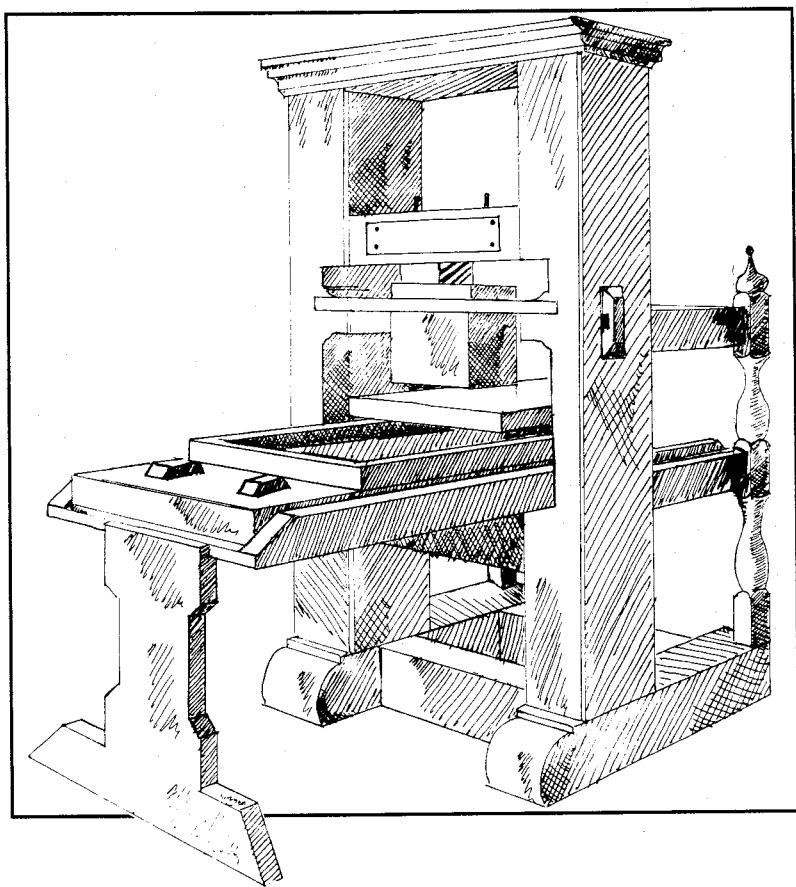


TUGBOAT

The Communications of the **TEX** Users Group



Volume 14, Number 4, December 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.

1994 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 443.

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.

TUGboat © Copyright 1993, TeX Users Group

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

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

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

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

Board of Directors

Donald Knuth, *Grand Wizard of TeX-arcana*[†]

Christina Thiele, *President**

Ken Dreyhaupt*, *Vice President*

Bill Woolf*, *Treasurer*

Peter Flynn*, *Secretary*

Peter Abbott, *Special Director for UKTEXUG*

Barbara Beeton

Alain Cousquer, *Special Director for GUTenberg*

Luzia Dietsche

Michael Ferguson

Roswitha Graham, *Special Director for the Nordic countries*

Yannis Haralambous

Doug Henderson

Alan Hoenig

Anita Hoover

Mimi Jett

David Kellerman

Kees van der Laan, *Special Director for NTG*

Joachim Lammarsch, *Special Director for DANTE*

Nico Poppelier

Jon Radel

Raymond Goucher, *Founding Executive Director*[†]

Hermann Zapf, *Wizard of Fonts*[†]

*member of executive committee
†honorary

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

TeX is a trademark of the American Mathematical Society.

Printing eventually slowed the pace of makeshift invention, forcing out many quaint superfluities, but novel [punctuation] marks, and surprising adaptations of old marks, may appear at any time.

Nicholson Baker

Survival of the Fittest, a review of
M. B. Parkes, *Pause and Effect:
An Introduction to the History
of Punctuation in the West*,
in *The New York Review of Books*
(Volume XI, Number 18,
4 November 1993)

TUGBOAT

COMMUNICATIONS OF THE *TEX* USERS GROUP

EDITOR BARBARA BEETON

VOLUME 14, NUMBER 4

PROVIDENCE

RHODE ISLAND

DECEMBER 1993

U.S.A.

TUGboat

During 1994, the communications of the *TeX* Users Group will be published in four issues. One issue (Vol. 15, No. 3) will contain the Proceedings of the 1994 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. 15, No. 1; deadlines for that issue will have passed by the time this issue is mailed. Deadlines for Vol. 15, No. 2 are February 15, 1994, for technical items, and March 15, 1994, for reports and similar items. Mailing dates for these two issues are scheduled for March and June. Deadlines for future issues are listed in the Calendar, page 438.

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. 369).

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. 369).

***TUGboat* Editorial Board**

Barbara Beeton, *Editor*

Victor Eijkhout, *Associate Editor, Macros*

Jackie Damrau, *Associate Editor, LATEX*

Alan Hoenig, *Associate Editor, Typesetting on Personal Computers*

See page 369 for addresses.

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 μ5 is a trademark of Autologic, Inc.

DOS and *MS/DOS* are trademarks of MicroSoft Corporation

METAFONT is a trademark of Addison-Wesley Inc.

PCTEX is a registered trademark of Personal *TeX*, Inc.

PostScript is a trademark of Adobe Systems, Inc.

TeX and *AMS-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
President, TeX Users Group

Well, here we are—end of the year. Publications for 1993 are back on track, as far as schedules go. This has been a difficult year for both Barbara and myself, as we have found more and more work, both volunteer and that-which-pays-our-bills piling up on our desks. You will have seen the ad in *TTN*, looking for a new editor; while I really enjoy working on the newsletter, I just have to pass the job on to a new person. Similarly, Barbara is making changes in how work on *TUGboat* is going to be re-distributed, as she also has to make some concessions to the fact that the day only has 24 hours in it. We acquire tasks, and make offers of assistance, and generally try to provide information when asked, and it just keeps on going—so we are looking to gently move some of our responsibilities onto new shoulders.

As we sit here at the end of the year (perhaps reading this issue, to avoid going out to shovel the snow yet again), it seems appropriate to reflect on my first year as TUG's president. I've seen a significant upsurge in activities in our user group: committee work, especially with respect to conferences; the Technical Council and its TWGs and SIGs. We have a new board in place, with a combination of old hands and new faces, if you'll pardon the mixed anatomy. The office is almost done with tidying up all the expected and unexpected loose ends from the move out to Santa Barbara. As a user group, I think we have improved our accountability and our responsiveness to our members. We've also begun to actively seek out opportunities to introduce a TeX presence outside our immediate community; I hope to see more of this in the new year.

1 Other happenings

For anyone in the Boston area in January (6th through 9th), there's a meeting of the Linguistic Society of America at the Sheraton Boston Hotel—and TeX will be there, the form of a poster session on TeX and linguistics. I hope to see the information which is gathered for the LSA meeting develop into a nice little package for linguists on how they can use TeX and all its add-ons in their work. Since this is an on-going project, I invite anyone who's interested to get in touch with me, and we'll add you to our group.

We're also hoping to have a TUG and TeX presence at the June meeting of the Society for Scholarly Publishing, which will be held in San Francisco.

2 Free-Net

Something which I've been meaning to write a bit about in this column is Free-Net. You may have heard this new buzz-word; you may have seen it as part of someone's e-mail address. All that's needed is a modem and a computer. There are no user fees, no connect fees—on the other hand, donations are never turned down! For a community such as ours, where so much of our work, our information, and many of our contacts are network-based, not being connected is a big problem; there's a sense of the haves and the have-nots, the privileged and the ordinary man/woman in the street. While the services available vary from site to site, all provide full service Internet e-mail. The opportunity to finally be connected—to reach colleagues, to find the files you want, the information you need—that alone is worth the price of asking. So find out if there's a Free-Net where you live!

3 Renew for 1994!

And remember to send in that TUG renewal form for 1994. You don't want to miss anything that's coming in the new year: new articles and tips in *TUGboat* and *TTN*; the annual meeting in Santa Barbara (if you want to submit a paper, your deadline is **February 1, 1994**; send queries to tug94@tug.org). Barbara's editorial has a list of some new ideas that are brewing for the coming year's issues of *TUGboat*. So stay tuned.

Tell a friend or colleague about the benefits which come with being a member in the TeX Users Group. If every current member brought in one new member, think of all the projects we could undertake. We aren't lacking for ideas; we're lacking funds, and volunteers. You can show your support by renewing your own membership; by letting people know what you do with TeX and lending a hand when someone asks for some advice or help. And you can always show your support by submitting items for publication in either *TUGboat* or *TTN*.

Have a safe and happy holiday season. And we'll see you next year!

- ◊ Christina Thiele
President, TeX Users Group
5 Homestead Street
Nepean, Ontario
K2E 7N9 Canada
cthiele@ccs.carleton.ca

Editorial Comments

Barbara Beeton

For this last issue of 1993, I'd like to indulge in a little wishing. Aside from grandiose wishes for improvements in the state of the world and hopes that people get some sense and learn to respect the beliefs and aspirations of others, I have some more mundane desires for TUG and *TUGboat*.

For TUG I wish many active and enthusiastic members as we enter our fifteenth year.

For *TUGboat* I hope for a deluge of interesting and informed authors, many hands to help, with some way of keeping them organized and directed toward a common goal, and the time to do my job as editor as it should be done. As for specific items, some suggestions are shown in the "wish list" on the next page; you, the readers, probably have some suggestions too—and you might also consider becoming an author or volunteering in some other way. Send in your suggestions, or declare your intentions, in a message to *TUGboat@math.AMS.org*. Happy holidays!

1 Reminder to potential *TUGboat* authors

We always welcome submissions to *TUGboat*. They can be on any topic related to *TeX* and its use. The net spreads rather broadly—typography, SGML, fonts, suitable hardware, . . . , you name it!

There are a few things that a potential author should keep in mind:

- Technical articles will be refereed.
- It's easier for the production staff (usually me) if a submission has already been tagged according to *TUGboat* style. The official and up-to-date plain and *LATeX* style files can be obtained by anonymous ftp from a CTAN site, in the directory *.../digests/tugboat*. The files are **tugboat.sty* and *tugboat.cmm*; instructions for their use are in *tubguide.tex* in the same area. For authors without net connections, the TUG office can supply the files on diskette.
- Actually test the file(s) as submitted. If additional macros or style options are required, send them along, or say where you obtained the version you are using. The same goes for fonts. Nothing is more discouraging than trying to send a file through (*LA*)*TeX* and finding out that something is missing, or a control sequence isn't defined (perhaps just because something is spelled wrong).
- An alternative to testing the files yourself is to ask a *TeX* friend, preferably one with a different *TeX* system, to run the article and read it before

you submit it. This would not only shake out any site-specific constraints, but would give you the benefit of a second pair of eyes checking your spelling, the flow of ideas, and so forth. This isn't a replacement for the referee process, but a good test of portability and lucidity.

A brief comment on the level to which articles might be directed: contrary to popular opinion, the desired level is not "by some great expert, for the edification of other great experts".¹ I continue to hope for good introductory and elementary material, though no one seems to want to write it, at least not for *TUGboat*. I'd like to be proven wrong! Remember—it isn't possible to publish something in *TUGboat* that hasn't been written or submitted.

2 Call for volunteers

As always, there are more tasks in producing *TUGboat* than can be done by just one person. Many, many thanks to all those people who have been working faithfully behind the scenes—associate editors, referees, and in particular, Ron Whitney, without whose assistance the July issue would have been even later than it was.

There are still areas not tended to as well as they might be. Some of the positions where skilled new volunteers might be of assistance are these:

- Referees. If you are interested in reading submissions to *TUGboat* before publication, and "assist[ing] authors in creating articles that are of maximum value to the *TUGboat* readership,"² this could be a job for you. Send a message to the *TUGboat* address stating your availability, listing your specific interests and experience, and identifying any restrictions.
- Columnists. *TUGboat* covers a wide variety of subject areas, only some of which appear in any particular issue. Yours truly comes across a lot of ideas through reading the *TeX*-related network discussions, but only rarely has time to follow them up. A volunteer with a strong interest in a particular subject and fewer distractions than the editor could follow up such leads and twist arms (gently, of course) to bring useful information into print.

There are two tracks that a columnist can follow: actually writing a regular or occasional column, or, for someone with a particularly solid background in the area, tactfully persuading someone else to do the work, and acting as midwife until the article is delivered ready to publish. After a suitable internship,

¹ Anna Russell, in her analysis of Wagner's *Ring der Nibelungen*

² Victor Eijkhout, *TUGboat* 11, no. 4, p. 605

TUGboat wish list

These are some of the topics on which the editor is looking for authors.
Add your own suggestions or volunteer!

Send e-mail to TUGboat@math.AMS.org with details.

- interviews with people who have influenced \TeX and TUG
- real product reviews of both commercial and PD \TeX implementations and other software, also macro packages like `pstricks`, etc.
- surveys of \TeX implementations for particular hardware/operating system combinations, with comparisons of features
- “road map” to the CTAN \TeX areas
- more tutorials and expository material, particularly for new users and users who aren’t intending to become \TeX wizards; one possibility—answers to the “top ten” questions sent to `comp.text.tex` by people writing dissertations
- “how to” articles—how to build your own style based on, say, `article.sty`, how to include an abstract and other stuff in the full-width block at the top of a two-column article, etc.
- comparative analyses of style files that address the same problem, e.g., crop marks
- crossword puzzles for the whole \TeX community

columnists of the latter variety may be promoted to associate editor (see the list on the reverse of the title page of this issue). If you are interested in either track, a message to *TUGboat* would be welcomed.

- Production assistance. This is a more problematic area, as the successful production of an issue of *TUGboat* requires that every file and every font be available to and compatible with the equipment on which the camera copy is generated. However, sometimes it’s useful to have someone to call on to generate fonts, vet macro files (I always assume that if the author doesn’t specify otherwise, the current version on CTAN will work properly, an assumption that isn’t always warranted), and help fight other fires. If you’re an experienced (LA) \TeX user and are interested in this sort of challenge, send a message to the *TUGboat* address with the details of the system you’re working on—computer, operating system, implementation and version of \TeX and `METAFONT`, output device(s) available. Previous production experience is a big plus, and a direct Internet connection a necessity.

By now, you’ve seen Christina’s solicitations for a new \TeX and *TUG NEWS* editor. The editor of *TUGboat* has been having similar thoughts off and on for several years, but hasn’t done anything se-

rious about it. After the nearly disastrous failures to meet the publication schedule this past year, it’s imperative that I do start looking toward the future. I know that *TUGboat* edited by someone else wouldn’t be quite the same, but there are many valid conceptions of what such a journal should be. The criteria that I’d value in a possible successor include, in no particular order:

- broad and thorough knowledge of \TeX and its relations;
- fascination with the typographic art and a desire always to learn more;
- literacy;
- a good (native) command of English and some ability to understand other human languages;
- tact;
- a comfortable familiarity with the electronic networks;
- the ability to bend a computer to one’s will;
- a well-developed sense of responsibility.

If you think you might be such a person, or know of someone else who is, please contact me directly: bnb@math.AMS.org.

◊ Barbara Beeton
American Mathematical Society
P. O. Box 6248
Providence, RI 02940 USA
bnb@Math.AMS.org

Dreamboat

NEXT_EX: A Personal View

Malcolm Clark

Explanation

Last year, at the Portland TUG Conference, I was invited to give the keynote address. What was printed in the conference proceedings was not what I talked about. This was perhaps a bit arrogant on my part, but since the conference preprints were available to those who wished to read the ‘official’ paper, I felt that it was not stretching the prerogative too far to talk about something which, at the time, I thought more important to the *T_EX* community. Perhaps unsurprisingly, the talk was mis-reported. Joachim Lammarsch, President of DANTE, the German-speaking heard it as an attack on NTS, the ‘New Typesetting System’ which his group had initiated. Since Lammarsch expressed his displeasure in DANTE’s ‘Die Technische Komödie’, reported in *TUGboat* 14(1) as ‘he (Lammarsch) expresses his strong disappointment over the statements on NTS (...) made by Malcolm Clark’, I feel it is appropriate to have the opportunity to see what was actually said. Naturally I cannot guarantee that what I said was exactly what is written below, but it is the text from which I was working (and one which I gave to Lammarsch later in 1992 so that he would have an accurate original which he might use). I have not included all the overhead slides I used, since they were a little too fragmentary, but they do not diverge from the argument developed below. I have corrected one or two grammatical errors, and added the footnotes. Nothing substantive has been changed.

It would have been difficult for me to say anything about NTS at the time, since it had hardly been reported in the English-speaking world, except in an email (*NTS-L*) list, where the status of the project was not particularly clear. It was not until September of 1992 that Philip Taylor [19] presented a paper at the Prague Euro*T_EX* conference in which details were given on a wider basis, but even this hardly amounts to widespread dissemination. Perhaps Taylor’s later exposition at the Aston’93 conference [20] will give the NTS project the exposure it warrants. Joachim Lammarsch [10] also accepted an invitation to talk on the subject.

1 Introduction

One of the consistent recurrent themes present at any gathering of two or more *T_EX*ies is the conversation about the deficiencies of the program, and the need to enhance by adding a number of features, both to do something in particular, but also to ensure that *T_EX* remains in the forefront of quality technical publishing.

On examination, it often, but not always, turns out that *T_EX* is well able to do the particular task which provided the perceived requirement to enhance the program, but that the code needed to achieve the result is not immediately obvious or intuitive (Spivak gives a good example [18]). There can be no doubt that *T_EX* is a very subtle beast and has depths that few of us will ever plumb. But equally, there are some things which *T_EX* does with great difficulty: a well-known example is the (almost) impossibility of finding out exactly where on the page you are (but see Hoenig’s solution [6]). Various people, with a deep understanding of the program, have listed some features that they would like to see enhanced: the papers of Stephan von Bechtolsheim [1], Frank Mittelbach [13] and David Salomon [15] are recent examples, but if we delve back into the *T_EX* literature (exemplified by *TUGboat*), we will find other examples. It is quite arresting to read Lynne Price’s words [14]: ‘One refreshing quality of the *T_EX* user community, and particularly the system’s creator, is that *T_EX* is viewed, in fact intended, to be the ancestor of an evolving family of document formatters rather than as a static piece of software that will be used for decades.’ In the same article, I was astounded to note an account of *L_AT_EX*: ‘a hybrid of *T_EX* and Lisp’, where text manipulations too difficult or impossible in *T_EX* are done in Lisp. (I had thought I had merely been joking when I had from time to time suggested implementing *T_EX* in Lisp for just this sort of reason!) As a result of this note by Price, proposals for future enhancements were given a column in *TUGboat*—the Dreamboat column (one recently revitalized by Barbara Beeton). In 1987 Lamport [11] bemoans the ‘idiosyncrasy’ of dvi format and suggests a switch to PostScript.

2 Change already

Looking at the problem historically, there have been two major jumps in *T_EX*. But not all jumps are alike: the first change was a major one—the change from *T_EX78* to *T_EX82*. *T_EX82* is the one with which most of us who have used *T_EX* will probably be familiar. It survived mostly unchanged save for bug fixes until 1988. The transition from *T_EX78* to *T_EX82* was radical. Some of the language primitives

changed: one of the most striking was in font handling: I was fortunate that I learned *T_EX* as *T_EX*78, when the manual was a scant 200 or so pages long. I doubt that I would have started if the manual had been 500 pages long. Internally, the changes were even more marked, since the language was changed from SAIL to Pascal. This also meant that *T_EX* became much more portable, inaugurating a whole new concept in software development.

The other change which will still be in our immediate memory is the change to the so-called *T_EX*3, which began in 1988. The magnitude of the change is much less great than the earlier change. In essence it was to enhance *T_EX* to handle eight-bit characters, instead of the seven-bit characters with which it originated. The immediate benefits of this change were felt mostly with respect to the ease with which accented characters could be dealt with—among other things, making it possible, at last, to hyphenate accented words properly. There were one or two other relatively minor changes too. I have to admit that the transition to *T_EX*3 has made hardly any difference at all to me, although I regularly use *T_EX*3 on Macintosh, UNIX and VAX/VMS.

In between times, there were a few other changes in the *T_EX* world, although not directly to *T_EX* itself. For example, *METAFONT* was upgraded in 1984, in rather the same way that *T_EX* had been: in general, the change was hardly noticed by the mass of *T_EX*ies, since they do not use *METAFONT* explicitly. Similarly the Computer Modern typeface started out as Computer Modern, reverted to *Almost Computer Modern*, and then re-asserted itself as Computer Modern (and as recently as 1992 was still being subtly altered). Those of us around in the days of this transition will recall the confusion caused between those machines which had the *Almost* fonts, and those with the more final version. In particular, PCs seemed to hang on to these older versions.

3 We are not alone

Naturally, *T_EX* does not live in noble isolation. In the years since its birth, we have seen a number of notable developments which have produced resonances within the somewhat hermetic *T_EX* universe. The dramatic rise in personal computing power spread the use of *T_EX* widely, and to some extent loosened the ties between *T_EX*ies. Reflect that the *T_EX* and *L_AT_EX* books have both sold into the hundreds of thousands. I think that the combined figure is now over 150,000—that's an expenditure of approximately \$5,000,000: if we take that as a crude measure of the number of *T_EX* and *L_AT_EX* users (and ex-users), and compare it with the num-

ber of TUG members (about 3,500), and then the number at the recent TUG conference (about 150), we see there may be a lot more people doing it than talking about it (maybe they are too embarrassed to talk about it).

In passing it is surprising just how long it took before the first non-canonical *T_EX* and *L_AT_EX* books appeared (my guess is that the first properly published follower was Norbert Schwarz [16], first in German in 1988, and then translated into English [17]). Maybe *The T_EXbook* really is crystal clear.

4 Diffuse

But this takes us away from the main theme I would like to develop. We have a vast increase in the number of users, and the majority have *T_EX* on their own individual machine with limited support from elsewhere. This has quite far-reaching consequences, especially when coupled with the near demise of commercial vendors outside the USA and the widespread availability of public domain implementations. To whom does the user turn? And how does she or he get information about changes and developments? To take a specific example, did you realise that the Computer Modern fonts had been tweaked earlier this year? The sub-text here is that changes may not diffuse too readily. A similar slowness of diffusion rates is experienced with *L_AT_EX* styles. The current version of *L_AT_EX* is 2.09. Most users seem to have this. But this version number is not sufficient. One must also know the date. The files should be dated February 1991.¹ Experience shows that this is not always the case. Similarly, the complete lack of clarity of the availability and distribution of the New Font Selection Scheme (seldom part of a vendor's offering) bodes ill for the acceptance and widespread availability of *L_AT_EX*3 (whenever it appears). There is a counter-example in the relative speed with which *T_EX*3 appears to have swept around the world.

5 Commercials

The rise of personal machines stimulated the widespread adoption of improved printing facilities, especially the 300 dpi laser printer. This was a development on which *T_EX* was well able to capitalise. But it is probably not a development which had been anticipated when *METAFONT* and Computer Modern were created. Laser printers were seen as low resolution devices used at a stage prior to the final high resolution photo-typesetting. Computer

¹ Wrong! Even at the time of writing, the latest release was March 25th, 1992, but since then *L_AT_EX*2e has been announced at Aston—let's watch its diffusion.

Modern fonts (like very many others) are not ideal 300 dpi fonts (and the even lower resolution screen versions leave much to be desired—sometimes the METAFONT rather falls apart). But the point being embroidered is that this identified two new foci, the laser printer, which quickly became identified with PostScript, Warnock & Geshke's page description language, and then direct manipulation word processing programs. Remember that *TeX*'s avowed aim was to assist publishing (masterpieces of the publishing art); the new generation of personal publishing was initially very happy with relatively low resolution laser printed copies. But in time quality and scope improved, up to the level where contemporary publishing packages, like Quark Xpress, PageMaker, InterLeaf, FrameMaker and 3B2 (to name a few) can arguably produce masterpieces.

Commercial software has some interesting qualities: it evolves. In order for the vendor to survive, it is essential that new versions of the software are released, correcting some of the bugs, introducing some new features, and basically keeping the software in the public's eye. *TeX* is not commercial software, except in a very limited sense. It is almost always possible to find a public domain implementation. But there is no development of the core software; there is no reason to keep releasing new versions. The only real exception to this rule is when a version for a new machine or version of an operating system is released.

In order to pay lip service at the altar of fair play, I have to admit that there is software around which is not commercial, and yet which has evolved. Kermit springs to mind, although I am not sure if it is still evolving now. I have versions of Kermit which work for the machines I use, and until they fall over badly, I won't bother replacing them. Much of the Gnu (Free Software Foundation) project's software is also still being developed. If we ignore the forbidding air of messianic fundamentalism surrounding the Gnu project (just as we expect everyone else to ignore our very own missionary position) we have to admit that they do provide a model of public domain software development.

I think there is a difference between this development and *TeX*, or some successor to it. There is a fixed mark, something to aim for: Kermit did develop along the way, but the main issue was to have something which worked on many platforms and performed a reasonably well-defined function. The Gnu project is aiming to provide substitutes for software which already exists (like a C compiler), and is therefore specified already (or even mis-specified already). The *TeX* successor will first have to decide what features it will encompass.

6 Quality

One of the arguments put forward for the need to develop *TeX* further is the quest for quality. It is said that there are areas where the highest quality is just not obtainable. I do not wish to challenge this statement, but rather to question the quest for quality. I appreciate that this is heretical. Currently, my organization,² a self-styled educational institution, is going through a sort of managerial restructuring. Part of the new baggage of management is the idea of 'total quality'. It is difficult to stand up and say that you do not believe in quality. But as far as a publishing system is concerned, I think it is possible to say that aspiration to the highest typeset quality is not the sole criterion.

I am not sufficient of an aesthete to recognise the highest quality. I think I can often find things which I consider to be pleasing to the eye, but when it comes to qualitative judgments, absolutes are so very elusive. Typographic quality at least has the advantage that there is often a function lurking underneath, and we can always appeal to the extent to which the form and function complement one another, or appeal to notions of 'fitness for purpose'. But sadly it often seems that the consensus for quality is a rather conservative one. Apparently, within a few years of Gutenberg's 42 line Bible being produced, there were vociferous complaints by the cognoscenti bemoaning the sad reduction in quality from traditional hand-lettered manuscripts. And we can see this pattern repeated again and again. We can be relatively confident that a departure from the norm is perceived as bad. In a few years it may become acceptable, but at the time, it is new and suspect. Of course, the iconoclasts will be prepared to pick up the new, for good and bad reasons. But even if we hedge around the problem of identifying the highest quality, we can usually acknowledge that some things are suspect.

But who actually worries? A few years ago, it was common to see typewritten manuscripts published by reputable publishers as whole books. The argument was usually that it was better to have something published at this lower quality than nothing published at all. It does seem to indicate that quality is only one of several issues, even among 'quality' publishers. Even today, using the same sort of argument, we often see books published from laser printed masters (even *TeX* books!). This is sad. The difference in cost is really not great. The publisher, for whatever reasons, economic or aesthetic, clearly feels that typographic quality commensurate with the book's 'worth' may be met with

² My ex-organisation!

inferior production. Let me take two contemporary examples. The quality of the paper used in the softback *TEXbook* has deteriorated over the years (in my opinion): I will not rise to the bait of the abysmal binding of the softback; even the hardback is not designed to last for ever—I was very disappointed when my Knuth-autographed hardback fell apart last year. And yes, I do look after books and take great care not to break the spines. Another example would be Victor Eijkhout's recent book [3]. Victor obviously spent a good deal of time and effort in the design of his book, even to the extent of eschewing the delightful (if traditional) Computer Modern typeface. Sadly, at least half the copies I have seen were under-inked. Both these examples emphasise that getting the marks on the paper in the right place is only one of the problems facing us.

In recent years, a number of word processing programs have acquired so-called mathematical ability. For example, MicroSoft Word even has an advert for Word 5 with some equations in it: they are acceptable, but not really of the highest quality: they are not even of the quality of *eqn*. Either quality is not an issue, or mathematics is such a strange pursuit that no-one recognises when it is done badly.

I have a problem with 'highest quality', as is probably evident. I expect *TEX* or whatever to be pretty good. I do not expect it to be perfect. Like a Persian rug, it ought to have at least one mistake in it. The fear of hubris is just too great. Even the concept that perfection could be achieved by a program worries me. I expect, indeed I am duty bound, to get in there and meddle. Obviously there are levels and magnitudes of meddling.

But there is an interesting question: why would anyone re-invent the mathematics typesetting wheel? or why would you not incorporate *TEX* mathematical typesetting in Word, or WordPerfect, or Ventura, or Frame, or Interleaf? Can anyone explain this? Sometimes we find *eqn* in there instead: sad. Having brought up *eqn*, we have to point out the presence of a computerised typesetting tool which seems to keep running, without moans and groans about its total inadequacy to face the future—*troff*: it just goes on as every UNIX system rolls off the production line. It doesn't aspire to excellence, it just comes as part of every system, and all the manuals expect its availability—for goodness sake, it isn't even device independent (well, it is now, but that took for ever to achieve—*ditroff* produces *dvi*!). It is surprising to see the longevity of the *nroff/troff* tools. They seldom produce anything very exciting, and they make no pretension towards quality. They seem to meet a very real need and in a very straightforward way, although I was sur-

prised to see a book produced recently which had as its topic *tbl* [12]. Maybe it's a subject area a whole lot more difficult than it seems.

7 Time

Let's briefly consider time spans. It isn't easy to work out just how much effort went into *TEX*. Somewhere, Knuth records that in 1977 he announced to Jill that he was going to take a year off his academic work to write a typesetting system. In fact we actually know when he started working predominantly on *TEX* (Thursday May 5th, 1977) (see [7] and [8]). Even more bizarre, we know what films he went to see that weekend (*Airport 77* and *Earthquake*). In the midst of this trivia, we have the estimate from Knuth, arguably one of the most talented programmers to have existed, that the program would take one year (or perhaps less) to complete. More realistically it appears to have taken at least four or five years in elapsed time (this is a wild guess: improved estimates would be appreciated): from this we might have to subtract the time spent on *METAFONT* and Computer Modern (and *WEB*), but on the other hand we should add in the efforts of his graduate students and all the others (like Art Samuel, David Fuchs, Luis Trabb Pardo, Frank Liang, Michael Plass, Arthur Keller...) who contributed to the program. I suspect that four or five man years is still a conservative estimate. Four or five man years of a small, highly motivated team, with one person in control who could decide what and what not to include.

