr.afm -T ec.enc -v ptmrq.vpl \
         pmtr0.tfm
   ```

2. Run `vpltovpl` on the `vpl` file, also telling it the name of the `afm` file so that it can work out accent corrections etc.

   ```
   vpltovpl ptmrq.vpl ptmr.afm
   ```

[1] This uses the 'standard' names proposed by Karl Berry, suffixing them with a variant letter indicating an encoding.

This adds virtual character entries to the end of the `.vpl` file

3. We can now run the normal `vptovf` program which creates the `tfm` file which TeX will actually use, and the `vf` file which drivers will access.

   ```
   vptovf ptmrq.vpl ptmrq.vf ptmrq.tfm
   ```

4. We have to tell `dvips` about the fonts we have created, so that it can resolve references to, e.g., `ptmrq` in the virtual font. This is done by lines in the control file `psfonts.map`, which also instructs `dvips` to send the (same) encoding file `ec.enc` to the PostScript interpreter to set up the font correctly. Some fonts may also need downloading. Some examples lines are given in Figure 5.

   If we use the public domain `ps2pk` program (created by Piet Tutelaars using IBM's library for rendering PostScript Type1 fonts), we could instead generate TeX `.pk` fonts using exactly the same encoding file.

The result is a `tfm` file for TeX (`ptmrq.tfm`), and a `vf` file for the driver (`ptmrq.vf`), which makes references to the 'raw' font `ptmr0`.

Before we can go ahead and use the font in TeX, there is one more job: rebuilding some of (LA)TeX's standard macros to do with accents and special characters which have hard-wired character positions hardwired. A sample set of code is given in Appendix A.

4 Using the result

The final product of the work is no more than a font table; Fig. 7 shows the result using Times Roman. The 'visible space' (which was absent in earlier versions on this system) has been created with SETRULE commands in the virtual font. In most of the modern Adobe fonts, the more awkward glyphs already exist, but some of the characters are rather badly fudged, as a comparison with the DC chart shows. A run with LucidaBright derived small caps (Fig. 8) shows that the procedure works with small caps as well as with normal roman.

5 Remaining problems

There are problems left for the TeX community to solve in their relationship with PostScript, even if we agree on the suggested layout — and it must be said that there are many people unhappy with the proposal. Thus the 'visible space' character is an extremely specialized brute, which, in the opinion of the author, has no place in a normal text font, and should be banished to a symbol font.

1. There remain some characters which are inadequately defined:
   (a) The 'Eng' (**Ŋ**) and 'eng' (**ŋ**) characters need total reworking; their creation from 'r' and 'j' is ridiculous; the original Cork table used this temporarily to show what was intended, but the combination was not designed to be useable.
   (b) The 'd bar' (**đ**) character is difficult because of the very short length of the ascender of the 'd' in some styles, such as Times Roman. In a more leggy font like Palatino (đ), it fits quite well. Someone more versed in font design than the writer should consider this.
   (c) The latter caveat applies even more strongly to the 'ij' characters (**ij** and **IJ**)
2. Not all Adobe fonts have the same characters; this may cause a problem in the future, though it is rather more likely that fonts will *increase* their ranges and acquire, for instance, an ffi ligature (ffi) or a Thorn (**Þ**). Rather more serious is the fact that the thousands of copies of, say, Palatino already in use are different versions, and symbols available on one printer's copy may be absent on another. 'y acute' (**ý**) is a good example of this, not being present in older copies of Adobe TimesRoman.
3. Some of the symbols that we are used to in TeX (such as † and ‡) have been squeezed out of the layout. We need to agree on a useful set of symbols as a companion to the extended TeX layout.
4. Lastly, is this the right approach? It might be more effective to do all the calculation of new composite characters in PostScript itself, so that no virtual font was needed at all. We must bear in mind, however, that at some point a font metric file must be created for TeX (possibly generated automatically from information in the PostScript font dictionary, which is relatively easy).

Some of these problems are soluble with a little patience; others need some careful expert advice; yet others may need manual intervention for each font. In the worst case, a font design program may have to be brought in to create the missing symbols.

6 Conclusions

The techniques presented in this article illustrate the skeleton of methods whereby fairly wide-ranging changes can be made to the effect of a PostScript font, beyond simple remapping. The same technique may be appropriate to create ISO Latin-1 layouts, for instance.

Readers who use their PostScript fonts with other software (such as Adobe Type Manager) may

```
pplr0 Palatino-Roman " ECEncoding ReEncodeFont " <ec.enc
ptmro0 Times-Roman   " ECEncoding ReEncodeFont " <ec.enc   ".167 SlantFont"
plcb0 Lucida-Bold <plcb.pfb " ECEncoding ReEncodeFont " <ec.enc
hlcdi40 LucidaFax-DemiItalic  " ECEncoding ReEncodeFont " <ec.enc <hlcdi4.pfb
```

Figure 5: Example additions to dvips file `psfonts.map`

```
%
/ECEncoding [           % now 256 chars follow
% 0x00
  /grave /acute /circumflex /tilde /dieresis /hungarumlaut /ring /caron
  /breve /macron /dotaccent /cedilla
  /ogonek /quotesinglbase /guilsinglleft /guilsinglright
% 0x10
  /quotedblleft /quotedblright /quotedblbase /guillemotleft
  /guillemotright /endash /emdash /compoundwordmark
  /perthousand /dotlessi /dotlessj /ff /fi /fl /ffi /ffl
% 0x20
  /visiblespace /exclam /quotedbl /numbersign
  /dollar /percent /ampersand /quoteright
  /parenleft /parenright /asterisk /plus /comma /hyphen /period /slash
```

Figure 6: Start of encoding file for `afm2tfm`

prefer to eschew this whole approach in favour of manipulating a copy of the Type1 font itself. Commercial programs are available which allow you not only to change the encoding of a font, but also to add new ligatures and composite characters in the same way as our virtual fonts do. This would provide PostScript fonts which can be used in any environment directly, and by TeX without virtual fonts.

The programs and header files used to create the examples in this article are available by electronic mail from the author, or from the UK TeX archive at Aston (`ftp.tex.ac.uk`). As this work is extremely derivative, the writer is very grateful to Norbert Schwarz (co-ordinator of the Cork table, and author of the dc fonts), Tom Rokicki and Donald Knuth (whose `afm2tfm` supplied the basis), Amanda Walker (who showed how to create missing characters), Alexander Samarin (some additions to `vpltovpl.c`), Karl Berry, and Berthold Horn (discussions of font encoding problems).

References

[1] Adobe Systems Incorporated 1985 PostScript *Language Reference Manual*, Addison-Wesley, Reading, Massachusetts

[2] Adobe Systems Incorporated 1987 PostScript *Language Tutorial and Cookbook*, Addison-Wesley, Reading, Massachusetts

[3] Bień, J. 1990. 'On standards for computer modern font extensions' Tugboat 11, no. 2, pp. 175–183

[4] Beebe, N. 1990. 'Character set encoding', Tugboat 11, no. 2, pp. 171–174

[5] Knuth, D. 1990. 'Virtual fonts: more fun for grand wizards' Tugboat 11, no. 1, pp. 13–23

◇ Sebastian Rahtz
ArchaeoInformatica
12 Cygnet Street
York Y01 2JR
U.K.
`spqr@minster.york.ac.uk`

	´0	´1	´2	´3	´4	´5	´6	´7	
´00x	`	´	ˆ	˜	¨	˝	˚	ˇ	″0x
´01x	˘	¯	˙	¸	˛	‚	‹	›	
´02x	“	”	„	«	»	–	—		″1x
´03x	‰	ı			fi	fl			
´04x	␣	!	"	#	$	%	&	’	″2x
´05x	(	)	*	+	,	-	.	/	
´06x	0	1	2	3	4	5	6	7	″3x
´07x	8	9	:	;	<	=	>	?	
´10x	@	A	B	C	D	E	F	G	″4x
´11x	H	I	J	K	L	M	N	O	
´12x	P	Q	R	S	T	U	V	W	″5x
´13x	X	Y	Z	[	\	]	^	_	
´14x	‘	a	b	c	d	e	f	g	″6x
´15x	h	i	j	k	l	m	n	o	
´16x	p	q	r	s	t	u	v	w	″7x
´17x	x	y	z	{	|	}	~	-	
´20x	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ	″8x
´21x	Ĺ	L’	Ł	Ń	Ň	r	Ő	Ŕ	
´22x	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů	″9x
´23x	Ÿ	Ź	Ž	Ż	IJ	İ	đ	§	
´24x	ă	ą	ć	č	d’	ě	ę	ğ	″Ax
´25x	ĺ	l’	ł	ń	ň	r	ő	ŕ	
´26x	ř	ś	š	ş	t’	ţ	ű	ů	″Bx
´27x	ÿ	ź	ž	ż	ij	¡	¿	£	
´30x	À	Á	Â	Ã	Ä	Å	Æ	Ç	″Cx
´31x	È	É	Ê	Ë	Ì	Í	Î	Ï	
´32x	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	Œ	″Dx
´33x	Ø	Ù	Ú	Û	Ü	Ý	þ	SS	
´34x	à	á	â	ã	ä	å	æ	ç	″Ex
´35x	è	é	ê	ë	ì	í	î	ï	
´36x	σ^{X}	ñ	ò	ó	ô	õ	ö	œ	″Fx
´37x	ø	ù	ú	û	ü	ý	þ	ß	
	″8	″9	″A	″B	″C	″D	″E	″F	

Figure 7: Extended TEX layout in Times Roman

	´0	´1	´2	´3	´4	´5	´6	´7	
´00x	`	´	ˆ	˜	¨	˝	˚	ˇ	″0x
´01x	˘	¯	˙	¸	˛	‚	‹	›	
´02x	“	”	„	«	»	–	—		″1x
´03x	‰	I	J	ff	fi	fl	ffi	ffl	
´04x	␣	!	"	#	$	%	&	’	″2x
´05x	(	)	*	+	,	-	.	/	
´06x	0	1	2	3	4	5	6	7	″3x
´07x	8	9	:	;	<	=	>	?	
´10x	@	A	B	C	D	E	F	G	″4x
´11x	H	I	J	K	L	M	N	O	
´12x	P	Q	R	S	T	U	V	W	″5x
´13x	X	Y	Z	[	\	]	^	_	
´14x	‘	A	B	C	D	E	F	G	″6x
´15x	H	I	J	K	L	M	N	O	
´16x	P	Q	R	S	T	U	V	W	″7x
´17x	X	Y	Z	{	|	}	~	-	
´20x	Ă	Ą	Ć	Č	Ď	Ě	Ę	Ğ	″8x
´21x	Ĺ	Ľ	Ł	Ń	Ň	ŋ	Ő	Ŕ	
´22x	Ř	Ś	Š	Ş	Ť	Ţ	Ű	Ů	″9x
´23x	Ÿ	Ź	Ž	Ż	Ĳ	İ	đ	§	
´24x	ă	ą	ć	č	ď	ě	ę	ğ	″Ax
´25x	ĺ	ľ	Ł	ń	ň	ŋ	ő	ŕ	
´26x	ř	ś	Š	ş	ť	ţ	ű	ů	″Bx
´27x	Ÿ	ź	Ž	ż	ĳ	¡	¿	£	
´30x	À	Á	Â	Ã	Ä	Å	Æ	Ç	″Cx
´31x	È	É	Ê	Ë	Ì	Í	Î	Ï	
´32x	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	Œ	″Dx
´33x	Ø	Ù	Ú	Û	Ü	Ý	Þ	SS	
´34x	À	Á	Â	Ã	Ä	Å	Æ	Ç	″Ex
´35x	È	É	Ê	Ë	Ì	Í	Î	Ï	
´36x	Đ	Ñ	Ò	Ó	Ô	Õ	Ö	Œ	″Fx
´37x	Ø	Ù	Ú	Û	Ü	Ý	Þ	SS	
	″8	″9	″A	″B	″C	″D	″E	″F	

Figure 8: Extended TEX layout in Lucida Bright derived small caps

A LaTeX macros to set up accented characters and ligatures

Practical use of the extended layout requires that we redefine some TeX or LaTeX macros which deal with accented characters. This is addressed perhaps more thoroughly in support files distributed with the New Font Selection Scheme for LaTeX, but a skeleton code is presented below:

```
%----------------------------------------------------
% ecacc.tex
%
% set up composite characters and accents
% assuming TeX Extended Layout (Cork September 1990)
%
% many combinations are included as ligatures
% in the virtual font
%
% Sebastian Rahtz October 10th 1990; August 20th 1992; Sep 9th 1992
%----------------------------------------------------
% some obvious ASCII characters used for other purposes
\chardef\%=`\%
\chardef\&=`\&
\chardef\#=`\#
\chardef\$=`\$
% a group of common extended characters
% this could probably be usefully expanded
\chardef\ss='377%  germandbls
\chardef\ae='346%  ae
\chardef\oe='367%  oe
\chardef\o='370%   oring
\chardef\AE='306%  AE
\chardef\OE='327%  OE
\chardef\O='330%   Oslash
\chardef\i='031%   dotless i
\chardef\j='032%   dotless j
\chardef\aa='345%  aring
\chardef\AA='305%  Aring
\chardef\l='252%   lslash
\chardef\L='212%   Lslash
%
%\def\_{\leavevmode\kern.06em\vbox{\hrule width.3em}}
\chardef\_=`\_
% some miscellaneous symbols
\chardef\pounds='277%
\chardef\guillemotleft='023%
\chardef\guillemotright='024%
\chardef\guilsinglleft='016%
\chardef\guilsinglright='017%
\chardef\quotesinglbase='015%
\chardef\quotedblbase='022%
\chardef\S='237%                  section mark
%
% but these are not in the new font:
% need agreement on a symbol font layout
% to include all this sort of thing
%\def\dag{{\symfont \char'207}}%     dagger
%\def\ddag{{\symfont \char'210}}%    doubledagger
%\def\P{{symfont\char'266}}%         paragraph mark
%
\ifx\protect\undefined\let\protect=\relax\fi
\def\pd#1{\oalign{#1\crcr\hidewidth.\hidewidth}}
\def\d{\protect\pd}%              dotunder accent
\def\pb#1{\oalign{#1\crcr\hidewidth
   \vbox to.2ex{\hbox{\char'055}\vss}\hidewidth}}
```

```
\def\b{\protect\pb}%              barunder accent
%
% accents for manual combination
\def\pc#1{\setbox\z@\hbox{#1}\ifdim\ht\z@=1ex\accent'013#1%
 \else{\ooalign{\hidewidth\char'013\hidewidth\crcr\unhbox\z@}}\fi}
\def\c{\protect\pc}%              cedilla
\def\`#1{{\accent'001 #1}}%           grave
\def\'#1{{\accent'000 #1}}%           acute
\def\v#1{{\accent'007 #1}}\let\^^_=\v%    hacek
\def\u#1{{\accent'010 #1}}\let\^^S=\u%    breve
\def\^#1{{\accent'002 #1}}\let\^^D=\^%    circumflex
\def\.#1{{\accent'012 #1}}%           dotaccent
\def\H#1{{\accent'005 #1}}%           hungarumlaut
\def\~#1{{\accent'003 #1}}%           tilde
\def\"#1{{\accent'004 #1}}%           dieresis
\def\=#1{{\accent'011 #1}}%           macron
%
% how do you specify ogonek ?
%
\def\acute{\mathaccent"7001 }   % acute
\def\grave{\mathaccent"7000 }   % grave
\def\ddot{\mathaccent"7004 }    % dieresis
\def\tilde{\mathaccent"7003 }   % tilde
\def\bar{\mathaccent"7009 }     % macron
\def\breve{\mathaccent"7008 }   % breve
\def\check{\mathaccent"7007 }   % caron
\def\hat{\mathaccent"7002 }     % circumflex
\def\dot{\mathaccent"700A }     % dotaccent

% upper and lowercase codes
\lccode223=255 % Germandbls
\uccode223=223
\uccode255=223
\lccode255=255
%  etc etc; rest omitted
```

B Creating new characters in PostScript

When I originally worked to produce EC-style PostScript fonts, I created a tiny four-character font which contained the characters commonly missing from PostScript fonts. This approach was clumsy, and in daily use of the fonts, I have simply ignored the missing characters, or used a font (like Lucida Bright) which had a full set of glyphs.

The code for deriving the characters is listed here out of interest:

```
%%BeginDocument: texchars.pro
%%
%% create a small new font with some extra characters
%% S Rahtz December 1990; stolen from Amanda Walker's font for TeX
%%
/TeXZZencoding 256 array def
0 1 255  { TeXZZencoding exch /.notdef put } for
TeXZZencoding dup 0
/ff put dup 1 /ffi put dup 2 /ffl put dup 3 /dotlessj put pop
/MakeTeXChars {
20 dict begin
        /FontType 3 def
        /FontMatrix [.001 0 0 .001 0 0] def
        /FontBBox [0 0 1000 1000] def
        /FontName exch def
        /BFont exch def
        /BaseFonts [ BFont findfont 1000 scalefont ] def
        /Encoding TeXZZencoding def
        /String 1 string def
        /CharProcs 5 dict def
        CharProcs begin
                /.notdef { } def
                /ff {
                        (ff) stringwidth exch 50 sub exch
                        0 0 moveto (ff) false charpath flattenpath pathbbox
                        exch 50 sub exch
                        6 copy setcachedevice
                        0 0 moveto (f) show -50 0 rmoveto (f) show
                } def
                /ffi {
                        (f\256) stringwidth exch 50 sub exch
                        0 0 moveto (f\256) false charpath flattenpath pathbbox
                        exch 50 sub exch
                        6 copy setcachedevice
                        0 0 moveto (f) show -50 0 rmoveto (\256) show
                } def
                /ffl {
                        (f\257) stringwidth exch 50 sub exch
                        0 0 moveto (f\257) false charpath flattenpath pathbbox
                        exch 50 sub exch
                        6 copy setcachedevice
                        0 0 moveto (f) show -50 0 rmoveto (\257) show
                } def
                /dotlessj {
                        (j) stringwidth
                        0 0 moveto (j) false charpath flattenpath pathbbox
                        6 copy setcachedevice newpath
                        0 0 moveto (\365) false charpath flattenpath pathbbox
                        newpath -1000 -1000 moveto
                        dup -1000 exch lineto
                        1000 exch lineto
                        1000 -1000 lineto
                        closepath clip
                        pop pop pop
                        0 0 moveto (j) show
                } def
        end
        /BuildChar {
                exch begin
                BaseFonts 0 get  setfont
                dup Encoding exch get
                dup CharProcs exch known
                        { CharProcs exch get exch pop exec }
                        { pop String exch 0 exch put
                          String stringwidth
                          newpath 0 0 moveto String false charpath
                          flattenpath pathbbox
                          setcachedevice
                          0 0 moveto
                          String show
                        } ifelse
                end
        } def
        currentdict
end
dup /FontName get exch definefont pop
} def
%%EndDocument
```

Zebrackets: A Pseudo-Dynamic Contextually Adaptive Font

Michael Cohen

Abstract

A system is introduced that increases the information density of textual presentation by reconsidering text as pictures, expanding the range of written expression. Indicating nested associativity with stripes, *Zebrackets* uses small-scale horizontal striations, superimposed on parenthetical delimiters. This system is implemented as an active filter that re-presents textual information graphically, using adaptive pseudo-dynamic character generation to reflect a context that can be as wide as the document.

0 Introduction

To represent parenthetical expressions, traditionally typewritten documents use parentheses, “( )”, for first-level subphrases, extended by (square) brackets, “[ ]”, for doubly nested phrases (parentheses within parentheses), and alternating the two sets of delimiters for the rare more deeply nested phrases. Parentheses and brackets are overloaded; they are used in prose for delimiting subordinate expressions, appositives, citations and cross-references, and in mathematical formulae and computer programs for associative precedence, array subscripts, numeric ranges, function parameters and arguments, as well as special interpretations (like “$((n))_m$” denoting “n mod[ulo] m”). Editors also use brackets to set off editorial substitution and interpolation (“[sic]”), ellipsis (“[. . .]”), elision (“[expletives deleted]”), etymology (“[MF *braguette* codpiece, fr. dim. of *brague* breeches, fr. OProv *braga*, fr. L *braca*, fr. Gaulish *brāca*, of Gmc origin; akin to OHG *bruoh* breeches – more at BREECH]”), etc. For literature and journalism, these conventions have been adequate, since the reader could usually parse the subphrases. Extended schemes have used (curly or set) braces (a.k.a. “bracelets”), “{}”, and angle brackets (a.k.a. “inequality signs”), “<>”, to indicate more deeply nested phrases.[1] But especially for non-natural languages, in which stacks of association are not only comprehensible but necessary and encouraged, a more extensible scheme is needed.

Using size to denote nesting (or any other kind of) level doesn't work, since juxtaposed expressions, at the same parenthetical depth, might require different sized delimiters for aesthetic (lexicographical) reasons. “(A * (B + C))” suffices, but “($\left(\begin{matrix} w & x \\ y & z \end{matrix}\right)$ * (B + C))” starts to degrade. Likewise, $\overbrace{\text{over}}$ / $\underbrace{\text{under}}$ –$\overline{\text{line}}$/ $\overbrace{\underbrace{\text{brace}}}$ schemes break down across line boundaries.

Bit-mapped terminals and high-resolution printers suggest the possibility of more elaborate presentations [Rub88] which exploit underutilized human visual acuity. Figure 1 shows some simple axes of variation under LaTeX/TeX [Lam86, Knu84]. Combining a computer filter (to analyze the text and automatically prepare appropriate semi-custom fonts) with an extra coding dimension orthogonal to standard display techniques, we can add more content to the ordinary printed display.

Zebrackets[2] [Coh92a, Coh92b] extends parentheses and square brackets by striping them systematically according to an index reflecting their order in the text. Each index of the respective delimiter pairs is cast into a binary pattern, which is superimposed on the characters as striations. The striping is adaptively chosen, so that the complexity of the parenthesized expression determines the spacing of the striations. Informal experiments suggest that users tend to look only for matching delimiters, but alternate encoding schemes are also possible. Some of these special-purpose modes are outlined later in the Discussion.

1 Examples

1.1 Chemical Compound

Figure 2 shows the the application of *Zebrackets* to the chemical formula for a popular roach control system, as shown on its box. What is the “matching bookend” to the parenthesis before “3-” in the second line? A quick scan of the zebracketed version finds the parenthesis closing the “(3-” association.

[1] Note that the open and close quotation marks are themselves, like yin and yang, pairwise delimiters.

[2] This paper uses the following orthographic conventions: the name of the described system is italicized; computer commands are in typewriter style; and most everything else (except for the bibliography, which has its own conventions) is in roman type style. Therefore, “*Zebrackets*” refers to the system described here meant to elegantly display nested associations, “`zebrackets`” to the eponymous computer filter for invoking this system, and “zebracketed” to the corresponding effect.

- ([{<delimiter shape>}])
- indentation
  - outlines
- $\boxed{\boxed{\boxed{\boxed{\text{frames}}}}}$
- (((((((parenthesis (or type)) size)))))))
- color, greyscale, or shading
- dynamic effects
  - momentary highlight (during editing)
  - synchronous flashing
- $\overset{1}{(}$explicitly$\overset{1}{)}$ tagged delimiters
- $\overline{\overline{\overline{\text{over}}\text{lines}}\text{ and }\underline{\underline{\text{under}}\text{lines}}}$
- $\overbrace{\overbrace{\text{(superscripted) }\overbrace{\overbrace{\text{overbraces}}}}^{1}\text{ and/or }\underbrace{\text{(subscripted) }\underbrace{\text{underbraces}}}_{2}}$
- typeface **weight**
- type style
  - serifs or sans serif
  - spacing: proportional (variable-spaced) or `fixed-pitch (monospaced)`
  - obliqueness: roman, *slanted*, or *italicized*

Figure 1: Some traditional and non-traditional ways of typographically indicating nested associativity

1.2 LISP

The LISP programming language relies on parentheses for delimiting lists,[3] the language's basic data structure. This LISP function performs a generalized "inclusive or":

```
(DEFUN ANY (LST)
 (COND ((NULL LST) NIL)
       ((CAR LST) T)
       (T (ANY (CDR LST))) ) )
```

Here is the same code, but written with zebrackets, which elucidates the associations:

```
(DEFUN ANY (LST)
 (COND ((NULL LST) NIL)
       ((CAR LST) T)
       (T (ANY (CDR LST))) ) )
```

[3] The name "LISP" is acronymic for "**l**ist **p**rocessing language", but has been jokingly expanded as "**l**ots of **i**ncessantly **s**illy **p**arentheses".

1.3 Objective-C

Figure 3 shows a line from an Objective-C program, with and without *Zebrackets*.

1.4 Logic

Figure 4 shows the first order predicate calculus notation indicating that symmetry and transitivity imply reflexivity. *Zebrackets* illuminates the precedence.

1.5 Association Beyond Single Parenthetical Pairs

Just as pairs of identically valued parentheses point at each other, like matching bookends, so do multiple sets of same-valued delimiters associate beyond the scope of a single pair of parentheses. Here the zebrackets are not just reinforcing patterns already present, but are adding new information, distributed across the text:

> An angle bracket ">" ("<") may be used to write (read) Unix file(s) to (from) standard output (input).

ACTIVE INGREDIENT: Hydramethylnon [tetrahydro-5, 5-dimethyl-2(1$\underline{H}$)-pyrimidinone(3-[4-(trifluoromethyl)phenyl]-1-(2-[4-(trifluoromethyl) phenyl]ethenyl)-2-propenylidene)hydrazone]

ACTIVE INGREDIENT: Hydramethylnon [tetrahydro-5, 5-dimethyl-2(1$\underline{H}$)-pyrimidinone(3-[4-(trifluoromethyl)phenyl]-1-(2-[4-(trifluoromethyl) phenyl]ethenyl)-2-propenylidene)hydrazone]

Figure 2: Chemical Compound, before and after *Zebrackets*

```
[inspectorPanel setAccessoryView:[[[[accessory contentView] subviews]
                                   objectAt:0] removeFromSuperview]];

[inspectorPanel setAccessoryView:[[[[accessory contentView] subviews]
                                   objectAt:0] removeFromSuperview]];
```

Figure 3: Objective-C code, before and after *Zebrackets*

∀P ∀v ∀w ∀x ∀y ∀z ((P(v,w) ⇒ P(w,v)) ∧ ((P(x,y) ∧ P(y,z)) ⇒ P(x,z)) ⇒ P(u,u))

∀P ∀v ∀w ∀x ∀y ∀z ((P(v,w) ⇒ P(w,v)) ∧ ((P(x,y) ∧ P(y,z)) ⇒ P(x,z)) ⇒ P(u,u))

Figure 4: First order predicate calculus, before and after *Zebrackets*

and

$$(\ln|(\ln|(x^2-2)|)|)' = (\ln|(x^2-2)|)^{-1}\ (x^2-2)^{-1}\ (x^2-2)'$$

The visual coupling of related expressions is also useful when parenthetical expressions cross, violating LIFO (last in ⇒ first out) protocol and subverting the "most recent unmatched" association. One domain in which this happens is when part of a character string's postfix is the same as another's prefix. (This recalls the [American TV show] Wheel of Fortune's "Before and After" category, or Emacs' "Dissociated Press" function [Sta86].)

⊙ associativity: A ⊙ B ⊙ C = (A ⊙ (B) ⊙ C)

(reference (manual) labor)

(Ze(bra)ckets)

(FM/(AM)/PM)

2 Implementation

Zebrackets glyphs are implemented as extensions of fonts in the Computer Modern typeface. The implementation of *Zebrackets* comprises two aspects: a filter to generate permuted invocations of the underlying parentheses and brackets, and the delimiter fonts themselves. The two-pass filter parses selected text. The first pass establishes the maximum number of stripes needed, and generates the necessary METAFONT [Knu86] files. (For the simplest mode, in which each pair of delimiters is assigned a unique index, the maximum number of stripes is the number of bits needed to represent its highest index, $\lceil \lg_2 |\text{delimiter pairs}| \rceil$.) Using the context established by the first pass, the second pass replaces each parenthesis or bracket with the LaTeX code invoking its respective zebracketed version.

For example, running the `zebrackets` filter on "`(a * (b + c))`" determines that only one potential stripe is needed, replaces the source text with "`{\pmcone \symbol{0}}a * {\pmcone \symbol{1}}b + c{\pmcone \symbol{3}}{\pmcone \symbol{2}}`", and generates the `pmcone12.mf` file, which together yield "(a * (b + c))" at image time.

By having indirected the glyphs one extra level, *Zebrackets* implements a meta-METAFONT. Dynamic fonts exploit what is sometimes called "dynamic programming", which is basically lazy evaluation of a potentially sparse domain. *Zebrackets* is implemented as a precompiler, like a macro processor, that replaces vanilla parentheses with zebracketed and emits METAFONT source that will be invoked at preview (`TeXview` on the NeXT) or printing (`dvips`) time. Since each character is determined at edit-time, this implementation is context-sensitive and adaptive, but not purely dynamic [AB89]. Because the locality disambiguates *Zebrackets*' indices, the scope of a zebracketed expression is typically a single formula or expression. Further, since each LaTeX `\newfont` must be declared at most once, it is easiest to simply include a list of all the potential *Zebrackets* fonts at the top of each file. Therefore I have termed *Zebrackets* 'pseudo-dynamic', since the automatic adaptive character specification can be conceptually lumped together with the compilation (via `latex`) and imaging, but the actual specification usually involves human-specified ranges for `zebracketsing`.

Zebrackets adopts a minimalist "less (ink) is more (data)" philosophy [Tuf83], adding information to regular parentheses and brackets by resetting some pixels in the characters. Because of the tendency of white features to bleed out into a black background, the striations need not be terribly thick to be perceivable. In the examples shown, the stripes are 1 pt. ($\frac{1}{72.27}$ inches) thick, subsuming about 2.6 arc-minutes ($\approx$ 0.044 degrees $\approx$ 0.77 milliradians) of visual angle (assuming a reading distance of 18 inches). This is greater than the accepted (if heuristic) industrial minimum for visual acuity, 1 arc-minute.

Since there are usually more ascending than descending characters, readers (of English) tend to look slightly above the line of print, deriving meaning from the "top coastline" (upper half) of the text. Therefore, the index of the parenthetical pair is encoded with the LSB (least significant bit) at the top of the parenthesis or bracket. *Zebrackets* are generated singly, although they occur pairwise in well-formed expressions. (When are parenthetical delimiters unmatched? In expressions like 'electroglyphs' or ASCII 'smilies' [Tem89] that suggest a rotated human face `:-)`.) For both square and curved delimiters, the stripes are drawn horizontally (rather than radially), so that the reader might imagine an invisible line drawn through the intermediate text. Since the delimiter pairs are symmetrical [Lan92], they can be conceptually associated, and since the bands are aligning, they can also be visually associated, like toothpicks holding bread around a thick sandwich.

3 Discussion

For single levels of nesting, the extended parentheses and brackets are identical to the unenhanced, since an index of zero leaves the delimiters unbanded in *Zebrackets*' default positively-coded scheme, chosen to preserve the backwards compatibility of the curve substrate.

For deeper levels, *Zebrackets* tries to maintain this backwards compatibility by being non-distracting to users who don't know about it. It is, by design, "just noticeable", right over the limen, or edge of perceptibility. The added information is meant to be clear to the user actively seeking it, and transparent to any user not actively seeking it. (Such an effect is like making something a little smaller to call attention to it.) By designing a scheme that is both noticeable and ignorable, one is obtained that is unambiguous but unobtrusive, unmistakable but unassuming.

In practice, however, *Zebrackets* has some problems: For linear reading without searching, zebrackets can actually be distracting, introducing high-frequency noise (that looks like printer spotting). On the other hand, zebracket striations can be difficult to see, especially for readers with less sharp eyes, and zebracketed documents are not robust under (perhaps repeated) copying by low-resolution devices (like most faxes). The useful limit of the current system seems to be around six (allowing $2^6 = 64$ different versions of each pair of delimiters). For overly rich expressions that exceed visual acuity, *Zebrackets* can be limited to a fixed stripe depth, wrapping around (repeating) the indexing scheme if the delimiters exhaust the range of uniquely encodable depths.

Variations that overcome these limitations are possible. For instance, grayscale striations (not yet implemented) might disappear at normal reading speed, but be visible when doing a detailed search. Alternatively, using black stripes to tick the parentheses, instead of dropping out (white) segments, is more legible, if less inconspicuous:

```
(DEFUN ANY (LST)
 (COND ((NULL LST) NIL)
       ((CAR LST) T)
       (T (ANY (CDR LST))) ) )
```

Further, invocation of different encoding modes and graphical rhythms, perhaps for special purposes, is straight-forward. Indexing the streaks "inside-out", (so that the outer delimiters pairs have a higher index (than inner)), orders the evaluation of expression trees. Displaying breadth (or depth (or both)), instead of an incremental index, is useful for visualizing expression complexity. Striping the delimiters "upside-down", from the bottom, might be better for languages (like Hebrew) that carry more information in the "lower coastline".

4 Conclusion

These utilities recall the counter-intuitive advice "To clarify, add detail" [Tuf90]. *Zebrackets* is a tool in a suite of prettyprinters that start to treat characters as pictures, with attendant increases in information/ink value: denser data without loss of legibility, intensifying without interfering.

Documents should look like what they mean: context after content, form after function, process after product, and style after substance. Creative orthography frees words from traditional (technologically imposed) constraints, allowing textual representation of multidimensional concepts by projecting multilayered structure into linear text. Extended electronic typography provides additional parsing cues and differentiates between heretofore duplicate symbols, stretching existent presentation styles without breaking into new modalities.

The handwritten "publishing" of pre-Gutenberg scribes was arbitrarily subtle, with its attendant human caprice (and mistakes). Printing can be thought of as having rigidified information transmission. The research described here loosens some of that determinism, not by randomizing the presented information, but by softening the digitized boundaries, thereby expanding the range of expression.

The notion of a fixed alphabet font is inherently limited, even one extended into a family by techniques like weighting, italicization, emboldening, and local contextual tools like ligature and kerning. Computers offer the potential of arbitrarily adaptable glyphs, permuted in subtle if significant ways, depending on their global context, to heighten legibility (readability, balance, or proportion) or evoke emotions that reinforce or complement the words and ideas.

Zebrackets' integral characterization of the text yields a discrete specification of a font; real numbers would allow for continuous variation, expanding even further the ability to custom-tailor a font for a context. Such variety might manifest as arbitrarily soft typefaces, perhaps employing greyscale or dynamic effects, or tuned by the user, to match visual acuity. Contextual fonts like *Zebrackets* indicate evolving modes of written representation, algorithmic descriptions driving adaptive displays, as style catches up to technology.

◇ Michael Cohen
The University of Aizu
Aza Kami-iawase 90, Oaza
Tsuruga
Ikki-machi, Aizu-Wakamatsu-shi
Fukushima-ken 965
Japan
`mcohen@u-aizu.ac.jp`

References

[AB89] Jacques André and Bruno Borghi. Dynamic fonts. In *Proc. Int. Conf. on Raster Imaging and Digital Typography*, pages 198–204, Lausanne, Switzerland, October 1989.

[Coh92a] Michael Cohen. Blush and Zebrackets: Large- and Small-Scale Typographical Representation of Nested Associativity. In *Proc.* IEEE *Workshop on Visual Languages*, pages 264–266, Seattle, September 1992.

[Coh92b] Michael Cohen. Large- and Small-Scale Typographical Representation of Nested Associativity. *Visible Language*, 23(3/4), Summer/Autumn 1992.

[Knu84] Donald E. Knuth. *The TEXbook*. Addison-Wesley, 1984. 0-201-13448-9.

[Knu86] Donald E. Knuth. *The METAFONTbook*. Addison-Wesley, 1986. 0-201-13444-6.

[Lam86] Leslie Lamport. *LATEX: A Document Preparation System.* Addison-Wesley, 1986. 0-201-15790-X.

[Lan92] John Langdon. *Wordplay: ambigrams & reflections on the art of ambigrams.* Harcourt Brace Jovanovich, 1992. 0-15-198454-9.

[Rub88] Richard Rubinstein. *Digital Typography: An Introduction to Type and Composition for Computer System Design.* Addison-Wesley, 1988. 0-201-17633-5.

[Sta86] Richard M. Stallman. *GNU Emacs Manual.* Free Software Foundation, 1000 Mass. Av.; Cambridge, MA 02138, fifth edition, December 1986.

[Tem89] Brad Templeton, editor. `Rec.Humor.Funny/TeleJoke` *Computer Network Humour Annual.* ClareNet Communications Corp., 1989.

[Tuf83] Edward R. Tufte. *The Visual Display of Quantitative Information.* Graphics Press, 1983.

[Tuf90] Edward R. Tufte. *Envisioning Information.* Graphics Press, 1990.

Graphics

From Observation to Publication

Theo A. Jurriens

Abstract

This article describes the use of TEX in publishing observations of variable stars observed by Dutch amateur-astronomers. The observations are published in the journal "Variabilia" and in the so-called Reports. In the latter the observations, collected in several years, are published and submitted to the professional astronomer. It includes tables and light-curves: plots of the changing magnitudes of stars versus time. In creating the light-curves, PICTEX is used. In preparing the files for PICTEX, simple TEX-coding is used for manipulating the data.

1 The Data

Each observation is characterized by the observed star, a date, the brightness of the star and a code representing the observer. To avoid calender problems the so-called Julian Date is used. For example the Julian Date of 1961, September 9, my date of birth, is equal to 2437552 .

Each observation is TEX-coded as follows:

```
\obs #1.#2.#3.#4.#5.
```

#1 is the whole part (i.e. integer part) of the Julian Date, #2 its fractional part. #3 is the integer part of the magnitude, #4 its fractional part, and #5 is the observer-code. (The last is not used for plotting.) TEX can calculate with integers so the observing date and magnitude (the apparent brightness of a star) are transformed to integers. In the case of date, simply the integer part of the Julian Date is used. The magnitude is expressed in units of 0.1 so the magnitude is $10 \times$ #3 + #4.

For each star we have a number of observations. The file of observations processed by plain-TEX produces a file suitable for PICTEX. This .plt file contains all the necessary information for a plot: the axes are properly scaled and labeled and contain information about the star. See the example on page 125. Figure 1 shows a typical file ready for TEX.

The macros used to create the .plt file are given below. Part II neglects the fraction of the Julian Date and calculates the magnitude, in units of 0.1. Both are written to a temporary file with the extension .obs. In part II also the minimum