This was not a democratic process, although it is clear that there was feedback. Even more recently, the transition to *TEX3* seems, to me, to have taken a shade longer than anticipated. There are probably many reasons for this. After all, Knuth was not really planning to change *TEX* in 1989. Forces conspired against him there, and marshalled some convincing arguments, and it is evident that he already had the feeling that seven-bit character representations were inadequate. The point here is that Knuth, with his intimate knowledge of the program, still appears to have taken longer than he expected to complete the changes.

One of the things that we have surely learned over the last fifty or so years of programming is that it takes longer than you expect. The folk-lore of computing (backed by some extremely readable books like Brook's *Mythical Man Month* [2]) knows that a project will take at least twice as long as you estimate; that doubling the estimated time has no effect on this inflation factor; and that the program will always be finished 'in another four weeks'. Changes to *TEX*, or a re-write, are going to take a

- the designer of a new kind of system must participate fully in the implementation
- writing software is much harder than writing books
- the designer should also write the first user manual

Figure 1: Knuth's lessons

long time. It will be a pity to have any new development labeled vapourware, but there will necessarily be a long time spent in development. It is unlikely that we will find some wealthy benefactor who will turn round and say 'take this million dollars: take your time: improve \TeX '.

Knuth [8] himself says 'If I had time to spend another ten years developing a system with the same ideas as \TeX —if I were to start all over again from scratch, without any considerations of compatibility with existing systems—I could no doubt come up with something that is marginally better.' My point here is the word *marginally*.

8 Or money

Because of \TeX 's public domain status, we sometimes lose sight of the fact that it did cost money to develop. Knuth [8] records 'generous financial backing' from a number of sources, including the System Development Foundation, the US National Science Foundation, and the Office of Naval Research. How much money is indeterminate, since it is unlikely that any of the funding detailed 'work on \TeX '.

Any future work will have to be done by interested individuals, probably working in their spare time, or, if we are exceptionally lucky, by graduate students working together on a funded project, although note Knuth's 'lessons' from the \TeX project ([8], Figure 1). I am not clear I see who to approach for the funds. Inter-disciplinary research has not been too well funded (certainly in the UK) in recessionary times. The core areas let the peripheral stuff go in times of crisis.

Where does computerised typesetting fit? Is it computer science; is it a branch of engineering; is it part of some typographic or fine arts discipline? Let's hope it isn't the latter, since they are particularly badly funded. But is this really research in the commonly accepted sense? What will we end up with? Something which is in some sense better than an existing program. How are we going to sell this? How will we convince some body with loose cash to support this? Do we indicate just how dreadful \TeX is, exposing all its warts and deficiencies? Why are we using it in the first place if it is so bad? Would a cheaper and easier solution not just

be to use an existing program which has none of these deficiencies? Never mind that there is no such paragon. The other contenders must offer some improved or needed features or they would not be in use at all. The chances are this proposal will have to go through a committee. If those on the committee have ever prepared their own documents (and remember there are still some oldsters out there who do not; their secretary does it), they will have their own favourite software. So we will end up telling a reasonably influential (maybe) bunch that \TeX is deficient and needs changing. In the end we are asking them to invest a fair chunk of money in order to benefit whom? This is one I find difficult.

8.1 Cui bono?

The people who seem most likely to benefit are book publishers: correct me if I am wrong here. But it appears to me that the principal beneficiaries are organisations like Elsevier, Springer Verlag, Addison Wesley and so on.

Oh dear. I confess that I would anticipate that printing and publishing organisations might reasonably be expected to underwrite research into the development of quality typesetting. There are research organisations founded and financed (at least in part) by them. In the UK, PIRA (Printing Industries Research Association) does just that, although in recent years it has become much more commercially oriented. There are others in other countries.

A ray of hope might be seen in some projects funded through initiatives which ultimately derive from Brussels and the EC. The Didot project is/was a three-year project set up to re-establish European pre-eminence in typography (in the sense of type design), and, from the outset, had a very strong digital component. It seems to have been successful in bringing type practitioners and computing people together (and maybe even a few engineers). The outcome of the project is to develop training programs, and an increased awareness and facility with digital type design. The project should finish in 1993. It does not quite do what we want, but it indicates that there are precedents. Although Didot started out with a rather strong chauvinist element (basically to prevent Europe being overwhelmed by the US, always a populist rallying call in Europe), it mellowed quite considerably and there is apparently effective interaction with North America now. But it remains a suspicion in my mind that an appeal to some external threat could be the most effective, if least ethical, way of appealing for funds.

9 I'll be in Scotland afore ye

I see two main routes towards a descendant of \TeX . One is an evolutionary approach, where the perceived deficiencies are remedied, and a few new features are added. Basically, \TeX itself changes only slightly, and in a well-defined way. Vulis' V \TeX [21] can be seen as an example, where the handling of fonts has been substantially changed, and arguably enhanced. Similarly, Ferguson's ML \TeX [4] which allowed multilingual hyphenation,³ falls into this category. It might even be reasonable to place Harrison's VOR \TeX project [5] into this model. I am quite a fan of the project, partly because I feel that the model they developed, of multiple views of documents, has much to commend it. The fact that the program itself was rather machine specific is a side issue. Almost five years or so ago, it accomplished at least some of the things that we presently feel we need.

There is probably not a single route, but several. If people go ahead and add some features to the underlying code, is there any guarantee that the full range of features added will be compatible with one another? I can envisage a whole cluster of similar but incompatible descendants. With luck an existing \TeX -encoded file will produce identical output, but there may be no way to use the extended features of more than one. Perhaps one will out-evolve the rest. There are examples of this happening. Tom Rokicki's DVIPS is arguably the *de facto* PostScript driver. This was not always so. There are, or have been, at least eight PostScript drivers, but Tom's has the advantage of being versatile, up-to-date, and runs on most platforms. It is also in the public domain.

If this is one route, what is the other? Why, a radical restructuring. Throw away the baby, bathtub and water, but keep the mission—that of creating a device for typesetting of the highest quality. I confess I find this a somewhat vague statement at best. How will the model be chosen? Who will be involved? In the worst possible case it may be totally democratic, and we can look forward to interminable referenda on desirable features. Let me quote from Knuth [8]:

I was constantly bombarded by ideas for extensions, and I was constantly turning a deaf ear to everything that did not fit well with \TeX as I conceived it at the time ...

I was perhaps able to save \TeX from the 'creeping featurism' that destroys systems whose users are allowed to introduce a patch-work of loosely connected ideas.

- an altered \TeX is not ' \TeX '
- will descendants be accepted widely?
- will they be public domain?
- who authorises or legitimises?
- will there be a trip test?
- may be multiple, mutually incompatible, descendants
- will they be widely ported?
- begins a tradition and expectation
- what time scales?

Figure 2: Some fears for a future development(s)

Apart from a warm and fuzzy glow, I am not too clear what I or any other existing \TeX or L \TeX user will get out of either route, apart from more upgrades. I feel I may even be tempted to do nothing, and just hang onto my working and apparently almost perfectly satisfactory current version of \TeX . For remember this: you will not be able to call this new beast ' \TeX '. This alone seems to me to mean that any small enhancements are likely to be still-born. It will be viewed with suspicion. It is \TeX , but it isn't ' \TeX '. Perhaps the highly \TeX literate will understand the differences, but the great unwashed will have to be sold the idea. How do you sell ideas when you are not commercial? and not very fashionable? Some of my fears are summarised in Figure 2.

I do not want to appear gloomy and despondent. I do not feel that way at all. I know that \TeX is not perfect. I can see several minor blemishes (and at least one major one). I would prefer the program to be truly modular, although that confers no immediate benefit. But I am not altogether convinced that the next generation will please me any more. What pleases me most about \TeX is its solidity. It has not changed much in the last eight or so years. And I do not feel too dissatisfied, although I think I have been using it seriously. Maybe I do not use it to its limits, but that is largely because its limits are pretty wide and the little I have learned about software indicates that when you push it to its limits, it breaks. That is not to say that developments will not take place, but like many others, I see them around the periphery (Figure 3).

This conclusion is awesome: in my self-view I like to feel I am some sort of radical, an iconoclast (in spite of my love of the Macintosh and its icons), and here I am saying do not change the core. This is so embarrassing. But equally it indicates that maybe it's a valid view. I may now go on and

³ Now, of course, superseded by $\text{\TeX}3$

- improve the support environment
 - editors
 - drivers
 - overall integration level
- widen the scope
 - additional macros/styles
 - dvi processors for increased functionality

Figure 3: Already suggested alternatives for development

show how many angels may stand on the head of a pin.⁴

A An editorial paraphrase

Lammarsch's editorial comments [9] in the German-speaking group's 'Die TeXnische Komödie' were published in August of 1992. They throw some useful light on what has been done, although the details are perhaps still unknown to those who do not read the Komödie. Paraphrased and translated (for which translation I am grateful to Peter Schmitt), Lammarsch stated the following

- Knuth is positive with regard to the project;
- funds, amounting to 20% of that required, have been secured already; in an earlier report, Lammarsch estimated that the project would cost DM 500 000, over 5 years;
- 'big publishers' have promised to support the project;
- commercial TeX dealers have accepted the project;
- the program will remain 'freeware'.

Like many others, I look forward to details of Knuth's endorsement, the extent of publishers' support, and the progress of the project. It is to be hoped they will be circulated widely.

References

- [1] Stephan von Bechtolsheim, 1990, TeX in practice: comments on a 4-volume, 1400-page series on TeX, *TUGboat* 11(3), pp. 409–412.
- [2] Frank P. Brooks, 1974, *The Mythical Man Month*, Addison-Wesley.
- [3] Victor Eijkhout, 1992, *TeX by Topic*, Addison-Wesley, 307 pp.
- [4] Michael Ferguson, 1985, A multilingual TeX, *TUGboat* 6(2), pp. 57–58.
- [5] Michael Harrison, 1989, News from the VORTEX project, *TUGboat* 10(1), pp. 11–14.
- [6] Alan Hoenig, 1990, Line-oriented layout with TeX, in *TeX, Applications, Uses, Methods* (editor, Malcolm Clark), Ellis Horwood, Chichester, pp. 159–183.
- [7] Donald E. Knuth, 1989, Remarks to celebrate the publication of *Computers & Typesetting*, *TUGboat* 7(2), pp. 95–98.
- [8] Donald E. Knuth, 1989, The errors of TeX, Software practice and experience 19(7), pp. 607–685.
- [9] Joachim Lammarsch, 1992, Grußwort, *Die TeXnische Komödie* 4(2), pp. 4–5.
- [10] Joachim Lammarsch, 1993, A new typesetting system: is it really necessary? *TUGboat* 14(3), pp. 167–170.
- [11] Leslie Lamport, 1987, TeX output for the future, *TUGboat* 8(1), p. 12.
- [12] Henry McGilton and Mary McNabb, 1991, *Typesetting tables on the UNIX system*, Addison-Wesley, 280pp.
- [13] Frank Mittelbach, 1990, E-TeX: guidelines for future TeX, *TUGboat* 11(3), pp. 337–245.
- [14] Lynne Price, 1981, Dreamboat, *TUGboat* 2(2), p. 58.
- [15] David Salomon, 1991, personal communication to TUG Board.
- [16] Norbert Schwarz, 1988, *Einführung in TeX*, Addison-Wesley.
- [17] Norbert Schwarz, 1989, *Introduction to TeX*, Addison-Wesley, 278pp.
- [18] Michael Spivak, 1991, A contrarian view on TeX extensions, *TeXline* 13, pp. 1–3.
- [19] Philip Taylor, 1992, The future of TeX, in *EuroTeX'92, Proceedings of the 7th European TeX Conference*, Prague (editor Jirí Zlatuška), pp. 235–254; reprinted in *TUGboat* 13(4), pp. 433–442.
- [20] Philip Taylor, 1993, NTS: the future of TeX? *TUGboat* 14(3), pp. 183–186.
- [21] Michael Vulis, 1990, VTEx enhancements to the TeX language, *TUGboat* 11(3), pp. 429–434.

◊ Malcolm Clark
Computing Services
University of Warwick
Coventry CV4 7AL, England, UK

⁴ As many as want to.

NTS Update

Philip Taylor

This is a report on the inaugural meeting of the NTS¹ project group, held during the Autumn DANTE meeting at Kaiserslautern (Germany) on 23rd and 25th September, 1993.

Present: Joachim Lammarsch (DANTE President, and instigator of the NTS project); Philip Taylor (Technical co-ordinator, NTS project); Marion Neubauer (minutes secretary); Prof. Dr. Peter Breitenlohner, Mariusz Olko, Bernd Raichle, Joachim Schrod, Friedhelm Sowa.

Background: Although the NTS project has been in existence for approximately eighteen months, there has not previously been a face-to-face meeting of members of the core group; at the Spring meeting of DANTE Rainer Schöpf announced his resignation as technical co-ordinator, and Philip Taylor was invited by Rainer and Joachim to take over as co-ordinator, which he agreed to do.

Joachim Lammarsch opened the Autumn meeting by reviewing the history of the project and the rationale which lay behind its creation; each member of the group then briefly reviewed his or her particular area of interest in the project, after which the group received an extended presentation from Joachim Schrod on one possible approach to the realisation of NTS. The members of the group were broadly in support of the approach outlined by Joachim Schrod, and it was *agreed* that this should form the basis for discussions at the meeting.

The approach proposed by Joachim may be summarised as follows: T_EX in its present form is not amenable to modification; the code, although highly structured in some ways, is also painfully monolithic in others, and any attempt to modify the present code in anything other than trivial ways is almost certainly doomed to failure. Accordingly, before attempting to modify T_EX in any way, it is first necessary to re-implement it, the idea behind such re-implementation being to eliminate the interdependencies of the present version and to replace these with a truly modular structure, allowing various elements of the typesetting process to be easily modified or replaced. This re-implementation should be undertaken in a language suitable for rapid prototyping, such as the Common Lisp Object System ('CLOS'). The primary reason for the re-implementation is to provide modularisation with specified internal interfaces and thereby provide a

test bed, firstly to ensure that T_EX has been properly re-implemented and subsequently to allow the investigation of new typesetting paradigms.

Once a working test bed has been created, and compatibility with existing T_EX demonstrated, a second re-implementation will be undertaken; this re-implementation will have the same modular structure as the test bed but will be implemented with efficiency rather than extensibility in mind, and will be undertaken using a combination of literate programming and a widespread language with a more traditional approach, such as 'C++'. When this second version has also been demonstrated to be compatible with T_EX, it will be made available to implementors around the world, the idea being to encourage people to migrate to NTS by demonstrating its complete compatibility with T_EX. (The test bed will also be made available if there is interest shewn in its use.) Thereafter new ideas and proposals will be investigated using the test bed, and if found to be successful these will be re-implemented in the distribution version.

The main problem which the group identified with the approach outlined by Joachim was simply one of resources: in order to accomplish two re-implementations within a reasonable time-scale, it would be essential to use paid labour, it being estimated that each re-implementation would require a minimum of four man-months work to produce a prototype, and eight man-months to reach the production stage. As this is far beyond the ability of members of the group to contribute in the short term, it is clearly necessary to employ a small team (between two and four members) to carry out the re-implementations, under the guidance and supervision of one or more members of the core group. Initial costings suggested that this could not be accomplished within the present financial resources of the group, and accordingly it was *agreed* that Joachim Lammarsch should seek further financial support. Subsequent investigations shewed that a quite significant reduction in costs could be achieved if the programming team were sited in a central or eastern European country, particularly if the members of the team were also residents of the country; this approach is being investigated.

As it was obvious that no immediate progress could be made with Joachim Schrod's proposal, even though the group agreed that it represented an excellent philosophical approach, it was also *agreed* that the group needed to identify some fallback approaches, which could (a) be commenced immediately, and (b) would be of significant benefit to the T_EX community at large. The group

¹ NTS: the 'New Typesetting System'

identified two such projects, these being (1) the specification of a canonical \TeX kit, and (2) the implementation of an extended \TeX (to be known as e- \TeX) based on the present WEB implementation. It was also *agreed* that Marek Ryćko & Bogusław Jackowski would be asked if they were willing to co-ordinate the first of these activities, and that Peter Breitenlohner would co-ordinate the second.

The ideas behind the two proposals are as follows.

- (1) The canonical \TeX kit: at the moment, the most that can be assumed of any site offering \TeX is (a) ini \TeX ; (b) plain \TeX ; (c) L \TeX ; and (d) at least sixteen Computer Modern fonts. Whilst these are adequate for a restricted range of purposes, it is highly desirable when transferring documents from another site to be able to assume the existence of a far wider range of utilities. For example, it may be necessary to rely on BIB \TeX , or on MakeIndex; it may be useful to be able to assume the existence of BM2FONT; and so on. Rather than simply say "all of these can be found on the nearest CTAN archive", it would be better if all implementations contained a standard subset of the available tools. It is therefore the aim of this project to identify what the elements of this subset should be, and then to liaise with developers and implementors to ensure that this subset is available for, and distributed with, each \TeX implementation.
- (2) Extended \TeX (e- \TeX): whilst the test bed and production system approach is philosophically very sound, the reality at the moment is that the group lacks the resources to bring it to fruition. None the less, there are many areas in which a large group of existing \TeX users believe that improvements could be made within the philosophical constraints of the existing \TeX implementation. E- \TeX is an attempt to satisfy their needs which could be accomplished without a major investment of resources, and which can pursued without the need for additional paid labour.

Finally the group agreed to individually undertake particular responsibilities; these are to be:

Peter Breitenlohner: Remove any existing incompatibilities between \TeX -XET and \TeX , with the idea of basing further e- \TeX developments on \TeX -XET; liaise with Chris Thompson concerning portability of the code; produce a catalogue of proposed extensions to e- \TeX .

Joachim Lammarsch: liaise with vendors and publishers in an attempt to raise money for the

implementation of NTS proper; arrange a further meeting of interested parties; liaise with Eberhard Mattes concerning the present constraints on the unbundling of em \TeX ; negotiate with leading academics concerning possible academic involvement in the project.

Mariusz Olko: take responsibility for the multilingual aspects of e- \TeX and NTS; discuss the possibility of siting the NTS programming team in Poland; discuss the possibility of academic involvement with leading Polish academics.

Bernd Raichle: endeavour to get \TeX -XET integrated into the standard UNIX distribution; prepare a list of proposed extensions to e- \TeX ; lead discussions on NTS-L.

Friedhelm Sowa: primary responsibility for finance; prepare proposals for a unified user interface and for unification of the integration of graphics; liaise with the Czech/Slovak groups concerning possible siting of the NTS programming team in the Czech Republic or Slovakia; discuss possible academic involvement with leading academics.

Philip Taylor: Overall technical responsibility for all aspects of the project; liaise with other potential NTS core group members; prepare and circulate a summary of the decisions of this and future meetings.

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

Software & Tools

Two Extensions to GNU Emacs that Are Useful when Editing \TeX Documents

Thomas Becker

Introduction

One of the most outstanding features of the GNU Emacs editor is the fact that it is customizable in the best and widest sense of the word. In this note, we present two extensions to GNU Emacs that are particularly useful when editing \TeX or L \TeX documents; these extensions were written by the author while typesetting a 574 page book

in L^AT_EX. The first package actually consists of a single function that provides an intelligent way of automatically blinking matching opening dollars each time a dollar sign is inserted. The second one improves an existing general feature of GNU Emacs, namely, keyboard macros. These are particularly but not exclusively interesting for mathematical typesetting with T_EX and L^AT_EX.

Super-tex-dollar

As a GNU Emacs user, you know that when you insert a closing delimiter such as) in a buffer, Emacs will blink the matching opening delimiter for one second or until new input arrives. In fact, you can declare any character to be a closing delimiter and tell Emacs what the matching opening delimiter is supposed to be. Emacs also knows that there is at least one self-matching delimiter known to humankind, namely, T_EX's dollar sign. Emacs' regular tex-mode makes the dollar sign a self-matching delimiter. The effect of this is that each time a dollar is inserted, the preceding dollar will blink. This blinking will skip a dollar that immediately precedes the one that is being inserted. This behavior is undoubtedly helpful when editing T_EX or L^AT_EX documents. I have also seen tex-modes for GNU Emacs that tried to be more intelligent about the dollar sign. However, everything that I have seen thus far along these lines has been, in one way or another, incomplete or outright annoying.

The function super-tex-dollar tries to provide a clean, safe, and intelligent way of dealing with the dollar sign when editing T_EX or L^AT_EX documents. The function is to be bound to the \$-key whenever a .tex file is being visited, so that it is invoked every time a dollar is inserted. (The mini-manual that comes with super-tex-dollar explains how to achieve this.) This is of course the kind of software that should not and does not require studying a manual before it can be used. You install it, continue to work as usual, and see if you like what is happening on your screen. The following short description of super-tex-dollar is meant to help you decide if you want to try this at all.

T_EX requires that all open dollars be closed at the end of a paragraph. Therefore, super-tex-dollar's basic strategy is to investigate the dollar situation between the beginning of the current paragraph and the current cursor position (*point* in Emacs terminology) and then decide what to do about the dollar that is being inserted. Now there are quite a few ways to start a paragraph in T_EX or

L^AT_EX, many of them unpredictable, so super-tex-dollar simply assumes that there is always at least one blank line between paragraphs. In order to get meaningful results and good performance, you must therefore make sure that a command like \chapter in L^AT_EX is always preceded or followed by a blank line. This is certainly not a bad idea anyway, but if you are not comfortable with it, then super-tex-dollar is not for you.

If super-tex-dollar finds that all opening dollars have been closed in the present paragraph up to the cursor position, then it will simply insert a dollar. When you type the closing dollar after having inserted your math formula, a dollar will be inserted and the opening dollar will blink for one second or until you continue typing. The next opening dollar will once again be inserted plainly. It should be clear that this behaviour gives you a lot more information than Emacs' default blinking as described above; in particular, if you have created a mess by deleting things in previously written text, you can locate the trouble by erasing and reinserting dollars.

Before we discuss super-tex-dollar's handling of \$\$'s, a few comments about displayed formulas in L^AT_EX are in order. If you are a L^AT_EX user, then you probably use

```
\begin{displaymath}
<formula>
\end{displaymath}
```

or \[(<formula>)\] to create displayed formulas. It is true that \begin{math}<formula>\end{math} and \(<formula>\) are both equivalent to \$<formula>\$, while

```
\begin{displaymath}
<formula>
\end{displaymath}
```

and \[(<formula>)\] are not exactly the same as \$\$<formula>\$\$. There are sometimes minuscule differences in vertical spacing, but I do not know of a situation where the double dollar produces something unwanted. The only real difference I can see is that the double dollar is more convenient to type and offers more flexibility because of the \eqno feature.

If you type an opening dollar and then another one immediately following it, then super-tex-dollar will insert this second one without any blinking: you have created an opening \$\$. Trying to insert a third dollar following the double dollar will have no effect whatsoever. When you type a dollar after having inserted your displayed formula, this dollar will automatically be doubled and the (first of the)

opening double dollars will blink. Trying to insert a third dollar after the closing double dollar will blink the opening one but not insert anything. In particular, if, out of habit, you close the opening double dollar by typing two dollars in succession, this will have the same effect as typing a single dollar.

If you have typed $\$(formula)$ and then decide that you really want this to be a displayed formula, then you can achieve this by typing two dollars at this point. The first one will of course be interpreted as the closing one for the opening dollar at the beginning of the formula. The second one, however, will cause that opening dollar to blink and be doubled automatically, so that you are now looking at $\$\$(formula)\$\$$. Again, trying to insert a third dollar will do nothing but blink the opening double dollar.

There is one situation in connection with double dollars for which there does not seem to be a perfect solution. Suppose you want to type

```
$$
y = \cases{x & if\quad x>0$\cr
           0 & otherwise.\cr}
$$
```

The first two dollars, i.e., the opening \$\$, will be inserted plainly. The third dollar will be seen by super-tex-dollar as an attempt to close the double dollar: it will be automatically doubled, and the opening double dollar will blink. To get what you want, you must now delete a character backwards. From then on, however, super-tex-dollar will once again know what is going on. The fourth dollar will be interpreted correctly as the closing for the preceding one. The attempt to insert another dollar immediately following the fourth one will be denied, and you will get the message "Dangling \$\$". Closing it now would leave an uneven number of \$'s in between." When the fifth dollar is inserted, this will again be interpreted as an attempt to close the opening double dollar and handled accordingly by automatic doubling and blinking. Deleting one character backwards will enable you to insert more pairs of single dollars, with the same behavior as in the case of the first pair. Instead of deleting a dollar backwards, you may of course always enforce plain insertion of single dollars by typing C-q \$.

How does super-tex-dollar cope with garbage encountered when checking the dollars in the current paragraph? When super-tex-dollar encounters a triple dollar, it concludes that no meaningful conclusions are possible. It assumes that all \$'s and \$\$'s have been closed at this point, continues its regular operation based on that assumption, and displays

an appropriate warning including the number of the line that contains the triple dollar. I do not know of a situation where the sequence \$xxx\$\$ — with the first dollar being an opening one — is meaningful in TeX. When super-tex-dollar encounters it, it will implicitly assume that the opening dollar has been closed before the double dollar. It will also display a warning that informs you of the problem and the number of the line where it occurs.

The handling of %, \%, and \\$ is as follows. If the cursor position is preceded by a % on the same line, then a \\$ is inserted like an ordinary character. When super-tex-dollar encounters a % earlier in the paragraph, it ignores the rest of that line. Moreover, it fully recognizes the fact that a \ quotes a \\$ as well as a %. However, it will see \\\$ and \\% as quoted \\$ and % as well.

The time that it takes super-tex-dollar to decide what to do increases linearly with the length of the region from the beginning of the paragraph to the cursor position, and with the number of dollars therein. A delay is not noticeable under normal circumstances, and it is negligible under all circumstances that are anywhere close to normal (i.e., on today's personal computers and workstations, and assuming that you do not write ridiculously long paragraphs with absurdly many dollar signs). As with Emacs' blinking of matching opening delimiters, the blinking is always interrupted when the user continues to type. The byte-compiled code of super-tex-dollar takes up 2.5 kB when loaded into Emacs. The space consumption of the program at runtime is always negligible: the position of each encountered opening \\$ or \$\$ will be forgotten as soon as it has been closed.

For information on how to obtain super-tex-dollar, see Section "Availability" below.

Emacs

When TeX is being criticized for not providing WYSIWYG, TeX buffs like to retort by saying that WYSIWYG is for wimps. I tend to agree. On the other hand, I have had some weak moments when I got tired of typing

```
$$
\begin{array}{rccc}
& : & \longrightarrow & \\
& & \longmapsto & \\
\end{array}
$$
```

for the umpteenth time just to get something like

$$\begin{aligned} f : [0, 1] &\longrightarrow [0, 1] \\ &x \longmapsto x^2. \end{aligned}$$

Even something like

```
\begin{corollary}

```

```
\end{corollary}
```

gets to be a drag after a while. There is of course the possibility of using *TeX* macros—with parameters if necessary—in this situation. On the other hand, there are very good reasons not to define a *TeX* macro every time you find yourself typing something more than three times. I was soon led to the conclusion that the appropriate solution in this situation is the use of *keyboard macros* on the editor level, where you issue some short, mnemonic command to insert a long and complicated string, with the cursor moving to a particular position if appropriate.

GNU Emacs provides keyboard macros.¹ However, I soon found out that Emacs' keyboard macros are the only feature that is somewhat underdeveloped in an otherwise perfect editor. I have therefore written a package called *Emacros* that adds a number of conveniences such as easy saving and reloading of macros and help with remembering macronames. A detailed manual comes with the package; in the sequel, we give a short general description of its capabilities.

Emacros' way of saving macro definitions to files is based on the idea that macro definitions should be separated by major modes to which they pertain. The macros used when editing a *TeX*-file, for example, will not be needed when working on a C-program. Moreover, within each mode, there will be macros that should be available whenever Emacs is in that mode, and others that are relevant for specific projects only. Consequently, each mode should allow one global macro file and several local ones in different directories as needed. This arrangement saves time and space and makes it easy to keep track of existing macro definitions.

A keyboard macro really consists of two components: the (complicated) string which is to be inserted and the (short) command which invokes this insertion. Here, we will refer to the string as the *macro*, and to the command as its *name*. In GNU Emacs, the key sequence **C-x)** starts the definition of a macro: the keystrokes following the command have the usual effect on the current buffer, while they are at the same time memorized to be inserted

automatically as a macro later on. The key sequence **C-x e**) ends this process; the macro can now be inserted before the cursor by typing **C-x e**. Note that a macro may not only contain self-insert commands, i.e., ordinary text, but arbitrary keyboard input. You can, for example, define a macro that creates

```
\begin{corollary}

```

```
\end{corollary}
```

on the screen, with the cursor, represented by the underscore, at the beginning of the blank line.

To be able to use the macro after defining another one, it must be given a name. This can be done by means of the Emacs function `name-last-kbd-macro`. This function is adequate if the macro is to be used in the current session only and if, moreover, there are very few macros around so that one can easily memorize them all. Otherwise, this is where *Emacros* comes in. The macro can now be named using the new function `emacros-name-last-kbd-macro-add`. This function first prompts the user for a name, enforcing appropriate restrictions. Next, the function saves the macro definition to a file named *mode-mac.el*, where *mode* is the current major mode, for reloading in future sessions. This file can be in the directory for global macros, in which case the macro will be available whenever *mode* is the major mode, or it can be in the current directory, in which case the macro will be locally available whenever *mode* is the major mode and the file that is being visited is from this directory. The function will ask you to choose between *l* for local and *g* for global. When the function is called with prefix argument, then you will be prompted to explicitly enter the name of a file to save the macro to.

Once a macro *macro* has a name *macroname*, this name is in fact a command which causes the macro to be inserted before the cursor: typing **M-x macroname RET** inserts *macro*. This has the disadvantage that completion takes into account all command names rather than just macro names. *Emacros* therefore provides a function specifically for executing keyboard macros. As a further convenience for the impatient (which was motivated by the attempt to make macro insertion no more tedious than using a *TeX* macro), there is a function called `emacros-auto-execute-named-macro`. This function will prompt for the name of a macro in the minibuffer. The cursor will stay at its position in the current buffer. As soon as the sequence that you have entered matches the name of a macro, this

¹ Using an editor like GNU Emacs to the full extent of its capabilities does of course require some effort and a certain computer maturity; but then, we are not wimps like the rest of them, remember?

macro is inserted and regular editing is resumed without the need to type a RET.

Every time you read a file into Emacs, *Emacros* invokes a function that will load those macros that have been saved to files named *mode-mac.el* in the current directory and in the directory for global macros. Here, *mode* is the major mode which Emacs has chosen for the visited file. Macro files that have been loaded before during the same session will be disregarded. If you have been editing a file and then read another one with a different mode and/or from a different directory, then the macros pertaining to the new file will be loaded, and all others that were loaded previously will remain active as well. If there are not too many macros around, this is probably what you want. In the long run, however, especially when you are one of those users that never leave Emacs, you would end up with all macros being loaded, thus rendering the separation into different files pointless. The function *emacros-refresh-macros* takes care of this problem. It will erase all previously loaded macros and load the ones pertaining to the current buffer, thus creating the same situation as if you had just started Emacs and read in the file that the current buffer is visiting.

There are three functions that allow you to manipulate macro definitions that have already been saved. The function *emacros-rename-macro* assigns a new name to a previously named macro, making the change effective in the current session and in the local or global macro file pertaining to the current buffer, as appropriate. The function *emacros-move-macro* moves macro definitions between the local and global file pertaining to the current buffer. Finally, the function *emacros-remove-macro* deletes macros from the current macro files and disables them in the current session.

Three functions provide help with keyboard macros. (The manual tells you how to make these available as help options.) The first of these will display in Emacs' help window a list of all currently defined macronames and the corresponding macros. The second one prompts you for a macro and then tells you its name. The third one acts like the second one, except that it also inserts the macro whose name you were asking for after the point in the current buffer, assuming that you were asking because you wanted to use the macro. The possibility to complete when entering the macro makes this an attractive way to insert, making it worthwhile using macros even if you never ever remember the name of one.

When I wrote *Emacros*, I made a strong effort to conform with Emacs' general style, both in terms of source code and in terms of look-and-feel. Completion is supported whenever an existing macro or macroname is to be entered, defaults are offered whenever there is the remotest chance of anticipating what the user wants to do next, and messages appear whenever the user tries to do something meaningless or dangerous. The byte-compiled code takes up 16 kB; otherwise, the space consumption is only a trifle more than what is needed to store your macros and their names.

Super-tex-dollar and Emacros Combined

There are two things that need to be said about using super-tex-dollar and *Emacros* together. When a dollar sign occurs in a keyboard macro, it should always be inserted as C-q \$ when defining the macro. That way, you do not get the blinking and, possibly, doubling of dollars when the macro is being executed. With this in mind, you will find that the unwanted doubling when placing single dollars between a pair of double dollars (see Section "Super-tex-dollar" above) becomes a rather rare occurrence. For example, I have a macro named *cas*, so that—with the function *emacros-auto-execute-named-macro* bound to M-\—I can type M-\ cas, and voilà, I have

```
\cases{_& if\quad $ \$\cr
      & if\quad $ \$\cr
      & otherwise.\cr}
```

on the screen, with the cursor in the position indicated by the underscore. All I have to do now is to fill in things and perhaps delete or copy the middle line. The whole thing is most likely to be in a displayed formula; the double dollars will now be handled correctly by super-tex-dollar.

Availability

Both the Superdollar package and the *Emacros* package are available via ftp from

alice.fmi.uni-passau.de

where they are to be found in the directory *pub/emacs_contrib*. The *Emacros* package will also be made part of the GNU Emacs distribution in the near future. Both packages come with manuals explaining installation and usage.

- ◊ Thomas Becker
Fakultät für Mathematik und Informatik
Universität Passau
94030 Passau
Germany
becker@alice.fmi.uni-passau.de

Icons for **TEX** and **METAFONT**

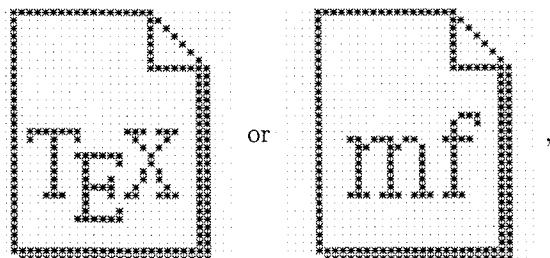
Donald E. Knuth

Macintosh users have long been accustomed to seeing their files displayed graphically in “iconic” form. I recently acquired a workstation with a window system and file management software that gave me a similar opportunity to visualize my own UNIX files; so naturally I wanted my **TEX**-related material to be represented by suitable icons. The purpose of this note is to present the icons I came up with, in hopes that other users might enjoy working with them and/or enhancing them.

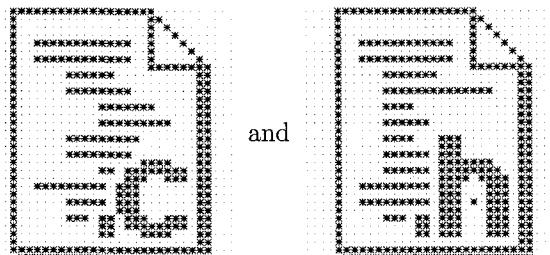
The file manager on my new machine invokes a “classing engine”, which looks at each file’s name and/or contents to decide what kind of file it is. Every file type is then represented by a 32×32 bitmap called its *icon*, together with another 32×32 bitmap called its *icon mask*. In bit positions where the icon mask is 1, the file manager displays one of two pixel colors, called the foreground and background colors, depending on whether the icon has 1 or 0 in that position. (The foreground and background colors may be different for each file type.) In other positions of the bitmap, where the icon mask is 0, the file manager displays its own background color.

Thus, I was able to fit my **TEX** and **METAFONT** files into the file manager’s scheme as soon as I designed appropriate icons and masks, once I had told the classing engine how to identify particular types of files.

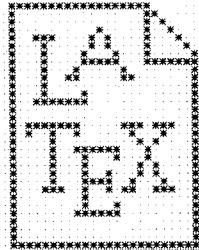
For example, I decided that each file whose name ends with **.tex** or **.mf** should be iconified with the bitmaps



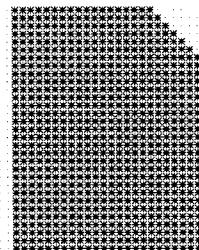
respectively; these are compatible with the existing scheme in which C program source and header files, identified by suffixes **.c** and **.h**, have



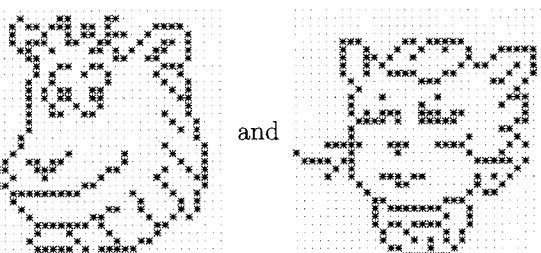
as icons. Similarly, a file named ***.ltx** will get the icon



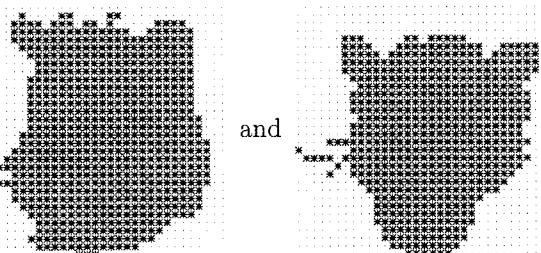
In each case the corresponding icon mask is one that the file manager already has built in as the `Generic_Doc_glyph_mask`, namely



The transcript files output by **TEX** and **METAFONT** provided me with a more interesting design problem. They’re both named ***.log** on my system, so they can’t be distinguished by file name. I decided that any file whose first 12 bytes are the ASCII characters ‘**This_is_TeX**,’ should be considered a **TEX** transcript, and any file that begins with ‘**This_is_METAfont**,’ should be considered a **METAFONT** transcript. The corresponding icons were fun to make; I based them on the illustrations Duane Bibby had drawn for the user manuals:



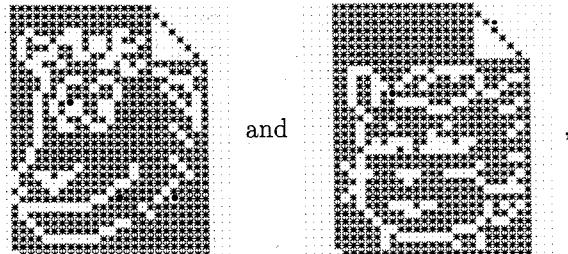
and



respectively.

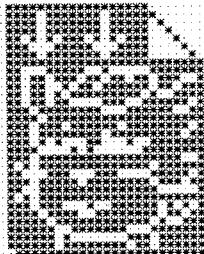
TEX’s main output is, of course, a device-independent (**.dvi**) file, and **METAFONT** produces

generic font (`gf`) files. I decided to represent such files by

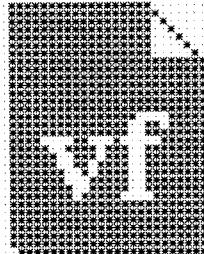


and

because they are analogous to photographic “negatives” that need to be “developed” by other software. When a `gf` file has been packed into a `pk` file, its icon will change to

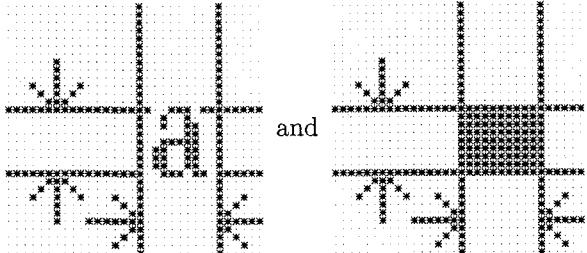


Virtual font files are represented by an analogous



These file types are identifiable by the respective names `*.dvi`, `*.gf`, `*.pk`, `*.vf`, and they can also be identified by content: The first byte always has the numerical value 247 (octal 367), then the next byte is respectively 2, 131, 89, 202 (octal 002, 203, 131, 312) for `dvi`, `gf`, `pk`, or `vf`.

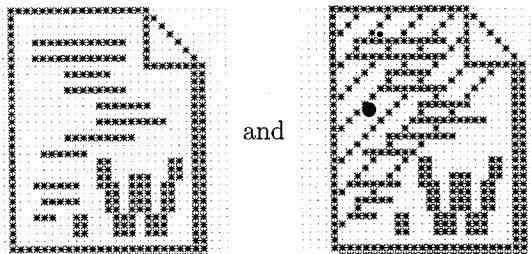
The other principal output of METAFONT is a font metric file, which can be identified by the suffix `.tfm` in its name. I assigned the following icon and mask to such files:



and

I do all my programming nowadays in the CWEB language [1, 2, 3, 4], hence I also accumulate lots of files of two additional types. CWEB source files

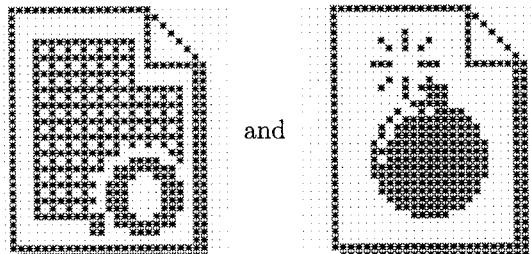
are identified by the suffix `.w`, and CWEB change files have the suffix `.ch`; the corresponding icons



and

are intended to blend with the system’s existing conventions for `.c` and `.h` files, mentioned above.

What foreground colors and background colors should be assigned to these icons? I’m not sure. At the moment I have a grayscale monitor, not color, so I don’t have enough experience to recommend particular choices. Setting all the foreground colors equal to basic black (RGB values $\langle 0, 0, 0 \rangle$) has worked fine; but I don’t want all the background colors to be pure white (RGB $\langle 255, 255, 255 \rangle$). I’m tentatively using pure white for the background color of the “negative” icons (`dvi`, `gf`, `pk`, and `vf`), and off-white (RGB $\langle 230, 230, 230 \rangle$) for the background of transcript icons. The `TEX` and `METAFONT` source file icons currently have background RGB values $\langle 200, 200, 255 \rangle$, corresponding to light blue; font metric icons and `LATeX` source icons have background RGB values $\langle 255, 200, 200 \rangle$, light red. (I should perhaps have given `METAFONT` source files an orange hue, more in keeping with the cover of *The METAFONTbook*.) On my grayscale monitor I had to lighten the background color assigned by the system software to C object files and to coredump files (`*.o` and `core*`); otherwise it was impossible for me to see the detail of the system icons

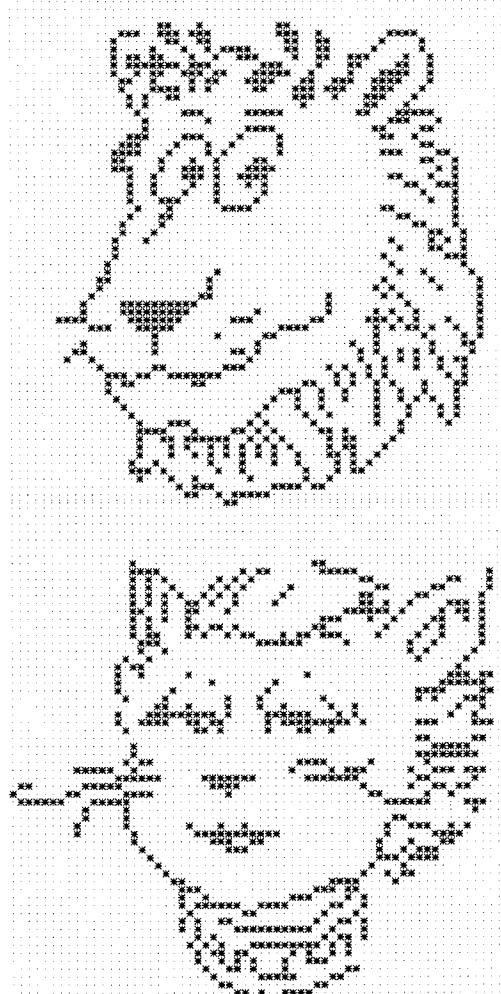


and

I expect other users will need to adjust foreground and background colors to go with the decor of their own desktops.

In 1989 I had my first opportunity to work with a personal graphic workstation, and I immediately decided to make 64×64-bit icons for `TEX` and `METAFONT`—for the programs, not for the files. But I’ve always found it more convenient to run `TEX` and `METAFONT` from UNIX shells, so I never have used

those early icons. Here they are, still waiting for their proper raison d'être:



All of the icons shown above, except for those already present in directory `/usr/openwin/share/include/images` of Sun Microsystem's OpenWindows distribution, can be obtained via anonymous ftp from directory `~ftp/pub/tex/icons` at `labrea.stanford.edu` on the Internet. That directory also contains a file called `cetex.ascii`, which can be used to install the icons into OpenWindows by saying '`ce_db_merge system -from_ascii cetex.ascii`'.

References

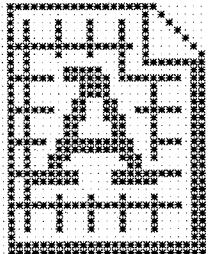
- [1] CWEB public distribution, available by anonymous ftp from directory `~ftp/pub/cweb` at `labrea.stanford.edu`.
- [2] Silvio Levy and Donald E. Knuth, The CWEB System of Structured Documentation, Stanford Computer Science report STAN-CS-1336

(Stanford, California, October 1990), 200 pp. An up-to-date version is available online in [1].

- [3] Donald E. Knuth, *Literate Programming* (Stanford, California: Center for the Study of Language and Information, 1992), xvi+368 pp. (CSLI Lecture Notes, no. 27.) Distributed by the University of Chicago Press.
- [4] Donald E. Knuth, *The Stanford GraphBase: A Platform for Combinatorial Computing* (New York: ACM Press, 1993).

◊ Donald E. Knuth
Stanford University

Editor's note: An additional icon, for files whose name ends with `.sty`, has been provided by Peter Flynn:



This icon has been deposited in the CTAN archives in the file `.../digests/tugboat/sty.icon`

bibview: A graphical user interface to BibTeX

Armin Liebl

Abstract

This paper describes an X Window application for manipulating BibTeX databases. The application provides the following facilities: creation of new entries, deletion and editing of entries, searching for entries, sorting and printing BibTeX databases or subsets of them. It is possible to work with several BibTeX databases simultaneously and copy entries between databases. Entry types other than the standard BibTeX types can be defined in a configuration file. The paper describes the features of the program. It contains a comparison of *bibview* with similar tools and discusses some useful improvements.

1 The windows of *bibview*

bibview uses the following types of windows:

- The *main window* contains five menus described in Section 1.1.
- A *bibliography window* is displayed for each BibTeX database loaded. It offers features to manipulate a single BibTeX database, such as making new entries, sorting the database, etc.
- A *list window* shows a list of all entries of a BibTeX database or the entries resulting from a search, respectively.
- A *card window* provides a template to edit the fields of an entry or to create a new entry.
- A *macro window* is used to edit the @STRING and @PREAMBLE parts of a BibTeX database.
- In a *search window* the user can specify (using regular expressions) the entries he/she is looking for.
- An *error window* can be used to correct syntax errors in a BibTeX database. An additional window contains information concerning the syntax errors.
- *Help windows* show help information.

1.1 Main window

The *main window* of *bibview* (see Figure 1) provides the following menus:

- **File** – Open a BibTeX database, create a new BibTeX database, or close/save an open BibTeX database. The name of the database is chosen through a file selection box.
- **Services** – The following services are available:
 - *Consistency Check*: The entries not containing all fields required by BibTeX are displayed in a *list window*. Note, however,

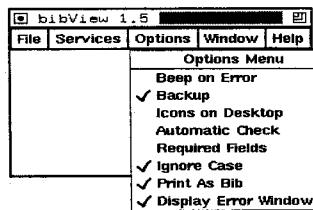


Figure 1: Main Window

that cross references are not checked, i.e., an entry is regarded as complete if it contains a cross reference.

- *Unify*: All entries of a BibTeX database are inserted into another loaded BibTeX database. If key conflicts occur, a new unique key is generated by appending a letter (a-z, A-Z) to the conflicting key.
- *Print*: Print a BibTeX database. The database can be printed as a LATEX file with \nocite commands or in a fixed format defined by *bibview*.
- *Edit Macros*: A window is displayed for editing the @STRING and @PREAMBLE parts of a BibTeX database.
- *Load Configuration*: A configuration file is evaluated. The structure of a configuration file is described in Section 2. The [Options] part of the configuration file is not evaluated.
- **Options** – *bibview* can be customized according to the preferences of the user. The default of the options can be changed in the configuration file. If an option is set, it is marked by a tick (see Figure 1). The following options are available:
 - *Beep on Error*: Beep if an error occurs (default: true).
 - *Backup*: Before an existing database is written to disk, a backup of the database is created with the suffix .bak.(i) where (i) is the number of the last backup incremented by one (default: true).
 - *Icons on Desktop*: Icons of list and card windows are placed within the corresponding bibliography window (default: false).
 - *Automatic Check*: A consistency check takes place whenever a BibTeX database is loaded (default: true).
 - *Required Fields*: A warning message is displayed if an entry is saved that does not contain all fields required by BibTeX (default: false).
 - *Ignore Case*: In a search the case of the letters is ignored (default: true).

- *Print As Bib*: When printing a database, a L^AT_EX file containing \nocite commands is created. The style file is alpha. This default can be changed in the [StyleFile] part of the configuration file.
- *Display Error Window*: If a syntax error is found while loading a BIBTEX database, a window is displayed for possible corrections of the error (default: true).
- **Window** - Windows belonging to the same BIBTEX database are grouped together.
- **Help** - A *help window* is displayed containing help information.

1.2 Bibliography Window

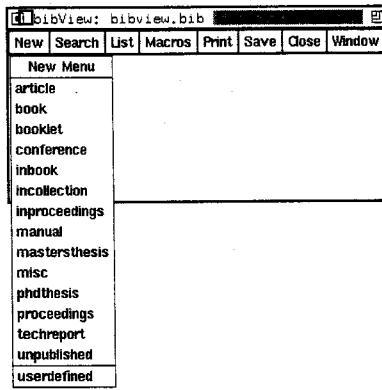


Figure 2: Bibliography Window

The *bibliography window* (see Figure 2) provides the following commands:

- **New**: Select the type of a new entry from a menu. Entries of a type other than the standard BIBTEX types or the types defined in the configuration file can be created by selecting *userdefined*. A *card window* is displayed for making the new entry.
- **Search**: A *search window* is displayed in which a search for entries can be initiated. The result of the search is shown in the *list window*.
- **List**: A list with all entries of the BIBTEX database is displayed.
- **Macros**: A *macro window* is displayed.
- **Print**: A L^AT_EX file with the entries of the database is produced.
- **Save**: The BIBTEX database is saved on secondary storage in a format conforming to BIBTEX's specification.
- **Close**: Close the BIBTEX database.

1.3 Card Window

The *card window* provides a template of the required as well as optional fields for each entry type defined by BIBTEX or in the configuration file. Figure 3 shows the card window for the type 'article'. Required fields are marked by bold lines. A card window is used to make a new entry or to edit an entry (eg to correct spelling errors). A card window for editing an entry has been clicked in the list window. It is possible to have several card windows displayed simultaneously. This is useful to cut and paste information between different entries.

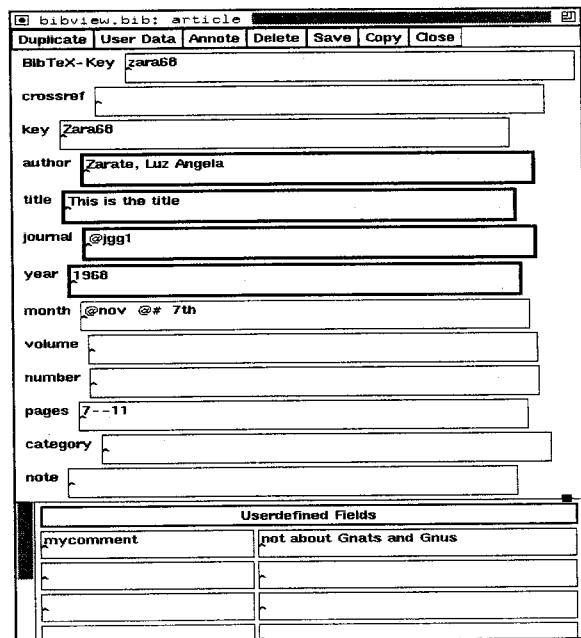


Figure 3: Card Window

Macros are entered with a preceding '@', otherwise *bibview* automatically surrounds the entry with brackets. In our example, *bibview* will produce

```
@article{zara68,
  key      = {Zara68},
  author   = {Zarate, Luz Angela},
  title    = {This is the title},
  journal  = jgg1,
  year     = {1968},
  month    = nov # {7th},
  pages    = {7--11},
  mycomment = {not about Gnats and Gnus}
}
```

The *card window* provides the following commands:

- **Duplicate:** Duplicate an (already saved) entry. The type of the new entry can be chosen in the menu. Fields that are not standard fields of the new type become user-defined fields. This feature is useful if one wants to make a new entry that has many fields in common with an existing entry. It can also be used to change the type of an entry.
- **UserData:** Additional fields not used by BIBTeX can be entered in the section *Userdefined Fields* of the *card window*.
- **Annote:** Edit the ‘annote’ field of an entry in a window. This field can be used to contain an abstract.
- **Delete:** Delete an entry. This is useful if one discovers a duplicate entry in a BIBTeX database.
- **Save:** Save an entry. If the option *Required Fields* is chosen, a warning message is displayed if not all fields required by BIBTeX were entered. If no BIBTeX key exists, *bibview* generates it. If key conflicts occur, a letter (a-z,A-Z) is appended in a unique way. This allows 52 different entries with the same BIBTeX key. As no check for syntax errors occurs when an entry is saved, it is the task of the user to care for the correctness of the entry.
- **Copy:** Insert an entry into another loaded BIBTeX database. Key conflicts are solved as described above.
- **Close:** Close the *card window*.

1.4 List Window

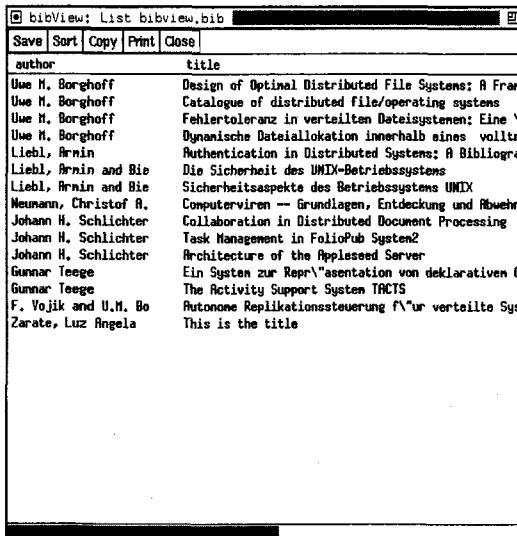


Figure 4: List Window

The *list window* is displayed after the *List* button of the *bibliography window* has been pressed or as a result of a search or consistency check. As there is not more than one list window per BIBTeX database, an already existing list is overwritten.

The list contains by default information concerning *author*, *title*, *year*, *BIBTeX key*, *category* and *BIBTeX type* of an entry. This is useful if one wants to browse through BIBTeX databases.

The information displayed in the list and the layout of the list can be changed in the configuration file.

The following commands are provided:

- **Save:** Save the entries of the list as a new BIBTeX database. This feature can be used to partition a BIBTeX database.
- **Sort:** Sort the entries of the list. By default the list can be sorted by all standard BibTeX fields. The author and editor fields are sorted by “last name”. The sort order is used when the list is saved or printed.
- **Copy:** Insert all entries of the list into another loaded BIBTeX database. Key conflicts are solved in the way described in Section 1.3.
- **Print:** Produce a LATEX file with the entries of the list.
- **Close:** Close the *list window*.

1.5 Search Window

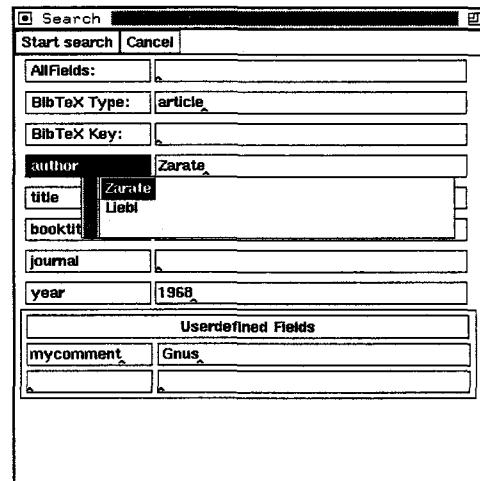


Figure 5: Search Window

bibview allows to search for entries matching regular expressions in certain fields. The result of the

search are the entries whose fields match all regular expressions specified in the search window.

If a regular expression is entered in the box *All Fields*, the entries that match the expression in any field (including the user-defined fields) are displayed in the list window. It is possible to use the *AllFields* box in combination with the other boxes.

It is possible to use (not more than two) user-defined fields in a search. In the left box of the "Userdefined Fields" part of the search window the exact name of the user-defined field is entered, in the right box a regular expression is entered.

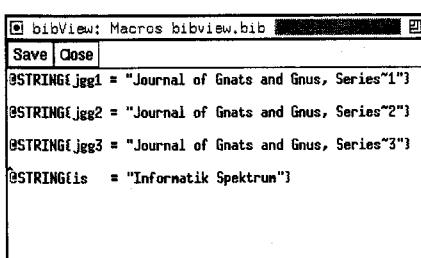
In our example (Figure 5), we search for all articles by author **Zarate** that were published in 1968 and for which the 'mycomment' field contains the string **Gnus**.

Regular expressions for each field can be defined in the configuration file. A predefined expression is selected by pressing the left mouse button in the box belonging to the field. In Figure 5 the expressions 'Zarate' and 'Liebl' were defined for the 'author' field.

The fields that are available in the search window can be defined in the configuration file. By default, all standard BIBTEX fields can be used.

1.6 Macro Window

A *macro window* is used to edit the **QSTRING** and **QPREAMBLE** parts of a BIBTEX file. As the content of the *macro window* is not checked for syntax errors when the database is written, it is the task of the user to care for a syntactically correct definition of the macros. An example of a *macro window* is shown in Figure 6.



```

@bibView: Macros bibview.bib
Save Close
\string{jgg1 = "Journal of Gnats and Gnus, Series"1"}%
\string{jgg2 = "Journal of Gnats and Gnus, Series"2"}%
\string{jgg3 = "Journal of Gnats and Gnus, Series"3"}%
\string{jggis = "Informatik Spektrum"}%

```

Figure 6: Macro Window

1.7 Error Window

An *error window* is used to correct syntax errors in BIBTEX databases. If syntax errors are found when a database is loaded, *bibview* reads the correct entries of the BIBTEX database, but the incorrect entries will be lost. To avoid this, the user should correct the syntax errors, save the database and load it again. A help window shows the BIBTEX keys of

the incorrect entries (see Figure 7). LINE refers to the line number in the database, and OFFSET refers to the line within the entry. With this information it is easy to correct errors by searching for the key and using OFFSET to find the erroneous line.