```
\input plot.tex
\harvard{000451}
\head{SS Cas}
\type{Mira  HIP}
\obs 47537.4.10.0.CMG.
\obs 47539.4.9.7.FJH.
\obs 47544.3.10.2.CMG.
\obs 47544.3.10.4.BMU.
\obs 47549.4.10.5.BMU.
\obs 47550.4.10.3.CMG.
\obs 47554.4.10.5.NWL.
\obs 47565.3.11.3.CMG.
\obs 47565.3.11.4.BMU.
\obs 47567.3.11.5.FJH.
\obs 47569.3.11.7.BMU.
\obs 47573.3.11.7.CMG.
\obs 47574.4.12.0.BMU.
\obs 47578.3.12.1.CMG.
\obs 47579.3.11.8.FJH.
\obs 47579.3.12.1.BMU.
\obs 47586.4.12.3.BMU.
\obs 47592.3.12.6.CMG.
\obs 47594.3.12.5.FJH.
\endhead
\bye
```

Figure 1: An input-file for TEX to create a file ready for PICTEX.

and maximum values along the axis are estimated. These values are used in Part III to calculate the proper sizes of the graph. In Part III the final .plt file is created for processing by PICTEX.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%
%% PART I plot.tex     %%
%%%%%%%%%%%%%%%%%%%%%%%%%%
\newcount\tempx    \newcount\tempy
\newcount\tempz    \newcount\maxx
\newcount\minx     \newcount\miny
\newcount\maxy     \newcount\numobs
\newwrite\plotfile \newwrite\obsfile
\def\head#1{
 \global\def\plotname{\sternum\ #1}
 \initplot}
\def\endhead{\immediate\closeout\obsfile
 \startplot}
\def\harvard#1{\global\def\sternum{#1}}
\def\type#1{\global\def\typename{#1}}
\def\initplot{
 \immediate\openout\obsfile=\jobname.obs
 \global\minx=99999
 \global\maxx=0
 \global\miny=99999
 \global\maxy=0
 \global\numobs=0}
\def\pplot#1#2{\tempx=#1
 \tempy=\maxy
 \tempz=#2
 \advance\tempy by-\tempz
 \advance\tempx by-\minx
 \immediate\write\plotfile{\noexpand
  \put {$\noexpand\bullet$} at
   {\the\tempx} {\the\tempy}}}
%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  PART II              %%
%%%%%%%%%%%%%%%%%%%%%%%%%%
\def\obs #1.#2.#3.#4.#5.{
 \message{#5}
 \global\advance\numobs by 1
 \tempx=#1
% estimate min max x
 \ifnum\minx>\tempx \global\minx=\tempx \fi
 \ifnum\maxx<\tempx \global\maxx=\tempx \fi
 \tempy=#3
 \tempz=#4
% calculating magnitude
 \multiply\tempy by 10
 \advance\tempy by\tempz
% estimate min max y
 \ifnum\miny>\tempy \global\miny=\tempy \fi
 \ifnum\maxy<\tempy \global\maxy=\tempy \fi
 \immediate\write\obsfile{\noexpand
  \pplot{\the\tempx}{\the\tempy}}}
%%%%%%%%%%%%%%%%%%%%%%%%%%
%%  PART III             %%
%%%%%%%%%%%%%%%%%%%%%%%%%%
\def\startplot{
% create nice numbers along axis
 \global\divide\minx by 10
 \global\multiply\minx by 10
 \global\divide\maxx by 10
 \global\multiply\maxx by 10
 \global\advance\maxx by 10
 \global\divide\miny by 10
 \global\multiply\miny by 10
 \global\divide\maxy by 10
 \global\multiply\maxy by 10
 \global\advance\maxy by 10
 \global\advance\maxx by-\minx
 \ifnum\numobs>15
% writing pictex commands
  \def\name{\sternum.plt}
  \immediate\openout\plotfile=\name
  \message{\name}
  \immediate\write\plotfile{\noexpand
   \beginpicture}
  \immediate\write\plotfile{\noexpand
   \setcoordinatesystem units <0.35mm,1mm> }
  \tempz=\maxy
  \advance\tempz by-\miny
  \immediate\write\plotfile{\noexpand
   \setplotarea x from 0 to {\the\maxx},
    y from 0 to {\the\tempz}}
  \immediate\write\plotfile{\noexpand
   \plotheading{\plotname\ \typename}}
  \immediate\write\plotfile{\noexpand
   \axis bottom label {JD-{\the\minx}}}
  \immediate\write\plotfile{ticks
   numbered from 0 to {\the\maxx} by 40 / }
  \immediate\write\plotfile{\noexpand
   \axis left label {} }
  \immediate\write\plotfile{ticks
   from 0 to {\the\tempz} by 10 / }
  \tempx=\miny
  \advance\tempx by-10
  \tempz=\maxy
  \loop\ifnum\tempz>\tempx
        \tempy=\maxy
        \advance\tempy by-\tempz
        \immediate\write\plotfile{\noexpand
         \put {\the\tempz} at
          -15 {\the\tempy}}
        \advance\tempz by-10
  \repeat
% find out where to put text
% along y-axis.
  \tempz=\maxy
  \advance\tempz by-\miny
  \divide\tempz by2
  \tempx=\tempz
  \divide\tempx by10
  \tempy=\tempz
  \multiply\tempx by10
  \advance\tempy by-\tempx
  \ifnum\tempy=0
        \advance\tempz by5
  \fi
  \immediate\write\plotfile{\noexpand
   \put {Magn.} at -15 {\the\tempz}}
  \input \jobname.obs
  \immediate\write\plotfile{\noexpand
   \endpicture}
  \immediate\closeout\plotfile
 \fi}
```

2 Macros

In Part I the initialization is done: counters and files are defined. Information about the stars is stored in the tokens `\plotname`, `\sternum` and `\typename`.

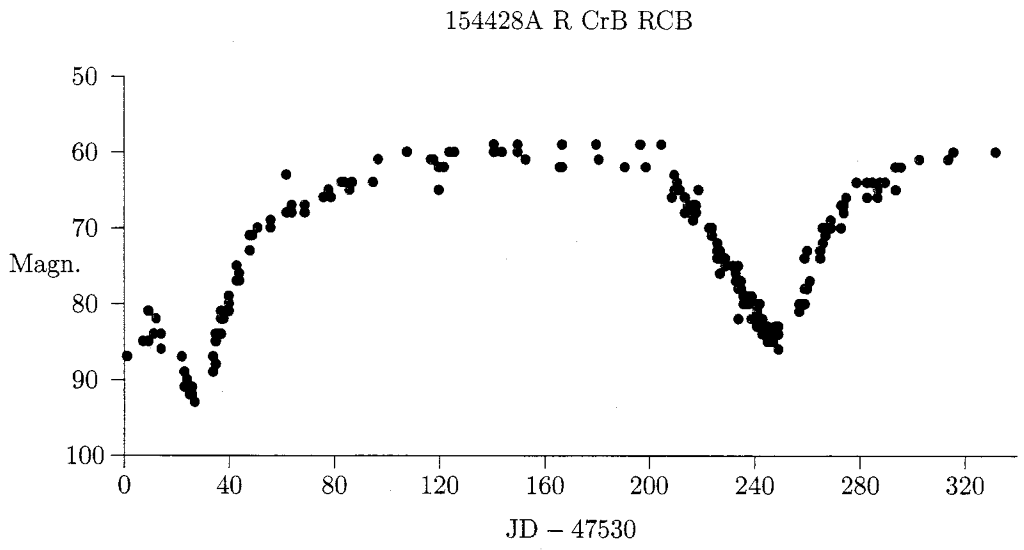


Figure 2: The light-curve of the star R CrB. The decrease in brightness can be seen clearly in this picture. The data are collected by Dutch amateur-astronomers.

Also some initial values for the minimum and maximum values along the axis are set. The macro `\pplot` is used in Part II and executed in Part III. The macro writes the re-scaled X and Y values to the `.plt` file. Again the latter is used with PICTEX to obtain the desired result.

In Part II `\obs` is defined: the magnitude and Julian Date are converted to integers as described above and the maximum and minimum values are estimated according to the simple algorithm:

if $x_i <$ min then min $:= x_i$;

if $x_i >$ max then max $:= x_i$;

Also the number of observations is counted. This number is stored in `\numobs`.

In Part III the final plotting commands are written to the `.plt` file. This part is only executed if the number of observations is bigger than 15. The minima and maxima found in Part II are re-calculated to have nice numbers along the axes. Commands for labels and numbering axes are also written into the file. After setting up the graph the data to be plotted are read from the `.obs` file. This file is created in Part II. Figure 3 shows such a `.plt` file.

```
\beginpicture
\setcoordinatesystem units <0.35mm,1mm>
\setplotarea x from 0 to {340},
 y from 0 to {50}
\plotheading {154428A\ R CrB\ RCB}
\axis bottom label {JD-{47530}}
 ticks numbered from 0 to {340} by 40 /
\axis left label {}
 ticks from 0 to {50} by 10 /
\put {100} at -15 {0}
\put {90} at -15 {10}
\put {80} at -15 {20}
\put {70} at -15 {30}
\put {60} at -15 {40}
\put {50} at -15 {50}
\put {Magn.} at -15 {25}
\put {$\bullet $} at {1} {13}
\put {$\bullet $} at {7} {15}
.
.
.
\endpicture
```

Figure 3: The start of a typical plot-file as created by Part III.

3 Publishing the Data

All the observations of one year are collected in one file with a structure as in Figure 1. This file contains 6000 to 10.000 lines depending on the weather conditions. The file is processed using `plot.tex` and a number of `.plt` files are created. The same file is used to create a multi-column tabular output of the observations; see the example on the next page. These macros are available on request.

◇ Theo A. Jurriens
Kapteyn Astronomical Institute
P. O. Box 800
9700 AV Groningen
The Netherlands
TAJ@RUGR86.RUG.NL

okt – nov – dec 1991

000451 SS Cas
Mira HIP

560.4	13.0	FJH
572.5	13.3	FJH
594.5	12.5	FJH

000928 UW And
Mira

533.4	11.7	FJH
556.3	12.7	FJH
570.5	13.7	FJH

001046 X And
Mira

532.4	10.9	FJH
551.4	9.9	FJH
572.4	9.1	FJH
596.4	9.1	FJH
611.4	9.8	FJH

001726 T And
Mira

557.4	10.6	FJH
572.4	9.5	FJH
596.4	8.2	FJH
614.4	8.5	FJH

001755 T Cas
Mira HIP

596.4	9.0	FJH
614.4	9.1	FJH

001838 R And
Mira HIP

540.5	9.1	FJH
557.3	9.8	FJH
572.4	10.3	FJH
596.3	10.9	FJH
611.4	11.2	FJH

002725 A TU And
Mira HIP

557.4	11.5	FJH

003162 TY Cas
Mira

576.3	13.5	FJH
594.5	11.5	FJH

003179 Y Cep
Mira

551.4	13.0	FJH
570.5	13.6	FJH
590.6	14.0	FJH

004047 U Cas
Mira

540.4	12.7	FJH
550.4	13.0	FJH
557.3	13.7	FJH
572.5	14.2	FJH
596.3	:15.5	FJH

004132 RW And
Mira

533.5	15.3	FJH
596.3	:15.5	FJH

004435 V And
Mira

557.3	14.1	FJH
569.5	14.5	FJH
596.3	14.8	FJH
615.3	14.3	FJH

004533 RR And
Mira

533.5	15.0	FJH
557.3	13.8	FJH
569.5	12.1	FJH
596.4	10.9	FJH
611.4	9.8	FJH

004746 A RV Cas
Mira

533.5	15.5	FJH
557.3	15.0	FJH
601.5	12.6	FJH
611.4	11.7	FJH

004958 W Cas
Mira HIP

556.6	:10.1	KKP

005840 RX And
UGZ

532.42	14.0	FJH
533.48	13.7	FJH
536.43	14.0	FJH
540.44	11.5	FJH
545.38	12.0	FJH
551.34	13.5	FJH
556.35	13.0	FJH
557.32	13.4	FJH
558.37	13.1	FJH
559.38	13.5	FJH
560.35	13.6	FJH
569.46	12.3	FJH
570.35	11.2	FJH
572.39	11.5	FJH
574.43	11.6	FJH
575.34	11.6	FJH
576.33	12.2	FJH
594.44	13.7	FJH
596.35	13.7	FJH
615.28	13.9	FJH

010621 A X Psc
Mira

536.4	14.5	FJH
558.4	13.9	FJH
570.5	13.5	FJH
596.4	12.9	FJH

010937 FO And
UG

533.47	14.9	FJH
536.45	<15.4	FJH
572.46	14.6	FJH
596.34	<15.4	FJH
615.28	14.2	FJH

010940 U And
Mira

540.5	9.6	FJH
557.4	10.2	FJH
569.5	11.1	FJH
594.4	11.9	FJH
614.4	12.3	FJH

011041 A UZ And
Mira

533.5	15.1	FJH
557.3	14.0	FJH
569.5	13.9	FJH
594.4	13.2	FJH
614.4	12.0	FJH

011055 A VZ Cas
Mira

560.4	10.6	FJH
611.4	11.0	FJH

011208 S Psc
Mira

540.5	11.0	FJH
557.5	11.8	FJH
570.5	12.2	FJH
596.3	12.7	FJH

011712 U Psc
Mira

536.4	14.2	FJH
558.4	14.4	FJH
570.5	13.6	FJH
596.3	12.2	FJH

012020 RX Psc
Mira

536.4	:15.1	FJH
596.3	14.5	FJH

012031 TY Psc
UGSU

594.45	12.1	FJH
596.34	12.4	FJH

012502 R Psc
Mira

536.5	13.8	FJH
559.3	13.9	FJH
570.5	13.9	FJH
596.3	13.1	FJH

012746 SX And
Mira

556.4	10.6	FJH
575.3	11.1	FJH
594.5	11.5	FJH
611.4	12.0	FJH

013050 KT Per
UGZ

536.44	12.4	FJH
540.46	14.4	FJH
556.35	12.2	FJH
557.32	11.9	FJH
558.36	11.8	FJH
559.38	11.8	FJH
560.35	12.0	FJH
572.46	14.4	FJH
575.35	12.4	FJH
576.33	12.4	FJH
590.52	11.9	FJH
601.46	11.9	FJH

013238 RU And
SRa

536.5	12.0	FJH
556.4	11.7	FJH
575.3	11.9	FJH

013338 Y And
Mira

532.4	14.2	FJH
540.5	14.0	FJH
551.3	13.2	FJH
569.5	11.3	FJH
594.5	9.9	FJH
611.4	9.2	FJH

013937 AR And
UGSS

532.42	14.2	FJH
533.47	15.3	FJH
556.36	12.5	FJH
557.32	12.8	FJH
558.36	13.5	FJH
559.39	<15.0	FJH
572.36	12.3	FJH
574.43	12.7	FJH
575.36	14.0	FJH
594.45	11.9	FJH
596.32	12.2	FJH
601.46	12.7	FJH

015254 U Per
Mira HIP

533.4	8.3	KKP
536.4	8.5	JOJ
556.3	8.2	JOJ
556.6	8.7	KKP
601.3	9.1	KKP
601.4	8.4	JOJ

015457 V666 Cas
Mira

556.4	11.2	FJH
576.3	11.2	FJH

015912 S Ari
Mira

533.5	13.8	FJH
558.4	14.5	FJH
570.5	14.6	FJH
596.3	:15.2	FJH

020227 Z Tri
Mira

532.5	14.7	FJH
540.5	14.7	FJH
596.3	13.7	FJH

020356 UV Per
UGSS

613.51	12.6	FJH
614.40	12.7	FJH
615.23	12.7	FJH

020657 A TZ Per
UGZ

532.50	12.9	FJH
536.44	13.8	FJH
540.46	14.3	FJH
551.35	12.8	FJH
557.33	13.9	FJH
558.36	14.2	FJH
559.39	14.2	FJH
560.48	13.7	FJH
570.35	13.5	FJH
572.47	13.8	FJH
575.36	13.7	FJH
590.51	13.6	FJH
596.46	13.6	FJH
601.46	12.7	FJH
615.27	13.4	FJH

021024 R Ari
Mira HIP

540.5	12.5	FJH
551.4	12.4	FJH
575.5	10.5	FJH

021143 A W And
Mira HIP

540.4	13.1	FJH
551.3	12.7	FJH
569.5	12.4	FJH
594.5	11.8	FJH
611.4	10.7	FJH

021281 Z Cep
Mira

536.5	14.1	FJH
551.4	13.5	FJH
570.5	11.9	FJH
581.3	11.6	FJH
590.6	11.8	FJH
613.5	12.3	FJH

0214-0 3 Mira
Mira HIP

536.5	4.0	SAQ
536.6	4.0	BMU
596.3	6.1	SAQ

021558 S Per
SRc HIP

556.4	12.0	FJH
557.3	11.8	HIL
576.3	12.2	FJH
594.5	12.2	FJH

0220-0 0 R Cet
Mira HIP

536.5	9.2	SAQ
601.3	8.3	SAQ

022150 RR Per
Mira

540.5	11.5	FJH
556.4	11.7	FJH
575.4	12.2	FJH
590.5	12.7	FJH

022980 RR Cep
Mira

536.5	14.1	FJH
570.5	13.1	FJH
581.3	12.5	FJH
590.6	11.8	FJH
613.5	11.2	FJH

Book Reviews

Review of recent LaTeX books

Nico Poppelier

Helmut Kopka and Patrick Daly, *A Guide to LaTeX, Document Preparation for Beginners and Advanced Users.* Addison Wesley, 1993. 436 pages (including indexes) ISBN 0-201-56889-6.

Antoni Diller, *LaTeX Line By Line, Tips and Techniques for Document Processing.* John Wiley & Sons, 1993. 291 pages (including index) ISBN 0-471-93471-2.

Whenever I read a book that describes a computer-related topic I ask myself these questions: does this book contain information I haven't seen elsewhere, does it explain things in a way that gives new insights? And I do this especially with books on TeX or LaTeX.

In April 1992 I gave a favourable review of two books about LaTeX, two *good* books, written by Helmut Kopka. However, they were written in German, which made them inaccessible to a large portion of the TeX user community. Fortunately the first of these two books is now available in an English version: *A Guide to LaTeX, Document Preparation for Beginners and Advanced Users.* This version was written by Helmut Kopka and his colleague Patrick Daly at the German Max-Planck-Institut für Aeronomie, and is based on the fourth edition of *LaTeX, eine Einführung.* The English version is not a mere translation, but an internationalized version, where parts specific to the German language have been replaced by descriptions of, e.g., the new font-selection scheme (NFSS) and the Babel system.

As in the German original, Kopka and Daly follow Lamport's basic notion that with LaTeX the user is freed from worrying about the layout while writing a piece of text. *A Guide to LaTeX* has chapters on document and page styles, displayed text, mathematical formulas, pictures, user-defined structures and a few advanced features of LaTeX. In the appendices the authors treat the `letter` document style, including possibilities for customization, BibTeX, SliTeX, LaTeX extensions, and the CM and DC fonts.

Their book provides a wealth of information, and if the updated English version of the companion volume *LaTeX, Possibilities for Extensions* will be published by Addison-Wesley soon enough, there is practically no need for further books about LaTeX, since almost everything beginning or advanced users need to know is in one of these two books.

In contrast with this, Antoni Diller does not present LaTeX as a system for the production of structured documents. LaTeX is a system that emphasizes structure over presentation, and that is ideally suited for the production of many instances of a certain type (class) of document, for example office memoranda or scientific articles. Instead, in *LaTeX Line By Line* LaTeX is presented as a collection of TeX macros, with which you can achieve all sorts of effects. Therefore, all examples and explanations in the book use a mix of LaTeX commands, plain TeX commands and TeX primitives.

There is of course nothing wrong with this approach, if the book is intended as a book on tips and tricks in LaTeX and plain TeX. However, the preface of *LaTeX Line By Line* clearly shows that the book is intended as a book for novice users, and attempts to explain *all* about LaTeX. In other words: it is intended as 'your first and only book on LaTeX'.

There are no glaring errors or omissions in the book, but it lacks structure, and the mix of LaTeX, plain TeX and TeX primitives will really confuse any novice user. Because of this, *LaTeX Line by Line* is not a book for beginners, even though the author writes in the preface 'This book can be read by someone who has no previous knowledge of either LaTeX or TeX.'

In the introduction of this review column I explained what the things are that I am looking for in a new computer book: what makes this book special or unique? What does it explain that I haven't seen before, or in a way I haven't read before? For what special group of people is it written? Antoni Diller's book on LaTeX isn't special in any sense; it is just another poorly written book about LaTeX, of which there are unfortunately a few too many already. *LaTeX Line by Line* contains a lot of useful tricks, especially in the area of mathematics, but they are presented in an unstructured and confusing way. Also, the title is not appropriate: it is not a book about LaTeX, but a book about how to combine LaTeX, plain TeX and TeX primitives to achieve certain special effects in layout.

A detail: both books give the old address of the TeX Users Group (that is a problem with putting addresses and similar factual information in a book).

Another detail: the book by Helmut Kopka and Patrick Daly has the nicest LaTeX logo I have ever seen — including the one on Lamport's book! No wonder, since the designer at Addison-Wesley called

Barbara Beeton and asked her what it should look like. And I must say: she did a very nice job!

The logo on Antoni Diller's book looks horrible, since the 'A' in LaTeX is *much* too far to the right. Not only that, but it is also reproduced like that many times on the front cover. Add to this the poor design of the book and the fact that it was reproduced from low-resolution output, I am afraid that there is another book about LaTeX on the market that I cannot recommend.

◇ Nico Poppelier
Elsevier Science Publishers
Amsterdam
The Netherlands
Internet:
n.poppelier@elsevier.nl

Typesetting on Personal Computers

ET — A TeX-Compatible Editor for MSDOS Computers

John Collins

Abstract

An editor for visually editing (LA)TeX files is described. It runs on MSDOS computers. A comparison with `Scientific Word` is given.