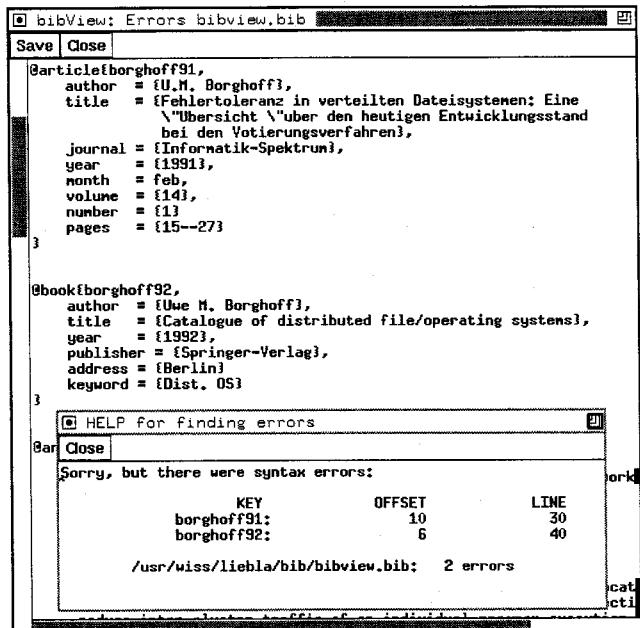


Figure 7: Error Window

2 The configuration file

When *bibview* is started, it looks for a configuration file with the name **.bibviewrc** in the user's home directory. In this file, some user-specific default values can be specified. The following sections are allowed:

- [Options]: The default options can be customized to the preferences of the user.
- [Types]: New BIBTEX types can be defined and additional fields can be added to existing types. An example of a type definition is:

```

t : isonorm
rf : title
rf : number
f : year
f : note

```

t is used to define a new type or to add fields to an already defined type. With **tc** the fields of an already defined type are undefined.

Additional fields to all already defined types can be defined with **t : all** followed by the descriptions of the fields.

rf indicates that the following field is a required field for the defined type.

In the example above, a type 'isonorm' is defined with fields 'year' and 'note' and required fields 'title' and 'number'.

- [ListFields]: The fields that are displayed in the list window and the layout of the list are defined. The definition for the list shown in Figure 4 is:

```
author : 20
title : 50
mainkey : 10
cardtype : 10
```

The field names `mainkey` and `cardtype` are used for the BIBTEX key and BIBTEX type of an entry. Following the colon, the number of characters is given that is used for displaying the field.

- [SearchFields]: The fields that are displayed in the search window are defined. With `$clear$` the fields that are displayed in a search window by default (all standard BIBTEX fields) are overridden. The definition for the search fields of Figure 5 is:

```
$clear$
author
title
booktitle
journal
year
```

- [SortFields]: The fields that are offered in the sort menu of the list window can be chosen. With `$clear$` the fields that are used by default (all standard BIBTEX fields) are overridden. A line in the [SortFields] part has the form

```
<field name>
or
<field name> : <sort order>
<sort order> can be n if the field contains a name or d if the field contains a date of the form dd.mm.yyyy.
```

- [UserFields]: User defined field names can be defined for certain types or for all types. A field name for `<type>` is defined by

```
<type> : <field name>
A field name for all fields is defined by
<type> : all
```

- [Predefines]: Data can be predefined for use in the *search window*. The field names `mainkey` and `cardtype` are used for the BIBTEX key and BIBTEX type of an entry. `allfields` defines data for the *Allfields* box of the search window. The definition for the predefined data of Figure 5 is:

```
author: Zarate
author: Liebl
```

- [LatexHeader]: A LATEX header is defined to be used in the LATEX file produced by the *Print* command.
- [LatexFooter]: A LATEX footer is defined to be used in the LATEX file produced by the *Print* command.
- [BibDir]: This section contains the directory that is initially used by the file select box.
- [StyleFile]: This section contains the name of the BIBTEX style that is used in the LATEX file produced by the *Print* command.
- [AnnoteField]: The name of the field that is used for annotations is entered. In most cases, this will be 'abstract' or 'annote'. The default name is 'annote'.
- [SortedBy]: The name of the field by which a BIBTEX file should be sorted by default.
- [Indent]: The format used when saving a BIBTEX database can be specified.

A configuration file can be loaded from within the *Services* menu of the *main window*. In this case the [Options] part is not evaluated.

The distribution of *bibview* contains an example of a configuration file. The syntax of the configuration file can be seen in this example.

3 Comparison with similar tools

In the last few years some tools have been developed for manipulating BIBTEX databases.

bibcard is a graphical interface to BIBTEX with features similar to *bibview*. Its user interface follows the OpenLook style. *bibview* provides additional facilities like printing a database and correcting syntax errors in an error window. An important advantage of *bibview* is the mechanism to define new types in a configuration file, because BIBTEX allows to create styles with new types. The searching and sorting facilities of *bibview* are more comfortable.

XBIBTEX (described in *TUGboat* 13, no. 4) is an X Window interface for inserting entries into a BIBTEX database. There are no features like sorting, searching or moving entries between databases.

The *lookbibtex* Perl script is intended for searching in BIBTEX databases. Its searching facilities are comparable to those of *bibview*.

bibadd, *bibsort* and *bibgrep* are tools for inserting entries, sorting BIBTEX databases by BIBTEX key and searching for entries with a given key word.

The advantage of *bibview* is that it offers the features of other tools in one single graphical user interface.

4 Limitations of *bibview*

bibview is intended for small personal databases. There may be problems with databases consisting of more than 1000 entries.

It is desirable to search in more than one database.

The consistency check should be more sophisticated and, for example, look for duplicate entries.

The algorithm for key generation is fixed. The user should be able to define his own method for key generation.

Comments in *BIBTEX* databases are accepted but ignored. They will be lost in the output produced by *bibview*.

5 How to Obtain *bibview*

The source for *bibview* can be obtained via anonymous ftp from `ftp.informatik.tu-muenchen.de` (current Internet address: 131.159.0.110) as `/pub/comp/typesetting/tex/bibview-1.5.tar.Z`. It must be transferred in 'binary' mode.

Acknowledgements

Holger Martin and Peter M. Urban implemented an earlier version of *bibview*. Prof. J. Schlichter and my colleagues helped to improve the tool.

- ◊ Armin Liebl
Technische Universität München
Fakultät für Informatik
Arcisstr. 21
80290 München
`liebla@informatik.tu-muenchen.de`

Bibliography Prettyprinting and Syntax Checking

Nelson H. F. Beebe

Contents

1	Introduction	395
2	<i>BIBTEX</i> needs improvement	396
3	Run-time options	397
4	Prettyprinting	399
5	Pattern matching and initialization files	400
6	Lexical analysis	403
7	Portability	404
8	SCRIBE bibliography format	405
9	Recommendations for <i>BIBTEX</i> design	405
10	A lexical grammar for <i>BIBTEX</i>	406
11	A parsing grammar for <i>BIBTEX</i>	409
12	Software availability	414
	References	414
	Index	415

List of Tables

1	Sample <i>bibclean</i> initialization file.	400
2	Escape sequences in quoted strings.	401
3	Initialization file pattern characters.	401

1 Introduction

BIBTEX [18, Appendix B] is a convenient tool for solving the vexing issue of bibliography formatting. The user identifies fields of bibliography entries via field/value pairs and provides a unique citation key and a document type for each entry. A simple string substitution facility makes it easy to reuse frequently-occurring strings. A typical example looks like this:

```
CString{pub-AW =
    "Ad{\-d}if{\-s}on-Wes{\-l}ey"}
```

```
@Book{Lamport:LDP85,
  author = "Leslie Lamport",
  title = "{$\backslash$LaTeX}---A Document Preparation System---User's Guide and Reference Manual",
  publisher = pub-AW,
  year = "1985",
  ISBN = "0-201-15790-X",
}
```

The *TeX* file contains citations of the form `\cite{Lamport:LDP85}`, together with a `\bibliographystyle` command to choose a citation and bibliography style, and a `\bibliography` command to specify which *BIBTEX* files are to be used. *TeX* records this information in an auxiliary file.

A subsequent **BIBTEX** job step reads this auxiliary file, extracts the requested bibliographic entries from the specified **BIBTEX** files, and outputs the entries into a bibliography file formatted according to the specified style. Several dozen such styles are currently available to help cope with the bizarre variations in bibliography formats that publishers have invented.

In a second **TeX** step, the `\cite` commands are not correctly expandable until the `\bibliography` command is processed and the bibliography file output by **BIBTEX** is read. However, at that point, the desired form of the citations is finally known, and at the end of the job, an updated auxiliary file is written.

A third **TeX** step finally has the necessary information from the auxiliary file and the bibliography file to correctly typeset the `\cite` commands and the bibliography in the specified style.

With the GNU Emacs text editor [7, 27], powerful **BIBTEX** editing support makes it simple to generate bibliography entry descriptions via templates that can be inserted with a couple of keystrokes, or on workstations, selected from a pop-up menu. This editor is freely available on UNIX, VAX VMS, and the larger members of the IBM PC family under PC-DOS.

The major benefits of using **BIBTEX** are the potential for data reuse, the separation of form and content (like the descriptive markup of **LATEX** and SGML[6, 31]), and the many stylistic variants of the typeset bibliography. During the preparation of this article, a scan of our Mathematics Department workstation file system located about 14 000 **TeX** files, and 445 **BIBTEX** files. The latter contained about 870 000 lines and almost 94 000 bibliography entries. These files form a valuable resource that authors and researchers can use to track and properly cite literature in their publications.

During my term as TUG President, I initiated a project to collect **BIBTEX** styles and bibliography data base files of material related to **TeX** and its uses, and electronic document production and typography in general. This dynamic collection also covers a few journals, including more than 1000 entries for *TUGboat*. A snapshot of part of the collection was published in the 1991 TUG Resource Directory [4, 5].

One drawback of **BIBTEX** is that errors in a bibliography file, such as unmatched quotation marks around a value string, can sometimes be hard to locate, because the current version of the program raises an error at the end of a scan when internal tables overflow after gobbling several thousand char-

acters of input. The result is that the error location is completely bogus, and actually lies much earlier in the file. We can hope that this serious deficiency will be remedied in the final version of **BIBTEX**, 1.0, which is expected to appear when the **LATEX** 3.0 development is completed.

Another drawback is that such bibliography files are normally prepared by human typists, and consequently there are formatting variations that reduce readability, and inconsistencies that persist into the final typeset bibliography. Some examples of such inconsistencies are variations in naming of publishers and journals, spacing around author and editor initials, and variations in letter case in titles. In addition, there are usually numerous typographical errors of omission, doubling, spelling, transcription, translation, and transposition.

In the fall of 1990, faced with a growing collection of **BIBTEX** files, I set out to write a software tool to deal with these problems. This program is called **bibclean**. It is a syntax checker, portability verifier, and prettyprinter, and was made freely available in 1991. In the fall of 1992, after considerable experience with the first version, I embarked on a set of enhancements that produced major version 2.0, and the purpose of this paper is to describe the new version, and to widely advertise its existence to the **TeX** community.

2 BibTEX needs improvement

BIBTEX, like **TeX**, assumes that its input is prepared correctly, and works best when that is the case. Both programs attempt to recover from errors, but that recovery may be unsuccessful, and errors may be detected only after lengthy processing. In neither case is the output of these programs suitable for input to them. That is, their knowledge of how their input streams are to be parsed is available only to them, and cannot be applied independently and used by other software. Both programs have a hazily-defined input syntax, and **TeX**'s is extensible, making it even harder to give a precise description to the user.

The trend of compiler technology development of the last two decades, largely on UNIX systems, has been to separate the compilation task into several steps.

The first is generally called *lexical analysis*, or lexing. It breaks the input stream up into identifiable tokens that can be represented by small integer constants and constant strings.

The second step is called *parsing*, which involves the verification that the tokens streaming

from the lexer conform to the grammatical requirements of the language, that is, that they make sense.

As parsing proceeds, an intermediate representation is prepared that is suitable for the third step, namely, *code generation* or *interpretation*.

This division into subtasks diminishes the complexity of writing a compiler, reduces its memory requirements, and importantly, partitions the job into two parts: a language-dependent, but *architecture-independent*, part consisting of lexing and parsing, and a language-independent, but *architecture-dependent*, part where code is generated or interpreted.

This makes it possible to write a front end for each language, and a back end for each architecture, and by combining them, obtain compilers for all languages and all architectures. The most successful example of this approach at present is almost certainly the Free Software Foundation's GNU Project compilers, which support all common computer architectures with the back ends, and C, C++, and Objective C with the front ends. Additional front ends for several other popular languages are in preparation.

When a lexer is available as a separate program, its output can be conveniently used by other programs for tasks such as database lookup, floating-point precision conversion, language translation, linguistic analysis, portability verification, prettyprinting, and checking of grammar, syntax, and spelling.

In response to a command-line request, `bibclean` will function as a lexer instead of as a prettyprinter. An example is given later in Section 6.

3 Run-time options

On several operating systems, `bibclean` is run by a command of the form

```
bibclean [options] bibfile(s) >newfile
```

One or more bibliography files can be specified; if none are given, input is taken from the standard input stream. A specific example is:

```
bibclean -no-fix-name mybib.bib >mybib.new
```

Command-line switches may be abbreviated to a unique leading prefix, and letter case is not significant. All options are parsed before any input bibliography files are read, no matter what their order on the command line. Options that correspond to a *yes/no* setting of a flag have a form with a prefix `no-` to set the flag to *no*. For such options, the last setting determines the flag value used. This is significant when options are also specified in initialization files (see Section 5).

On VAX VMS and IBM PC-DOS, the leading hyphen on option names may be replaced by a slash; however, the hyphen option prefix is always recognized.

-author Display an author credit on the standard error unit, `stderr`. Sometimes an executable program is separated from its documentation and source code; this option provides a way to recover from that.

-error-log filename Redirect `stderr` to the indicated file, which will then contain all of the error and warning messages. This option is provided for those systems that have difficulty redirecting `stderr`.

-help or -? Display a help message on `stderr`, giving a sample command usage, and option descriptions similar to the ones here.

-init-file filename Provide an explicit value pattern initialization file. It will be processed after any system-wide and job-wide initialization files found on the PATH (for VAX VMS, `SYS$SYSTEM`) and `BIBINPUTS` search paths, respectively, and may override them. It in turn may be overridden by a subsequent file-specific initialization file. The initialization file name can be changed at compile time, or at run time through a setting of the environment variable `BIBCLEANINI`, but defaults to `.bibcleanrc` on UNIX, and to `bibclean.ini` elsewhere. For further details, see Section 5.

-max-width nnn Normally, `bibclean` limits output line widths to 72 characters, and in the interests of consistency, that value should not be changed. Occasionally, special-purpose applications may require different maximum line widths, so this option provides that capability. The number following the option name can be specified in decimal, octal (starting with 0), or hexadecimal (starting with 0x). A zero or negative value is interpreted to mean unlimited, so `-max-width 0` can be used to ensure that each field/value pair appears on a single line.

When `-no-prettyprint` requests `bibclean` to act as a lexical analyzer, the default line width is unlimited, unless overridden by this option.

When `bibclean` is prettyprinting, line wrapping will be done only at a space. Consequently, an extremely long non-blank character sequence may result in the output exceeding the requested line width. Such sequences are extremely unlikely to occur, at least in English-language text, since even the 45-letter

giant [16, p. 451] *pneumonoultramicroscopicsilicovolcanoconiosis* will fit in `bibclean`'s standard 72-character output line, and so will 58-letter Welsh city names.

When `bibclean` is lexing, line wrapping is done by inserting a backslash-newline pair when the specified maximum is reached, so no line length will ever exceed the maximum.

- [no-]`check-values` With the positive form, apply heuristic pattern matching to field values in order to detect possible errors (e.g. `year = "192"` instead of `year = "1992"`), and issue warnings when unexpected patterns are found.

This checking is usually beneficial, but if it produces too many bogus warnings for a particular bibliography file, you can disable it with the negative form of this option. Default: `yes`.

- [no-]`delete-empty-values` With the positive form, remove all field/value pairs for which the value is an empty string. This is helpful in cleaning up bibliographies generated from text editor templates. Compare this option with -[no-]`remove-OPT-prefixes` described below. Default: `no`.

- [no-]`file-position` With the positive form, give detailed file position information in warning and error messages. Default: `no`.

- [no-]`fix-font-changes` With the positive form, supply an additional brace level around font changes in titles to protect against downcasing by some BIBTEX styles. Font changes that already have more than one level of braces are not modified.

For example, if a title contains the Latin phrase `{\em Dictyostelium Discoideum}` or `\em {D}ictyostelium {D}iscoideum`, then downcasing will incorrectly convert the phrase to lower-case letters. Most BIBTEX users are surprised that bracing the initial letters does not prevent the downcase action. The correct coding is `{}{\em Dictyostelium Discoideum}`. However, there are also legitimate cases where an extra level of bracing wrongly protects from downcasing. Consequently, `bibclean` will normally not supply an extra level of braces, but if you have a bibliography where the extra braces are routinely missing, you can use this option to supply them.

If you think that you need this option, it is strongly recommended that you apply `bibclean` to your bibliography file with and without `-fix-font-changes`, then compare the two output files to ensure that extra braces are not

being supplied in titles where they should not be present. You will have to decide which of the two output files is the better choice, then repair the incorrect title bracing by hand.

Since font changes in titles are uncommon, except for cases of the type which this option is designed to correct, it should do more good than harm. Default: `no`.

- [no-]`fix-initials` With the positive form, insert a space after a period following author initials. Default: `yes`.

- [no-]`fix-names` With the positive form, reorder author and editor name lists to remove commas at brace level zero, placing first names or initials before last names. Default: `yes`.

- [no-]`par-breaks` With the negative form, a paragraph break (either a formfeed, or a line containing only spaces) is not permitted in value strings, or between field/value pairs. This may be useful to quickly trap runaway strings arising from mismatched delimiters. Default: `yes`.

- [no-]`prettyprint` Normally, `bibclean` functions as a prettyprinter. However, with the negative form of this option, it acts as a lexical analyzer instead, producing a stream of lexical tokens. See Section 6 for further details. Default: `yes`.

- [no-]`print-patterns` With the positive form, print the value patterns read from initialization files as they are added to internal tables. Use this option to check newly-added patterns, or to see what patterns are being used.

When `bibclean` is compiled with native pattern-matching code (the default), these patterns are the ones that will be used in checking value strings for valid syntax, and all of them are specified in initialization files, rather than hard-coded into the program. For further details, see Section 5. Default: `no`.

- [no-]`read-init-files` With the negative form, suppress loading of system-, user-, and file-specific initialization files. Initializations will come only from those files explicitly given by `-init-file filename` options. Default: `yes`.

- [no-]`remove-OPT-prefixes` With the positive form, remove the OPT prefix from each field name where the corresponding value is not an empty string. The prefix OPT must be entirely in upper-case to be recognized.

- This option is for bibliographies generated with the help of the GNU Emacs **BIBTEX** editing support, which generates templates with optional fields identified by the **OPT** prefix. Although the function **M-x bibtex-remove-OPT** normally bound to the keystrokes **C-c C-o** does the job, users often forget, with the result that **BIBTEX** does not recognize the field name, and ignores the value string. Compare this option with **-[no-]delete-empty-values** described above. Default: *no*.
- [no-]scribe** With the positive form, accept input syntax conforming to the **SCRIBE** document system. The output will be converted to conform to **BIBTEX** syntax. See Section 8 for further details. Default: *no*.
 - [no-]trace-file-opening** With the positive form, record in the error log file the names of all files which **bibclean** attempts to open. Use this option to identify where initialization files are located. Default: *no*.
 - [no-]warnings** With the positive form, allow all warning messages. The negative form is not recommended since it may mask problems that should be repaired. Default: *yes*.
 - version** Display the program version number on **stderr**. This will also include an indication of who compiled the program, the host name on which it was compiled, the time of compilation, and the type of string-value matching code selected, when that information is available to the compiler.

4 Prettyprinting

A prettyprinter for any language must be able to deal with more than just those files that strictly conform to the language grammar. For programming languages, most compilers implement language extensions that prettyprinters must recognize and try to deal with gracefully. **bibclean** recognizes two such input languages: **BIBTEX** and **SCRIBE**.

Ideally, a prettyprinter should be able to produce output even in the presence of input errors, displaying it in such a way as to make the location of the errors more evident. **bibclean** provides detailed error and warning messages to help pinpoint errors. With the **-file-position** command-line option, it will flag the byte, column, and line, positions of the start and end of the current token in both input and output files.

Here is a summary of the actions taken by **bibclean** on its input stream.

- Space between entries is discarded, and replaced by a single blank line.

- Space around string concatenation operators is standardized.
- Leading and trailing space in value strings is discarded, and embedded multiple spaces are collapsed to a single space.
- String lengths are tested against the limit in standard **BIBTEX**, and warnings issued when the limit is exceeded. The standard limit has proven to be too small in practice, and many sites install enlarged versions of **BIBTEX**. Perhaps **BIBTEX** version 1.0 will use more realistic values, or eliminate string length limits altogether.
- Outer parentheses in entries are standardized to braces.
- Braced value strings are standardized to quoted value strings.
- Field/value pairs are output on separate lines, wrapping long lines to not exceed a user-definable standard width whenever possible.
- A trailing comma is supplied after the last field/value assignment. This is convenient if assignments are later reordered during editing.
- **-fix-font-changes** provides for protecting value string text inside font changes from downcasing.
- Brace-level zero upper-case acronyms in titles are braced to protect from downcasing.
- **-no-par-breaks** provides a way to check for blank lines in string values, which may be indicative of unclosed delimiter errors.
- Umlaut accents, **\"x**, inside value strings at brace-level zero are converted to **{\"x}**. This has been found to be a common user error. **BIBTEX** requires embedded quotes to be nested inside braces.
- Letter-case usage in entry and field names is standardized, so for example, **mastersthesis** and **MASTERSTHESIS** become **MastersThesis**.
- ISBN and ISSN checksums are validated. **BIBTEX** style files that recognize field names for them are available in the TUG bibliography collection, and the bibliography for this document uses them.
- Name modifiers like Jr, Sr, etc., are recognized and handled by **-fix-names**, and names are put into a standard order, so that **Bach, P. D. Q.** becomes **P. D. Q. Bach**.
- With **-fix initials**, uniform spacing is supplied after brace-level zero initials in personal names.

- With **-check-values**, citation key and field values are matched against patterns to catch irregularities and possible errors.
- Dates of the month, like "July 14", are converted to use month abbreviations, jul # " 14".
- Page number ranges are converted to use en-dashes, instead of hyphens or em-dashes.
- With **-check-values**, year numbers are checked against patterns, then if no match is found, the year values are checked against reasonable limits.
- With **-trace-file-opening**, file open attempts are logged. This helps in the diagnosis of problems such as missing files, or incorrect file permissions.
- On lexing or parsing errors, **bibclean** attempts to resynchronize by flushing the input until it finds the next line containing an initial @ character preceded by nothing other than optional white space.
- When an @ character begins a line, a new bibliography entry is assumed to have started. The current brace balance is then tested to make sure it is zero. A non-zero brace level is strongly suggestive of an error, so **bibclean** issues an error message, and zeros the brace level.
- At end-of-file, the brace level is tested. A non-zero brace level is very likely an error, and occasions an error message.

5 Pattern matching and initialization files

bibclean can be compiled with one of three different types of pattern matching; the choice is made by the installer at compile time:

- The original version uses explicit hand-coded tests of value-string syntax.
- The second version uses regular-expression pattern-matching host library routines together with regular-expression patterns that come entirely from initialization files.
- The third version uses special patterns that come entirely from initialization files.

The second and third versions are the ones of most interest here, because they allow the user to control what values are considered acceptable. However, command-line options can also be specified in initialization files, no matter which pattern-matching choice was selected.

When **bibclean** starts, it searches for initialization files, finding the first one in the system executable program search path (on UNIX and IBM

PC-DOS, PATH) and the first one in the **BIBINPUTS** search path, and processes them in turn. Then, when command-line arguments are processed, any additional files specified by **-init-file filename** options are also processed. Finally, immediately before each named bibliography file is processed, an attempt is made to process an initialization file with the same name, but with the extension changed to **.ini**. The default extension can be changed by a setting of the environment variable **BIBCLEANEXT**. This scheme permits system-wide, user-wide, session-wide, and file-specific initialization files to be supported.

When input is taken from **stdin**, there is no file-specific initialization.

For precise control, the **-no-init-files** option suppresses all initialization files except those explicitly named by **-init-file filename** options, either on the command line, or in requested initialization files.

Recursive execution of initialization files with nested **-init-file filename** options is permitted; if the recursion is circular, **bibclean** will finally get a non-fatal initialization file open failure after opening too many files. This terminates further initialization file processing. As the recursion unwinds, the files are all closed, then execution proceeds normally.

An initialization file may contain empty lines, comments from percent to end of line (just like **TEX**), option switches, and field/pattern or field/pattern/message assignments. Leading and trailing spaces are ignored. This is best illustrated by the short example in Table 1. Long logical lines can be

Table 1: Sample **bibclean** initialization file.

```
%% Start with our departmental patterns
-init-file /u/math/bib/.bibcleanrc

%% Make some small additions
chapter = "\"D\""
          %% 23

pages   = "\"D--D\""
          %% 23--27

volume  = "\"D \\an\\d D\""
          %% 11 and 12

year    = \
          "\"ddd, ddd, ddd\""
          "Multiple years specified."
          %% 1989, 1990, 1991

-no-fix-names %% do not modify
               %% author/editor lists
```

split into multiple physical lines by breaking at a backslash-newline pair; the backslash-newline pair is discarded. This processing happens while characters are being read, before any further interpretation of the input stream.

Each logical line must contain a complete option (and its value, if any), or a complete field/pattern pair, or a field/pattern/message triple.

Comments are stripped during the parsing of the field, pattern, and message values. The comment start symbol is not recognized inside quoted strings, so it can be freely used in such strings.

Comments on logical lines that were input as multiple physical lines via the backslash-newline convention must appear on the last physical line; otherwise, the remaining physical lines will become part of the comment.

Pattern strings must be enclosed in quotation marks; within such strings, a backslash starts an escape mechanism that is commonly used in UNIX software. The recognized escape sequences are given in Table 2. Backslash followed by any other character produces just that character. Thus, \" produces a quotation mark, and \\ produces a single backslash.

Table 2: Escape sequences in quoted strings.

\a	alarm bell (octal 007)
\b	backspace (octal 010)
\f	formfeed (octal 014)
\n	newline (octal 012)
\r	carriage return (octal 015)
\t	horizontal tab (octal 011)
\v	vertical tab (octal 013)
\ooo	character number octal ooo (e.g. \012 is linefeed). Up to 3 octal digits may be used.
\0hh	character number hexadecimal hh (e.g. \0x0a is linefeed). hh may be in either letter case. Any number of hexadecimal digits may be used.

An ASCII NUL (\0) in a string will terminate it; this is a feature of the C programming language in which `bibclean` is implemented.

Field/pattern pairs can be separated by arbitrary space, and optionally, either an equals sign or colon functioning as an assignment operator. Thus, the following are equivalent:

```
pages = "\"D--D\""
pages : "\"D--D\""
pages  "\"D--D\""
```

```
pages = "\"D--D\""
pages : "\"D--D\""
pages  "\"D--D\""
```

Each field name can have an arbitrary number of patterns associated with it; however, they must be specified in separate field/pattern assignments.

An empty pattern string causes previously-loaded patterns for that field name to be forgotten. This feature permits an initialization file to completely discard patterns from earlier initialization files.

Patterns for value strings are represented in a tiny special-purpose language that is both convenient and suitable for bibliography value-string syntax checking. While not as powerful as the language of regular-expression patterns, its parsing can be portably implemented in less than 3% of the code in a widely-used regular-expression parser (the GNU `regexp` package).

The patterns are represented by the special characters given in Table 3.

Table 3: Initialization file pattern characters.

	one or more spaces
a	exactly one letter
A	one or more letters
d	exactly one digit
D	one or more digits
r	exactly one Roman numeral
R	one or more Roman numerals (i.e. a Roman number)
w	exactly one word (one or more letters and digits)
W	one or more words, separated by space, beginning and ending with a word
.	one ‘special’ character, one of the characters !#()**,-./:;?[]~, a subset of punctuation characters that are typically used in string values
:	one or more ‘special’ characters
X	one or more ‘special’-separated words, beginning and ending with a word
\x	exactly one x (x is any character), possibly with an escape sequence interpretation given earlier
x	exactly the character x (x is anything but one of these pattern characters: aAdDrRwW.:!~\)

The **X** pattern character is very powerful, but generally inadvisable, since it will match almost anything likely to be found in a **BIBTEX** value string. The reason for providing pattern matching on the value strings is to uncover possible errors, not mask them.

There is no provision for specifying ranges or repetitions of characters, but this can usually be done with separate patterns. It is a good idea to accompany the pattern with a comment showing the kind of thing it is expected to match. Here is a portion of an initialization file giving a few of the patterns used to match **number** value strings:

```
number = "\D\%"      %% 23
number = "\A AD\%"   %% PN LPS5001
number = "\A D(D)\%" %% RJ 34(49)
number = "\A D\%"    %% XNSS 288811
number = "\A D\\.D\%" %% Version 3.20
number = "\A-A-D-D\%" %% UMIAC-TR-89-11
number = "\A-A-D\%"   %% CS-TR-2189
number = "\A-A-D\\.D\%" %% CS-TR-21.7
```

For a bibliography that contains only **Article** entries, this list should probably be reduced to just the first pattern, so that anything other than a digit string fails the pattern-match test. This is easily done by keeping bibliography-specific patterns in a corresponding file with extension **.ini**, since that file is read automatically. You should be sure to use empty pattern strings in this pattern file to discard patterns from earlier initialization files.