1 Introduction

Many of us are familiar with the advantages of (LA)TeX for scientific word processing and typesetting. It provides a high quality of typesetting of complex mathematical material that is fully acceptable for the best publishers. The software runs on a wide variety of host computers. Document files are plain ASCII files that can be readily transmitted without problems between disparate computers and over network connections. In the kinds of scientific collaboration that many of us are involved in, these last advantages are very important. The portability allows electronic submission of papers to journals and electronic bulletin boards.

However, the user interface of TeX shows its great age of well over a decade. For LaTeX to output

$$\phi(x,Q) = \frac{Z_2^{(Q)}}{Z_2^{(\Lambda)}} \int^{Q^2} \frac{dk_\perp^2}{16\pi^2} \psi^{(\Lambda)}(x,k_\perp), \qquad (1)$$

the user has to put up with typing

```
\begin{equation}
\phi(x,Q) = \frac {Z_2^{(Q)}}{Z_2^{(\Lambda)}}
            \int^{Q^2}
            \frac {dk_\perp^2}{16\pi^2}
            \psi^{(\Lambda)} (x,k_\perp).
\end{equation}
```

Even though this formula is quite simple, the source text is relatively complicated to edit. The reason is that one's mind has to translate the TeX source to something like Eq. (1).

On the other hand, one mostly edits scientific papers in terms of content rather than appearance. Thus it is not essential to have an editor with a full WYSIWYG ('What You See Is What You Get') display that shows on-screen exactly what one will get on paper. Remember that a handwritten equation in a colleague's draft of a paper is much easier to read than the corresponding TeX source, at least if the equation is complicated and the handwriting is neat. But one should be able to edit in terms of the content of a document, not in terms of the low-level TeX commands. This is particularly important when one is composing a document at the keyboard, as I often do, for example from the transparencies of a talk at a conference.

To handle this problem, I have written an editor called ET (for Edit TeX). It satisfies the following requirements:

- ET should allow one to edit visually the most common mathematical constructs directly, and should immediately show them on-screen. ET in fact works with Greek letters, sub- and superscripts and built-up fractions, showing them as mathematics, not as TeX control sequences.
- ET should make it easy to type and edit these objects from the keyboard. It is frustrating to navigate a menu to perform a common and simple operation like entering a single Greek character.
- ET should generate standard, or almost-standard, TeX and LaTeX files, with at most a small number of special macro definitions.
- ET should be able to import an ordinary (LA)TeX file and edit it. That is to say, it can handle files generated by a collaborator who neither uses ET nor knows about it.

- It should be simple or trivial to enter ordinary TeX and LaTeX commands for constructs that ET does not handle.
- ET should run with adequate performance on even a low-grade computer like the cheap IBM clone many people have at home for simple word processing. (Of course, one would probably do the TeX printout using a fancier machine at the office.)

For the purposes of the editor, a LaTeX file is an ordinary TeX file: the differences between TeX and LaTeX are almost entirely in the commands that concern the overall structure of the document. Indeed, I find it relatively unimportant to have a WYSIWYG display for such 'structural commands', at least in scientific papers. To see the characters `\section` is as useful as seeing a section heading formatted identically to the final hard copy.

I find ET to be a comfortable editor to use for composing (LA)TeX papers. When I am working on a collaboratively written paper and receive a draft from a co-author, I find it best to use ET in preference to using a regular editor on the TeX file when I am revising the draft.

One of the great advantages of using ET is that many trivial TeX errors are avoided. Errors with unbalanced braces and with misspelled control sequence names are restricted to those parts of the file that are typed in ordinary TeX.

ET runs on most MSDOS PCs — typical IBM clones. It will also run under DOS Windows and SunPC on Sun workstations (which emulate an IBM PC).

To obtain ET, it is easiest to use anonymous ftp to `ftp.phys.psu.edu` — see a later section for details. If that is not possible, contact me — most easily by e-mail to `collins@phys.psu.edu`.

2 What ET Does

As regards its non-mathematical features, ET is a fairly standard editor for ASCII text files. It has the usual features for insertion and deletion, for searching and replacing of text, for cutting and pasting, and so forth. Editing functions are invoked by the PC keys in the usual way. Control-key combinations can also be used; the default is to assign them in the manner of the WordStar word processor that is familiar to many veteran users of personal computers. The assignments of the keyboard commands are completely reconfigurable. There is also a pull-down menu and on-line help. At present, there is no mouse support, although I plan to add this in a future version, if I get sufficiently many requests from users.

Two sizes of screen font are provided. The smaller of these allows over 43 lines on a VGA screen. The larger only gives 33 lines on a VGA, but is easier to read.

The editor contains an alternate font of Greek letters and common mathematical symbols. These are entered very simply: For example `alt/G` followed by `f` results in a ϕ on the screen. Characters in the Greek/symbol font are of course treated as single characters by ET, for example by the delete key.

There are simple keyboard short-cuts for opening subscripts, superscripts and fractions (`alt/L`, `alt/H`, and `alt/F`, respectively). The results are immediately visible on screen. Special short-cuts speed up entry of simple formulae — specifically `alt/+` superscripts the previous character and `alt/-` subscripts the previous character. Thus there is no need to access the menu for common operations, and ET is well suited to rapid touch typing.

Eq. (1) was entered using ET mathematics only. The only regular TeX commands (LaTeX actually) consisted of the `\begin{equation}` and the `\end{equation}` lines.

Ordinary (LA)TeX commands may be freely intermixed with ET's mathematics: they are simply typed as in a normal text editor.

Although ET writes to the screen in graphics mode, it is fast and responsive on even the slowest of PCs. Care has been taken in the writing of the screen driver routines to ensure this. (The contrast with the `Scientific Word` [1] program is quite noticeable.)

Since TeX syntax is not very convenient for fast parsing, ET works with its own file format (which is regular ASCII text with interspersed control characters). Programs are provided for conversion in both directions, to and from regular TeX. Suppose I receive a file `paper.tex` from a colleague. I first convert it to ET:

```
textoet paper
```

This generates a file `paper.et`. Then I edit this file:

```
et paper
```

An extra step of conversion to TeX is needed before (LA)TeXing the file:

```
ettotex paper
latex paper
```

This process can be automated by a batch file, of course. The `ettotex` conversion is much faster than the processing by TeX, so the extra conversion does not give a significant time penalty.

It is the `paper.tex` file that results from the `ettotex` conversion that I send back to my collaborators.

Very rarely, the `textoet` conversion produces an undesired result. (This can happen in a macro definition, for example.) There are 'metacommands' that can be put in the `.tex` file to turn the conversion to ET off and on. These have the form of TeX comments, so that they do not interfere with normal processing by TeX. The untranslated part of the file is passed through as normal text. This does not preclude the use of normal TeX comments; it simply means that those few TeX comments that have the form of metacommands have the side effect of being instructions to the translation program.

The conversion from TeX is simple-minded, but robust. This enables it to handle (or ignore) TeX syntax errors, such as those generated, for example, by an older colleague who is rather inexpert at using TeX.

I have used ET and the conversion programs for a few years on all my scientific papers. Most of these have been written in collaboration with colleagues at other institutions round the world. A typical paper is transmitted by e-mail dozens of times. The conversion programs in particular have given no trouble with papers written by other people by normal methods (i.e., without ET) and without taking any precautions because of my use of ET. This is in sharp contrast to the `Scientific Word` program, which almost always crashes the operating system when importing even a syntactically correct LaTeX file, and cannot handle plain TeX at all.

3 Compatibility with (LA)TeX

ET is fully compatible with LaTeX. For other varieties, including plain TeX, the most that is needed is a definition of the macro `\frac`, viz.,

```
\def\frac#1#2{{#1 \over #2}}
```

4 Hardware and Software Requirements

The ET program and the `ettotex` and `textoet` converters run on any MSDOS (i.e., IBM compatible) computer.[1] This includes computers running Windows and OS/2. (ET will need to be run as a full-screen non-Windows program if you are running Windows, and it will need to run in an 'MSDOS compatibility box' under OS/2.)

The only restrictions are that there should be sufficient memory, which is normally present on current MSDOS computers, and that there should be a suitable video card, which at present means: EGA, VGA, CGA, Hercules. It would be fairly simple for me to add to this list any other graphics-capable card.

[1] Such compatible computers include emulators of IBM PCs, in particular, DOS Windows and SunPC on Sun workstations. To ensure compatibility in such a mixed UNIX-MSDOS environment, both ET and `textoet` will correctly read text files in the standard UNIX format.

5 Comparison with `Scientific Word`

Recently, a scientific word processor, `Scientific Word` [1], has become available whose announced purpose is identical to that of ET.

`Scientific Word` is a much fancier program than ET, and it aims to take much more of the burden of using TeX from the user. As such it is an excellent program. Unfortunately `Scientific Word` has some severe disadvantages, which leave ET a very useful tool in many situations.

The simplest disadvantages are that `Scientific Word`'s editor is slow and that it lacks some basic features like search and replace. It also runs only under Windows (which is not a great disadvantage nowadays).

The main problem I have found is that `Scientific Word` relies on its own extensive package of LaTeX macros. In a collaborative environment, where one may receive a first draft of a paper written in some other TeX dialect (and in particular not in LaTeX), this is a fatal problem, especially when `Scientific Word`'s utility to import LaTeX files typically crashes the operating system without so much as an error message from the filter. The authors of `Scientific Word` explicitly say that they do not regard the importing of TeX to be an important function. The philosophy I have adopted with ET is essentially the opposite, and this is based on the actual use I and all my colleagues make of TeX. If we did not collaborate and did not submit papers electronically to journals and to bulletin boards, a regular WYSIWYG editor with good equation handling facilities would be very tempting.

Another problem is that `Scientific Word` does not allow free intermixture of regular TeX commands in a simple way. For example, one is forced to edit an `eqnarray` environment entirely in TeX in a separate window, and entirely without the benefit of the mathematics editing features of `Scientific Word`. Again, I see one of the important features of TeX to be its ability to define new commands and to redefine old ones. Therefore ET is designed to handle such definitions. Or rather, it is designed not to handle them, but to treat as regular text any constructs it has not been programmed to know about, and this treatment needs no special precautions or special configuration.

6 How to Obtain ET

ET and its associated programs and files are freely available by anonymous ftp or by request on floppy disk. (Note that the program is copyrighted and is not public domain.)

The following is a typical session for obtaining ET by anonymous ftp. User input is underlined.

```
>:  ftp ftp.phys.psu.edu
Name (ftp.phys.psu.edu:collins):
anonymous
331 Guest login ok,
send e-mail address, e.g.
user@host", as password.
Password:  _ ...
YOUR NAME@NODE GOES HERE

ftp> cd pub/dos_tools/et
ftp> binary
ftp> prompt
ftp> mget *.*
ftp> bye
```

Some changes may be necessary if you are using a different version of ftp. The `image` command, or its equivalent[2] is essential to avoid mangling the files, especially the executable files.[3] Remember that if you download these files to a non-MSDOS machine, and then download them to a PC by a terminal emulator program (e.g., KERMIT), then you will have to set the file type to `image` or `binary`, or whatever is the equivalent for your program.

The files can also be obtained in a compressed form in a file `etzip.exe`, which is available from the same anonymous ftp location. This file is in the subdirectory `zip` of the directory where the uncompressed files are held. It is a self-unzipping file: execute the file on an MSDOS computer, and it will generate the files for ET.

A `aaaread.me` file among those you have downloaded will tell you how to install ET. There is also a documentation file `etdoc.et` that will give you a manual for ET when printed out.

If for some reason you are unable to use ftp to obtain ET, I can send it to you on a floppy disk. Send me a request at the address shown at the end of this article. Include $5 for shipping etc., and state the MSDOS disk size and density that you can use.

[2] E.g., the use of the command `ftp /image` to invoke `ftp` on some VAX/VMS systems.

[3] The text files can usually be downloaded safely in ASCII mode, but may suffer from extra carriage return characters at the ends of lines.

In any case, if you use the program and find it useful, send me a brief note (preferably by e-mail to the address given below), so that I can send information about new versions. Also, let me know of any requests you have for new features, and of course let me know about any bugs. I am aware of some of the improvements that can be made. But there is only a point to spending some of my limited time on improving the program if there is an audience for it (and if I know about the audience).

Acknowledgments

Some of the time spent in developing this program was supported by the U.S. Department of Energy under grant DE-FG02-90ER-40577.

References

[1] `Scientific Word`, TCI Software Research, Inc., Las Cruces, NM.

◇ John Collins
Physics Department
Pennsylvania State University
104 Davey Lab
University Park, PA 16802
U.S.A.
Internet: `collins@phys.psu.edu`
Bitnet: `jcc8@psuvm`

Tutorials

Essential NFSS2, version 2

Sebastian Rahtz

Abstract

This article offers a user's view of the New Font Selection Scheme, version 2. It describes the reasons for using the NFSS; the differences a user will encounter between NFSS and old LaTeX; what it will be like installing and using NFSS2; some special situations in mathematics; and an overview of the advanced NFSS2 commands for describing new fonts.

1 Introduction

TeX has a very general model of using different typefaces; all it needs is to have a set of metric information for each font, and it leaves the actual setting up to the dvi driver. Within the TeX input file, the user associates a name with a given font at a particular size, and uses that name when she wishes to set in the font. In Knuth's descriptive 'plain' macros, this is easy to work with; in LaTeX, however, the user works at a higher level of commands like `\bf` or `\huge`, and the system is supposed to work out which particular font is meant. In the early 1980s when LaTeX was being developed, there was not much choice in typefaces (everyone used Computer Modern) or sizes, so Lamport simply 'hard-wired' all the allowed combinations of size and style, and assigned them to specific external fonts.

Unfortunately, LaTeX's font selection is not as general as it sometimes seems. The commands do not add together to produce an intuitive effect; thus the commands `\it` and `\bf`, which separately produce italic and bold, do *not* produce bold italic when used together, but have the effect of whichever command comes last. Similarly, a size change always reverts to a normal weight font, so that `\bf\Large` does *not* produce large bold, whereas `\Large\bf` does. What is much worse is that if one wishes to use a different typeface throughout a document wherever (say) sans-serif is used, it requires major surgery to the innards of LaTeX; the font assignments are made in the file `lfonts.tex` which is read only when one is building the format file for LaTeX, something which most users never do.

The average user who simply uses the fonts which come with a typical TeX installation is not usually aware of the underlying lack of flexibility; however, now that many more fonts are available for use (a greater choice of METAFONT sources, and the ubiquitous PostScript fonts), this area has been a bottleneck in the use of LaTeX in general typesetting. The user who wanted Times Roman as the default typeface in a document had three choices:

1. To load many new fonts with the `newfont` command, and use them explicitly;
2. To heavily edit the file `lfonts.tex` and build a new version of LaTeX; or
3. To take large chunks out of `lfonts.tex` and replicate their effect in a style file.

Other style files were written to specifically implement features like bold sans-serif; but an altogether more general solution was needed, and this was met in 1989 with the New Font Selection Scheme, which was subsequently also used in AMS-LaTeX. The NFSS is a complete replacement for `lfonts.tex`; it contains a generic concept for varying font attributes individually and integrating new font families easily into LaTeX.

Since the first NFSS, considerable progress has been made towards a completely new version of LaTeX itself, and a new version of NFSS is one of the first useable results; this is what is described here.

2 Overview of NFSS2

The NFSS concept is based on five *attributes* of a font; the user changes any of the attributes, and it is the job of LaTeX to identify the appropriate external font which fits the new situation. The attributes are

Encoding The way the font is laid out; when only CM fonts are used, this does not vary, because all of Knuth's fonts have more or less the same layout. But users of PostScript fonts, or the recent 256-character 'DC' fonts (described in detail in 2), will meet different layouts. This is discussed in more detail in Appendix 1.

Family The basic design of the typeface, e.g. Times Roman, Computer Modern, or Stone Informal.

Shape The main divisions of the typeface; most have at least two variants, 'normal' and italic, and possibly something specialized like small caps.

Series Many typefaces come in a range of 'weights' (light, normal, bold, etc.), determined by the thickness of the strokes of letters; they also vary in width, as we see in Computer Modern, which has a normal bold and an 'extended' bold. The two attributes of width and weight are combined in the NFSS, and called 'series'.

Size The base height of the letter shapes (10pt, 14pt, etc.). These different sizes may be achieved in two ways — by having a separate font design for each size, or by scaling of a single (usually) 10pt design. Typical Computer Modern and PostScript setups use a combination of methods, but we do need to worry about this for the NFSS.

At the start of a LaTeX job, each of the five attributes has a default value, typically as follows:

encoding	`OT1`	Normal TeX encoding
family	`cmr`	Computer Modern Roman
shape	`n`	Normal (upright roman)
series	`m`	Medium weight
size	`10pt`	

Inside LaTeX, this combination is looked up in a table to see which external font this situation corresponds to, and the result will be the familiar `cmr10`. Suppose we now wish to move into italics; we simply change the 'shape' attribute to 'italic', and LaTeX reconsults its table to find this means it should load and use the font `cmti10`. A subsequent request to change the 'series' to 'bold' will *not* change the italic *shape*, but search the table for a new combination of 'bold series' and 'italic shape', coming up with `cmbxti10`. Similarly, a request to change the font size at this point to 24pt will leave all other characteristics unchanged, and simply load `cmbxti10` at a larger size (since that particular font is available only in a 10pt design).

In practice, all this selection is hidden from the user behind familiar commands like `\em` and `\Large`, but it is important to realize that the scheme is completely general. If, for instance, we had available the full Univers family, which has a great range of weights, we could select any of them by changing the 'series' characteristic, and leaving LaTeX to find the right font metric file.

How does LaTeX know how to match a combination of `OT1+cmr+m+n+10pt` to `cmr10`? By means of special `fd` (*font description*) files (with a suffix of `.fd` in most operating systems). There is one of these for every combination of *font encoding* and *family*, which defines the possibilities of the other three attributes. Thus `OT1cmr.fd` classifies all the normal Computer Modern fonts by means of the font attributes, and says which external font they correspond to. When LaTeX starts, it does not have *any* fonts loaded[1], but waits until some font selection is needed, and then loads the appropriate `fd` file; as new combinations of font attributes occur, `fd` files are loaded. We will see later what the format of these font files is like.

[1] Not quite true; it needs some fallback defaults if all else fails.

Perhaps the most dramatic effect of attribute changing is when a different *family* is selected, and with a single command the whole document changes from, e.g., Computer Modern to Lucida Bright.

3 Normal usage

The vast majority of your existing LaTeX documents will run through the new system with no further change. The commonest area for incompatibilities is in math (see below, section 4), but you may also be 'tripped' by the new orthogonality in shape and weight selection. If your document contains:

```
See {\bf bold and then \tt typewriter}
```

under the old LaTeX, the `\tt` completely overrides the effect of `\bf`, but in the new system, they are different attributes, so the command `\tt` changes the *family* to (perhaps) `cmtt`, but the bold series attribute is unchanged, so the system will try to load a bold typewriter font (which you may or may not have). A rather more worrying situation occurs when you have defined a new command without thinking through the ramifications. Thus if in a style file or document preamble, we have said

```
\newcommand{\PS}{{\sc PostScript}}
```

so that \PS produces PostScript, and then later we write

```
\section{How \PS\ changed my life}
```

we may have a problem, if the `\section` command is defined to set its argument in a bold sans-serif typeface. The definition of `\PS` says that the word is in small caps. This is a change in *shape*, so the *family* (a sans-serif font) stays unchanged, as does the *series* (bold). So the system tries to load a small caps bold sans-serif font, which you may not have (you don't in Computer Modern, for instance). Here a very important feature of NFSS2 comes into action: *font substitution*. This follows a set of rules (which the user can change) to find the closest possible match. This may not be at all what the user intended, and it will probably produce a different effect from the old LaTeX. However, NFSS2 always puts out clear warnings on the terminal and log file when it is substituting, and the problem is usually obvious when you see your printout. Some more careful discipline is required when writing new documents, and emending old ones. Bear in mind that the strange effects are not *mistakes* by LaTeX, but simply a stricter interpretation of the input than the old LaTeX.

How do we persuade LaTeX to choose the attributes we want? The same commands which we already have work as expected, with two new ones available; these are listed in Table 1.