The value strings passed to the pattern matcher contain surrounding quotes, so the patterns should also. However, you could use a pattern specification like **"\D"** to match an initial digit string followed by anything else; the omission of the final quotation mark **\"** in the pattern allows the match to succeed without checking that the next character in the value string is a quotation mark.

Because the value strings are intended to be processed by **TEX**, the pattern matching ignores braces, and **TEX** control sequences, together with any space following those control sequences. Spaces around braces are preserved. This convention allows the pattern fragment **A-AD-D** to match the value string **TN-K\slash 27-70**, because the value is implicitly collapsed to **TN-K27-70** during the matching operation.

bibclean's normal action when a string value fails to match any of the corresponding patterns is to issue a warning message similar to this: **Unexpected value in "year = "192"**. In most cases, that is sufficient to alert the user to a problem. In some cases, however, it may be desirable to associate a dif-

ferent message with a particular pattern. This can be done by supplying a message string following the pattern string. Format items **%%** (single percent), **%e** (entry name), **%f** (field name), **%k** (citation key), and **%v** (string value) are available to get current values expanded in the messages. Here is an example:

```
chapter = "\D:D\%" \
"Colon found in '%f = %v'" %% 23:2
```

To be consistent with other messages output by **bibclean**, the message string should not end with punctuation.

If you wish to make the message an error, rather than just a warning, begin it with a query (?), like this:

```
chapter = "\D:D\%" \
"?Colon found in '%f = %v'" %% 23:2
```

The query will *not* be included in the output message.

Escape sequences are supported in message strings, just as they are in pattern strings. You can use this to advantage for fancy things, such as terminal display mode control. If you rewrite the previous example as

```
chapter = "\D:D\%" \
"?\\033[7mColon found \
in '%f = %v'\\033[0m" %% 23:2
```

the error message will appear in inverse video on display screens that support ANSI terminal control sequences. Such practice is not normally recommended, since it may have undesirable effects on some output devices. Nevertheless, you may find it useful for restricted applications.

For some types of bibliography fields, **bibclean** contains special-purpose code to supplement or replace the pattern matching:

- ISBN and ISSN field values are handled this way because their validation requires evaluation of checksums that cannot be expressed by simple patterns; no patterns are even used in these two cases.
- When **bibclean** is compiled with pattern-matching code support, **chapter**, **number**, **pages**, and **volume** values are checked only by pattern matching.
- month values are first checked against the standard **BIBTEX** month name abbreviations, and only if no match is found are patterns then used.
- year values are first checked against patterns, then if no match is found, the year numbers are found and converted to integer values for testing against reasonable bounds.

Values for other fields are checked only against patterns. You can provide patterns for any field you like, even ones `bibclean` does not already know about. New ones are simply added to an internal table that is searched for each string to be validated.

The special field, `key`, represents the bibliographic citation key. It can be given patterns, like any other field. Here is an initialization file pattern assignment that will match an author name, a colon, an alphabetic string, and a two-digit year:

```
key = "A:Add" %% Knuth:TB86
```

Notice that no quotation marks are included in the pattern, because the citation keys are not quoted. You can use such patterns to help enforce uniform naming conventions for citation keys, which is increasingly important as your bibliography data base grows.

6 Lexical analysis

The command-line option `-no-prettyprint` requests `bibclean` to function as a lexical analyzer instead of as a prettyprinter. Its output is then a stream of lines, each of which contains one token. For the bibliography entries shown in Section 1, here is what the output looks like; the long lines have been wrapped by a backslash-newline to fit in these narrow journal columns:

```
# line 1 "stdin"
2     AT      "@"
18    STRING   "String"
11    LBRACE  "{"
1     ABBREV  "pub-AW"
6     EQUALS  "="
# line 2 "stdin"
19    VALUE   "\"Ad{\\"-d}i{\\"-s}on-Wes\\
\ey\""
15    RBRACE  "}"
# line 4 "stdin"
13    NEWLINE "\n"
13    NEWLINE "\n"
2     AT      "@"
5     ENTRY   "Book"
11    LBRACE  "{"
10    KEY     "Lamport:LDP85"
3     COMMA  ","
13    NEWLINE "\n"
# line 5 "stdin"
7     FIELD   "author"
6     EQUALS  "="
19    VALUE   "\"Leslie Lamport\""
3     COMMA  ","
13    NEWLINE "\n"
# line 6 "stdin"
```

```
7     FIELD   "title"
6     EQUALS  "="
# line 8 "stdin"
19    VALUE   "\"{\\"LaTeX}---{A} Document Preparation System---User's Guide and Reference Manual\""
3     COMMA  ","
13    NEWLINE "\n"
# line 9 "stdin"
7     FIELD   "publisher"
6     EQUALS  "="
1     ABBREV  "pub-AW"
3     COMMA  ","
13    NEWLINE "\n"
# line 10 "stdin"
7     FIELD   "year"
6     EQUALS  "="
19    VALUE   "\"1985\""
3     COMMA  ","
13    NEWLINE "\n"
# line 11 "stdin"
7     FIELD   "ISBN"
6     EQUALS  "="
19    VALUE   "\"0-201-15790-X\""
3     COMMA  ","
13    NEWLINE "\n"
# line 12 "stdin"
15    RBRACE  "}"
# line 13 "stdin"
13    NEWLINE "\n"
```

Each line begins with a small integer token type number for the convenience of computer programs, then a token type name for human readers, followed by a quoted token string.

Lines beginning with a sharp, `#`, are ANSI/ISO Standard C preprocessor line-number directives [3, Section 3.8.4] to record the input line number and file name.

There are currently 19 token types defined in the documentation that accompanies `bibclean`. Because `BiTeX` styles can define new field names, there is little point in the lexical analyzer of attempting to classify field names more precisely; that job is left for other software.

Inside quoted strings, the ANSI/ISO Standard C [3, Section 3.1.3.4] backslash escape sequences shown in Table 2 on page 401 are used to encode non-printable characters. In this way, a multi-line string value can be represented on a single line. This is convenient for string-searching applications. If the long output lines prove a problem on some systems, the `-max-width nnn` command-line option can be used to wrap lines at a specified column number by the insertion of a backslash-newline pair.

As a simple example of how this token stream might be processed, the UNIX command pipeline

```
bibclean -no-prettyprint mylib.bib | \
awk '$2 == "KEY" {print $3}' | \
sed -e 's//g' | \
sort
```

will extract a sorted list of all citation keys in the file `mylib.bib`.

As a more complex example, consider locating duplicate abbreviations and citation keys in a large collection of bibliography files. This is a daunting task if it must be done by visual scanning of the files. It took me less than 10 minutes to write and debug a 35-line `nawk` [1] program (15 lines of comments, 20 of code) that processed the token stream from `bibclean` and printed warnings about such duplicates.

The processing steps can be represented by the simple UNIX pipeline

```
bibclean -no-prettyprint bibfiles | \
tr '[A-Z]' '[a-z]' | \
nawk -f bibdup.awk
```

which is most conveniently encapsulated in a command script so that it can be invoked more simply as

```
bibdup *.bib
```

to produce output like this:

```
Duplicate string abbreviation ["pub-aw"]:
    # line 1 "ll.bib"
    # line 141 "master.bib"
Duplicate key ["lamport:ldp85"]:
    # line 4 "ll.bib"
    # line 4172 "master.bib"
...
```

`BIBTeX`'s grammar is somewhat hazy, so it is not easy to perform a lexical analysis without some context sensitivity. `bibclean` therefore produces the lexical token stream merely as an alternate output format. In particular, this means that any requested run-time formatting options will have been applied to the tokens *before* they are output to the lexical token stream. For example, a `SCRIBE` bibliography file can be converted to a `BIBTeX` token stream so that software that processes `bibclean`'s output need not be `SCRIBE`-aware.

7 Portability

`bibclean` is written in ANSI/ISO Standard C [3] with great care taken to produce maximum portability. It has been successfully tested with more than 30 different compilers on all major workstation, and one mainframe, UNIX systems, plus VAX VMS, PC-DOS, OS/2, and Atari TOS.

The C programming language has become the language of choice today for most personal computer and UNIX software development, and the increasing availability of C implementations conforming to the 1989 Standard [3] makes it easier to write code that will compile and run without modification on a wide variety of systems.

C does not have Pascal's problems with character strings and dynamic memory allocation that forced Don Knuth to implement the `WEB` string pool feature and to use compile-time array allocation in the `TEX` software development. C's rich operator syntax, its powerful run-time library, and generally excellent operating-system interfaces have made it widely popular. More than a million copies of the first edition of *The C Programming Language* book [13] have been sold, and the second edition [14] may do even better.

Nevertheless, C has some serious problems. Philippe Kahn, the founder of Borland International, has called C a *write-only* language. Two books have been written about its syntactical peculiarities [9, 17], and one of them has already appeared in a second edition.

The only way to overcome these problems is meticulous care in programming, and experience with as many compilers and computer architectures as possible. Several books offer valuable advice on C portability [10, 11, 19, 23, 24, 26, 29].

C++ [8, 30] is an extension of C to support object-oriented programming, and has an enthusiastic following. ANSI/ISO standardization efforts are in progress, sadly while the language is still evolving.

From the point of view of a C programmer, the advantage of C++ over C is its much stricter checking of type conversions and intermodule interfaces. `bibclean` has been carefully written to be compilable under C++ as well as C, and to date, has been tested with more than a dozen C++ and Objective C (another C superset) compilers.

All of the extra features of the C++ language are strictly avoided, because using them would seriously limit `bibclean`'s portability. Not only is the syntax of the C++ language under evolution, but the C++ class libraries are for the most part *completely dependent* on the particular implementation. Microsoft's 1020-page documentation of its C++ class library is 10% larger than that of its C run-time library.

Nevertheless, I *strongly recommend* use of C++ compilers in preference to C compilers, so as to catch bugs at compile time that would otherwise not be found until post-mortem dump time, or when the code is ported to a new architecture.

8 Scribe bibliography format

The SCRIBE document formatting system [25] greatly influenced LATEX and BIBTEX, as well as the GNU Emacs TEXinfo system.

With care, it is possible to share bibliography files between SCRIBE and BIBTEX. Nevertheless, there are some differences, so here is a summary of features of the SCRIBE bibliography file format. We record them because they are difficult to determine from the published manual, and because readers may sometimes acquire files in this format without having prior exposure to SCRIBE.

1. Letter case is not significant in field names and entry names, but case is preserved in value strings.
2. In field/value pairs, the field and value may be separated by one of three characters: =, /, or \sqcup (space). Space may optionally surround these separators.
3. Value delimiters are any of these seven pairs: { }, [], (), < >, ' ', " ", and ` '.
4. Value delimiters may not be nested, even though with the first four delimiter pairs, nested balanced delimiters would be unambiguous.
5. Delimiters can be omitted around values that contain only letters, digits, sharp (#), ampersand (&), period (.), and percent (%).
6. Outside of delimited values, a literal at-sign (@) is represented by doubled at-signs @@.
7. Bibliography entries begin with @name, as for BIBTEX, but any of the seven SCRIBE value delimiter pairs may be used to surround the values in field/value pairs. As in (4), nested delimiters are forbidden.
8. Arbitrary space may separate entry names from the following delimiters.
9. @Comment is a special command whose delimited value is discarded. As in (4), nested delimiters are forbidden.
10. The special form

```
@Begin{comment}
...
@End{comment}
```

permits encapsulating arbitrary text containing any characters or delimiters, other than @End{comment}. Any of the seven delimiter pairs may be used around the word comment following the @Begin or @End; the delimiters in the two cases need not be the same, and consequently, @Begin{comment}/@End{comment} pairs may not be nested.

11. The key field is required in each bibliography entry.
12. A backslashed quote in a string will be assumed to be a TEX accent, and braced appropriately. While such accents do not conform to SCRIBE syntax, SCRIBE-format bibliographies have been found that appear to be intended for TEX processing.

Because of this loose syntax, bibclean's normal error detection heuristics are less effective, and consequently, SCRIBE mode input is not the default; it must be explicitly requested.

9 Recommendations for BibTEX design

The documentation available for BIBTEX leaves several points about the input syntax unclear, and I had to obtain answers to the following questions by experiment:

- Can an at-sign occur inside a @Comment{...}? *No.*
- Can string abbreviation names be used on the right-hand side of string definitions? *Yes.*
- Can the argument of @String be empty? *No.*
- Can a citation key be omitted in an entry? *No.*
- Can the list of assignments in an entry be empty? *Yes.*
- Can a @Comment{...} occur between arbitrary tokens? *No.*
- Are newlines preserved in the argument of a @Preamble{...}? The answer is relevant if the user includes TEX comments in the preamble material. *No.*

I view the experimental answers to these questions as pure happenstance, and could reasonably argue for the opposite answers to the ones obtained.

Grammar

The most important recommendation that I can make for the next version of BIBTEX is that it *must* have a rigorous grammar, including a well-defined comment syntax.

The grammar can almost be of the simple class LL(0) [2], requiring no lookahead during parsing, and one-character lookahead during lexical analysis. However, the presence of the string concatenation operator complicates things sufficiently to require at least an LL(1) grammar.

Such grammars are straightforward to handle with either hand-coded parsers, or with parsers automatically generated from grammar files by compiler development tools like the UNIX lex [20] and yacc [12, 21, 22, 28] programs, or the Free Software Foundation equivalents, flex and bison.

`yacc` and `bison` implement LALR(1) parsers; the acronym stands for “Look-Ahead at most 1 token with a Left-to-Right derivation”. These are simpler than the LR(k) grammars introduced by none other than the author of `TeX` in the fundamental paper on the theory of parsing [15]. Nevertheless, they are sufficient for a broad class of language grammars, including most major programming languages, and importantly, they produce compact, efficient, fast, and reliable parsers. LL(1) grammars are a special case of LALR(1) grammars, and we will later define a `BIBTeX` grammar in LALR(1) form in Section 11.

Comment syntax

The comment syntax should preferably be identical to that of `TeX`, so that a comment runs from percent to end-of-line, and then *additionally gobbles all leading horizontal space on the next line, up to, but not including, its end-of-line*. This permits breaking of long lines without having to destroy indentation that is so necessary for readability. Percent-initiated comments are already supported in `BIBTeX` style files, though such comments end after the first following newline.

For `SCRIBE` compatibility, `BIBTeX` should also support a `@Comment{...}` entry type. This will require additions to *all* `BIBTeX` style files, since the entry types are known there, and not in the `BIBTeX` code itself. `BIBTeX` 0.99c already knows about `@Comment{...}`, but the `WEB` code section “Process a comment command” will have to be extended to deal with the grammar changes.

It is important that `BIBTeX` not discard `@Comment{...}` entries, because it would then not be possible to write a `BIBTeX` style file that converted a bibliography file to another format without loss of information. One such style already exists to convert `BIBTeX` files to UNIX `bib/refer` format.

Characters in names

The characters that can appear in key, entry, and field names *must* be defined by enumeration, rather than by exclusion, as is currently done [18, Section B.1.3]. The reason is that character sets vary between computers, and the new, and very much larger, ISO10646M character set may be widely available in this decade. These variations make the set of admissible name characters vary between systems, compromising portability. I strongly recommend following the conventions for identifiers in widely-used programming languages to define the grammar of key, entry, and field names. It seems to me that letters, digits, colon, hyphen, and possibly plus and slash, should be adequate, and names

should be required to begin with a letter. ‘Letter’ here should include *only* the 26 Roman letters ‘A’ through ‘Z’, because allowing letters from other alphabets opens a horrid can of worms that will seriously impact portability of bibliography files until the computer world has a single uniform character set.

I tested this set of characters against 92 500 entries in local bibliography files, and found only a few keys that used other characters: the new ones were period and apostrophe (e.g. O’Malley:TB92). They might therefore be permitted as well, though I would prefer to omit them, and retrofit changes in a few citation keys.

The characters permitted in citation keys should be the same as those in entry and field names, so as to avoid user confusion.

Error reporting

When `BIBTeX` begins to collect a token, it should record the current line number. When an unclosed string later causes internal buffer overflow, it could report something like `String buffer overflow on input lines 24--82` that would better help locate the offending string by giving its starting and ending line numbers.

To simplify error recovery in such cases, `BIBTeX` could additionally require that the `@` character that starts a new entry must be the first non-space character on a line.

File inclusion

`BIBTeX` sorely needs a file inclusion facility. With `BIBTeX` 0.99c, this feature is available in a crude fashion by listing several files in the `\bibliography` command. However, this is not sufficiently general, and requires unnecessary knowledge on the part of the user of the bibliography.

The author of a `BIBTeX` file should be free to restructure it into subfiles without requiring modifications to all documents that use it. File inclusion is important to allow sharing of common material, such as `@String{...}` definitions.

`SCRIBE` uses the form

`@Include{filename}`

and `BIBTeX` should too. It must be possible to nest file inclusions to a reasonable depth, at least five levels.

10 A lexical grammar for `BibTeX`

To test the recommendations of Section 9, I wrote and tested a `lex` grammar for `BIBTeX`. It took just 22 rules to identify the 19 basic token types. The

complete **lex** file was about 510 lines long, with about 340 lines of C code mostly concerned with the input and output of strings, and 120 lines of function and variable declarations. After **lex** processing, the complete C program was about 1130 lines long; with **flex**, it is 1700 lines long. This program is named **biblex**, and its output is compatible with that of **bibclean** with the **-no-prettyprint** option. However, it offers none of **bibclean**'s other services.

The **lex** grammar is presented in this section in the style of literate programming, with grammar rules interspersed with descriptive text. The index at the end of this document provides an essential feature of a literate program. To my knowledge, no WEB facility yet exists for **lex** and **yacc**, so this literate program must be handcrafted.

File structure

A **lex** file has this general structure:

```
definitions
%
rules
%
user functions
```

C declarations and definitions can be included in the definitions part if they are enclosed in %{ and %}. Such text is copied verbatim to the output code file, together with additional **lex**-supplied header code.

Running **lex** on this file produces a C file that can be compiled and linked with a main program from the **lex** library to produce a working lexical analyzer. Alternatively, the user can write a customized main program which is linked with the **lex**-generated code to make a functional lexer.

In the following subsections, we describe the contents of the definitions and rules parts, but omit the user functions, since they are not relevant to understanding the grammar.

Macro definitions

The **lex** grammar begins with macro definitions. **lex** macros are single letters followed by a regular expression that defines them.

In regular expressions, square brackets delimit sets of characters, hyphen is used for character ranges inside sets, asterisk means zero or more of the preceding pattern, and plus means one or more. A period represents any character other than a newline.

lex macro names are braced to request expansion when they are used in grammar rules.

The first macro, N, represents the set of characters permitted in **BIBTEX** names of abbreviations, citation keys, entries, and fields. If this set is ever modified, this is the *only* place where that job has to be done.

N [A-Za-z] [--A-Za-z0-9:.+/*]*

It is not reasonable to make this set differ for these four different uses, because the differences are insufficient to distinguish between them lexically. We'll see later that we have to examine surrounding context to tell them apart.

Macro O represents the set of open delimiters that start a **BIBTEX** entry argument. We could extend this grammar for **SCRIBE** by adding additional characters to the set.

O [({]

Macro W represents a single horizontal space character.

W [\f\r\t\013]

Notice that we include formfeed, \f, and vertical tab, \v, in the set of horizontal space characters, even though they produce vertical motion on an output device. The reason is that we want to treat them just like blanks, and distinguish them from newlines, which are handled separately. **lex** does not recognize the escape sequence \v, so we have to reencode it in octal as \013.

Carriage return, \r, is not normally used in UNIX text files, but is common in some other operating systems. On the Apple Macintosh, carriage return is used instead of newline as an end-of-line marker. Fortunately, this will be transparent to us, because the C language requires [3, Section 2.2.2] that the implementation map host line terminators to newline on input, and newline back to host line terminators on output, so we will never see carriage returns on that system.

The last macro, S, represents optional horizontal space.

S {W}*

Format of grammar rules

The remainder of the grammar consists of pairs of regular expression patterns and C code to execute when the pattern is matched. **lex** uses a "maximal munch" strategy in matching the longest possible sequence to handle the case where two rules have common leading patterns.

In the grammar file, the pairs are each written on a single line, but we wrap lines here to fit in the narrow journal columns, with the backslash-newline convention used earlier.

@ token

The first grammar rule says that an @ character should be recognized as the token named TOKEN_AT.

```
[@]      RETURN (out_token(TOKEN_AT));
```

On a successful match, the output function optionally emits the token, then returns its argument as a function value which the lexer in turn returns to the parser.

The C return statement is hidden inside the RETURN macro, because for yacc and bison, we need to bias bibclean's small integer token values to move them beyond the range of character ordinals.

Comment, Include, Preamble, and String tokens

The next four rules ignore letter case in matching the words Comment, Include, Preamble, or String. If they follow an @ character, they are identified as special tokens; otherwise, they are regarded as string abbreviation names.

```
[Cc] [Oo] [Mm] [Mm] [Ee] [Nn] [Tt] \
    RETURN ((last_token == TOKEN_AT) ?
            out_token(TOKEN_COMMENT) :
            out_token(TOKEN_ABBREV));
```

```
[Ii] [Nn] [Cc] [Ll] [Uu] [Dd] [Ee]/{S}{O} \
    RETURN ((last_token == TOKEN_AT) ?
            out_token(TOKEN_INCLUDE) :
            out_token(TOKEN_ABBREV));
```

```
[Pp] [Rr] [Ee] [Aa] [Mm] [Bb] [Ll] [Ee]/{S}{O} \
    RETURN ((last_token == TOKEN_AT) ?
            out_token(TOKEN_PREAMBLE) :
            out_token(TOKEN_ABBREV));
```

```
[Ss] [Tt] [Rr] [Ii] [Nn] [Gg]/{S}{O} \
    RETURN ((last_token == TOKEN_AT) ?
            out_token(TOKEN_STRING) :
            out_token(TOKEN_ABBREV));
```

Although lex supports examination of trailing context in order to identify tokens more precisely, the presence of arbitrary whitespace and in-line comments in this grammar makes it impossible to use this feature. The output routines remember the last non-space, non-comment token seen in order to make use of leading context to assist in token identification.

Abbreviation, entry, field, and key tokens

Several token types are recognized by a match with the name macro, N. Since the same set of characters can occur in abbreviations, entry names, field names, and key names, we have to use the record of

leading context to distinguish between the various possibilities.

```
{N} {
    if (last_object == TOKEN_STRING)
        RETURN(out_token(TOKEN_ABBREV));
    switch (last_token)
    {
        case TOKEN_COMMA:
            RETURN(out_token(TOKEN_FIELD));
        case TOKEN_LBRACE:
            RETURN(out_token(TOKEN_KEY));
        case TOKEN_AT:
            RETURN(out_token(TOKEN_ENTRY));
        default:
            RETURN(out_token(TOKEN_ABBREV));
    }
}
```

In the event of errors in the input stream, this identification of token types may be unreliable; such errors will be detected later in the parsing program.

Digit string

A *digit string* is an undelimited value string. The output function will supply the missing quotation mark delimiters, so that all strings take a standard form.

```
[0-9]+   RETURN \
    (out_protected_string( TOKEN_VALUE));
```

In-line comment token

A percent initiates an *in-line comment* that continues to the end of line and then over all leading horizontal space on the next line.

```
[%].*[\n]{S} \
    RETURN (out_token(TOKEN_INLINE));
```

Because this pattern marks the start of a new token, the previous token has already been terminated. Thus, an line-line comment *cannot* split a token. The same is true for TeX macros, though not for ordinary TeX text.

String concatenation token

A sharp sign is the BIBTeX *string concatenation operator*.

```
[#]      RETURN (out_token(TOKEN_SHARP));
```

Delimited string token

A quotation mark initiates a *delimited string*.

```
["]      RETURN (out_string());
```

The complete string must be collected by the C function out_string() because regular expressions cannot count balanced delimiters.

BIBTEX's quoted string syntax is a little unusual, in that an embedded quote is not represented by double quotes, as in Fortran, or by an escape sequence, as in C, but rather by putting the quote character in braces.

Brace tokens

Left and right *braces* are recognized as single tokens.

```
[{}]    RETURN (out_lbrace());
[{}]    RETURN (out_rbrace());
```

The output functions keep track of the current brace level to distinguish between outer braces delimiting a **BIBTEX** entry, and inner braces delimiting a string value, and return TOKEN_LBRACE, TOKEN_LITERAL, TOKEN_RBRACE, or TOKEN_STRING, depending on preceding context.

TOKEN_LITERAL is used for the argument of **Comment** and **Include** entries, and contains the delimiting braces.

Parenthesis tokens

In order to simplify the parser grammar, we remap outer *parentheses* delimiting arguments of **BIBTEX** entries to *braces*. However, if the parentheses are not preceded by a valid entry name, they are output instead as single-character tokens of type TOKEN_LITERAL. They cannot legally occur in this context, but that error will be detected during the parsing stage. During lexical analysis, we do not want to have any error conditions.

```
[()]   RETURN (out_lparen());
[()]   RETURN (out_rparen());
```

To support **SCRIBE**, we would need to add patterns for other delimiters here.

Assignment and separator tokens

The *assignment operator* and *assignment separator* are returned as single tokens.

```
[=]    RETURN (out_token(TOKEN_EQUALS));
[,]    RETURN (out_token(TOKEN_COMMA));
```

Newline token

A *newline* is returned as a separate token because we want to be able to preserve line boundaries so that filter tools that make minimal perturbations on the input stream can be constructed.

```
[\\n]   RETURN (out_token(TOKEN_NEWLINE));
```

Horizontal space token

Consecutive horizontal space characters are returned as a single space token, for the same reason that newlines are recognized as distinct tokens by the preceding rule.

```
{W}+    RETURN (out_token(TOKEN_SPACE));
```

Unclassifiable tokens

Finally, we have a catch-all rule: any character not recognized by one of the preceding rules is returned as a literal single-character token, and will cause an error during the parsing. The regular-expression character period matches anything but a newline, and we already have a rule for newline.

```
.        RETURN (out_token(TOKEN_LITERAL));
```

Lexical grammar summary

We now have a complete lexical grammar suitable for **lex** that can complete tokenize an arbitrary input stream containing any character values whatever.

The associated C code functions normalize entries by changing outer parentheses to braces, brace string delimiters to quotes, and undelimited digit strings to quoted strings.

All string tokens of type TOKEN_VALUE output by the lexer will contain surrounding quotes, and any nested quotes will be braced, with proper care taken to handle \" accent control sequences properly.

All special characters inside the quoted strings will be represented by the escape sequences given in Table 2 on page 401. Thus, even with a binary input stream, the output of the lexer will contain only printable characters.

It must be observed that **lex** is not capable of handling all 256 8-bit characters. In particular, it treats an ASCII NUL (\0) in a string as an end-of-file condition. Older versions of **lex** are not 8-bit clean; they will not reliably handle characters 128-255. This latter deficiency is being remedied by the X/Open Consortium activities to internationalize and standard UNIX applications [32].

11 A parsing grammar for **BibTEX**

To complete the job, I wrote a yacc grammar for **BIBTEX**. This was considerably more work than the **lex** grammar, mostly due to my relative inexperience with writing LALR(1) grammars, and it took several days to understand the process well enough to eliminate the grammatical ambiguities that initially plagued me.

The final complete yacc program is about 270 lines long, and produces a parser of 760 (yacc) to 1000 (bison) lines, excluding the lexer. The grammar contains just 35 rules. Ten of these rules could be eliminated if we arranged for the lexer to discard space and in-line comments, but for a pretty-printer and other BIBTEX tools, they must be preserved. This parsing program is called **biparse**; it can be used with the output of either **bibclean -no-prettyprint**, or **biblex**.

The complete BIBTEX grammar is given below, expressed as yacc rules, again in literate programming style. It must be augmented by about 180 lines of C code to provide a working parser.

File structure

A yacc file has this general structure:

declarations

%

rules

%

user functions

C declarations and definitions can be included in the declarations part if they are enclosed in **{** and **}**. Such text is copied verbatim to the output code file, together with additional yacc-supplied header code.

Running yacc on this file produces a C file that can be compiled and linked with the lexical analyzer code to produce a working parser.

In the following subsections, we describe the contents of the declarations and rules parts, but omit the declaration C code and the user functions, since they are not relevant to understanding the grammar.

Format of grammar rules

The grammar rules will be presented in top-down order, from most general, to most particular, since this seems to be the best way to understand the overall structure of the grammar, and to ensure that it describes current BIBTEX usage, plus our suggested extensions and clarifications.

The colon in a grammar rule should be read “is” or “produces”, because the rule is also known as a *production*. A vertical bar separates alternatives, and can be read “or”. A semicolon terminates the rule.

Lower-case letters are used for *non-terminals*, which are names of rules in the parser grammar. Upper-case letters are used for *terminals*, which are names of tokens recognized by the lexer.

The spacing shown is arbitrary, but conventional for yacc grammars: each rule starts a new line, with the right-hand side indented from the margin, and the semicolon occupies a separate line.

Token declarations

The %token declarations merely provide symbolic names for the integer token types returned by the lexer. The values are arbitrary, except that they must exceed 257, and must agree with the definitions in the lexer code. We simply increment the token types output from **bibclean** by 1000, matching the offset added in the RETURN macro in the lexer.

%token TOKEN_ABBREV	1001
%token TOKEN_AT	1002
%token TOKEN_COMMA	1003
%token TOKEN_COMMENT	1004
%token TOKEN_ENTRY	1005
%token TOKEN_EQUALS	1006
%token TOKEN_FIELD	1007
%token TOKEN_INCLUDE	1008
%token TOKEN_INLINE	1009
%token TOKEN_KEY	1010
%token TOKEN_LBRACE	1011
%token TOKEN_LITERAL	1012
%token TOKEN_NEWLINE	1013
%token TOKEN_PREAMBLE	1014
%token TOKEN_RBRACE	1015
%token TOKEN_SHARP	1016
%token TOKEN_SPACE	1017
%token TOKEN_STRING	1018
%token TOKEN_VALUE	1019

Precedence declarations

The %nonassoc declaration makes the assignment operator non-associative, so input of the form **a = b = c** is illegal.