Command	*Meaning*	*Effect*
`\rm`	normal family	usually a serif font
`\sf`	sans family	a sans font
`\tt`	typewriter family	`a monospaced typewriter font`
`\bf`	bold series	**bold typeface**
`\it`	italic shape	*italic typeface*
`\sl`	slanted shape	*slanted typeface*
`\sc`	small-caps shape	SMALL-CAPS TYPEFACE
`\normalshape`	normal shape	back to normal
`\mediumseries`	normal series	normal weight

Table 1: User commands to change attributes

Command	*Example*	*Effect*
Changing *family*		
`\textrm`	`\textrm{Whales}`	Whales
`\textsf`	`\textsf{Whales}`	Whales
`\texttt`	`\texttt{Whales}`	`Whales`
Changing *series*		
`\textbf`	`\textbf{Whales}`	**Whales**
`\textmedium`	`\textmedium{Whales}`	Whales
Changing *shape*		
`\textit`	`\textit{Whales}`	*Whales*
`\textsl`	`\textsl{Whales}`	*Whales*
`\textsc`	`\textsc{Whales}`	WHALES
`\textnormal`	`\textnormal{Whales}`	Whales
`\textem`	`\textem{Whales}`	*Whales*

Table 2: New font commands with arguments

The size changing commands have an important difference compared to old LaTeX: they change *only* the font *size* attribute (in the old LaTeX they changed the series and shape back to 'normal roman').

The font changing commands in LaTeX declarations have always been rather anomalous, in that they affect the text within the `{` and `}` group where they are used, instead of having an argument like most other commands. LaTeX beginners often mistakenly type `\em{hello}` when trying to typeset *hello*. It is therefore a very good idea to start using a different set of commands, which are provided by the style option `fontcmds`, listed in Table 2.

If you want to change the *default* action associated with any of the above commands, you can do so with the `\renewcommand` macro; each of the declarations above has a corresponding command with a suffix of `default`. Thus you could change the effect produced by `\tt` by saying (in the document preamble or a style file):

```
\renewcommand{\ttdefault}{courier}
```

if you had a font family called 'courier'. There may be problems with the encoding, if this is a PostScript font, but we will discuss that in Appendix 1.

'How,' the suspicious reader will ask, 'do I know what values are allowed for the font attributes? How do I know that boldness is indicated by a series of "bx"?' In fact, more or less *any* value for the attributes is permitted, but if you want your document to be useable by others, it would be as well to stick to a conventional set. If you ask for a *shape* of 'grotesque', you will get the right font *if* the `fd` contains an entry for that combination of attributes. Conventional values are as follows (for the Computer Modern family):

Family cmr, cmss, cmtt

Shape n, it, sl, sc

Series m, b, bx (differentiating between normal bold and extended bold)

Most users will not worry about this, but simply use the high-level commands and get the effect they intended. Different font *families* will commonly be loaded via a style file which changes the default families looked for by `\rm`, `\sf` and `\tt`. A `palatino.sty`, for instance, will set things up so that the roman font is Palatino-Roman, the sans font is Helvetica and the typewriter font is Courier. A set of suitable `fd` files and style files for common PostScript fonts is distributed with NFSS2. The only problem here is agreeing on *family* names for fonts, and having suitable `fd` files, but this is done for a great many typefaces in the standard NFSS2 distribution, with the family names listed in Table 3.

4 Mathematical work

Mathematical typesetting with NFSS2 works in a very different way to text, as the fonts do *not* vary according to the current font attributes in the main body of the document. There are some incompatibilities with old LaTeX, and some new facilities. The principal difference is that font *declarations* like `\bf` no longer work, but we must rely on two different concepts, *math versions* and *math alphabets*. The

Computer Modern			
\cmr	Computer Modern Roman	\cmss	Computer Modern Sans
\cmtt	Computer Modern Typewriter	\cmm	Computer Modern math italic
\cmex	Computer Modern math extension	\cmsy	Computer Modern math symbols
\cmdh	Computer Modern Dunhill	\cmfib	Computer Modern Fibonacci
\cmfr	Computer Modern Funny		
LaTeX			
\lasy	LaTeX symbols		
AMS			
\msa	AMS symbol font 1	\msb	AMS symbol font 2
Concrete			
\ccmi	Concrete math italic	\ccr	Concrete Roman
Euler			
\euex	Euler math extension	\euf	Euler Fraktur
\eur	Euler Roman	\eus	Euler Script
PostScript			
\pag	Avant Garde	\pbk	Bookman
\pcr	Courier	\phv	Helvetica
\ppl	Palatino	\psy	Symbol
\ptm	Times	\pzc	Zapf Chancery
\pzd	Zapf Dingbats		

Table 3: Common font family names

mathversion changes the appearance of the whole formula (all the fonts change), while the *alphabet* is used to set a particular set of of characters in a chosen font. The normal text commands like `\rm`, `\sf` or `\bf` are now completely illegal in math, and a new set of commands is provided:

Example	*Effect*
`\mathcal`	calligraphic style
`\mathrm`	upright text
`\mathbf`	bold text
`\mathsf`	sans-serif
`\mathit`	italic text

These are *commands* with an argument, not declarations. So to get a calligraphic ABC, we say `$\mathcal{ABC}$`, and *not* `$\mathcal ABC$`. The effect of the latter will be to set just the A in calligraphic, since just the first token after `\mathcal` is taken as the argument.

You can define *new* math alphabets for yourself easily, with the `\DeclareMathAlphabet` command, which associates a particular font family, encoding, shape, and series with the command you want to use. So if we want to declare a typewriter math alphabet, we could say (in the document preamble):

```
\DeclareMathAlphabet
        {\mathtt}{OT1}{cmtt}{m}{n}
```

(`OT1` is the name of the original TeX font layout).

What about new math symbol fonts? To illustrate some of the commands available here, let us look at how a style file looks which sets up the AMS symbol fonts. Assuming that the relevant `fd` files exist on our system for the fonts (named *msa* and *msb* in Table 3 above), we can declare the existence of them as symbol fonts:

```
\DeclareSymbolFont{AMSa}{U}{msa}{m}{n}
\DeclareSymbolFont{AMSb}{U}{msb}{m}{n}
```

where we define the names (`AMSa` and `AMSb`) by which we are going refer to them in future, the *encoding* (U is for 'undefined', where there is no standard layout), *family* (msa and msb), *series* (m) and *shape* (n). We can use the new fonts in two ways:

1. By declaring named math symbols, e.g.:

   ```
   \DeclareMathSymbol\lozenge
           {\mathord}{AMSa}{"06}
   ```

   which looks more fearsome than it really is. We are defining a new math macro `\lozenge`, and saying it comes from the `AMSa` font we have defined earlier, at position `"06` (this hexadecimal numbering notation is described in 4, p. 116). The tricky bit is `\mathord`, which says what *type* of symbol it is. The possibilities are listed in Table 4, but the serious user is advised to consult 3, p. 158.[2]

2. By saying that we want to use a symbol font also as a *math alphabet*:

   ```
   \DeclareSymbolFontAlphabet
           {\bbold}{AMSb}
   ```

[2] Note that `\mathalpha` is an NFSS2 addition to the list of possible types.

This defines a new *math alphabet* command `\bbold`, which picks up characters from the `AMSb` fonts (where the AMS has placed 'blackboard bold' letters).

The use of the three math alphabet and symbol commands here is used to construct most of the standard math interface; the supplied style file `euler.sty`, which redefines math to use the Euler fonts, is a good example of its use.

I have not dealt much here with *math versions*; suffice it to say that each of the commands described above has a corresponding command which allows the user to select a specific symbol or alphabet font for each different math version.

5 Going further, and towards LaTeX3

This article has not described all the features of NFSS2, or the style files which are distributed with it. For a detailed description of the whole system (and a great many other useful topics in LaTeX), the reader is referred to 1.

I advise all LaTeX users to switch to the new NFSS2 standard for LaTeX as soon as possible; the work done in NFSS2 is central to the development of the next version of LaTeX, written by the official maintainers of LaTeX, and if you get used to the NFSS way of thinking now, you will be in a good position to take advantage of LaTeX3 when it appears. The few remaining common style files which rely on the old behaviour of LaTeX are rapidly being converted, and many new font-related styles are being written now that it is so much easier to do so.

References

[1] Goossens, M., Mittelbach, F., and Samarin, A. (1993). *The LaTeX Companion.* Addison-Wesley, Reading, Massachusetts.

[2] Haralambous, Y. (1992). TeX conventions concerning languages. *TeX and TUG News*, 1(4):3–10.

[3] Knuth, D. E. (1984). *The TeXbook.* Addison-Wesley, Reading, Massachusetts.

[4] Lamport, L. (1986). *LaTeX Users Guide and Reference Manual.* Addison-Wesley, Reading, Massachusetts.

Appendix 1: Encoding, and PostScript fonts

It is an unfortunate fact of life that we do not have a completely accepted standard for layout of fonts; everyone agrees on a certain number of characters, and where they are to be found (the ASCII character set), but this is not adequate for any serious typesetting. Whenever we type character 234 or a TeX macro like `\c{c}`, we want the effect to be ç, and TeX has to know where to find this in a font. We are likely to meet at least three situations, or five if we use PostScript fonts:

1. The layout of Knuth's original Computer Modern fonts (3, Appendix F).
2. The 'Cork' layout, discussed by 2.
3. An extended ASCII layout, as used (for instance) in many DOS or Macintosh applications (not necessarily the same!).
4. 'Raw' PostScript layout, the default defined by Adobe.
5. A re-coded PostScript layout; an example is in the font metric files supplied with Tom Rokicki's *dvips*, which is basically the same as Knuth's layout, but has some small differences.

This means that the definition of the `\c` macro is going to vary according to the font we are using; hence the font attribute of *encoding.*[3] NFSS2 provides *hooks*, so that we can supply TeX code which will be activated when the encoding changes. Alternatively, if we know we are going to use a different encoding throughout a document, we can just make changes in a style file.

The normal user will have to make one (important) choice: is she going to use the current Computer Modern Roman layout, or the new Cork layout? If she chooses the latter (and this means the 'dc' fonts must be available), then the style file `dclfont.sty` is supplied to set up all the necessary macros.[4] If use of PostScript fonts is proposed, slight complications ensue. In the first case (original TeX layout), then the PostScript fonts must be in exactly the right layout; users of Rokicki's metric files and *dvips* will need to use the style file `psnfss.sty`, and use the command `\RokickiExtras` in the document preamble to fix some macros. In the case of using Cork encoding, it will be necessary to get metrics for the PostScript fonts which remap the layout correctly, and to ensure that the PostScript interpreter in the printer knows about it too. This is discussed in detail in the *PSNFSS2* sub-package supplied with NFSS2.

Assuming the encoding question has been sorted out, the important thing is to change all the family defaults. Setting up a document to use (say) PostScript Times Roman throughout is trivial; we simply write:

```
\renewcommand{\rmdefault}{ptm}
\renewcommand{\sfdefault}{phv}
\renewcommand{\ttdefault}{pcr}
```

[3] This is an addition to the NFSS since version 1.

[4] This will become the *default* in LaTeX3.

Type	*Meaning*	*Type*	*Meaning*
`\mathord`	ordinary	`\mathop`	large operator
`\mathbin`	binary operator	`\mathrel`	relation
`\mathopen`	opening	`\mathclose`	closing
`\mathpunct`	punctuation	`\mathalpha`	alphabetic

Table 4: Math symbol types

in a style file or in the document preamble, and we will have changed the standard for normal, sans-serif and typewriter setting.

Some users will not be working with anything like the ASCII layout at all, but will be typesetting in something like Cyrillic, Devanagari or Elvish. The *encoding* attribute for fonts makes it easy to swap between different systems in the same document, and call up whatever macros are needed for a particular situation. New encodings can be defined, and old ones redefined. The detailed commands for declaring new encodings, and for declaring new font families, are described in the NFSS2 documentation.

Appendix 2: Installation

This section is relevant to the reader only if NFSS2 has not already been installed on her computer.

One of the characteristics of the new work being undertaken on LaTeX is that Knuth's principles of literate programming are being applied, which means that a single documented source is supplied from which the user can either generate printed documentation, or produce a useable file for TeX input. The NFSS2 distribution cannot therefore be used until it has been unpacked and installed; this is all done using TeX itself, and the only additional package needed by the first time user is the *docstrip* macros, which should always be supplied with NFSS2. The installation is in two parts:

1. We must first get a set of `fd` (and some LaTeX `.sty`) files ready for use; this is done by running plain TeX or LaTeX on the file `main.ins` which generates all the files we need. It will probably ask some questions as it runs about what fonts you have, but it won't matter much if you get the answers wrong (it may create `fd` files for strange fonts you don't have, but if you never try to use them, that does not matter).
2. Now we need to create a new LaTeX *format file* (usually with a suffix of `.fmt`); this can seem a slightly forbidding procedure, but should be explained in the documentation with your TeX; what happens is that a special version of TeX is run, called iniTeX, which reads the basic LaTeX macros, and hyphenation patterns, and dumps a fast loading version which you place where normal TeX can find it. On some systems there is actually an `initex` command, but on others it is an option to normal TeX. Thus, if you use the emTeX package, you type `tex -i` to use the right portion of the program. You *must* use this special TeX, or you cannot use the hyphenation system.

   Creating the NFSS2 format is straightforward — just run iniTeX on the file `nfss2ltx`, and type `\dump` in response to the `*` prompt when it finishes loading files.[5]

If you want to test your new LaTeX format now, work in the directory where you placed the NFSS2 files; otherwise move `nfss2ltx.fmt` to the directory where your TeX looks for format files, and copy all the `fd` files to a directory which TeX searches for inputs (along with any `.sty` files created in stage 1 above). Depending on how your TeX system works, you may have to do some more work to access the format file; thus we might edit the `latex.bat` file[6] under DOS, or make a new symbolic link to `virtex` if we run the *web2c* Unix TeX.

If you wish to find out more about NFSS2, and maybe print the documentation of its internal workings (not for the faint of heart!), read the file `readme.mz8` for details of how to proceed.

◇ Sebastian Rahtz
ArchaeoInformatica
12 Cygnet Street
York Y01 2JR
UK
`spqr@minster.york.ac.uk`

[5] Unless you are a rather advanced guru, and commonly add extra features to your format files.

[6] It is strongly advised that you do *not* make your new LaTeX an option, by making a new command, but that you use it for all your work from now on.

A Pragmatic Approach to Paragraphs

Philip Taylor

Something that never ceases to amaze me is just how many TeX users (including some who are quite eminent!) are familiar with the most arcane areas of TeX, yet when faced with what should be the simplest task of all — that of persuading a given paragraph to set correctly, without either generating `underfull \hbox` messages or abusing `\spaceskip` and its ilk — seem unable to achieve anything approaching a satisfactory solution. Whenever I receive a TeX document from such a user, and process it to find a dozen or so warning messages which the author seems quite happy to ignore, or to find the whole document set with grossly exaggerated interword space and flexibility, I ask myself why this apparently simple task should seem so difficult to so many.

To be fair, part of the blame must be said to lie with Knuth, for while Boson SlowCoach and its rivals would seem happy to set a paragraph with t h e m o s t a p p a l l i n g l e t t e r s p a c i n g, or to set a line with just three words and the most enormous   interword   space, Don took the decision that TeX would have none of this: either a paragraph would set correctly, or it would not set at all! And of course, most if not all of us agree with Don; for why else would we eschew the WYSIWYG wonders of Boson for the cabalistic complexity of the TeX language (unless, of course, we are all intellectual masochists, which I sometimes suspect). But also, to continue to be fair, part of the blame must be said to lie with *The TeXbook*; for whilst it more than adequately describes the various parameters which govern TeX's setting of paragraphs, it is somewhat less forthcoming about the methods by which suitable values for those parameters may be established.

In this article, I hope to present what I term '*a pragmatic approach to paragraphs*', for until some mathematical genius comes up with a formula or an algorithm by which suitable values for these parameters may be determined for any given combination of font, measure,[1] indentation, hyphenation patterns, etc., etc., etc., lesser mortals such as I will continue to have the unenviable task of typesetting text to some apparently arbitrary combination of these variables, and of convincing those for whom we are typesetting that it will *not* go to bromide until it meets our own somewhat exacting standards of æsthetic excellence, as well as those of Don and TeX...

[1] I use 'measure' in the typesetters' sense, meaning the width of the printed page excluding margins; for multi-column work, it refers to the width of a single column excluding margins and gutters.

We should start by considering those parameters which (a) most closely affect whether or not a given paragraph will set correctly, and (b) may reasonably be deemed to be within the ægis of the typesetter, as opposed to those which affect the setting but which are strictly under the control of the designer. The following table, although not exhaustive, lists some of the more important of the parameters which come under these two headings:

Typesetter specified	Designer specified
`\emergencystretch`	`\font`
`\pretolerance`	`\hsize`
`\tolerance`	`\patterns`
`\hbadness`	`\parindent`
`\hfuzz`	`\fontdimen` *n*

Of course, the designer will probably want to set upper bounds on even the entries in the typesetter's column, whilst the typesetter would be well advised indeed not to meddle with the entries in the designer's column (if he or she ever wants to be employed again!).

As to the method, I believe it to be simplicity itself, albeit somewhat complex to explain:

1. Process the text with sensible default values for the five 'typesetter's parameters'. Suitable defaults might be:

   ```
   \emergencystretch = 0 pt
   \pretolerance = 150
   \tolerance = 250
   \hbadness = 150
   \hfuzz = 0 pt
   ```

   If no error message issues from TeX, all is well and the task is complete.

   If not, then the error messages must be classified into two sets: those representing `overfull \hbox`es (and therefore true errors), and those representing `underfull \hbox`es (and therefore warnings rather than errors). Of these, the true errors must be addressed first.

2. Re-process the text, but this time specify `\tolerance = 9999`; this is most easily done by removing the assignment to `\tolerance` from the preamble, and initialising it on the command-line itself, as in:

   ```
   TeX "\tolerance = 9999 \input text"
   ```

Again the error messages must be classified into true errors and warnings, on the same basis as before.

If there are no longer any true errors, go to step 7; if there are still true errors, then the text and format may reasonably be considered to be pathologically bad, and further action will be required.

3. Examine the magnitude of the `overfull \hbox`es; if it is not more than the designer's specified upper bound on `\hfuzz`, set `\hfuzz` to the maximum overrun and go to step 7.
4. There are *still* `overfull \hbox`es. Sigh deeply. Examine the log to ascertain the line(s) in which these occur, and determine whether the problem is one of inadequate hyphenation (which is usually soluble), or one of over-wide tables, unbreakable formulæ, etc. If the problem is caused by inadequate hyphenation, go to step 6.
5. The problem lies outside the scope of this paper! Consider setting the offending table in a smaller font, reducing `\tabskip`, etc. Ask the author if the unsplittable formula *could*, in fact, be split. Apply your own heuristics to the task, and re-join this procedure from step 7 when you have resolved the difficulty.
6. The problem is caused by inadequate hyphenation. Ascertain by inspection whether an addition is needed to the `\hyphenation` list, or whether the offending word needs explicit discretionary hyphens to be added. An addition to the hyphenation list would be in order if there were nothing unusual about the word, but insufficient or poorly placed hyphenation points were indicated; explicit discretionary hyphens would be required if the word contained some hyphenation-inhibiting character, such as an accent, or if it were not preceded by glue. Augment the hyphenation and repeat from step 2.
7. Success! There are no more true errors. Now all that remains is to optimise the document, such that the final version represents the 'best possible setting', in some vague sense.

   Note the worst instance of badness in the `underfull \hbox`es. Set `\tolerance` to this value, and `\hbadness` to one less, and re-process the document. There should be no true errors, and no `underfull \hbox`es whose badness exceeds `\tolerance`.

   Now comes the surprising part. It might reasonably be thought that we have determined a lower bound on `\tolerance`, yet for many documents this proves not to be the case. Set `\tolerance` to one less than the value set in the step above, and re-process the document. It *may* still set correctly! The reasons for this are subtle, but may easily be understood by realising that TeX's concept of an 'ideal' paragraph is one in which the *overall* badness is minimised; TeX is *not* interested in minimising the badness of any one line. Thus we may now have a paragraph in which two or more lines are 'bad' in some sense, whilst the badness of the worst line has been reduced. Overall the paragraph is 'worse', but æsthetically it may *appear* 'better' (a paragraph consisting entirely of loose lines may look better than one in which one line stands out as being extremely loose).