%nonassoc TOKEN_EQUALS

The first %left declaration makes space, in-line comment, and newline tokens left associative, and of equal precedence.

**%left TOKEN_SPACE TOKEN_INLINE **
TOKEN_NEWLINE

The second %left declaration makes the BIBTEX string concatenation character, #, left associative, and of higher precedence than space, in-line comment, and newline.

%left TOKEN_SHARP

These precedence settings are crucial for resolving conflicts in this grammar which arise in assignments when the parser has seen an assignment operator and a value. Without the operator precedences, it cannot decide whether to complete the assignment, or to read ahead looking for a concatenation operator.

BibTEX file

The beginning of the grammar rules is indicated by a pair of percent characters.

%%

The first rule defines what we are going to parse, namely, a *BibTEX file*. The left-hand side of the first rule is known as the grammar's *start symbol*.

```
bibtex_file:
    opt_space
    | opt_space object_list opt_space
    ;
```

This rule says that a BibTEX file contains either optional space, or optional space followed by a list of objects followed by optional space. This definition permits a file to be empty, or contain only space tokens, or have leading and trailing space.

Object lists

A *list of objects* is either a single object, or a list of such objects, separated by optional space from another object.

```
object_list:
    object
    | object_list opt_space object
    ;
```

For LL(1) parsers, usually implemented by hand-coded recursive descent programs, this kind of left-recursive rule must be rewritten by standard methods [2, pp. 47–48, 176–178] to avoid an infinite loop in the parser. In this rule, we would instead define a list as an object, separated by optional space from another list. However, for LALR(1) parsers, left-recursive definitions are preferable, because they avoid parser stack overflow with long lists.

Objects

An *object* is one of the BibTEX `@name{...}` constructs. Notice that we allow optional space between the `@` and the `name`.

```
object:
    TOKEN_AT opt_space at_object
    ;
```

In this grammar, we will consistently allow optional space between *any* pair of BibTEX tokens; space is described more precisely below. This convention is easy to remember, and easy to implement in the grammar rules.

While it would be possible to include the `@` as part of the `name`, making `@name` a single lexical token, both BibTEX and SCRIBE permit intervening space, so we cannot collapse the two into a single token.

Entry types and error recovery

Here are the possibilities for the `name` following an `@`, which we call an `at_object`.

`at_object:`

```
comment
| entry
| include
| preamble
| string
| error TOKEN_RBRACE
;
```

`Comment`, `Include`, `Preamble`, and `String` must be handled separately from other types of entries, like `Article` and `Book`, because their braced arguments have a different syntax.

The rule with `error` is a special one supported by yacc and bison. It says that if an `at_object` cannot be recognized at the current state of the parse, then the input should be discarded until a right brace is found. An error message will be issued when this happens, but recovery will be attempted following that right brace. Without this error handling, any input error will immediately terminate the parser, hardly a user-friendly thing to do.

This is the only place where we will attempt error repair, although we could certainly do so in other rules, such as in the assignment rule below. The goal here is to present a rigorous complete grammar, without additional embellishments that would complicate understanding.

Comment entry

A BibTEX `@Comment{...}` is special in that the only requirement on the argument is that delimiters be balanced. The lexer returns the delimited argument as a single literal string, including the delimiters, and standardizes the delimiters to braces.

`comment:`

```
TOKEN_COMMENT opt_space
TOKEN_LITERAL
;
```

Bibliography entry

A *BIBTEX bibliography entry* is braced text containing a citation key, a comma, and a list of assignments. The rules provide for an optional assignment list, and for an optional trailing comma. To shorten the rules, we introduce a subsidiary rule, *entry_head*, to represent their common prefix.

```

entry: entry_head
      assignment_list
      TOKEN_RBRACE
| entry_head
      assignment_list
      TOKEN_COMM opt_space
      TOKEN_RBRACE
| entry_head TOKEN_RBRACE
;

entry_head:
      TOKEN_ENTRY opt_space
      TOKEN_LBRACE opt_space
      key_name opt_space
      TOKEN_COMM opt_space
;
;
```

There is no *opt_space* item following *assignment_list* because it is included in the definition of the latter. This infelicity seems to be necessary to obtain a grammar that conforms to the LALR(1) requirements of yacc and bison.

Key name

Because of intervening newlines and in-line comments, the lexical analyzer cannot always correctly recognize a *citation key* from trailing context. It might instead erroneously identify the token as an abbreviation. We therefore need to account for both possibilities:

```

key_name:
      TOKEN_KEY
| TOKEN_ABBREV
;
```

Include entry

The *Include entry* is followed by a file name enclosed in balanced braces.

```

include:
      TOKEN_INCLUDE opt_space
      TOKEN_LITERAL
;
```

Because file names are operating-system dependent, the only restrictions that are placed on the file name are that it cannot contain unbalanced braces, and that it cannot contain leading or trailing space.

However, the file name can have embedded space if the operating system permits.

BIBTEX should discard the delimiting braces and surrounding space in the *TOKEN_LITERAL* to isolate the file name. It should search for this file in its standard input path, so that the file name need not contain an absolute directory path. This feature is not supported in *BIBTEX* 0.99c, but *bibclean* and the lexer and parser recognize it in anticipation of its eventual incorporation.

Preamble entry

The *Preamble entry* argument is a braced *BIBTEX* string value. *BIBTEX* outputs the argument verbatim, minus the outer delimiters, to the *.bb1* file for *TEX* to process.

```

preamble:
      TOKEN_PREAMBLE opt_space
      TOKEN_LBRACE opt_space
      value opt_space
      TOKEN_RBRACE
;
```

String entry

The *String entry* argument is a braced single assignment.

```

string:
      TOKEN_STRING opt_space
      TOKEN_LBRACE opt_space
      assignment opt_space
      TOKEN_RBRACE
;
```

Value string

A *BIBTEX value* is a string, which may be a simple value, or a list of strings separated by the string concatenation operator.

```

value: simple_value
| value opt_space
      TOKEN_SHARP opt_space
      simple_value
;
```

Simple values

A *simple value* is either a delimited string, returned by the lexer as a *TOKEN_VALUE*, or a string abbreviation, returned as a *TOKEN_ABBREV*.

```

simple_value:
      TOKEN_VALUE
| TOKEN_ABBREV
;
```

The lexer can distinguish between these two because of the string delimiters. It is up to the parser support code to verify that an abbreviation is actually defined before it is used.

Assignment list

The body of most BIBTEX entries consists of a list of one or more assignments, separated by commas. Notice that this definition does not provide for an optional trailing comma after the last assignment. We handled that above in the rules for `entry`.

```
assignment_list:
    assignment
  | assignment_list
    TOKEN_COMMA opt_space
    assignment
;
```

Assignment

An `assignment` has a left-hand side separated from a value by the assignment operator, `=`.

```
assignment:
    assignment_lhs opt_space
    TOKEN_EQUALS opt_space value
    opt_space
;
```

Trailing optional space is included here, and omitted before the comma in `assignment_list`, in order to allow the LALR(1) parser to successfully distinguish between space between a value and a comma, and space between a value and a string concatenation operator.

My initial version of this grammar did not have this optional space item, and the resulting parser proved unable to recognize input in which a space separated a value from a comma or closing brace; it took quite a bit of experimentation to determine how to rewrite the grammar to remove this problem.

The left-hand side of an assignment is either a field name, like `author` or `title`, or a string abbreviation name. The lexer must distinguish between the two by remembering the last entry type seen, because they are made up of exactly the same set of possible characters.

```
assignment_lhs:
    TOKEN_FIELD
  | TOKEN_ABBREV
;
```

Optional space

Optional space is either an empty string, here indicated by the `/*...*/` comment, or space.

```
opt_space:
    /* empty */
  | space
;
```

Space

`Space` is an important part of the grammar. It is one or more single spaces.

```
space: single_space
    | space single_space
;
```

We include space handling to support tools that process BIBTEX files and wish to preserve the input form. In normal compiler design, space is recognized by the lexer, and discarded, so the parser never has to deal with it, and the grammar can be considerably simpler.

Single space

The final rule of the grammar defines a *single space* as a literal space character, or an in-line comment, or a literal newline character.

```
single_space:
    TOKEN_SPACE
  | TOKEN_INLINE
  | TOKEN_NEWLINE
;
```

Although we could arrange for the lexer to merge `TOKEN_SPACE` and `TOKEN_NEWLINE` into a single token, this would interfere with heuristics used by a prettyprinter to detect empty lines inside string values, which are possibly indicative of missing delimiters.

Parsing grammar summary

We have now completed a yacc grammar for BIBTEX that provides a rigorous grammatical analysis of a stream of tokens recognized by the lexers in Sections 6 and 10.

Notice that there is no character-string processing whatever in the parser, because it has all been done in the lexer. Parsing operations just manipulate small integer values.

In this version, no actions have been supplied as C code fragments in the yacc grammar. The only output of the parser will be the token stream from the lexer, interspersed by error messages when the input fails to match a grammar rule.

Error recovery has been kept simple: input is flushed to the next closing brace, which is presumably the end of an entry. Braces of type `TOKEN_LBRACE` and `TOKEN_RBRACE` do not occur except around apparent entries in the lexer output;

other braces are returned as tokens of type TOKEN_LITERAL.

No more than one token of lookahead is required by this grammar, although the lexer often looked several characters ahead to examine trailing context in order to distinguish between otherwise similar tokens.

BIBTEX users should be able to read this grammar and decide whether a questionable **BIBTEX** construct is legal or not, without having to resort to software experiments as I did to clarify fuzzy grammatical points.

12 Software availability

The source code and documentation for **bibclean** are in the *public domain*, in the interests of the widest availability and greatest benefit to the **TeX** community. Commercial vendors of **TeXware** are encouraged to include **bibclean** with their distributions.

The distribution also includes the separate complete lexer and parser grammar and code, which can be processed by **lex** or **flex**, and **yacc** or **bison**, respectively. The output C code from these tools is included so that recipients need not have them installed to actually compile and run the lexer and parser.

If you have Internet anonymous **ftp** access, you can retrieve the distribution in a variety of archive formats from the machine **ftp.math.utah.edu** in the directory **pub/tex/bib**. Major **TeX** Internet archive hosts around the world will also have **bibclean**, but the author's site will always have the most up-to-date version. If you lack **ftp** capability but have electronic mail access, a message to **tuglib@math.utah.edu** with the text

```
help
send index from tex/bib
```

will get you started.

The **bibclean** distribution includes a substantial collection of torture tests that should be run at installation time to verify correctness. As with the **TeX** **trip** and **METAFONT** **trap** tests, this testing has proved valuable in uncovering problems before the code is installed for general use.

References

- [1] Alfred V. Aho, Brian W. Kernighan, and Peter J. Weinberger. *The AWK Programming Language*. Addison-Wesley, Reading, MA, USA, 1988. ISBN 0-201-07981-X.
- [2] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers—Principles, Techniques, and Tools*. Addison-Wesley, Reading, MA, USA, 1986. ISBN 0-201-10088-6.
- [3] American National Standards Institute, 1430 Broadway, New York, NY 10018, USA. *American National Standard Programming Language C, ANSI X3.159-1989*, December 14, 1989.
- [4] Nelson H. F. Beebe. Publications about **TeX** and typography. *TUGBoat*, 12(2):176–183, May 1991.
- [5] Nelson H. F. Beebe. Publications prepared with **TeX**. *TUGBoat*, 12(12):183–194, May 1991. *TUGBoat*, 12(2):183–194, May 1991.
- [6] Martin Bryan. *SGML—An Author's Guide to the Standard Generalized Markup Language*. Addison-Wesley, Reading, MA, USA, 1988. ISBN 0-201-17535-5.
- [7] Debra Cameron and Bill Rosenblatt. *Learning GNU Emacs*. O'Reilly & Associates, Inc., 981 Chestnut Street, Newton, MA 02164, USA, 1991. ISBN 0-937175-84-6.
- [8] Margaret A. Ellis and Bjarne Stroustrup. *The Annotated C++ Reference Manual*. Addison-Wesley, Reading, MA, USA, 1990. ISBN 0-201-51459-1.
- [9] Alan R. Feuer. *The C Puzzle Book*. Prentice-Hall, Englewood Cliffs, NJ 07632, USA, second edition, 1989. ISBN 0-13-115502-4.
- [10] Samuel P. Harbison and Guy L. Steele Jr. *C—A Reference Manual*. Prentice-Hall, Englewood Cliffs, NJ 07632, USA, third edition, 1991. ISBN 0-13-110933-2.
- [11] Rex Jaeschke. *Portability and the C Language*. Hayden Books, 4300 West 62nd Street, Indianapolis, IN 46268, USA, 1989. ISBN 0-672-48428-5.
- [12] Steven C. Johnson. Yacc: Yet another compiler compiler. In *UNIX Programmer's Manual*, volume 2, pages 353–387. Holt, Reinhart, and Winston, New York, NY, USA, 1979. AT&T Bell Laboratories Technical Report, July 31, 1978.
- [13] Brian W. Kernighan and Dennis M. Ritchie. *The C Programming Language*. Prentice-Hall, Englewood Cliffs, NJ 07632, USA, 1978. ISBN 0-13-110163-3.
- [14] Brian W. Kernighan and Dennis M. Ritchie. *The C Programming Language*. Prentice-Hall, Englewood Cliffs, NJ 07632, USA, second edition, 1988. ISBN 0-13-110362-8.
- [15] Donald E. Knuth. On the translation of languages from left to right. *Information and Control*, 8(6):607–639, 1965. This is the original paper on the theory of LR(k) parsing.

- [16] Donald E. Knuth. *The TeXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986. ISBN 0-201-13447-0.
- [17] Andrew Koenig. *C Traps and Pitfalls*. Addison-Wesley, Reading, MA, USA, 1989. ISBN 0-201-17928-8.
- [18] Leslie Lamport. *LATEX—A Document Preparation System—User’s Guide and Reference Manual*. Addison-Wesley, Reading, MA, USA, 1985. ISBN 0-201-15790-X.
- [19] J. E. Lapin. *Portable C and UNIX Programming*. Prentice-Hall, Englewood Cliffs, NJ 07632, USA, 1987. ISBN 0-13-686494-5.
- [20] Michael E. Lesk and Eric Schmidt. Lex—a lexical analyzer generator. In *UNIX Programmer’s Manual*, volume 2, pages 388–400. Holt, Reinhart, and Winston, New York, NY, USA, 1979. AT&T Bell Laboratories Technical Report in 1975.
- [21] John R. Levine, Tony Mason, and Doug Brown. *lex & yacc*. O’Reilly & Associates, Inc., 981 Chestnut Street, Newton, MA 02164, USA, second edition, 1992. ISBN 1-56592-000-7.
- [22] Tony Mason and Doug Brown. *lex & yacc*. O’Reilly & Associates, Inc., 981 Chestnut Street, Newton, MA 02164, USA, 1990. ISBN 0-937175-49-8.
- [23] P. J. Plauger. *The Standard C Library*. Prentice-Hall, Englewood Cliffs, NJ 07632, USA, 1992. ISBN 0-13-838012-0.
- [24] Henry Rabinowitz and Chaim Schaap. *Portable C*. Prentice-Hall, Englewood Cliffs, NJ 07632, USA, 1990. ISBN 0-13-685967-4.
- [25] Brian Reid. *Scribe User’s Manual*. Carnegie-Mellon University, Pittsburgh, PA, USA, third edition, 1980.
- [26] Marc J. Rochkind. *Advanced UNIX Programming*. Prentice-Hall, Englewood Cliffs, NJ 07632, USA, 1985. ISBN 0-13-011818-4 (hardback), 0-13-011800-1 (paperback).
- [27] Michael A. Schoonover, John S. Bowie, and William R. Arnold. *GNU Emacs: UNIX Text Editing and Programming*. Addison-Wesley, Reading, MA, USA, 1992. ISBN 0-201-56345-2.
- [28] Axel T. Schreiner and H. George Friedman, Jr. *Introduction to Compiler Construction Under UNIX*. Prentice-Hall, Englewood Cliffs, NJ 07632, USA, 1985. ISBN 0-13-474396-2.
- [29] W. Richard Stevens. *UNIX Network Programming*. Prentice-Hall, Englewood Cliffs, NJ 07632, USA, 1990. ISBN 0-13-949876-1.
- [30] Bjarne Stroustrup. *The C++ Programming Language*. Addison-Wesley, Reading, MA, USA, second edition, 1991. ISBN 0-201-53992-6.
- [31] Eric van Herwijnen. *Practical SGML*. Kluwer Academic Publishers Group, Norwell, MA, USA, 1990. ISBN 0-7923-0635-X.
- [32] X/Open Company, Ltd. *X/Open Portability Guide, XSI Commands and Utilities*, volume 1. Prentice-Hall, Englewood Cliffs, NJ 07632, USA, 1989. ISBN 0-13-685835-X.

Index

- ., 401
- .bb1, 412
- .bibcleanrc, 397
- .ini, 400, 402
- /* empty */ rule, 413
- /*...*/, 413
- : , 401
- ? , 402
- @, 400
- %{ , 407, 410
- %} , 407, 410
- %%, 402, 407, 410
- %e, 402
- %f, 402
- %k, 402
- %left, 410
- %nonassoc, 410
- %token, 410
- %v, 402
- \", 401, 402, 409
- \c, 405, 406, 408, 411
- \Begin, 405
- \Begin{ comment } , 405
- \Comment, 405
- \Comment{ ... } , 405, 406, 411
- \End, 405
- \End{ comment } , 405
- \Preamble{ ... } , 405
- \String, 405
- \String{ ... } , 406
- \c, 405
- @name, 405, 411
- @name{ ... } , 411
- \\", 401
- \0x0a, 401
- \0xhh, 401
- 8-bit clean, 409
- \0, 401, 409
- \012, 401
- \013, 407
- \A, 401
- \a, 401
- \a, 401
- abbreviation, 408, 412, 413
- accent control sequence, 409
- Aho, Alfred V., 404, 405, 411

anonymous **ftp**, 414
ANSI/ISO Standard C, 403, 404,
 407
 apostrophe
 in citation key, 406
Apple Macintosh, 407
 archive hosts
 Internet, 414
Article, 411
 assignment, 413
 list, 412
 operator, 409, 413
 associativity of, 410
 rule
 error recovery in, 411
 separator, 409
 assignment rule, 412, 413
 assignment_lhs rule, 413
 assignment_list, 412, 413
 assignment_list rule, 412, 413
 associativity, 410
 at-sign, 405
 at_object rule, 411
 Atari, 404
 -author, 397
 author, 413
 author name
 period after initials, 398
 reordering, 398
 auxiliary file, 395

\b, 401
 Bach, P. D. Q., 399
 back end, 397
 backslash-newline, 398, 401, 403,
 407
 backslash-quote, 405
 Beebe, Nelson H. F., 396
bib, 406
bibclean, 395–405, 407, 408, 410,
 412, 414
bibclean.ini, 397
BIBCLEANEXT, 400
BIBCLEANINI, 397
bibdup, 404
bibdup.awk, 404
BIBINPUTS, 397, 400
biblex, 407, 410
 bibliography
 entry, 412
 file, 396, 397
 style, 395, 396
\bibliography, 395, 396, 406
 bibliography-specific pattern, 402
\bibliographystyle, 395
biparse, 410
bibtex_file rule, 411
bison, 405, 406, 408, 410–412, 414
Book, 411

Borland International, 404
 brace, 409
 ignored in pattern matching,
 402
 space around, 402
 Brown, Doug, 405
 Bryan, Martin, 396
 buffer overflow, 406

C++, 397, 404
 Cameron, Debra, 396
 carriage return, 407
chapter, 402
-check-values, 398, 400
 checksum
 in ISBN and ISSN, 402
 citation
 key, 395, 403, 412
 problems in recognizing,
 412
 style, 395
\cite, 396
 class library, 404
 code generation, 397
 colon, 410
 comma, 412
 optional after assignment, 413
 command-line options, *see* options
Comment, 408, 409, 411
 comment
 entry, 411
 in-line, 408, 410, 412, 413
 associativity of, 410
 precedence of, 410
 syntax, 406
comment, 405
comment rule, 411
 concatenation, *see* string
 control sequence
 \bibliography, 395, 396, 406
 \bibliographystyle, 395
 \cite, 396
 core dump, 404

D, 401
d, 401
 decimal, 397
-delete-empty-values, 398, 399
 delimited string, 408, 412
 delimiters
 in SCRIBE, 405
 mismatched, 398, 413
 digit string, 408
 documentation, 414
 dump
 post-mortem, 404

 editor, *see* Emacs
 editor name

period after initials, 398
 reordering, 398
 electronic mail server, 414
 Ellis, Margaret A., 404
 Emacs, 396, 399, 405
 embedded quote, 409
 empty
 pattern, 402
 string, 413
 values
 deleting, 398
 entry
 name, 408
entry rule, 411–413
entry_head rule, 412
 environment variable, 397
 error
 log file, 399
 message, 411
 redirecting, 397
 recovery, 411
 reporting, 406
error rule, 411
-error-log filename, 397
 escape sequence, 401, 403, 405, 409
 in message text, 402
 table, 401

\f, 401, 407
 Feuer, Alan R., 404
 field name, 408, 413
 file
 .bbl, 412
 .bibcleanrc, 397
 .ini, 400, 402
bibclean, 404
bibclean.ini, 397
BIBCLEANEXT, 400
BIBCLEANINI, 397
bibdup, 404
bibdup.awk, 404
BIBINPUTS, 397, 400
 bibliography, 396, 397
 error log, 399
ftp.math.utah.edu, 414
 inclusion, 406, 412
 initialization, 397, 398, 400
 locating, 399
 nested, 400
 pattern characters, 401
 patterns in, 398
 name
 space in, 412
 syntax of, 412
nawk, 404
PATH, 397, 400
pub/tex/bib, 414
regexp, 401
 sample initialization, 400

stderr, 397, 399
stdin, 400
SYS\$SYSTEM, 397
tr, 404
tuglib@math.utah.edu, 414
-file-position, 398, 399
-fix-font-changes, 398, 399
-fix-initials, 398, 399
-fix-names, 398, 399
flex, 405, 407, 414
font changes
fixing, 398
format
item, 402
`%%`, 402
`%e`, 402
`%f`, 402
`%k`, 402
`%v`, 402
of grammar rules, 407, 410
formfeed, 407
Free Software Foundation, 397, 405
Friedman, Jr., H. George, 405
front end, 397
ftp, 414
ftp.math.utah.edu, 414
function
`out_lbrace()`, 409
`out_lparen()`, 409
`out_protected_string()`, 408
`out_rbrace()`, 409
`out_rparen()`, 409
`out_string()`, 408
`out_token()`, 408, 409

GNU
Emacs, 396, 399, 405
regexp package, 401
TeXinfo, 405
grammar, 405
format of rules, 407, 410
formatting conventions, 410
LALR(1), 406, 412
lexical, 406
LL(0), 405
LL(1), 405, 406
LR(*k*), 406
parsing, 409
size of, 407, 410

Harbison, Samuel P., 404
help, 414
-help or -?, 397
Herwijnen, Eric van, 396
hexadecimal, 397
horizontal space character, 407, 409
in-line comment, 408, 410, 412, 413

associativity of, 410
precedence of, 410
Include, 408, 409, 411, 412
include rule, 411, 412
-init-file filename, 397, 398, 400
initialization file, 397, 398, 400
locating, 399
nested, 400
pattern characters, 401
patterns in, 398
sample, 400
Internet archive hosts, 414
interpretation of code, 397
ISBN, 402
ISBN (International Standard Book Number), 399
ISO10646M character set, 406
ISSN, 402
ISSN (International Standard Serial Number), 399

Jaeschke, Rex, 404
Johnson, Steven C., 405

Kahn, Philippe, 404
Kernighan, Brian W., 404
key, 403, 405
key name, 408, 412
key_name rule, 412
Knuth, Donald E., 398, 404, 406
Koenig, Andrew, 404

LALR(1)
grammar, 406, 412
parser, 406
Lamport, Leslie, 395, 403, 406
Lapin, J. E., 404
last_object, 408
last_token, 408
%left, 410
left-recursive rule, 411
Lesk, Michael E., 405
Levine, John R., 405
lex, 405–409, 414
lexer, *see* lexical analyzer
lexical analysis, 396
lexical analyzer, 397, 398, 403
lexical grammar, 406
line
number, 406
number directive, 403
width limit, 397
wrapping, 397, 403, 407
list
of assignments, 412
of objects, 411
literate programming, 407, 410
LL(0) grammar, 405

LL(1)
grammar, 405, 406
parser, 411
LR(*k*) grammar, 406

Macintosh
Apple, 407
macro, *see also* control sequence
N, 407, 408
O, 407
RETURN, 408–410
S, 407
W, 407
macro definition
lex, 407
macro use
lex, 407
Mason, Tony, 405
-max-width 0, 397
-max-width nnn, 397, 403
menu
pop-up, 396
message
disabling warning, 399
error, 411
help, 397
redirecting, 397
mismatched delimiters, 398, 413
month, 402

N, 407, 408
\n, 401
name, 411
nawk, 404
newline, 409, 412, 413
associativity of, 410
-no-check-values, 398
-no-delete-empty-values, 398, 399
-no-file-position, 398
-no-fix-font-changes, 398
-no-fix-initials, 398
-no-fix-names, 398
-no-init-files, 400
-no-par-breaks, 398, 399
-no-prettyprint, 397, 398, 403, 407, 410
-no-print-patterns, 398
-no-read-init-files, 398
-no-remove-OPT-prefixes, 398
-no-scribe, 399
-no-trace-file-opening, 399
-no-warnings, 399
non-terminal, 410
/ empty */*, 413
assignment, 412, 413
assignment_lhs, 413
assignment_list, 412, 413

at_object, 411
bibtex_file, 411
comment, 411
entry, 411–413
entry_head, 412
error, 411
include, 411, 412
key_name, 412
object, 411
object_list, 411
opt_space, 411–413
preamble, 411, 412
simple_value, 412
single_space, 413
space, 413
string, 411, 412
value, 412, 413
%nonassoc, 410
NUL (0)
 in string, 401, 409
number, 402

0, 407
object, 411
 list, 411
object rule, 411
object-oriented programming, 404
object_list rule, 411
Objective C, 397, 404
octal, 397
\ooo, 401
operator
 assignment, 409, 413
 string concatenation, 408,
 410, 412
OPT- prefix
 removing, 398
opt_space, 412
opt_space rule, 411–413
option
 -author, 397
 -check-values, 398, 400
 -delete-empty-values, 398,
 399
 -error-log filename, 397
 -file-position, 398, 399
 -fix-font-changes, 398, 399
 -fix-initials, 398, 399
 -fix-names, 398, 399
 -help or -?, 397
 -init-file filename, 397,
 398, 400
 -max-width 0, 397
 -max-width nnn, 397, 403
 -no-check-values, 398
 -no-delete-empty-values,
 398, 399
 -no-file-position, 398
 -no-fix-font-changes, 398

-no-fix-initials, 398
-no-fix-names, 398
-no-init-files, 400
-no-par-breaks, 398, 399
-no-prettyprint, 397, 398,
 403, 407, 410
-no-print-patterns, 398
-no-read-init-files, 398
-no-remove-OPT-prefixes,
 398
-no-scribe, 399
-no-trace-file-opening, 399
-no-warnings, 399
-par-breaks, 398
-prettyprint, 398
-print-patterns, 398
-read-init-files, 398
-remove-OPT-prefixes, 398
-scribe, 399
-trace-file-opening, 399,
 400
-version, 399
-warnings, 399
options, 400
OS/2, 404
out_lbrace(), 409
out_lparen(), 409
out_protected_string(), 408
out_rbrace(), 409
out_rparen(), 409
out_string(), 408
out_token(), 408, 409
overflow of string buffer, 406

pages, 402
-par-breaks, 398
parenthesis, 409
parser
 LALR(1), 406
 LL(1), 411
parsing, 396
parsing grammar, 409
Pascal, 404
PATH, 397, 400
pattern
 bibliography-specific, 402
 changing warning message,
 402
 empty, 402
 quotes in, 402
pattern matching, 400
 brace ignored in, 402
 regular expression, 400
PC-DOS, 396, 397, 400, 404
period
 in citation key, 406
 in regular expression, 407, 409
pipeline, 404
Plauger, P. J., 404

pop-up menu, 396
portability, 404
post-mortem dump, 404
Preamble, 408, 411, 412
preamble rule, 411, 412
precedence declaration, 410
preprocessor, 403
-prettyprint, 398
prettyprinter, 397, 398, 403
prettyprinting, 399
-print-patterns, 398
program
 search path, 400
 version, 399
pub/tex/bib, 414

query (?)
 in messages, 402
quote
 embedded, 409
 in pattern, 402

R, 401
\r, 401, 407
r, 401
Rabinowitz, Henry, 404
-read-init-files, 398
recovery
 from error, 411
recursion, 400
refer, 406
regexp, 401
regular expression
 pattern matching, 400
 syntax of, 407
Reid, Brian, 405
-remove-OPT-prefixes, 398
RETURN, 408–410
return, 408
Ritchie, Dennis M., 404
Rochkind, Marc J., 404
Rosenblatt, Bill, 396
run-time options, *see* options
runaway string argument, 398, 406

S, 407
Schaap, Chaim, 404
Schickele, Peter, 399
Schmidt, Eric, 405
Schreiner, Axel T., 405
SCRIBE, 395, 399, 404–407, 409,
 411, 416
-scribe, 399
search path, 400
semicolon, 410
send, 414
separator
 assignment, 409
Sethi, Ravi, 405, 411

- SGML, 396
sharp (#), 403, 408
simple value, 412
simple_value rule, 412
single space, 413
single_space rule, 413
source code, 414
space, 410, 413
 - associativity of, 410
 - between tokens, 411
 - precedence of, 410**space rule**, 413
standard error unit, 397
stderr, 397, 399
stdin, 400
 Steele Jr., Guy L., 404
 Stevens, W. Richard, 404
String, 408, 411, 412
string
 - concatenation operator, 408, 410, 412
 - pool, 404
 - runaway, 398, 406
 - substitution, 395**string rule**, 411, 412
 Stroustrup, Bjarne, 404
style
 - bibliography, 396**SYS\$SYSTEM**, 397

\t, 401
template
 - editor, 396**terminal**, 410
 - TOKEN_ABBREV, 403, 408, 410, 412, 413
 - TOKEN_AT, 403, 408, 410, 411
 - TOKEN_COMMMA, 403, 408–410, 412, 413
 - TOKEN_COMMENT, 408, 410, 411
 - TOKEN_ENTRY, 403, 408, 410, 412
 - TOKEN_EQUALS, 403, 409, 410, 413
 - TOKEN_FIELD, 403, 408, 410, 413
 - TOKEN_INCLUDE, 408, 410, 412
 - TOKEN_INLINE, 408, 410, 413
 - TOKEN_KEY, 403, 408, 410, 412
 - TOKEN_LBRACE, 403, 409, 410, 412, 413
 - TOKEN_LITERAL, 409–412, 414
 - TOKEN_NEWLINE, 403, 409, 410, 413
 - TOKEN_PREAMBLE, 408, 410, 412
 - TOKEN_RBRACE, 403, 409–413
 - TOKEN_SHARP, 408, 410, 412
 - TOKEN_SPACE, 409, 410, 413
 - TOKEN_STRING, 403, 408–410, 412
 - TOKEN_VALUE, 403, 408–410, 412**TOS**, 404
tr, 404
 - trace-file-opening**, 399, 400**trailing context**, 412, 414
trap, 414
trip, 414
TUG bibliography collection, 396, 399
TUG Resource Directory, 396
TUGboat, 396
tuglib@math.utah.edu, 414

Ullman, Jeffrey D., 405, 411
unclassifiable token, 409
UNIX, 396, 397, 400, 401, 404–407, 409

\v, 401, 407
value, 412
value rule, 412, 413
van Herwinnen, Eric, 396
variable
 - last_object**, 408
 - last_token**, 408**VAX**, 396, 397, 404
version
 - of program, 399**-version**, 399
vertical
 - bar, 410
 - tab, 407**VMS**, 396, 397, 404
volume, 402

W, 401, 407
w, 401
warning message
 - changing, 402
 - disabling, 398, 399
 - redirecting, 397**-warnings**, 399
WEB, 404, 406, 407
 Weinberger, Peter J., 404
wrapping
 - of long lines, 397, 403, 407
X, 401, 402
\x, 401
x, 401
X/Open Consortium, 409

yacc, 405–414
year, 402

◊ Nelson H. F. Beebe
 Center for Scientific Computing
 Department of Mathematics
 University of Utah
 Salt Lake City, UT 84112
 USA
 Tel: +1 801 581 5254
 FAX: +1 801 581 4148
 Internet: beebe@math.utah.edu

Graphics

A Tough Table Becomes Easy with PICTEX

Kevin Carmody

A comment was made in *TUGboat* [2, p. 437] to the effect that *TEX* does not allow one to typeset a table or anything else by specifying page positions. This made me think of my own experience typesetting the table below. This table with its "gnomons" (L-shaped corridors) had defeated my best efforts to typeset it in plain *TEX*. Fortunately, I remembered that I was already familiar with a way to place text and draw lines by coordinates: *PICTEX*.

Infinite Rectangular Array

1	3	5	7	9	11	13	15	17	19
1	4	7	10	13	16	19	22	25	28
1	5	9	13	17	21	25	29	33	37
1	6	11	16	21	26	31	36	41	46
1	7	13	19	25	31	37	43	49	55
1	8	15	22	29	36	43	50	57	64
1	9	17	25	33	41	49	57	65	73
1	10	19	28	37	46	55	64	73	82
1	11	21	31	41	51	61	71	81	91
1	12	23	34	45	56	67	78	89	100

The *PICTEX* commands for this table are as follows:

```
$$\begin{picture}
\setcoordinatesystem units <20pt,20pt>
\setplotarea x from 1 to 10,
y from 1 to -10
\put { 1} [r] at 1 -1
\put { 3} [r] at 2 -1
\put { 5} [r] at 3 -1
...
\putrule from 0.3 -1.5 to 1.3 -1.5
\putrule from 1.3 -1.5 to 1.3 -0.5
...
\end{picture}$$
```

This table appears in a famous Russian puzzle book [1]. Among its properties is the fact that the sum of the numbers in each gnomon is a perfect cube.

This small example reminds us once more that proper macros can accomplish the seeming impossible. It further shows us how to do coordinate-based layouts in *TEX*.

References

- [1] Kordemsky, Boris A. *The Moscow Puzzles*. New York, Scribner's, 1972.
- [2] Taylor, Philip. "The Future of *TEX*", *TUGboat* 13, no. 4, (December 1992), pp. 433-442.
- [3] Wichura, Michael. *The PICTEX Manual*. (*TEX-techniques Series*, No. 6.) Providence, R.I., *TEX* Users Group, 1987.

◊ Kevin Carmody
R.O.W. Sciences
1104 Arcola Avenue
Wheaton, MD 20902

Book Reviews

Book review: *TEX per l'impaziente*

Claudio Beccari

P. W. Abrahams, K. Berry and K. Hargreaves, *TEX per l'impaziente*. (Translation of *TEX for the impatient*; translated by Gaia and Guido Franchi.) Milano: Addison-Wesley, 1991. 396 pp. ISBN 88-7192-022-8.

Although in Italy there is no national TUG association, the *TEX* users community is pretty broad since *TEX* is widely used in academic environments as well as in commercial activities and in public services.

In the summer of 1991 the Italian branch of the well known publishing house Addison-Wesley published a translated version of *TEX for the impatient* by P. W. Abrahams, K. Berry and K. Hargreaves. The new title, *TEX per l'impaziente*, closely reflects the original one, and the same applies for the contents, although the sitting White Rabbit of Alice in Wonderland is reproduced only on the front cover, not in the chapter front pages.

The translators, Gaia and Guido Franchi, did a very good job with the translation, but they had to face the lack of professional phototypesetters capable of setting a book with Computer Modern fonts; in my country there are no problems with phototypesetters that use the classical PostScript standard fonts and a large variety of other outline fonts, since most of the machinery is imported or is adapted from U.S. hardware and software. Therefore the Franchis had to rely on their 300 dpi laser printer and have the publisher print the whole book from the translators' originals; the result is fairly good but compares unfavorably with the English version.

On the other hand the Italian version is free from that annoying bug that infested the original book, when the Optima font with a different character layout was used for the command headings so that open and closing braces were substituted with en-dashes and closing double quotes respectively; the Computer Modern sans serif font is used in its place, but when you find \ you don't know if it means \l or \I.

The translation is quite good, and some of the small errors of the English version are eliminated (for example in the Edible Mushroom table *Boletus edulis* is spelled correctly) and the translators succeeded in rendering all the examples in Italian, even the one that explains the \parshape command

with the paragraph shaped as the silhouette of a wine glass; it is not simple at all considering that Italian words are significantly longer on the mean than their English counterparts.

It is not the purpose here to praise or disparage the book: it has more or less the same advantages and faults that Victor Eijkhout pointed out in his review (*TUGboat*, vol. 11 (1990), pp. 572–573) but it has some features that are specific for the Italian orthography that were not listed (of course) in the original English text; this is a reason why the translators must be praised for the good job they did. At the same time (a book review must always contain some criticism) there are some points that leave me unsatisfied, and I think it is very important to point them out because, besides this translation, they might be overlooked also in other circumstances.

- From the very beginning (page 15) the translators introduce the possibility of assigning a category code 13 to the accented characters à, è, é, ì, ò, ù that have individual keys on the Italian keyboard, and to define them so as to correspond to the \`a, ..., \`u commands. This is certainly possible if the *TEX* implementation in use accepts input characters with ASCII codes higher than 127; but this is not always the case. Sometimes this is just impossible, sometimes it requires a special initialization with a suitable *codepage* file that establishes the necessary correspondences for the input and the output of these characters with the internal codes that *TEX* uses; no warning is given with this regard.
- Italian hyphenation, or better, the patterns that were used for the Italian hyphenation of this book, are reported in an appendix (pp. 381–382); the rules are taken from an unspecified Italian grammar that I suppose was a junior high school level textbook. The rules specified in such initial level grammars reduce to the simple statement that "you can put the hyphen wherever the syllable to the right of the hyphen starts with one or more letters that may be found at the beginning of another Italian word." This statement probably holds true for the totality of the words a junior high school student might encounter, but is completely unsatisfactory with grownup people's vocabulary.

Apparently Guido Franchi listed the groups of two consonants that could be found at the beginning of words¹ and then prepared patterns

¹ He lists also v1 but I do not know of any common Italian word starting with this group: there

with all combinations of one and two consonants of the form

1K2 4B3C4 4s3s4

where K is any of the 16 “Italian” consonants², and B and C are the sets of consonants such that $B = K \setminus h$ and $C = K \setminus \{h, l, m, n, r, s\}$. They obtain a total of 210 simple patterns that do a pretty good job with the setting of the book.

Unfortunately this set of patterns has several drawbacks (and one advantage):

1. the set contains a large number of combinations that never occur in Italian (for example all those of the series 4q3C4, and many more);
2. the set is incomplete in the sense that it cannot split vocalic clusters into their component diphthongs and “tripthongs”; although TeX minimizes the number of hyphenated line breaks, this is a major point with Italian where vowels play a more important rôle than in several other languages;
3. the set contains some errors in the sense that the groups pn and ps should be split, even if there are some Italian words starting with such groups; fortunately enough these groups occur very rarely;
4. separable prefixes are ignored; the national regulations allow prefixed words to be hyphenated with common hyphenation rules, but there are some prefixes, used mostly in technical writing, that it is better to separate according to etymology;
5. (advantage) the method Franchi used, although incomplete and error prone, is suitable for a “formal hyphenation” grammar for many languages provided that sets of vowels, semivowels, consonants and semi-consonants are properly defined. It would be a pleasure if TeX could deal with “generalized” patterns so that the hyphenation

is Vladimiro, but this is an italianization of a foreign proper name, and in my dictionary I found the word “vladika” that comes from (actually is) Serbo-Croatian and means bishop in the orthodox church.

² Take the 26 letter Latin alphabet, eliminate the vowels and the letters j, k, x, y, w and you are left with the consonants that occur in ordinary Italian words. The adjective “Italian” is quoted because even today many Italian grammars stick to the obsolete autharchic axiom that the Italian alphabet contains just 21 letters.

table for each language could consist of a very limited number of entries, such as the three patterns above, without the need of expanding the combinations.

Moreover the Franchi state that hyphenation patterns should be written one per line (which, unless they refer to a particular implementation of TeX and initex, is completely new to me, and is not documented in *The TeXbook*); in addition, before defining the Italian patterns they establish the codes for the apostrophe in this way:

```
\catcode`\'=11
\lccode`\'=11
\uccode`\'=11
```

and after the list of patterns they reset the codes this way:

```
\catcode`\'=12
\lccode`\'=12
\uccode`\'=12
```

The *TeXbook* states that patterns can be constructed with any character of category 11 or 12 provided it has a nonzero \lccode; therefore the above definitions are mostly superfluous and may lead to errors if the primitives \uppercase and \lowercase were used.

Aside from the above comments, the book is well translated and should prove very useful among the Italian users and, may be, help the further diffusion of our favorite text processor.

◊ Claudio Beccari
Dipartimento di Elettronica
Politecnico di Torino
Turin, Italy
beccari@polito.it

Hints & Tricks

Ten *TeX* Tricks for the Mathematician

Helmer Aslaksen

TeX has changed the face of mathematical typesetting. If you look at the proceedings from a conference published ten years ago, you will probably find that most of the articles were prepared with a typewriter. Today, most of them will be done by *TeX*. More and more monographs are also produced using the author's *TeX* file. Is this a step forward?

For proceedings, I would definitely say yes. The typewriter will go the way of the dinosaurs, and I'm not going to miss it. But when it comes to monographs, the author's camera ready copy must be compared to professionally set books. An expert *TeX*nician can produce output of the highest standard, but the average *TeX* author/typist fails miserably when compared to professional typesetting. Most authors/typists are not very knowledgeable about *TeX* or mathematical typography. They tend to make the same common mistakes. The purpose of this brief article is to try to point out some such errors. This list reflects my personal choice. I would like to thank the referee for helpful comments.

All page references are to the seventh printing of you-know-which book. I don't always give details about how to achieve the different effects. This is partially because the syntax would be different depending on which dialect of *TeX* you use.

1. *Set operator names in roman.* My head goes into a spin whenever I read about *Spin*(*n*). Look at the spacing! Math italics uses special spacing (p. 164). As a general rule, *every* mathematical term with more than one letter should be set in roman, whether or not it is in Knuth's list (p. 162 and p. 361). So please write *Spin*(*n*). If you use *AMS-LATEX*, you can write \$\operatorname{Spin}(n)\$, or you can define \Spin to be \mathop{\rm Spin}\nolimits. A clever trick (due to the referee) is to define a macro like

```
\def\newop#1
  {\expandafter\def\csname #1\endcsname
   {\mathop{\rm #1}\nolimits}}
```

Then \newop{\Spin} will define a command \Spin that can be used throughout the paper.

2. *Scale the delimiters.* Constructions like

$$S = \left\{ \begin{pmatrix} \pm 1 & 0 \\ 0 & \pm 1 \end{pmatrix} \right\}$$

are the sign of a true *TeX*-novice. Please write

$$S = \left\{ \begin{pmatrix} \pm 1 & 0 \\ 0 & \pm 1 \end{pmatrix} \right\}.$$

I also find $[[X, Y], Z]$ easier to read than $[[X, Y], Z]$.

3. *Use / more often.* Always write a/b in text. Big fractions like $\frac{a}{b}$ can mess up a whole paragraph. This also raises another issue. You should understand the difference between display style and text style. *TeX* has a tendency to use text style when I feel display style would be better. I prefer

$$f(x) = \frac{g(x)}{h(x)} \quad \text{to} \quad f(x) = \frac{g(x)}{h(x)}.$$

4. *Use the right kind of dots.* This is slightly controversial. Everybody I know writes $1, \dots, n$ and $x_1 \dots x_n$, but Knuth (p. 172) wants $x_1 \dots x_n$. Anyway, don't write

$$x_1 + \dots + x_n.$$

5. *Should you break before or after +'s?* The rule is simple (p. 195): you break *after* binary operators in text and *before* binary relations in displays. And when you break before a +, remember to write {}+x, so *TeX* knows that the + is a binary operator (p. 196).

6. *Be generous with space.* Watch for places to put \, (pp. 167–169). Don't you think (,) looks better than (,)? Learn how to insert space between formulas in display, or use constructions that do it for you. Compare

$$\begin{array}{ll} f(x) = x & f(x) = x \\ g(x) = x^2 & \text{and} \\ & g(x) = x^2. \end{array}$$

Notice how the parentheses almost touch in the first one.

8. *Get your bibliography right.* Don't write *Notices Amer. Math. Soc.*, write *Notices Amer. Math. Soc.* (Use ._ to get proper spacing.) And write pp. 1–40 instead of pp. 1-40 (remember to use --, see p. 4).

9. *Don't use symbols for visual effects.* Learn to use the proper commands. On a typewriter, people must use logical symbols like < for visual effects, like writing $\langle x, y \rangle$ to denote an inner product. Fortunately, *TeX* has a huge supply of mathematical symbols and delimiters. In this case you should use the so-called angle brackets, to get $\langle x, y \rangle$ ($\$\\langle x, y \\rangle\$$). And remember to write \langle, \rangle ($\$\\langle, \\rangle\$$) and not \langle, \rangle .

Similarly, don't write $x \epsilon S$ ($\$x \\epsilon S\$$), but $x \in S$ ($\$x \\in S\$$). First of all, \epsilon is the wrong symbol, and secondly the spacing is wrong. When you use \in, *TeX* knows that you want a binary relation, so it puts in the proper amount of space.

I'm also tired of seeing "this" ("this"). It should be "that" ("that")! Notice how " will always give the wrong result on the left. When using Computer Modern fonts, " gives the right result on the right, but it may not work for other fonts.

And I think that \tilde{S} looks too wimpy. Beef it up with a \widetilde to get \tilde{S} .

10. *Read Chapter 18.* Just do it!

◊ Helmer Aslaksen
 Department of Mathematics
 National University of Singapore
 Singapore 0511, Republic of
 Singapore
 mathelmr@nusunix.nus.sg

Macros

The bag of tricks

Victor Eijkhout

Hello all. From Jonathan Kew I received the following useful macros. Their purpose is to make testing hyphenation patterns easier. We all know about \showhyphens, but for long lists of words using this is tedious. The macro \printhypens takes a list of words and prints their hyphenation on consecutive lines. T_EX hackers will get a kick out of these macros. In particular the nested use of \everypar is neat.

```
\def\printhypens{
  \everypar{%
    \setbox0\lastbox
    \setbox1\hbox{\strut}
    \vbox\bgroup
      \everypar{\setbox0\lastbox
        \nobreak\hskip0pt
        \relax}
      \dimen0=\hsize
      \hsize=0pt \hfuzz\maxdimen
      \def\par{\endgraf \hsize=\dimen0
        \getlastline \egroup \endgraf}}
    \offinterlineskip\breakafterword}
```

```
\def\breakafterword
  {\catcode'`^M\active\catcode`\ \active}
{\breakafterword%
\gdef`^M{\par}\global\let `^M}
\def\getlastline{\setbox0\lastbox
  \ifvoid0 \let\next\nomorelines
  \else \unskip\unpenalty
    \setbox1\hbox
      {\unhbox0\strut\discretionary{}{}{}%
       \unhbox1}
    \let\next\getlastline
  \fi \next}
\def\nomorelines{\unhbox1}

A small test

\begin{group}
\printhypens
photograph photography photographer
photographical photographically
hypersupersuperdupercali%
fragilisticexpihalidocious
Eijkhout
\end{group}

gives
```

```
photo-to-graph
photo-to-gra-phy
photo-to-gra-pher
photo-to-graph-i-cal
photo-to-graph-i-cally
hy-per-su-per-su-perduper-cal-ifrag-ilis-tic-ex-pi-
hali-do-cious
Eijk-hout
```

The \discretionary{}{}{} is an addition to the original macros; I took it from the macro by Oliver Schoett that is used for the hyphenation exception list of TUB. Its purpose is to make line breaks possible for long words.

And that's it for this time. More contributions from readers are welcome!

◊ Victor Eijkhout
 Department of Computer Science
 University of Tennessee at
 Knoxville
 Knoxville TN 37996-1301
 Internet: eijkhout@cs.utk.edu

LATEX

The “operational requirement” (?) for support of bibliographic references by LATEX3

David Rhead

Abstract

It is suggested that:

- LATEX3 should aim to support the principal citation schemes used in conventional publishing
- consideration be given to a *modus vivendi* between LATEX3 and mainstream bibliography-formatting software.

Contents

1	Introduction	425
2	Doing it yourself	425
3	Using bibliography-formatting software	427
3.1	Background	427
3.1.1	Software available	427
3.1.2	<i>Modus vivendi</i> with the main 4?	428
3.1.3	Preferred interface	428
3.1.4	Hybrid approaches	430
3.1.5	The user’s choice	431
3.2	OR for LATEX3	431
4	Miscellaneous	431
4.1	“Local names” for keys	431
4.2	Reference-lists that are also indexes	432
A	Some suppliers of mainstream bibliographic software	432
B	E-mail discussion lists about bibliographic software	432

1 Introduction

Ideally, when writing software, it is a good idea to write down what the software is intended to achieve —the “operational requirement” — before writing any code.

This article attempts to take an “operational requirement” approach to the “bibliographic reference” aspects of LATEX3.¹ The objective is to stimulate debate—if you don’t agree with my suggestions, please suggest specific alternatives! (In the

¹ Obviously, there are limits to the what the OR approach can achieve. For example, it is difficult to quantify “usability”. Nevertheless, the approach should facilitate debate about objectives *before* the “user interface” has been fixed.

remainder of the article, “operational requirement” is abbreviated to “OR”).

Generalizing the approach taken by the LATEX 2.09 manual [1, pp. 73–74], it is convenient to divide the topic into “doing it yourself” and “using bibliography-formatting software”.

2 Doing it yourself

In effect, the only scheme that is “fully supported” by LATEX 2.09 is “reference by number, where the sequence of numbers is determined by position in the reference-list”.

By contrast, for “real-world publishing”, my impression is that:

1. only a minority of “instructions to authors” specify anything like the default LATEX 2.09 scheme. This minority consists of those journals that specify “reference by number, with the reference-list in alphabetical order of author’s names”.
2. the majority of “instructions to authors”, style-books, etc., specify one of the following:
 - (a) reference by number, with the reference-list in order of first citation
 - (b) author-date
 - (c) “short-form in footnotes”. For publications in the humanities, there seem to be two main variants of this scheme, depending on whether or not there is a reference-list.²

ISO 690 [3, sec. 9] provides a convenient specification of the details of these schemes. The default LATEX 2.09 system gives no particular help to anyone wanting to use them.³

3. a few publishers specify alternative schemes. E.g.,

² If there is no reference-list, the convention is usually “first citation gives full bibliographic details, subsequent citations give cross-reference to first citation”. This variant is common in law publications, when it is used in conjunction with numerous law-specific citation conventions [2].

³ BIBTEX can help with (a). Anyone wishing to use (b) will probably grope around in archives looking for style-options that: arrange for \cite to give (...) rather than [...]; omit [...] from the reference-list; support date-only citations when the author’s name appears naturally in a sentence. Apart from the law-specific LexiTEx [4], I’m not aware of any 2.09-related software that helps people who wish to use scheme (c).

- some Springer journals⁴ accept citations of the form “first letter of author’s surname, in square brackets”
- Butcher [5] mentions a variation of the reference-by-number system in which there is a separate numerical sequence for each letter, and a variation of the author-date system in which a number is used instead of a date
- a scheme like the BIBTEX alpha style is sometimes used (for example, in the journal *Formal Aspects of Computing*).

Therefore, I suggest that the OR for LATEX3:

- should include support⁵ for the schemes mentioned in items 1 and 2 above, i.e.,
 - a 2.09-like scheme aimed at journals that specify “reference by number, with the reference-list in alphabetical order of author’s names”
 - the schemes specified in ISO 690, namely: “reference-by-number in order of first citation”, author-date, and 2 variations of “short-form in footnotes”.⁶
- should bear in mind the possibility of a “plug-in module” to support law conventions. Since such conventions are crucial only to lawyers, it would probably be inappropriate to delay LATEX3 while law-specific commands were finalised, or to increase the bulk of the LATEX3

⁴ See the “instructions for authors” in, for example, *Mathematische Zeitschrift*.

⁵ I assume that “sorting a reference-list” will be beyond the scope of LATEX3. Thus, in practice, the LATEX3 “support” might be minimal (a “better than nothing” warning that a reference-list needs human intervention, perhaps). People who want anything better would be advised to use bibliography-formatting software.

⁶ To support these schemes, it is probably desirable that LATEX3 should be able to determine whether a citation of a source is “the first citation” of that source. Clearly this would help to provide support for “reference by number in order of first citation”. In the author-date case, it would allow support for the convention [6, sec. 3.87] that, when there are multiple authors, they should all be named in the first citation but “et al.” should be used subsequently. It might also help to provide support for the variant of the short-form scheme in which a “subsequent citation” uses the short-form and gives a cross-reference to the footnote containing the “first citation” (where full details of the source can be found).

manual by including law-specific material. Nevertheless, it might be worth simultaneous experiments with a prototype LATEX3 and a prototype law-support module, in the hope that the law-specific commands in such a module might end up with a similar “look and feel” to those for the mainstream “short-form in footnotes” commands.

- need not include support for the alternative schemes mentioned in item 3 above (although the possibility of “plug-in modules” to support these schemes might be borne in mind).

In addition, the following features are desirable:

- for situations where several bibliographic sources are cited simultaneously:
 - a syntax that permits a particular division of each source to be pin-pointed [7, sec. 15.25]. (The LATEX 2.09 \cite[...]{...} syntax only supports pin-pointing within a single source.)
 - a mechanism for sorting reference-by-number citations into ascending numerical order [8, p. 106].⁷
 - a mechanism for sorting author-date citations⁷ into alphabetical order of author’s surnames (or, ideally, the order in which the sources appear in the reference-list) [6, sec. 3.91] or into “date of publication” order [7, sec. 15.24].
- support for types of bibliography that, although not as common as a single undivided list, are appropriate in particular circumstances, namely:
 - a list divided into sections according to kinds of material, subject matter or other appropriate categories
 - an annotated bibliography
 - a bibliographical essay.

See, for example, the *Chicago Manual of Style* [7, chap. 15].

(End-users get confused if they try using LATEX 2.09’s `\thebibliography` environment for such bibliographies.)

The above might provide the major elements of an OR. Minor elements may be more difficult to specify, but can perhaps be summarized as

⁷ Alternatively, if it is not feasible to sort reference-by-number and author-date citations into a desired order, mechanisms for giving warnings if simultaneous citations are in the wrong order would be “better than nothing”.

LATEX3 should be able to survive β -testing of whether it can conveniently deliver bibliographic details formatted as specified by influential style-books and “instructions for authors”.

See [2, 3, 5, 6, 7, 9, 10, 11, 12, 13, 14].

3 Using bibliography-formatting software

3.1 Background⁸

3.1.1 Software available

The bibliography-formatting software that is “advertised” in the *LATEX 2.09* manual is *BIBTEX* [1, 15]. *Tib* [16] is also sometimes mentioned in *TeX* circles.

In fact, there are a large number of bibliography-formatting programs available. A recent review article [17] names 52 such programs.

Judging by comments on the *bibsoft* list, the most important bibliographic programs (from the point-of-view of professional librarians and bibliographers) seem to be *EndNote*, *Library Master*, *Papyrus*, *ProCite* and *Reference Manager*. (Appendices A and B give details of the *bibsoft* list and of the relevant vendors.)

Of these, *EndNote*, *Papyrus*, *ProCite* and *Reference Manager* have procedures for processing a “manuscript”, filling in the in-text citations and generating the corresponding reference-list. Although I understand that a similar facility is planned for the next version of *Library Master*, I don’t know what form this will take. Therefore, when referring to these programs, I will use:

“main 4” to mean the programs (*EndNote*, *Papyrus*, *ProCite* and *Reference Manager*) whose procedures for filling in the in-text citations are currently known

“main 5” to mean the “main 4” plus *Library Master*.

From a *LATEX*-er’s point-of-view, the public-domain *BIBTEX* and *Tib* are obviously attractive, since they were *designed* to work with *TeX/LATEX*, and are available for most of the platforms on which *TeX/LATEX* are available. By contrast, the “main 5” are:

- proprietary

⁸ Warning: I do not currently have “hands on” experience of using *LATEX* in conjunction with software other than *BIBTEX* (although I have browsed through as many of the relevant manuals as I could find). Hence, the ideas given in this section, and in section 4, are theoretical and speculative.

- currently aimed at “wordprocessor” users⁹
- only available on a restricted selection of platforms. (All are available for MS-DOS. Some are available for Macintosh or VAX/VMS.)

Nevertheless, there are many things about the “main 5” that are of interest:

- The programs have standard procedures for importing information from standard database programs, online information services, CD-ROMs and library catalogues.
- They generally have good facilities for maintenance of a “personal bibliographic database”, and for searching such a database for entries that satisfy particular criteria.
- It seems likely that the programs will continue to be developed and supported into the future. (By contrast, my understanding is that *BIBTEX* will be “frozen” when version 1.0 has been finished.)
- There is a choice. If one program has underlying assumptions that don’t match the assumptions that are usual in your discipline, you can look for an alternative!

Even if you don’t regard the “main 5” as of positive interest, you may be unable to avoid them. If a research-group contains a *LATEX*-ing minority and a non-*LATEX*-ing majority:

- the “majority” may choose one of the “main 5” as the group’s “standard bibliography-formatting software”
- the *LATEX*-ers will then be at a serious disadvantage if they cannot use the group’s bibliographic databases.

Also, if your librarian is providing bibliographic information in electronic form (e.g., from a computerized library catalogue), s/he may offer an off-the-shelf way to get the information into a database for one of the “main 5”, but be unable to help you if you use *BIBTEX*.

Overall, it seems to me desirable that, as well as having standard procedures for inter-working with *BIBTEX* and *Tib*, *LATEX3* should have standard procedures for inter-working with the “main 5”. Such procedures are unlikely to be perfect, but it should be possible to agree on some *modus vivendi*.¹⁰

⁹ Certain vendors state that *TeX* is one of their program’s “supported wordprocessors”. You may or may not regard this as a hopeful sign!

¹⁰ It is unlikely that the vendors will re-focus their products to concentrate on *LATEX* users—and equally unlikely that *LATEX*-ers will start to think of themselves as “wordprocessor users”. Nevertheless,

3.1.2 *Modus vivendi* with the main 4?

Before considering how L^AT_EX might co-operate with the “main 4”¹¹ it is convenient to contrast BIBL^ETEX’s approach with that of Tib.

BIBL^ETEX’s approach involves searching a L^AT_EX .aux file for details of in-text citations, and then writing out a .bb1 file. The .bb1 file defines a reference-list that is read in when L^AT_EX is next applied to the root file.

Tib’s approach is different. It starts with a .tex file that contains “incomplete or keyed citations” within citation-delimiters, and produces another .tex file that contains proper in-text citations plus (optionally) a reference-list.