   If the paragraph set correctly with `\tolerance` one less than the apparent lower bound, it may well do so again! Every time that the paragraph sets without true error, set `\tolerance` to one less than the reported worst badness and repeat. Eventually no further reduction in `\tolerance` will be possible, and an `overfull \hbox` will occur; re-instate the previous value and stop. Global optimisation is now complete, and the optimal value for `\tolerance` has been determined. If is this less than or equal to the designer's specified upper bound, then our work is done; if not, we will need to invoke `\emergencystretch`, and then, perhaps, to proceed to local optimisation.
8. Reduce `\tolerance` to the designer's specified upper bound, and set `\emergencystretch` to a small positive dimension. The behaviour of `\emergencystretch`, and in particular its interaction with other related parameters, is poorly understood, and indeed is the subject of research for the LaTeX3 project. However, a sound rule of thumb is to set it to `1 em` (based on the primary text font); this value, strange as it may seem, appears equally suitable for both wide and narrow measures.

   Set `\hbadness` to one less than the value of `\tolerance` and re-process the document. If `overfull` boxes are reported, then we have a problem: we *could* increase the value of `\emergencystretch`, but this rapidly leads to severely `underfull` boxes and appalling æsthetics; it is probably better to proceed to local optimisation in these circumstances, which is the next step anyway.

For each line reported as being either `overfull` or `underfull`, consider the associated paragraph and see if additional discretionary hyphens might enable TeX to pack a few extra or a few less glyphs on the line; consider also whether one or more of the parameters listed in Appendix A (e.g. `\doublehyphendemerits`, `\exhyphenpenalty`) might be forcing TeX to adopt a less-than-perfect setting for this paragraph. If necessary, consider modifying one or more of these parameters just for the duration of the paragraph, by enclosing the latter between a `\begingroup`/`\endgroup` pair, modifying the parameter(s) immediately after the `\begingroup`. Remember that the paragraph will need to terminate with a `\par` or blank line *before* the `\endgroup` if the parameter change(s) is/are to have any effect.

Continue local optimisation, with no adjustments to `\tolerance` or `\hbadness`, until no further improvement can be achieved; if `overfull` boxes remain, the best option is to invite the author to re-cast the paragraph(s); if only `underfull` boxes remain, discretion is called for: visually inspect the offending paragraph(s), and invite the author to re-cast if and only if æsthetic considerations warrant it.

Remember that `\tolerance` has been reduced from its 'optimal' value to the designer's upper bound; if the author is unwilling to recast an offending paragraph, then bracket that paragraph in a `\begingroup`/`\endgroup` pair, as above, and for the duration of that paragraph only, re-set `\tolerance` to its 'optimal' value (and advise the designer that this was necessary).

Local optimisation is now also complete. Modify the preamble to incorporate the experimentally determined values for `\tolerance` and `\hbadness`. The justification for setting `\hbadness` to one less than `\tolerance` was not given above: it is simply that by setting it to the recommended value, it is possible to check that the experimentally determined value for `\tolerance` was in fact necessary, whilst suppressing TeX's reporting of lesser warnings; if TeX fails to report an `underfull \hbox` of badness equal to `\tolerance`, some error has been made. Of course, for distribution, `\hbadness` should be set equal to `\tolerance` so as to eliminate spurious warning messages.

This procedure may appear complex, but it is in fact very straightforward, and is certainly intuitive once the subtlety of repeated reductions in `\tolerance` is fully appreciated. The steps involved require a bare minimum of editing, and maximum advantage is taken of the fact that TeX's behaviour can be influenced by command-line parameters. The procedure has been consistently applied as this (and several previous) articles were written, and is in regular day-to-day use at my College, where productivity is valued even more than æsthetic excellence!

◇ Philip Taylor
The Computer Centre, RHBNC,
University of London, U.K.
<P.Taylor@Vax.Rhbnc.Ac.Uk>

Appendix A: Summary of paragraph-related parameters

Integer parameters:

```
\adjdemerits
\doublehyphendemerits
\exhyphenpenalty
\finalhyphendemerits
\hbadness
\hyphenpenalty
\linepenalty
\looseness
\pretolerance
\tolerance
```

Dimension parameters:

```
\emergencystretch
\hangindent
\hfuzz
\hsize
\parindent
```

Glue parameters:

```
\leftskip
\parfillskip
\rightskip
\spaceskip
\xspaceskip
```

Miscellaneous parameters:

```
\fontdimen 2
\fontdimen 3
\fontdimen 4
\fontdimen 7
\parshape
```

Letters

Truth in Indexing

Jonathan Fine reports in *TUGboat* 13 #4, page 495, under the heading "Too Many Errors", that Knuth is in error when he uses the index-reminder scheme described on pages 424–425 of *The TEXbook*. Padding the page numbers is irrelevant; check Knuth's original version in the lower display on page 424, where you will surely notice the `\noexpand` to which Fine objects. As a matter of fact, Knuth himself produced just the "error" Fine describes when preparing the index for *Concrete Mathematics*, at least in the first impression of that volume. Look up Christian Goldbach in the index for the first printing; the page number given there is 583 but Goldbach is only mentioned in the first line of page 584.

I wonder if Fine has overlooked the paragraph divided between pages 424 and 425 of *The TEXbook*, in which Knuth describes his philosophy regarding index construction. He may also have overlooked several references in the tutorials in volumes 10, 11, and 12 of *TUGboat*, to my belief that automation can sometimes be carried to excess. Indeed, the effort required to avoid the necessity for proof reading one's document by automating all of its components that might be subject to automation will often be significantly greater than the effort needed to accomplish the task with reasonable restraint in this regard. Wherever we concentrate our efforts, we are still required to pay close attention to the results generated by them.

The only printings of *Concrete Mathematics* that I have seen, so far, are the first and sixth. Studying the differences between them can be both entertaining and informative, even in connection with the bibliography and the index. Between these two printings, two additional items were interpolated before the entry that originally appeared at the top of page 584 of the bibliography, hence the problem with which Fine was so concerned vanished as a result of natural causes. There are at least two possibilities: One's book is so popular that it must be reprinted frequently and in the process trivial errors that do not vanish automatically are easily fixed; if the book doesn't require reprinting, it may be that it has few readers or none, in which case the distinction between gross errors and trivial errors simply evaporates.

Let me propose, for further discussion in this context, what I would like to call the Occam-Ludd Razor: *Entities should not be multiplied beyond necessity, and automation should be encouraged when it simplifies things and avoided when it does not.*

Lincoln Durst
46 Walnut Road
Barrington, RI 02806 U.S.A.
`LKD@Math.AMS.org`

Macros

Letter-Spacing in TEX

Philip Taylor

One of the joys of looking at a page of TEXset material, particularly when compared to the majority of today's magazines, is the uniform grayness of the page. Whereas many of today's top-end DTP packages frequently achieve justification through the use of letter-spacing, TEX prefers to distribute any spare white space between words rather than between letters. Indeed, there *are* no intrinsic facilities within TEX which would permit the use of letter-spacing, even were it desired.

And yet, there *are* times when letter-spacing is effective: in running heads, for example, or for mastheads or titles. In some languages, letter-spacing (then more properly termed *Sperrsatz*) is used for stress or emphasis, much as we use italicisation in English. For these purposes, then, rather than as a general letter-spacing tool, I have developed the following code, which allows at most a single line of text to be letter-spaced. It is worth pointing out straight away that there are some restrictions on the text, although considerably fewer than in earlier releases: it should not, for example, contain unprotected `\accents`, (neither explicit, using the `\accent` primitive, nor implicit, through the use of control symbols as `\'`), although either form may be used provided that the accent and its accompanying letter are concealed within a brace-delimited group; control-sequences without arguments may occur in the text to be typeset, but if they expand to text, that text will not be letter-spaced, and thus it is difficult, although not impossible, to typeset T H E J O Y O F L E T T E R - S P A C E D T E X ! (And, of course, it should contain no lower-case text: "A man who would letter-space lower-case text would probably steal sheep [Goudy]"). But these are the only restrictions: provided that the text is straightforward, considerable flexibility is offered in the degree of letter-spacing achieved.

The design desideratum was very simple: create a `\letterspace` macro which would provide *at least* the same degree of flexibility in determining the degree of letter-spacing as TEX already provides for the specification of `\hbox`es and `\vbox`es, and using the same syntax. Thus we must allow `\letterspace {`*text*`}`, `\letterspace to` *dimen* `{`*text*`}` and `\letterspace spread` *dimen* `{`*text*`}`. In addition, I sought to provide even greater flexibility, by providing the user with additional information concerning the text to be letter-spaced: in particular, its natural height, depth and width. These are all made available through reserved TEX control sequences: `\naturalheight`, `\naturaldepth` and `\naturalwidth`. By combining the two, a very elegant syntax is provided for letter-spacing text: for example, `\letterspace to 1.5 \naturalwidth {`*text*`}`, or `\letterspace spread 0.5 \naturalwidth {`*text*`}` (both of which have the same effect).

The code is presented twice: once as a monolithic entity, so that the reader can take in its structure at a glance, and once in annotated form, so that its inner workings can be clarified; first the code as monolithic entity:

```
%%% Open control sequences

\newdimen \naturalwidth
\newdimen \naturaldepth
\newdimen \naturalheight


%%% Concealed control sequences

\expandafter \chardef
        \csname \string @code\endcsname =
                \the \catcode '\@
\catcode '\@ = 11


\newbox \l@tterspacebox
\newtoks \l@tterspacetoks


\def \sp@ce { }
\def \hsss
    {\hskip 0 pt plus 1 fill minus 1 fill\relax}


\let \@nd = \empty
\let \@x = \expandafter


\let \sp@cetoken = \relax
\edef \t@mp {\let \sp@cetoken = \sp@ce}
\t@mp \let \t@mp = \undefined


%%% Primary (user-level) macro

\def \letterspace #1#%
    {\def \hb@xmodifier {#1}%
     \afterassignment \l@tterspace
     \l@tterspacetoks =
    }


%%% Secondary (implementation-level) macro

\def \l@tterspace
    {\setbox \l@tterspacebox = \hbox
                {\the \l@tterspacetoks}%
     \naturalwidth =  \wd \l@tterspacebox
     \naturaldepth =  \dp \l@tterspacebox
     \naturalheight = \ht \l@tterspacebox
     \hbox \hb@xmodifier
     \bgroup
           \hss
           \@x \l@tt@rspace
           \the \l@tterspacetoks {}\@nd
           \hss
     \egroup
    }


%%% Tertiary (implementation-level) macro

\def \l@tt@rspace #1\@nd
    {\ifx  \@nd #1\@nd
           \let \n@xt = \relax
     \else
           \p@rtition #1\@nd
           \ifx \h@ad \sp@ce \hsss \h@ad \hsss
           \else
                \h@ad
                \ifx \t@il \@nd \else \hsss \fi
           \fi
           \@x \def \@x \n@xt \@x
                {\@x \l@tt@rspace \t@il \@nd}%
     \fi
     \n@xt
    }


%%% Adjunct macros -- list partitioning

\def \p@rtition #1\@nd
    {\m@kespaceexplicit #1\@nd
     \@x \p@rtiti@n \b@dy \@nd
    }

\def \p@rtiti@n #1#2\@nd
    {\def \h@ad {#1}%
     \def \t@il {#2}%
    }


%%% Adjunct macros -- <space>... -> {<space>}...
```

```
\def \m@kespacexplicit #1\@nd
    {\futurelet \h@ad \m@kesp@cexplicit #1\@nd}

\def \m@kesp@cexplicit #1\@nd
    {\ifx \h@ad \sp@cetoken
          \@x \def \@x \b@dy
                      \@x {\pr@tectspace #1\@nd}%
     \else
          \def \b@dy {#1}%
     \fi
    }%

\@x \def \@x \pr@tectspace \sp@ce #1\@nd {{ }#1}


%%% re-instate category code of commercial-at

\catcode `\@ = \the \@code
```

The same code is now presented in annotated form, each group of declarations, and each individual macro, being preceded by a discussion of their purpose and functionality:

As far as is possible, 'inaccessible' control sequences (i.e. control sequences containing one or more commercial-at s) are used to minimise the risk of accidental collision with user-defined macros; however, the control sequences used to access the natural dimensions of the text are required to be user accessible, and therefore must contain only letters.

```
%%% Open control sequences

\newdimen \naturalwidth
\newdimen \naturaldepth
\newdimen \naturalheight
```

All control sequences defined hereafter are inaccessible to the casual user, including the control sequence used to store the category code of commercial-at; this catcode must be saved in order to be able to re-instate it at end of module, as we have no idea in what context the module will be `\input`. Having saved the catcode of commercial-at, we then change it to `11` (i.e. that of a letter) in order to allow it to be used in the cs-names which follow.

```
%%% Concealed control sequences

\expandafter \chardef
        \csname \string @code\endcsname =
                \the \catcode `\@
\catcode `\@ = 11
```

We will need a box register in order to measure the natural dimensions of the text, and a token-list register in which to save the tokens to be letter-spaced.

```
\newbox \l@tterspacebox
\newtoks \l@tterspacetoks
```

We will need to test whether a particular macro expands to a space, so we will need another macro which does so expand with which to compare it; we will also need a 'more infinite' variant of `\hss`.

```
\def \sp@ce { }
\def \hsss
    {\hskip 0 pt plus 1 fill minus 1 fill\relax}
```

Many of the macros will use delimited parameter structures, typically terminated by the reserved control sequence `\@nd`; we make this a synonym for the empty macro (which expands to nothing), so that should it accidentally get expanded there will be no side effects. We also define a brief synonym for `\expandafter`, just to keep the individual lines of code reasonably short.

```
\let \@nd = \empty
\let \@x = \expandafter
```

We will also need to compare a token which has been peeked at by `\futurelet` with a space token; because of the difficulty of accessing such a space token (which would usually be absorbed by a preceding control word), we establish `\sp@cetoken` as a synonym. The code to achieve this is messy, because of the very difficulty just outlined, and so we 'waste' a control sequence `\t@mp`; we then return this to the pool of undefined tokens.

```
\let \sp@cetoken = \relax
\edef \t@mp {\let \sp@cetoken = \sp@ce}
\t@mp \let \t@mp = \undefined
```

The user-level macro `\letterspace` has exactly the same syntax as that for `\hbox` and `\vbox`, as explained in the introduction; the delimited parameter structure for this macro ensures that everything up to the open brace which delimits the beginning of the text to be letter-spaced is absorbed as parameter to the macro, and the brace-delimited text which follows is then assigned to the token-list register `\l@tterspacetoks`; `\afterassignment` is used to regain control once the assignment is complete.

```
%%% Primary (user-level) macro

\def \letterspace #1#%
    {\def \hb@xmodifier {#1}%
     \afterassignment \l@tterspace
     \l@tterspacetoks =
    }
```

Control then passes to `\l@tterspace`, which starts by setting an `\hbox` containing the text to be typeset; the dimensions of this box (and therefore of the text) are then saved in the open control sequences previously declared, and the `\hbox` becomes of no further interest.

A new `\hbox` is now created, in which the same text, but this time letter-spaced, will be set; the box starts and ends with `\hss` glue so that if only a single character is to be letter-spaced, it will be centered in the box. If two or more characters are to be letter-spaced, they will be separated by `\hsss` glue, previously declared, which by virtue of its greater degree of infinity will completely override the `\hss` glue at the beginning and end; thus the first and last characters will be set flush with the edges of the box.

The actual mechanism by which letter-spacing takes place is not yet apparent, but it will be seen that it is crucially dependent on the definition of `\l@tt@rspace`, which is expanded as the box is being set; the `\@x` (`\expandafter`) causes the actual tokens stored in `\l@tterspacetoks` to be made available as parameter to `\l@tt@rspace` rather than the token-list register itself, as it is these tokens on which `\l@tt@rspace` will operate. The `\@nd` terminates the parameter list, and the `{}` which immediately precedes it ensures that that there is always a null element at the end of the list: without this, the braces which are needed to protect an accent/character pair would be lost if such a pair formed the final element of the list, at the point where they are passed as the second (delimited) parameter to `\p@rtiti@n`; by definition, TEX removes the outermost pair of braces from both simple and delimited parameters during parameter substitution if such braces form the first and last tokens of the parameter, and thus if a brace-delimited group ever becomes the second element of a two-element list, the braces will be irrevocably lost. The `{}` ensure that such a situation can never occur.

```
%%% Secondary (implementation-level) macro

\def \l@tterspace
    {\setbox \l@tterspacebox = \hbox
                 {\the \l@tterspacetoks}%
     \naturalwidth =  \wd \l@tterspacebox
     \naturaldepth =  \dp \l@tterspacebox
     \naturalheight = \ht \l@tterspacebox
     \hbox \hb@xmodifier
     \bgroup
           \hss
           \@x \l@tt@rspace
           \the \l@tterspacetoks {}\@nd
           \hss
     \egroup
    }
```

The next macro is `\l@tt@rspace`, which forms the crux of the entire operation. The text to be letter-spaced is passed as parameter to this macro, and the first step is to check whether there is, in fact, any such text; this is necessary both to cope with pathological usage (e.g. `\letterspace {}`), and to provide an exit route, as the macro uses tail-recursion to apply itself iteratively to the 'tail' of the text to be letter-spaced; when no elements remain, the macro must exit.

Once the presence of text has been ensured, the token-list representing this text is partitioned into a head (the first element), and the tail (the remainder); at each iteration, only the head is considered, although if the tail is empty (i.e. the end of the list has been reached), special action is taken.

If the first element is a space, it is treated specially by surrounding it with two globs of `\hsss` glue, to provide extra stretchability when compared to the single glob of `\hsss` glue which will separate consecutive non-space tokens; otherwise, the element itself is yielded, followed by a single glob of `\hsss` glue. This glue is suppressed if the element is the last of the list, to ensure that the last token aligns with the edge of the box.

When the first element has been dealt with, the macro uses tail recursion to apply itself to the remaining elements (i.e. to the tail); `\@x` (`\expandafter`) is again used to ensure that the tail is expanded into its component tokens before being passed as parameter.

```
%%% Tertiary (implementation-level) macro

\def \l@tt@rspace #1\@nd
    {\ifx  \@nd #1\@nd
           \let \n@xt = \relax
     \else
           \p@rtition #1\@nd
           \ifx \h@ad \sp@ce \hsss \h@ad \hsss
           \else
                \h@ad
                \ifx \t@il \@nd \else \hsss \fi
           \fi
           \@x \def \@x \n@xt \@x
                {\@x \l@tt@rspace \t@il \@nd}%
     \fi
     \n@xt
    }
```

The operation of token-list partitioning is conceptually simple: one passes the token list as parameter text to a macro defined to take two parameters, the first simple and the second delimited; the first element of the list will be split off as parameter-1, and the remaining material passed as parameter-2. Unfortunately this naïve approach fails when the first element is a bare space, as the semantics of TEX prevent such a space from ever being passed as a simple parameter (it could be passed as a delimited parameter, but as one does not know what token follows the space, defining an appropriate delimiter structure would be tricky if not impossible). The technique used here relies upon the adjunct macro `\m@kespacexplicit`, which replaces a leading bare space by a space protected by braces; such spaces may legitimately be passed as simple parameters. Once that operation has been completed, the re-constructed token list is passed to `\p@rtiti@n`, which actually performs the partitioning as above; again `\@x` (`\expandafter`) is used to expand `\b@dy` (the re-constructed token list) into its component elements before being passed as parameter text.

```
%%% Adjunct macros -- list partitioning

\def \p@rtition #1\@nd
    {\m@kespacexplicit #1\@nd
     \@x \p@rtiti@n \b@dy \@nd
    }

\def \p@rtiti@n #1#2\@nd
    {\def \h@ad {#1}%
     \def \t@il {#2}%
    }
```

The operation of making a space explicit relies on prescience: the code needs to know what token starts the token list before it knows how to proceed. Prescience in TEX is usually accomplished by `\futurelet`, and this code is no exception: `\futurelet` here causes `\h@ad` to be `\let` equal to the leading token, and control is then ceded to `\m@kesp@cexplicit`.

The latter compares `\h@ad` with `\sp@cetoken` (remember the convolutions we had to go through to get `\sp@cetoken` correctly defined in the first place), and if they match (i.e. if the leading token is a space), then `\b@dy` (the control sequence through which the results of this operation will be returned) is defined to expand to a protected space (a space surrounded by braces), followed by the remainder of the elements; if they do not match (i.e. if the leading token is *not* a space), then `\b@dy` is simply defined to be the original token list, unmodified. If the leading token *was* a space, it must be replaced by a protected space: this is accomplished by `\pr@tectspace`.

The `\pr@tectspace` macro uses a delimited parameter structure, as do most of the other macros in this suite, but the structure used here is unique, in that the initial delimiter is a space. Thus, when a token-list starting with a space is passed as parameter text, that space is regarded as matching the space in the delimiter structure and removed; the expansion of the macro is therefore the tokens remaining once the leading space has been removed, preceded by a protected space `{ }`.

```
%%% Adjunct macros -- <space>... -> {<space>}...

\def \m@kespacexplicit #1\@nd
    {\futurelet \h@ad \m@kesp@cexplicit #1\@nd}

\def \m@kesp@cexplicit #1\@nd
    {\ifx \h@ad \sp@cetoken
          \@x \def \@x \b@dy
                     \@x {\pr@tectspace #1\@nd}%
     \else
          \def \b@dy {#1}%
     \fi
    }%

\@x \def \@x \pr@tectspace \sp@ce #1\@nd {{ }#1}
```

The final step is to re-instate the category code of commercial-at.

```
%%% re-instate category code of commercial-at

\catcode '\@ = \the \@code
```

Thus letter-spacing is accomplished. The author hopes both that the code will be found useful and that the explanation which accompanies it will be found informative and interesting.

◇ Philip Taylor
The Computer Centre, RHBNC,
University of London, U.K.
<P.Taylor@Vax.Rhbnc.Ac.Uk>

Abstracts

Les Cahiers GUTenberg
Contents of Recent Issue

Numéro 15 — avril 1993*

A. COUSQUER, Éditorial : changements...
[Changes]; pp. 1–2

In his first editorial as GUTenberg's president, Alain Cousquer discusses some of the user group's activities, changes in personnel (Bernard Gaulle has taken a leave of absence; Olivier Nicole has been posted overseas), changes in the focus for TEX users and therefore for GUTenberg. An introduction to each of the five articles in the issue is also provided. He concludes with an outline of upcoming activities, and a call for volunteers.

DANIEL TAUPIN, Commentaires sur la portabilité de TEX [Comments on TEX portability]; pp. 3–31

This article is preceded by an explicit statement by the editorial board that the opinions expressed are those of the author, not of GUTenberg, nor of the editorial board, who say reactions to the article will be published in future issues.

The paper is a long discussion of various aspects of the portability issue — how and where it is threatened, and what actions can be taken by authors to avoid undermining this extremely significant feature of TEX. The various points of concern are elaborated in detail while remaining accessible and understandable to most users of TEX. The author concludes with a summary of nine points to keep in mind in order to help preserve TEX's portability, and some example code to help achieve some of these goals.

A World-Wide Window on TEX: TUG '93 Aston University, UK ; p. 32

JACQUES ANDRÉ, Titres: à lire, à voir ou à dire ? [Titles: to read, to see, or to say out loud?]; pp. 33–42

Author's summary: "This article provides authors with some French typographic rules concerning titles. As well, examples are given which demonstrate that titles are more 'seen' than 'read'. The author also outlines a good way to decide where to split long titles: say them out loud first."

The article concludes with a list of 16 references, which should prove useful to those who deal with French-language documents.

BERNARD GAULLE, Nos outils du bureau : les arguments du choix [Office tools: the selection]; pp. 43–45

Author's summary: "A few years ago, the computing centre decided to use LATEX and other tools to produce its own documents. The author presents an overview of his adventures during the installation of TEX, and preparations for its use. This first article discusses the arguments, pro and con, which went into the decision to select LATEX for the typesetting of in-house documents. Next episode: the [startup] hiccups."

O. NICOLE, J. ANDRÉ et B. GAULLE, Notes en bas de pages : commentaires [Footnotes: comments]; pp. 46–52

This article is a follow-up by the editorial staff of the *Cahiers* to an article which appeared in *Cahiers* 12 (pp. 57–70), written by J. André and P. Louarn. The current article, on LATEX footnotes, has an interesting presentation — each section is written by one of the authors, with his own particular comment or trick.

These include: ensuring that notes to a table appear on the same page as the table (Nicole); a special case of embedded `description` environments (Nicole); suppressing footnote flags in Table of Contents entries for titles which *do* have footnotes attached (André); changing the characters invoked by `\fnsymbol` (André); an alternative solution for multiple footnote calls to the same footnote text (Gaulle); and mention of a style option `footnpag` which renumbers footnotes on every page (Gaulle).

YANNIS HARALAMBOUS, Conventions concernant les polices DC et les langues [Conventions concerning DC fonts and languages]; pp. 53—61

This is an updated French version of an article which appeared originally in *TEX and TUG NEWS* 1,4:3–10. Included with this version is a new section outlining the current activities and areas of concern of the Technical Working Group which Yannis chairs: TWG on Multiple Language Coordination.

(Summaries by Christina Thiele)

* Issue 15 of the *Cahiers* was accompanied by a letter explaining that the recently mailed copies of the Prague proceedings (*EuroTEX'92*) were actually no. 14 of the *Cahiers*. Through an error, that information was not printed on copies of the proceedings.

Calendar

1993

Aug	17	***TUGboat* Volume 14, 3rd regular issue:** Deadline for receipt of *technical* manuscripts.
Aug	23 – 27	**TUG Course:** Intensive LaTeX, Ottawa, Canada
Aug	25	TeX–Stammtisch, Hamburg, Germany. For information, contact Reinhard Zierke (`zierke@informatik.uni-hamburg.de`; telephone (040) 54715-295).
Sep	2	TeX–Stammtisch at the Universität Bremen, Germany. For information, contact Martin Schröder (`115d@alf.zfn.uni-bremen.de`; telephone 0421/628813).
Sep	14	***TUGboat* Volume 14, 3rd regular issue:** Deadline for receipt of news items, reports.
Sep	20	TeX-Stammtisch in Bonn, Germany. For information, contact Herbert Framke (`Herbert_Framke@BN.MAUS.DE`; telephone 02241 400018).
Sep	21	TeX-Stammtisch in Duisburg, Germany. For information, contact Friedhelm Sowa (`tex@ze8.rz.uni-duesseldorf.de`; telephone 0211/311 3913).
Sep	23 – 24	**TUG Course:** Book and Document Design with TeX, Boston, Massachusetts
Sep	23 – 24	9th Meeting of DANTE eV., Kaiserslautern. For information, contact Klaus Uttler (`uttler@rhrk.uni-kl.de`).
Sep	29	TeX–Stammtisch, Hamburg, Germany. (For contact information, see Aug 25.)
Oct	4 – 8	CyrTUG meeting, Pereslavl′-Zalesskiĭ, near Pleshchevo Lake. For information, contact Irina Makhovaya (`cyrtug@mir.msk.su`).
Oct	7	TeX–Stammtisch at the Universität Bremen, Germany. (For contact information, see Sep 2.)
Oct	18 – 22	**TUG Course:** Beginning/Intermediate TeX Santa Barbara, California
Oct	18	TeX-Stammtisch in Bonn, Germany. (For contact information, see Sep 20.)
Oct	19	TeX-Stammtisch in Duisburg, Germany. (For contact information, see Sep 21.)
Oct	27	TeX–Stammtisch, Hamburg, Germany. (For contact information, see Aug 25.)

TUG Courses, Santa Barbara, California

Oct	25 – 29	Intensive LaTeX
Nov	1 – 5	Advanced TeX and Macro Writing
Nov	8 – 9	Practical SGML and TeX

Nov	4	TeX–Stammtisch at the Universität Bremen, Germany. (For contact information, see Sep 2.)
Nov	10	**TUG Course:** SGML and TeX for Publishers, New York City
Nov	12	**TUG Course:** TeX for Publishers, Washington, DC
Nov	15	***TUGboat* Volume 15, 1st regular issue:** Deadline for receipt of *technical* manuscripts.
Nov	15	TeX-Stammtisch in Bonn, Germany. (For contact information, see Sep 20.)
Nov	16	TeX-Stammtisch in Duisburg, Germany. (For contact information, see Sep 21.)
Nov	18	12th NTG Meeting, "(LA)TeX user environment", Oce, Den Bosch, Netherlands. For information, contact Gerard van Nes (`vannes@ecn.nl`).
Nov	23	***TUGboat* Volume 14, 3rd regular issue:** Mailing date (tentative).

Status as of 15 August 1993

Nov 24	TEX–Stammtisch, Hamburg, Germany. (For contact information, see Aug 25.)
Dec 2	TEX–Stammtisch at the Universität Bremen, Germany. (For contact information, see Sep 2.)
Dec 13	***TUGboat* Volume 15, 1st regular issue:** Deadline for receipt of news items, reports.
Dec 20	TEX-Stammtisch in Bonn, Germany. (For contact information, see Sep 20.)
Dec 22	TEX–Stammtisch, Hamburg, Germany. (For contact information, see Aug 25.)
Dec 21	TEX-Stammtisch in Duisburg, Germany. (For contact information, see Sep 21.)

1994

TUG Courses, Santa Barbara, California

Jan 31 – Feb 4	Intensive LATEX
Feb 7 – 11	Beginning/Intermediate TEX
Feb 14 – 18	Advanced TEX and Macro Writing
Feb 28 – Mar 2	Modifying LATEX Style Files

Feb 15	***TUGboat* Volume 15, 2nd regular issue:** Deadline for receipt of *technical* manuscripts (tentative).
Mar 9	***TUGboat* Volume 15, 1st regular issue:** Mailing date (tentative).
Mar 15	***TUGboat* Volume 15, 2nd regular issue:** Deadline for receipt of news items, reports (tentative).
Apr 11 – 15	Four conferences, Darmstadt, Germany: • EP94, Electronic Pubishing, Document Manipulation and Typography (for information, contact `ep94@gmd.de`); • RIDT94, Raster Imaging and Digital Typography (for information, contact `ridt94@irisa.fr`); • TEP94, Teaching Electronic Publishing (for information, contact `ltsdyson@reading.ac.uk`); • PODP94, Principles of Document Processing (for information, contact `podp94@cs.umd.edu`). Deadline for submission of papers: 15 August 1993
May 23	***TUGboat* Volume 15, 2nd regular issue:** Mailing date (tentative).
Jul 24 – 29	SIGGRAPH'94: 21st International ACM Conference on Computer Graphics and Interactive Techniques. Orlando, Florida (for information, contact `siggraph-94@siggraph.org`, telephone 312-321-6830).

For additional information on the events listed above, contact the TUG office (805-963-1338, fax: 805-963-8358, email: `tug@tug.org`) unless otherwise noted.

Index of Advertisers

153	American Mathematical Society
153	ArborText
Cover 3	Blue Sky Research
154, 155	Kinch Computer Company
151	Micro Programs, Inc.
156	Y&Y

Late-Breaking News

Production Notes

Barbara Beeton

Input and input processing

Electronic input for articles in this issue was received by e-mail, and on diskette. In addition to text amd various code files processable directly by TeX, the input to this issue includes METAFONT source code. and several encapsulated PostScript files. More than 150 files were required to generate the final copy; over 100 more contain earlier versions of articles, auxiliary information, and records of correspondence with authors and referees. These numbers represent input files only; `.dvi` files, device-specific translations, and fonts (`.tfm` files and rasters) are excluded from the total.

Most articles as received were fully tagged for *TUGboat*, using either the `plain`-based or LaTeX conventions described in the Authors' Guide (see *TUGboat* 10, no. 3, pages 378–385). The macros are available from CTAN (the Comprehensive TeX Archive Network); see p. 100.

Almost 75% of the articles in this issue are in LaTeX, accounting for more than 80% of the pages.

Font work was required for three articles: Knappen" "Fonts for Africa" (p. 104), Cohen: "Zebnrackets" (p. 118), and Rahtz on PS fonts (p. 107).

Test runs of articles were made separately and in groups to determine the arrangement and page numbers (to satisfy any possible cross references). A file containing all starting page numbers, needed in any case for the table of contents, was compiled before the final run. Final processing was done in 2 runs of TeX and 3 of LaTeX, using the page number file for reference.

Beginning with Taylor's article on paragraphs (p. 138), all articles were processed with the `plain`-based `tugboat.sty`; all articles before that were prepared with LaTeX.

Output

The bulk of this issue was prepared at the American Mathematical Society from files installed on a VAX 6320 (VMS) and TeX'ed on a server running under Unix on a Solbourne workstation. Output was typeset on the Math Society's Compugraphic 9600 Imagesetter, a PostScript-based machine, using the Blue Sky/Y&Y PostScript implementation of the CM fonts, with additional fonts downloaded for special purposes.

No pasteup of camera-ready items or illustrations was required for this issue.

The output devices used to prepare the advertisements were not usually identified; anyone interested in determining how a particular ad was prepared should inquire of the advertiser.

Coming Next Issue

Bibliography prettyprinting and syntax checking

Nelson Beebe presents techniques for "solving the vexing issue of bibliography formatting" with reusable strings in BibTeX. This is the full text of his presentation at the 1993 TUG annual meeting at Aston, with additional details and copious examples.

The "operational requirement" for support of bibliographic references for LaTeX 3

David Rhead suggests that:

- LaTeX 3 should aim to support the principal citation schemes used in conventional publishing;
- consideration be given to a *modus vivendi* between LaTeX 3 and mainstream bibliography-formatting software.

Typesetting of ancient languages

The visual characteristics of ancient languages were based originally on manuscript traditions, not those of printing. Claudio Beccari provides some history and proposes an approach that, while not adhering strictly to ancient traditions, may be more suitable for modern presentations of ancient works.

Icons for TeX and METAFONT

Donald Knuth's new workstation has a graphical user interface instead of the operating system prompts he was used to previously. Using Duane Bibby's drawings, the alphabet, and other simple shapes, he has developed icons to represent the various file types used in the TeX system.

Institutional Members

The Aerospace Corporation, *El Segundo, California*

Air Force Institute of Technology, *Wright-Patterson AFB, Ohio*

American Mathematical Society, *Providence, Rhode Island*

ArborText, Inc., *Ann Arbor, Michigan*

ASCII Corporation, *Tokyo, Japan*

Brookhaven National Laboratory, *Upton, New York*

Brown University, *Providence, Rhode Island*

California Institute of Technology, *Pasadena, California*

Calvin College, *Grand Rapids, Michigan*

Carleton University, *Ottawa, Ontario, Canada*

Centre Inter-Régional de Calcul Électronique, CNRS, *Orsay, France*

CERN, *Geneva, Switzerland*

College Militaire Royal de Saint Jean, *St. Jean, Quebec, Canada*

College of William & Mary, Department of Computer Science, *Williamsburg, Virginia*

Communications Security Establishment, Department of National Defence, *Ottawa, Ontario, Canada*

Cornell University, Mathematics Department, *Ithaca, New York*

CSTUG, *Praha, Czech Republic*

E.S. Ingenieres Industriales, *Sevilla, Spain*

Elsevier Science Publishers B.V., *Amsterdam, The Netherlands*

European Southern Observatory, *Garching bei München, Germany*

Fermi National Accelerator Laboratory, *Batavia, Illinois*

Florida State University, Supercomputer Computations Research, *Tallahassee, Florida*

GKSS, Forschungszentrum Geesthacht GmbH, *Geesthacht, Germany*

Grinnell College, Computer Services, *Grinnell, Iowa*

Grumman Aerospace, Melbourne Systems Division, *Melbourne, Florida*

GTE Laboratories, *Waltham, Massachusetts*

Hungarian Academy of Sciences, Computer and Automation Institute, *Budapest, Hungary*

Institute for Advanced Study, *Princeton, New Jersey*

Institute for Defense Analyses, Communications Research Division, *Princeton, New Jersey*

Intevep S. A., *Caracas, Venezuela*

Iowa State University, *Ames, Iowa*

Los Alamos National Laboratory, University of California, *Los Alamos, New Mexico*

Louisiana State University, *Baton Rouge, Louisiana*

MacroSoft, *Warsaw, Poland*

Marquette University, Department of Mathematics, Statistics and Computer Science, *Milwaukee, Wisconsin*

Masaryk University, *Brno, Czechoslovakia*

Mathematical Reviews, American Mathematical Society, *Ann Arbor, Michigan*

Max Planck Institut für Mathematik, *Bonn, Germany*

National Research Council Canada, Computation Centre, *Ottawa, Ontario, Canada*

Naval Postgraduate School, *Monterey, California*

New York University, Academic Computing Facility, *New York, New York*

Nippon Telegraph & Telephone Corporation, Software Laboratories, *Tokyo, Japan*

Observatoire de Genève, Université de Genève, *Sauverny, Switzerland*

The Open University, Academic Computing Services, *Milton Keynes, England*

Personal TEX, Incorporated, *Mill Valley, California*

Politecnico di Torino, *Torino, Italy*

Princeton University, *Princeton, New Jersey*

Rogaland University, *Stavanger, Norway*

Ruhr Universität Bochum, Rechenzentrum, *Bochum, Germany*

Rutgers University, Computing Services, *Piscataway, New Jersey*

St. Albans School, *Mount St. Alban, Washington, D.C.*

Smithsonian Astrophysical Observatory, Computation Facility, *Cambridge, Massachusetts*

Space Telescope Science Institute, *Baltimore, Maryland*

Springer-Verlag, *Heidelberg, Germany*

Springer-Verlag New York, Inc., *New York, New York*

Stanford Linear Accelerator Center (SLAC), *Stanford, California*

Stanford University, Computer Science Department, *Stanford, California*

Texas A & M University, Department of Computer Science, *College Station, Texas*

United States Military Academy, *West Point, New York*

Universität Augsburg, *Augsburg, Germany*

University of British Columbia, Computing Centre, *Vancouver, British Columbia, Canada*

University of British Columbia, Mathematics Department, *Vancouver, British Columbia, Canada*

University of California, Berkeley, Space Astrophysics Group, *Berkeley, California*

University of California, Irvine, Information & Computer Science, *Irvine, California*

University of California, Santa Barbara, *Santa Barbara, California*

University of Canterbury, *Christchurch, New Zealand*

University College, *Cork, Ireland*

University of Crete, Institute of Computer Science, *Heraklio, Crete, Greece*

University of Delaware, *Newark, Delaware*

University of Exeter, Computer Unit, *Exeter, Devon, England*

University of Groningen, *Groningen, The Netherlands*

University of Heidelberg, Computing Center, *Heidelberg, Germany*

University of Illinois at Chicago, Computer Center, *Chicago, Illinois*

Universität Koblenz–Landau, *Koblenz, Germany*

University of Manitoba, *Winnipeg, Manitoba*

University of Maryland, Department of Computer Science, *College Park, Maryland*

Università degli Studi di Trento, *Trento, Italy*

University of Oslo, Institute of Informatics, *Blindern, Oslo, Norway*

University of Salford, *Salford, England*

University of South Carolina, Department of Mathematics, *Columbia, South Carolina*

University of Southern California, Information Sciences Institute, *Marina del Rey, California*

University of Stockholm, Department of Mathematics, *Stockholm, Sweden*

University of Texas at Austin, *Austin, Texas*

University of Washington, Department of Computer Science, *Seattle, Washington*

Uppsala University, *Uppsala, Sweden*

Villanova University, *Villanova, Pennsylvania*

Virginia Polytechnic Institute, Interdisciplinary Center for Applied Mathematics, *Blacksburg, Virginia*

Vrije Universiteit, *Amsterdam, The Netherlands*

Washington State University, *Pullman, Washington*

Wolters Kluwer, *Dordrecht, The Netherlands*

Yale University, Department of Computer Science, *New Haven, Connecticut*

FOR YOUR TEX TOOLBOX

CAPTURE
Capture graphics generated by application programs. Make LaserJet images compatible with TEX. Create pk files from pcl or pcx files. $135.00

texpic
Use texpic graphics package to integrate simple graphics—boxes, circles, ellipses, lines, arrows—into your TEX documents. $79.00

Voyager
TEX macros to produce viewgraphs—including bar charts—quickly and easily. They provide format, indentation, font, and spacing control. $25.00

FOR YOUR TEX BOOKSHELF

TEX BY EXAMPLE NEW!
Input and output are shown side-by-side. Quickly see how to obtain desired output. $19.95

TEX BY TOPIC NEW!
Learn to program complicated macros. . . . $29.25

TEX FOR THE IMPATIENT
Includes a complete description of TEX's control sequences. $29.25

TEX FOR THE BEGINNER NEW!
A carefully paced tutorial introduction. . . $29.25

BEGINNER'S BOOK OF TEX
A friendly introduction for beginners and aspiring "wizards." $29.95

MR Micro Programs Inc. 251 Jackson Ave. Syosset, NY 11791 (516) 921-1351

TEX Consulting & Production Services

North America

Abrahams, Paul

214 River Road, Deerfield, MA 01342;
(413) 774-5500

Development of TEX macros and macro packages. Short courses in TEX. Editing assistance for authors of technical articles, particularly those whose native language is not English My background includes programming, computer science, mathematics, and authorship of *TEX for the Impatient.*

American Mathematical Society

P. O. Box 6248, Providence, RI 02940;
(401) 455-4060

Typesetting from DVI files on an Autologic APS Micro-5 or an Agfa Compugraphic 9600 (PostScript). Times Roman and Computer Modern fonts. Composition services for mathematical and technical books and journal production.

Anagnostopoulos, Paul C.

433 Rutland Street, Carlisle, MA 01741;
(508) 371-2316

Composition and typesetting of high-quality books and technical documents. Production using Computer Modern or any available PostScript fonts. Assistance with book design. I am a computer consultant with a Computer Science education.

ArborText, Inc.

1000 Victors Way, Suite 400, Ann Arbor, MI 48108; (313) 996-3566

TEX installation and applications support. TEX-related software products.

Archetype Publishing, Inc., Lori McWilliam Pickert

P. O. Box 6567, Champaign, IL 61821;
(217) 359-8178

Experienced in producing and editing technical journals with TEX; complete book production from manuscript to camera-ready copy; TEX macro writing including complete macro packages; consulting.

The Bartlett Press, Inc., Frederick H. Bartlett

Harrison Towers, 6F, 575 Easton Avenue, Somerset, NJ 08873; (201) 745-9412

Vast experience: 100+ macro packages, over 30,000 pages published with our macros; over a decade's experience in all facets of publishing, both TEX and non-TEX; all services from copyediting and design to final mechanicals.

Cowan, Dr. Ray F.

141 Del Medio Ave. #134, Mountain View, CA 94040; (415) 949-4911

Ten Years of TEX and Related Software Consulting, Books, Documentation, Journals, and Newsletters. TEX & LATEX macropackages, graphics; PostScript language applications; device drivers; fonts; systems.

Electronic Technical Publishing Services Co.

2906 Northeast Glisan Street, Portland, Oregon 97232-3295;
(503) 234-5522; FAX: (503) 234-5604

Total concept services include editorial, design, illustration, project management, composition and prepress. Our years of experience with TEX and other electronic tools have brought us the expertise to work effectively with publishers, editors, and authors. ETP supports the efforts of the TEX Users Group and the world-wide TEX community in the advancement of superior technical communications.

NAR Associates

817 Holly Drive E. Rt. 10, Annapolis, MD 21401; (410) 757-5724

Extensive long term experience in TEX book publishing with major publishers, working with authors or publishers to turn electronic copy into attractive books. We offer complete free lance production services, including design, copy editing, art sizing and layout, typesetting and repro production. We specialize in engineering, science, computers, computer graphics, aviation and medicine.

Ogawa, Arthur

1101 San Antonio Road, Suite 413, Mountain View, CA 94043-1002;
(415) 691-1126;
`ogawa@applelink.apple.com`.

Specialist in fine typography, LATEX book production systems, database publishing, and SGML. Programming services in TEX, LATEX, PostScript, SGML, DTDs, and general applications. Instruction in TEX, LATEX, and SGML. Custom fonts.

Pronk&Associates Inc.

1129 Leslie Street, Don Mills, Ontario, Canada M3C 2K5;
(416) 441-3760; Fax: (416) 441-9991

Complete design and production service. One, two and four-color books. Combine text, art and photography, then output directly to imposed film. Servicing the publishing community for ten years.

Quixote Digital Typography, Don Hosek

349 Springfield, #24, Claremont, CA 91711; (714) 621-1291

Complete line of TEX, LATEX, and METAFONT services including custom LATEX style files, complete book production from manuscript to camera-ready copy; custom font and logo design; installation of customized TEX environments; phone consulting service; database applications and more. Call for a free estimate.

Richert, Norman

1614 Loch Lake Drive, El Lago, TX 77586;
(713) 326-2583

TEX macro consulting.

TEXnology, Inc., Amy Hendrickson

57 Longwood Ave., Brookline, MA 02146;
(617) 738-8029

TEX macro writing (author of MacroTEX); custom macros to meet publisher's or designer's specifications; instruction.

Type 2000

16 Madrona Avenue, Mill Valley, CA 94941;
(415) 388-8873; FAX (415) 388-8865

$2.50 per page for 2000 DPI TEX camera ready output! We have a three year history of providing high quality and fast turnaround to dozens of publishers, journals, authors and consultants who use TEX. Computer Modern, Bitstream and METAFONT fonts available. We accept DVI files only and output on RC paper. $2.25 per page for 100+ pages, $2.00 per page for 500+ pages.

Outside North America

TypoTEX Ltd.

Electronical Publishing, Battyány u. 14. Budapest, Hungary H-1015;
(036) 11152 337

Editing and typesetting technical journals and books with TEX from manuscript to camera ready copy. Macro writing, font designing, TEX consulting and teaching.

Information about these services can be obtained from:

TEX Users Group
P. O. Box 869
Santa Barbara, CA 93102-0869
Phone: (805) 963-1388
Fax: (805) 963-8538

TEX Publishing Services

From the Basic:
The American Mathematical Society offers you two basic, low cost TEX publishing services.

- You provide a DVI file and we will produce typeset pages using an Autologic APS Micro-5 phototypesetter. $5 per page for the first 100 pages; $2.50 per page for additional pages.
- You provide a PostScript output file and we will provide typeset pages using an Agfa/ Compugraphic 9600 imagesetter. $7 per page for the first 100 pages; $3.50 per page for additional pages.

There is a $30 minimum charge for either service. Quick turnaround is also provided... a manuscript up to 500 pages can be back in your hands in one week or less.

To the Complex:
As a full-service TEX publisher, you can look to the American Mathematical Society as a single source for any or all your publishing needs.

Macro-Writing	TEX Problem Solving	Non-CM Fonts	Keyboarding
Art and Pasteup	Camera Work	Printing and Binding	Distribution

For more information or to schedule a job, please contact Regina Girouard, American Mathematical Society, P. O. Box 6248, Providence, RI 02940, or call 401-455-4060.

NEW! NEW!

ArborText Inc.

- Silicon Graphics Iris or Indigo
- Solaris 2.1
- DVILASER/HP3
- Motif and OPEN LOOK Preview

Complete TEX packages
Ready to use, fully documented and supported.

Also Available For: Sun-4 (SPARC), IBM RS/6000, DEC/RISC-Ultrix, HP 9000, and IBM PC's

Call us for more information on our exciting new products!

1000 Victors Way ▲ Suite 400 ▲ Ann Arbor, MI 48108 ▲ (313) 996–3566 ▲ FAX (313) 996–3573

AP-TEX Fonts

**TEX-compatible Bit-Mapped Fonts
Identical to
Adobe PostScript Typefaces**

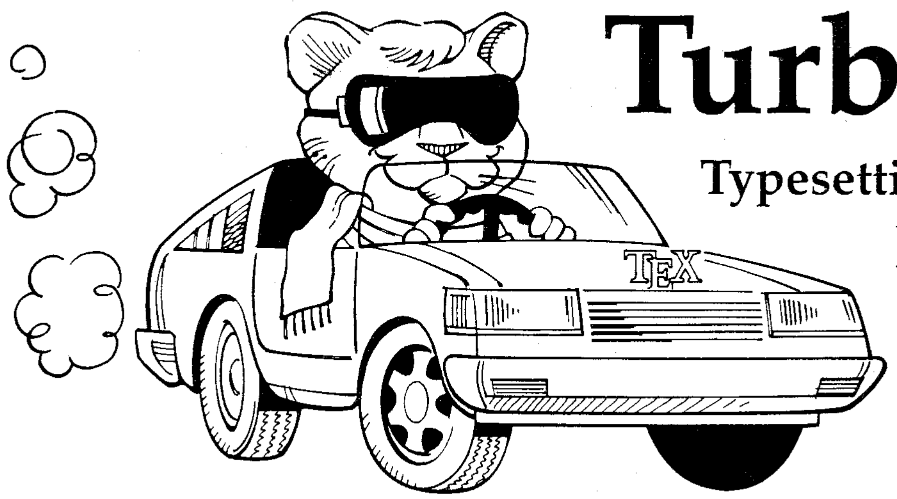
If you are hungry for new TEX fonts, here is a feast guaranteed to satisfy the biggest appetite! The AP-TEX fonts serve you a banquet of gourmet delights: 438 fonts covering 18 sizes of 35 styles, at a total price of $200. The AP-TEX fonts consist of `PK` and `TFM` files which are exact TEX-compatible equivalents (including "hinted" pixels) to the popular PostScript name-brand fonts shown at the right. Since they are directly compatible with any standard TEX implementation (including kerning and ligatures), you don't have to be a TEX expert to install or use them.

When ordering, specify resolution of 300 dpi (for laser printers), 180 dpi (for 24-pin dot matrix printers), or 118 dpi (for previewers). Each set is on ten 360 KB 5-1/4" PC floppy disks. The $200 price applies to the first set you order; order additional sets at other resolutions for $60 each. A 30-page user's guide fully explains how to install and use the fonts. Sizes included are 5, 6, 7, 8, 9, 10, 11, 12, 14.4, 17.3, 20.7, and 24.9 points; headline styles (equivalent to Times Roman, Helvetica, and Palatino, all in bold) also include sizes 29.9, 35.8, 43.0, 51.6, 61.9, and 74.3 points.

The Kinch Computer Company
PUBLISHERS OF TURBOTEX
501 South Meadow Street
Ithaca, New York 14850
Telephone (607) 273-0222
FAX (607) 273-0484

Helvetica, Palatino, Times, and New Century Schoolbook are trademarks of Allied Linotype Co. ITC Avant Garde, ITC Bookman, ITC Zapf Chancery, and ITC Zapf Dingbats are registered trademarks of International Typeface Corporation. PostScript is a registered trademark of Adobe Systems Incorporated. The owners of these trademarks and Adobe Systems, Inc. are not the authors, publishers, or licensors of the AP-TEX fonts. Kinch Computer Company is the sole author of the AP-TEX fonts, and has operated independently of the trademark owners and Adobe Systems, Inc. in publishing this software. Any reference in the AP-TEX font software or in this advertisement to these trademarks is solely for software compatibility or product comparison. LaserJet and DeskJet are trademarks of Hewlett-Packard Corporation. TEX is a trademark of the American Math Society. TurboTEX and AP-TEX are trademarks of Kinch Computer Company. Prices and specifications subject to change without notice. Revised October 9, 1990.

Avant Garde	Bold
Avant Garde	Bold Oblique
Avant Garde	Demibold
Avant Garde	Demibold Oblique
Bookman	Light
Bookman	Light Italic
Bookman	Demibold
Bookman	Demibold Italic
Courier	
Courier	Oblique
Courier	Bold
Courier	Bold Oblique
Helvetica	
Helvetica	Oblique
Helvetica	Bold
Helvetica	Bold Oblique
Helvetica Narrow	
Helvetica Narrow	Oblique
Helvetica Narrow	Bold
Helvetica Narrow	Bold Oblique
Schoolbook	New Century Roman
Schoolbook	New Century Italic
Schoolbook	New Century Bold
Schoolbook	New Century Bold Italic
Palatino	Roman
Palatino	Italic
Palatino	Bold
Palatino	Bold Italic
Times	Roman
Times	Italic
Times	Bold
Times	Bold Italic
Zapf Chancery	Medium Italic
Symbol ΔΦΓϑΛΠΘ	
Zapf Dingbats ✂☞❑	



TurboTEX

Typesetting Software

Executables $150
With Source $300

MICROSOFT® WINDOWS™ COMPATIBLE

THE MOST VERSATILE TEX ever published is breaking new ground in the powerful and convenient graphical environment of Microsoft Windows: TurboTEX Release 3.1E. TurboTEX runs on all the most popular operating systems (Windows, MS-DOS, OS/2, and UNIX) and provides the latest TEX 3.14 and METAFONT 2.7 standards and certifications: preloaded plain TEX, LATEX, AMS-TEX and AMS-LATEX, previewers for PC's and X-servers, METAFONT, Computer Modern and LATEX fonts, and printer drivers for HP LaserJet and DeskJet, PostScript, and Epson LQ and FX dot-matrix printers.

■ **Best-selling Value:** TurboTEX sets the world standard for power and value among TEX implementations: one price buys a complete, commercially-hardened typesetting system. *Computer* magazine recommended it as "the version of TEX to have," *IEEE Software* called it "industrial strength," and thousands of satisfied users around the globe agree.

TurboTEX gets you started quickly, installing itself automatically under MS-DOS or Microsoft Windows, and compiling itself automatically under UNIX. The 90-page User's Guide includes generous examples and a full index, and leads you step-by-step through installing and using TEX and METAFONT.

■ **Classic TEX for Windows.** Even if you have never used Windows on your PC, the speed and power of TurboTEX will convince you of the benefits. While the TEX command-line options and *TEXbook* interaction work the same, you also can control TEX using friendly icons, menus, and dialog boxes. Windows protected mode frees you from MS-DOS limitations like DOS extenders, overlay swapping, and scarce memory. You can run long TEX formatting or printing jobs in the background while using other programs in the foreground.

■ **MS-DOS Power, Too:** TurboTEX still includes the plain MS-DOS programs. Virtual memory simulation provides the same sized TEX that runs on multi-megabyte mainframes, with capacity for large documents, complicated formats, and demanding macro packages.

■ **Source Code:** The portable C source to TurboTEX consists of over 100,000 lines of generously commented TEX, TurboTEX, METAFONT, previewer, and printer driver source code, including: our WEB system in C; PASCHAL, our proprietary Pascal-to-C translator; Windows interface; and preloading, virtual memory, and graphics code, all meeting C portability standards like ANSI and K&R.

■ **Availability & Requirements:** TurboTEX executables for IBM PC's include the User's Guide and require 640K, hard disk, and MS-DOS 3.0 or later. Windows versions run on Microsoft Windows 3.0 or 3.1. Order source code (includes Programmer's Guide) for other machines. On the PC, source compiles with Microsoft C, Watcom C 8.0, or Borland C++ 2.0; other operating systems need a 32-bit C compiler supporting UNIX standard I/O. Specify 5-1/4" or 3-1/2" PC-format floppy disks.

■ **Upgrade at Low Cost.** If you have TurboTEX Release 3.0, upgrade to the latest version for just $40 (executables) or $80 (including source). Or, get either applicable upgrade free when you buy the AP-TEX fonts (see facing page) for $200!

■ **No-risk trial offer:** Examine the documentation and run the PC TurboTEX for 10 days. If you are not satisfied, return it for a 100% refund or credit. (Offer applies to PC executables only.)

■ **Free Buyer's Guide:** Ask for the free, 70-page Buyer's Guide for details on TurboTEX and dozens of TEX-related products: previewers, TEX-to-FAX and TEX-to-Ventura/Pagemaker translators, optional fonts, graphics editors, public domain TEX accessory software, books and reports.

Ordering TurboTEX

Ordering TurboTEX is easy and delivery is fast, by phone, FAX, or mail. Terms: Check with order (free media and ground shipping in US), VISA, Mastercard (free media, shipping extra); Net 30 to well-rated firms and public agencies (shipping and media extra). Discounts available for quantities or resale. International orders gladly expedited via Air or Express Mail.

The Kinch Computer Company
PUBLISHERS OF TURBOTEX
501 South Meadow Street
Ithaca, New York 14850 USA
Telephone (607) 273-0222
FAX (607) 273-0484

Lucida Bright + Lucida New Math

The Lucida Bright + Lucida New Math

font set is the first alternative to Computer Modern complete with math fonts in ATM compatible Adobe Type 1 format. Lucida Bright + Lucida New Math will give your papers and books a bright new look!

The Lucida New Math

fonts (Lucida New Math Italic, Lucida New Math Symbol, Lucida New Math Extension and Lucida New Math Arrows) include the mathematical signs and symbols most used in mathematical and technical composition, including italic Greek capitals and lower case. The Lucida New Math fonts contain the math characters that are standard in the TeX math composition software, which, in its various forms, is one of the most popular mathematical composition packages used worldwide. In addition to the standard Computer Modern math font character sets, Lucida New Math fonts also include the characters in the American Mathematical Society ($\mathcal{AMS}$) symbol fonts.

Switching to Lucida Bright + Lucida New Math is as easy as adding an input statement to the head of your TeX source file. Aside from four styles of each of the expected seriffed, sans serif, and fixed width text faces, the font set also contains Lucida Blackletter, Lucida Calligraphy and Lucida Handwriting.

The Lucida Bright + Lucida New Math

font set is available in fully hinted ATM compatible Adobe Type 1 format for Macintosh, IBM PC compatibles, as well as Unix/NeXT.

We also carry the other font sets commonly used with TeX in fully hinted ATM compatible Adobe Type 1 format, but we are most excited about the new Lucida Bright + Lucida New Math fonts. The finest tools for working with scalable outline fonts in TeX are DVIPSONE and DVIWindo (on IBM PC compatibles).

Y&Y, Inc.
106 Indian Hill
Carlisle, MA 01741
(800) 742-4059 (508) 371-3286 (voice) (508) 371-2004 (fax)

Lucida is a registered trademark of Bigelow & Holmes. Adobe Type Manager is a registered trademark of Adobe Systems Inc.

"Lightning Textures works wonders. This has saved me many hours of corrections...I have three books to be completed and this new gadget certainly gives me courage." anonymous graduate student, University of Ottawa

"I find that Lightning sets type on my Quadra 700 faster than the VAX in the Lab can do it. It's a very nice program, indeed." John C. Allred

"I'll spread the word---Lightning Textures is great!" Prof. Anthony Siegman, Stanford University

"...a fantastic product and I can't imagine going back to an ordinary TeX environment." Chuck Bouldin, Naval Research Laboratory

"I used TeX on a PC for years, but now that I've used Textures I wouldn't (want to) go back to that for anything." George Killough, Hendecagon Corporation

LIGHTNING TEXTURES

"I just typeset 300 pages of two-column heavy math in 7.45 minutes on a Mac IIci -- almost a 4X speedup (actually 3.8). I am very impressed. Good job." Stanley Rabinowitz, Editor of Index to Mathematical Problems

"A few weeks ago, I received a copy of Lightning Textures from you. I was rather skeptical when I ordered it. I could hardly believe that Lightning would work the way you described...But, after testing the program, I am fully satisfied, even more, I am excited about Lightning: Congratulations!" Dr.Bernd Fischer, Universität Hamburg

"I have Textures installed and running -- it's great! Who could prefer other implementations of TeX, I wonder?." Mark Seymour, Springer-Verlag Heidelberg

"It's heaven." Ken Dreyhaupt, Springer-Verlag

"I know that Blue Sky Research did not intend Lighting Textures as a hackers's tool, but it sure can be used that way. It is strange how a simple speedup in the turnaround cycle (of about a factor of 10 to 100) changes your perspective. Lightning Textures is a joy to use." Victor Eijkhout, University of Tennessee

"I am absolutely amazed by Lightning Textures. It beats everything else hands down. I can't imagine what you could be putting in next." Kaveh Bazarghan, Focal Image

503 222 9571 • 800 622 8398 **BLUE SKY RESEARCH** FACSIMILE • 503 222 1643

TUGboat

Volume 14, Number 2 / July 1993