When the procedures used by the “main 4” are interpreted in terms of L^AT_EX, they seem to be more akin to Tib’s approach than to BIBL^ETEX’s. It looks as though the end-user would start with a .tex file containing keys, etc., within citation-delimiters, and use the bibliography-formatting program to produce a near-duplicate .tex file that contains proper in-text citations plus a reference-list.

In fact, Tib’s citation-delimiters are chosen so that:

The escape characters of Tib do not interfere with T_EX processing. If T_EX is applied to the original pre-Tib document, the escape characters and incomplete citations will appear as written.

I.e., the pre-Tib .tex file and the post-Tib .tex file are both valid L^AT_EX input files.

This seems a useful precedent. If L^AT_EX could inter-work with the “main 4” in an analogous way, it would not be necessary to

Apply bibliography-formatting software.

Then apply L^AT_EX.

every time that a .dvi file is required. For example, if someone is concentrating on getting their equations typeset correctly, they might want to get .dvi files quickly without always having to go through the bibliography-formatting step. At the equation-

with a few minor changes (which might involve the L^AT_EX end, the bibliographic program end and/or the documentation), it should be possible for L^AT_EX3 and the mainstream bibliographic software to work reasonably well together.

“*Modus vivendi*”, i.e., “an arrangement between peoples who agree to differ”, seems to fit the situation quite well.

¹¹ Hopefully, it will be possible to use the same general ideas for Library Master when its procedure for “filling in the in-text citations and generating the reference-list” becomes known.

checking stage, they may just want a .dvi file that shows their equations, and not be worried about the appearance of their in-text citations or reference-list.

A potential problem for any L^AT_EX-er trying to follow the Tib precedent, is that EndNote and ProCite use # to identify “number within database”. Hence the end-user may need to put a # (which is one of L^AT_EX’s 10 “special characters”) within the relevant citation-delimiters. (See Table 1 for details of the programs’ default citation-delimiters, and the alternatives available.)

One way of imitating Tib (in spite of the possibility of # characters) might be to arrange delimiters such that the proprietary program’s “start delimiter” is interpreted by L^AT_EX as being equivalent to L^AT_EX 2.09’s \verb+, and its “end delimiter” is interpreted as equivalent to the + that terminates the text introduced by \verb+.

- if L^AT_EX is applied to the original .tex file, the citation keys will be typeset “as is” in a typewriter font (to remind the L^AT_EX-er that the bibliographic software needs applying before the document can be regarded as finished)
- if the bibliographic software is applied to the original .tex file, a new .tex file will be produced that, when L^AT_EX-ed, has proper in-text citations and a reference-list.

Overall, the L^AT_EX-er will be able to apply L^AT_EX and the bibliographic software in either order (in much the same way that L^AT_EX and Tib can be applied in either order).

This approach could be the major element of a *modus vivendi* between L^AT_EX3 and the “main 4”. Table 2 shows some delimiters that might be suitable.

A *modus vivendi* would also need to incorporate an approach to the “root file and \include-ed files” situation. Although I don’t have any specific suggestions at this stage, I speculate that support for this feature might be obtained by reference to the bibliographic software’s support for analogous features in wordprocessors (e.g., WordPerfect’s “master document and subdocument” scheme, and Microsoft Word’s “include” scheme).

3.1.3 Preferred interface

The suggestions in Table 2 are intended as part of a *modus vivendi* between L^AT_EX3 and the *current* versions of the “main 4”. Although the general approach is the same, the details differ from product to product.

It would be open to L^AT_EX-ers to decide on a preferred interface, and to inform the vendors of

Software	Citation delimiters	Notes
L ^A T _E X 2.09 with BIBT _E X	\cite{ }	
Tib	[. .]	The delimiters <. .> are used in some circumstances
EndNote	Default: []	You can tell EndNote to look for alternative 1-character delimiters (e.g., < >)
Library Master	Not known	I understand that a facility for "given the in-text citations, compile a reference-list" is in preparation
Papyrus	Default: %% %%	You can tell Papyrus to look for alternative delimiters (but "start delimiter" must be the same as "end delimiter")
ProCite	Default: ()	You can tell ProCite to look for [] rather than for ()
Reference Manager	Default: { }	You can tell Reference Manager to look for alternative delimiters. "Start delimiter" and "end delimiter" can each have up to 7 characters.

Table 1: Citation-delimiters: defaults and alternatives

Biblio. software	Tell bib. software	Tell L ^A T _E X3	Notes
EndNote	Delimiters are < ... >	< ... > is equivalent to 2.09's \verb+...+	
Papyrus	Delimiter is "	" ... " is equivalent to 2.09's \verb+...+	
ProCite			No obvious alternative to "always apply ProCite before L ^A T _E X"
Reference Manager	Delimiters are \bsoft{ ... } and }	\bsoft{ ... } is equiv. to 2.09's \verb+...+	

Note: Clearly the default Papyrus and Reference Manager delimiters (see Table 1) must be changed if the end-user is to have the option of applying L^AT_EX without having previously dealt with citations, etc. However, the Papyrus and Reference Manager keys are not liable to contain a # character. Hence, it is not crucial whether Papyrus and Reference Manager keys are "hidden" from L^AT_EX.

Table 2: Choice of delimiters for *modus vivendi*?

their preference in the hope that it may be possible to implement the approach more consistently at some time in the future. We wouldn't lose anything by asking!

For example, if the preferred interface involved `\bsoft{key}` (as shown in Table 2 for Reference Manager), it would be open to us to ask the other vendors to relax their rules on citation-delimiters so that future versions of the "main 5" will all accept `\bsoft{key}`. If we were lucky enough to get the vendors' agreement, we might be able to produce notes about "using proprietary bibliographic software with L^AT_EX" that would appear simpler to the end-user than Table 2.

Note It might be possible to have a *modus vivendi* (e.g., with Reference Manager) involving `\verb+key+`, rather than having an additional command such as `\bsoft` (which would, in any case, be implemented in much the same way as `\verb`). The bibliographic software will probably ignore things within `\verb+ +` and `+ +` that don't look like citation keys. Nevertheless, I would be inclined to introduce an extra command (e.g., `\bsoft`) so that .tex files can be "marked up logically" to distinguish between:

- delimiters for a key that is intended for processing by bibliographic software
- delimiters for text that is intended to appear in a `typewriter` font in the final document.

3.1.4 Hybrid approaches

One can envisage schemes that embed a proprietary bibliographic system's mechanism for dealing with citations and reference-lists within L^AT_EX's mechanism (or *vice versa*). Examples might include:

- telling Papyrus to use `!!` as its delimiter, and putting the Papyrus citation markers inside a L^AT_EX `\cite` command, thus `\cite{!! ... !!}`.¹²
- trying to get proprietary bibliographic software to read an .aux file, and write a .bb1 file, as BIBTEX does. (Perhaps this could be done by a shell script which invokes the proprietary software in a suitable way.)

Generally, I fear that such hybrid schemes may lead to confusion, and I would not be inclined to pursue them:

- Anyone constructing a hybrid scheme will have to be very careful about "which software is in charge when" (e.g., whether citation numbers are incremented by L^AT_EX, by the proprietary system, or by "one shadowing the other").

¹² Bernard J. Treves Brown, of Manchester University, is experimenting with this technique.

The hybrid scheme will need maintenance (e.g., someone will need to verify that the scheme still works with each new release of the proprietary system). There may be three lots of documentation for the end-user to study: that about L^AT_EX3, that about the proprietary system, and that about the hybrid scheme's subtle combination of elements of both. If anything goes wrong, it may be in "a grey area", which is neither the responsibility of the L^AT_EX3 project, nor the responsibility of the bibliographic software vendor.

- The proprietary systems seem more akin to Tib than to BIBTEX. To try and force them into the BIBTEX stereotype when they are not designed to work like BIBTEX seems like "asking for trouble". I doubt whether the T_EX community has the resources to produce interfaces that "make proprietary systems work like BIBTEX", and I doubt whether the vendors have the inclination to commit such resources.

My instinct is that it would be better to have a simple interface (e.g., conventions such as those outlined in Table 2), so as to put the end-user in a situation where responsibilities are clear:

- typesetting is the responsibility of L^AT_EX3
- bibliography-generation is the responsibility of the bibliographic software.

Hence, if using a proprietary bibliographic system, the end-user should ignore the L^AT_EX3 manual's descriptions of commands to support the DIY-er (i.e., ignore the L^AT_EX commands envisaged in section 2), and ignore anything that is provided to support the BIBTEX-er.

- The proprietary system will be "in charge" of bibliography generation. The method used will be that envisaged by the vendor, and documented in the vendor's manual: if it's good, the vendor will get the credit; if it's bad, the vendor will get the blame.
- The delimiters in the .tex file will be delimiters for the proprietary system (chosen, if possible, in such a way that the .tex file is acceptable to L^AT_EX even before processing by the proprietary system). They might be as shown in Table 2. The "keys", etc., inside the delimiters will follow the rules given in the vendor's manual (*not* the rules given in the L^AT_EX3 manual about keys that the DIY-er can use).
- The proprietary system will be "told to produce T_EX output". How good or bad they are at this will be the responsibility of the proprietary system (although interested L^AT_EX-ers might advise the vendors about what is required).

Overall, the end-user will get in-text citations filled in, and reference-lists generated, in the standard way that is described in the manual that describes the proprietary system. If this standard way does not suit a L^AT_EX-er's requirements, it may be better for him/her to seek alternative bibliography-formatting software rather than spending time trying to circumvent the problems.

Of course, if people want to put effort into developing hybrid schemes, and happen to get good *modus vivendi* between L^AT_EX and proprietary bibliographic systems, I would be delighted to find that my instinct is wrong!

3.1.5 The user's choice

Given some *modus vivendi*, end-users would be able to make their own assessments of which bibliographic software suits their needs.

- Cost is obviously a factor.
- An end-user who wants software that has been designed specifically for use in conjunction with L^AT_EX, will probably be inclined to choose BIB^AT_EX or Tib.
- BIB^AT_EX's approach makes good use of disk-space. A `.bb1` file will be smaller than "near-duplicates of `.tex` files".
- An end-user who wants ready-made methods of downloading information from commercial bibliographic databases, library catalogues, etc., will probably favour one of the proprietary programs. The proprietary systems also offer database administration and searching facilities.
- Anyone who does not have the time and patience to deduce (from a proprietary system's wordprocessor-oriented documentation/menus) what the L^AT_EX-er should do might prefer to wait until someone else has deduced what is required, and has documented the tricks involved.
- The end-user's choice may be constrained by the platform on which they are using L^AT_EX (e.g., they may need bibliographic software for a Unix system).
- Wordprocessor-oriented systems may not support typesetting subtleties to the degree that L^AT_EX-ers would like.
- Support (or lack of it) for non-English languages may be another factor.¹³

¹³ Decisions may be needed about whether to try using a proprietary system's support for diacritics, in the hope of being able to share a database with colleagues who use wordprocessors. The alternative would be to have database entries that use TeX encoding for diacritics.

- End-users may be constrained to use the same system as other people in their research group (e.g., so that the group can share databases).

It is unlikely that anyone will find bibliographic software that is perfect for their needs. However, people are more likely to find something that suits them if they have a choice than if they have no choice.

3.2 OR for L^AT_EX3

Given the situation outlined in section 3.1, I suggest the following as the OR for L^AT_EX3's relationship with bibliography-formatting software:

- As far as practicable, L^AT_EX3 should be neutral towards the end-user's choice of bibliography-formatting software. Ideally, people should be able to choose typesetting software for typesetting reasons, and bibliographic software for bibliographic reasons — their choice of typesetting software should not restrict their choice of bibliographic software.
- Hence, a *modus vivendi* between L^AT_EX3 and each of the "main 5" should be thought up, tested and documented.¹⁴
- There might be "a preferred interface" between L^AT_EX3 and proprietary bibliographic software. If vendors can be persuaded to support this interface, L^AT_EX-ers will get a consistent interface to proprietary bibliographic software. If not, things will stay inconsistent (e.g., as shown in Table 2).
- In line with the neutrality suggested above, BIB^AT_EX will continue to be supported, but L^AT_EX3 documentation will not be particularly pro-BIB^AT_EX. It is desirable that `.bst` files should be updated so that BIB^AT_EX produces L^AT_EX3 commands (designed to satisfy the requirements listed in section 2) rather than L^AT_EX 2.09 commands.

4 Miscellaneous

4.1 "Local names" for keys

If you are "doing it yourself", choice of keys is unlikely to be a problem. For example, you could equally well use `lampart-86` or `latexbook` as a key for the L^AT_EX manual. There is no particular need for consistency from one document to another: you

¹⁴ The *modus vivendi* might be along the lines shown in Table 2, or might be something else that emerges from practical experience. It doesn't matter much whether the documentation is provided by the L^AT_EX3 project or by the bibliography software vendor, as long as someone provides it!

can use `lamport-86` as the key in one document, and use `latexbook` as the key in another.

However, if you have a large bibliographic database (perhaps shared with a group of colleagues), it may be impracticable to keep track of keys assigned on an *ad hoc* basis, and difficult to guarantee that keys will stay unique whenever a new item is added to the database.

Moreover, a `.tex` file to be `\input` may contain bibliographic details and L^AT_EX commands that are generated automatically by bibliographic software (even though L^AT_EX will have no way of distinguishing the file from a one typed in by a DIYer). Such bibliographic software might be programmed to assign keys automatically. For example, software might write a `.tex` file that contains L^AT_EX 2.09 `\bibitem` commands, with keys of the form `lamport-86` constructed automatically from two fields in the database.¹⁵

To help cater for such situations, it might be useful if L^AT_EX3 allowed "local names" for keys, i.e., some mechanism whereby an author could declare (e.g., in a document's root file) that, for the duration of a document, a particular "informal key" (to be used in in-text citation commands) should be treated as a synonym for a "formal key" (which appears in an entry in an automatically generated reference-list). For example, it might be useful to be able to declare that `latexbook` can be used as a "local name" for `lamport-86`.

4.2 Reference-lists that are also indexes

Another requirement that needs to be borne in mind is for reference-lists which, as well as providing bibliographic details of sources, provide an index to the pages on which the sources are cited:

- in mainstream academic publications, the requirement will probably be for a "combined list of references and author index" [5, pp. 198 & 258]
- in law books, the requirement is usually for "front matter" units such as "table of cases", "table of statutes" and "table of treaties". In a typical "table of cases", each entry tells the reader

¹⁵ Some thought would need giving to any method of assigning keys automatically. If a bibliographic database is continually growing, there may be no guarantee that keys of the form `lamport-86` will stay unique when new items are added to the database. It might be safer to assign less memorable keys that can be guaranteed to stay distinct, e.g., the "record number" in the database, or a book's ISBN

- where further details of the case can be found (e.g., the relevant law report)
- which pages in the book's main text mention the case.

The other types of tables are analogous.

A Some suppliers of mainstream bibliographic software

EndNote Niles and Associates. 2000 Hearst St, Berkeley, CA 94709, USA. E-mail: nilesinc@well.sf.ca.us.

Library Master Balboa Software, P.O. Box 3145, Station D, Willowdale, Ontario, M2R 3G5, Canada. E-mail: hahne@epas.utoronto.ca.

Papyrus Research Software Design, 2718 S. W. Kelly St, Suite 181, Portland, Oregon 97201, USA. E-mail: RSD@applelink.apple.com.

ProCite Personal Bibliographic Software, P.O. Box 4250, Ann Arbor, Michigan 48106. E-mail: sales@pbsinc.com or support@pbsinc.com.

Reference Manager Research Information Systems, Camino Corporate Center, 2355 Camino Vida Roble, Carlsbad, CA 92009, USA. E-mail: sales@ris.risinc.com.

B E-mail discussion lists about bibliographic software

The `bibsoft` list provides a forum for general discussion of personal bibliographic database management systems. You can subscribe by sending a one-line e-mail message of the form

```
subscribe bibsoft last-name,first-name
to
listserv@indycms.iupui.edu.
```

There are also specific discussion lists for EndNote, Library Master and ProCite. See [17].

In the United Kingdom, there is a discussion list for Higher Education institutions that have taken up the CHEST Papyrus deal. You can subscribe by sending a one-line message of the form

```
subscribe
chest-papyrus first-name last-name
to
mailbase@mailbase.ac.uk.
```

References

- [1] Leslie Lamport. *L^AT_EX: A Document Preparation System*. Addison-Wesley, 1986.
- [2] *The Bluebook: A Uniform System of Citation*. Harvard Law Review Association, 15th edition, 1991. Obtainable from: Harvard Law Review Association, 1511 Massachusetts Avenue, Cambridge, Massachusetts 02138.

- [3] Documentation — bibliographic references — content, form and structure. ISO 690, International Organization for Standardization, 1987.
- [4] Frank G. Bennett, Jr. LexiT_EX: a L_AT_EX macro package for lawyers. Document deposited in electronic archives, 1993.
- [5] Judith Butcher. *Copy-editing*. Cambridge University Press, 3rd edition, 1992.
- [6] *Publication Manual of the American Psychological Association*. American Psychological Association, 3rd edition, 1983. Obtainable from: American Psychological Association, P. O. Box 2710, Hyattsville, MD 20784.
- [7] *The Chicago Manual of Style*. University of Chicago Press, 13th edition, 1982.
- [8] Janet S. Dodd. *The ACS Style Guide*. American Chemical Society, 1986.
- [9] *MHRA Style Book*. Modern Humanities Research Association, 4th edition, 1991.
- [10] Joseph Gibaldi and Walter S. Achert, editors. *MLA Handbook for Writers of Research Papers*. Modern Language Association of America, 3rd edition, 1988.
- [11] International Committee of Medical Journal Editors. Uniform requirements for manuscripts submitted to biomedical journals. *British Medical Journal*, 302:340–341, February 1991. Note: This article was also published in the *New England Journal of Medicine* (7th Feb. 1991). It specifies the “Vancouver style” for manuscript-preparation, which is accepted by over 400 journals.
- [12] Citing publications by bibliographic references. BS 5605, British Standards Institution, 1978.
- [13] References to published materials. BS 1629, British Standards Institution, 1989.
- [14] Citation of unpublished documents. BS 6371, British Standards Institution, 1983.
- [15] Oren Patashnik. BibTeXing. Document deposited in electronic archives, January 1988.
- [16] James C. Alexander. Tib: A T_EX bibliographic preprocessor. Document deposited in electronic archives, 1989.
- [17] Sue Stigleman. Bibliography formatting software: an update. *Database*, February 1993.

◊ David Rhead
 Cripps Computing Centre
 University of Nottingham
 University Park
 Nottingham NG7 2RD England,
 U.K.
David_Rhead@vme.ccc.nottingham.ac.uk

Relative moves in L_AT_EX pictures

Richard Bland

1 Introduction

In this note I hope to do three things:

1. Make a number of observations about why picture-drawing in L_AT_EX, as described by Lamport, is so difficult and unpleasant.
2. Put forward a suggestion for a very simple mechanism to overcome at least some of these difficulties.
3. Show one way of implementing this suggestion, using the Unix utility `m4`. This particular implementation is presented only to demonstrate the simplicity of the underlying mechanism: no claim is made that it is an optimal implementation.

2 An example

Consider the simple picture in Figure 1. As is obvious, this picture has no meaning: it is just a collection of graphic elements such as labelled shapes, text strings, lines and arrows: but it does exemplify the kind of output which many users have in mind when they set out to draw a picture in L_AT_EX. Such users want some form of diagrammatic representation in which different shapes are used to represent types of entity, lines and arrows are used to connect the entities, and labelling is used to give some domain-specific meaning. Often these pictures are conceptually quite simple.

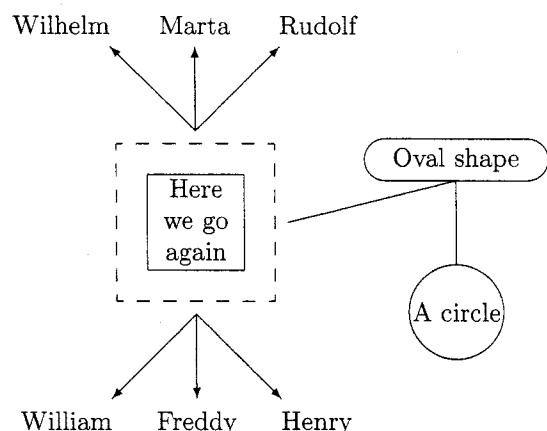


Figure 1: A L_AT_EX picture

How does one produce pictures like this? Many people would suggest using an interactive drawing tool (on some suitable hardware) to produce an intermediate file which can be incorporated into the L^AT_EX source of a document (or added at some appropriate point downstream). I've never found this an agreeable approach, for two main reasons: first, as an occasional user I find it hard to come to grips with the supposedly intuitively-obvious interfaces which these tools present. After a certain point in one's career the fun of learning *another* system begins to diminish: the busy user who has learned one set of syntactic and semantic ideas (like those of L^AT_EX) would like to get results from those ideas rather than adding a new set. Second, in using an interactive drawing tool one often abandons or jeopardises some of the main reasons for using a markup system like L^AT_EX in the first place. These are, of course, portability, device-independence, and the ability to manipulate the source indefinitely with any number of the myriad tools which handle ASCII text. This last point is particularly important: because L^AT_EX source is just a character file it can be edited, cut, pasted, searched, burgled, extended, all without limit. This is certainly not the case with the behind-the-scenes formats of many drawing packages.

These considerations suggest that there are good reasons for trying to produce pictures with the L^AT_EX tools described by Lamport.

Now consider the commands which produced Figure 1. Slightly edited, they are as follows:

```
\begin{figure}[htb]
\setlength{\unitlength}{1pt}
\centering
\begin{picture}(216,216)
\put(48,78){\dashbox{5}(60,60){
\begin{tabular}{|c|}\hline Here \\ we go \\ again \\ \hline
\end{tabular}}}
\put(78,73){\vector(1,-1){32}}
\put(110,36){\makebox(0,0)[t1]{Henry}}
\put(78,73){\vector(-1,-1){32}}
\put(46,36){\makebox(0,0)[tr]{William}}
\put(78,73){\vector(0,-1){32}}
\put(78,36){\makebox(0,0)[t]{Freddy}}
\put(78,143){\vector(1,1){32}}
\put(110,180){\makebox(0,0)[bl]{Rudolf}}
\put(78,143){\vector(-1,1){32}}
\put(46,180){\makebox(0,0)[br]{Wilhelm}}
\put(78,143){\vector(0,1){32}}
\put(78,180){\makebox(0,0)[b]{Marta}}
\put(113,108){\line(4,1){64}}
\put(177,132){\oval(70,16)}\put(177,132)
```

```
{\makebox(0,0){Oval shape}}
\put(177,124){\line(0,-1){32}}
\put(177,74){\circle{36}}\put(177,74)
{\makebox(0,0){A circle}}
\end{picture}
\caption{A LATEX picture
\label{exampfig}}
\end{figure}
```

What can we say about this? Well, readers of *TUGboat* presumably have strong stomachs, but even those who read *The T_EXbook* for fun will surely realise that these instructions are awful. Users who set out to produce pictures using this sort of apparatus will very soon become discouraged. In the next section we try to analyse the problem with these instructions.

3 The difficulties

Looking at the code above, we can draw three main conclusions. First, the *syntax* of the instructions is very complicated and very hard to remember, making the instructions extremely hard to write unless one has a model immediately to hand. Also, there seem to be inconsistencies. For example, the parameters for `\oval` are in round brackets while the parameter for `\circle` is in curly brackets, although they are semantically equivalent — in each case the parameter(s) give the size of the shape to be drawn.

Second, the code is stuffed full of literal numeric *constants*. This immediately makes users with a programming background uneasy. After all, one of the things we are always told (or are telling others) is *not* to use constants. Because they convey no semantic information they make code hard to read: because we have to change *every semantically-equivalent instance* of a constant in order to edit code, the code is hard to change without making mistakes. In this case there is the additional difficulty that we suspect that the author of the code must have sweated blood in order to work out what all these constants ought to be in the first place: in our mind's eye is an image of Lamport crouched over his quadrille paper, cursing.

Third, the picture is composed in terms of *absolute* positions rather than *relative* positions. We realise that if we were to try to move the components of the picture in relation to one another, it would be very hard to do so by editing the absolute positions (the pairs of values in all the `\put` instructions).

4 Existing remedies

Some of these problems can, of course, be dealt with by sensible use of existing L^AT_EX facilities. We can make the literal constants into symbolic constants (using `\newcommand`), we can tinker a bit

with the syntax of repeated constructions (also using `\newcommand`) and we can modularise the picture (using nested `\picture` environments) and move the modules in relation to one another using offsets.

These solutions only go so far, however. Defining symbolic constants is fine: but one soon needs a facility for *arithmetic* in these definitions, which L^AT_EX lacks. For example, if one has a symbolic constant for the width of a box, you may need one for half the width as well. There's no easy way (that I know of) of defining one constant as a function of a previously-defined constant, so you must define them both literally: once again this makes changes difficult. Also, the scope for simplifying the syntax is quite limited because each command is still quite complicated semantically: 'at the point (177,132) draw an oval of size (70,16) and within it centre the string "Oval shape"' could certainly be more simply expressed, but not very much more simply. Finally, the method of modularising the picture by nesting `\picture` environments is useful, but has to be set out *very* carefully if the human reader is not to become hopelessly lost about the scope of the environments and hence about the offset to be applied to any particular position.

5 New remedies

There are two remedies which I wish to propose: one minor and one major. The minor one is to *make it easier to define symbolic constants as functions of other constants*. The major one is to *remove the 'position' information from the drawing commands*. The minor remedy really needs no further discussion at this stage: the only question to be settled is the method of implementation. The idea of taking the 'position' information out of the drawing commands is more complex.

The basic notion is to introduce the idea of a *current position* at which the next drawing action is to be done. Using macros, we re-package all the drawing operations which we wish to use, so as to

- Make them all draw at the current position. The re-packaged commands can now be simpler, because they no longer need position parameters.
- Give each of them a defined effect on the current position. For entity-representing shapes (boxes, ovals, circles, strings) the command will leave the current position where it is: for connectors (lines, arrows) the command will start drawing the line or arrow at the current position and end by moving the current position to the other end of the line or arrow.

Obviously we also need to add new commands to manipulate the current position: these will include

- An absolute `jump`, to move the current position to some new point.
- A relative `move`, to move the current position by an offset (it turns out to be convenient to have a family of these: four single-parameter moves, `up`, `down`, `left` and `right`, as well as a full two-parameter move).
- A method of 'remembering' the current position, and of resetting the current position to some remembered point.

One way of thinking about these commands, and of implementing the 'remembering' mechanism, is that we have introduced *position variables*. There's a behind-the-scenes position variable, the current position, which is global to all commands, and as many explicitly-named position variables as the user wishes. The only defined operations (so far, anyway) are those of assigning from a user-declared position variable to the current position and vice-versa.

The payoff turns out to be quite considerable. Our repackaged commands can be much simpler (for example, the command for an oval has three parameters instead of five). More importantly, the whole business of absolute positions (which gives the user so much difficulty) has now disappeared and been replaced by a much more natural idea of drawing one thing, moving relative to that thing and drawing another thing. This is what we do when we sketch naturally, on the backs of envelopes: we certainly don't work as Lamport recommends, "first pick[ing] the slopes of all lines, then ... calculat[ing] the position of each object before drawing it on the graph paper" ([3], page 110). The naturalness of this new approach is particularly obvious when the graph of the entities and connectives is a tree: in this case the new approach makes the picture simple to draw and very easy to change.

An example is needed here, but before we can present one, an implementation is needed. This is discussed in the next section.

6 Implementation

No doubt in an ideal world I would now present an implementation in T_EX macros. In fact I shall not do this. For many years I have used the Unix macroprocessor `m4` ([1, 2]), which comes free with Unix and is in the public domain for MS-DOS. It has the facilities which we need (including arithmetic) and I know how to use it. Unfortunately I don't know how to write T_EX macros.

Is this a problem? I believe not. My intention in this note is not to advertise a product but to discuss an approach. Although I shall of course be happy to share my few lines of code with anyone who wants them, my purpose here is to demonstrate that a particular approach can be made to work very easily and can greatly simplify a particular task. I hope that readers will be stimulated to suggest better or fuller implementations of the idea.

Using `m4` means that the source file (a mixture of `m4` statements and `LATEX` statements) must be run through `m4` before processing by `LATEX`, but this step is easily arranged and has a negligible penalty in processing time.

In the following account I shall not show the full details of the implementation in `m4` (although this is only a few dozen lines): I shall concentrate instead on explaining the commands which a user would need to know in order to draw the picture of Figure 1. In this account, I shall show macro names defined by me in capital letters, for clarity, and will follow the `m4` convention of describing macro parameters as `$1`, `$2`, etc., rather than the `TEX` convention of `#1`, `#2`, etc. I shall not attempt to give a rigorous account of `m4`, which is completely defined in [2]. As a working label, I'll refer to the set of macros written by me as the Macro Library for `LATEX` Pictures MLLP, although this perhaps conveys an undue air of importance for a very few lines of code.

We begin by noting that in `m4` we define a macro using the `define` macro, which takes two arguments, as in `define(HEIGHT,216)` which sets up the symbolic constant `HEIGHT` to be 216. This is the intended height of the picture—216 points (which is about three inches). We also define other useful dimensions for the picture, whose meanings should be obvious: `WIDTH`, `BOXHEIGHT`, `BOXWIDTH`, `CIRCLEDIAM`, `OVALHEIGHT`, `OVALWIDTH` and `XARROWLEN`. This is done in the same way as for `HEIGHT` (but with different values, of course, the exact values of which aren't important for the exposition). We also define a useful quantity `SEPARATION`, which is defined as 5 (points) and is used as a general spacing parameter in the picture.

We can now write

```
\begin{picture}(WIDTH,HEIGHT)
```

as the start of the environment. The first thing we should like to do is draw the most significant element of the picture, the dashed box, slightly to the left of the midpoint of the picture. We can calculate this using the `m4` built-in macro `eval`, which takes a conventionally-formed arithmetic expression as its argument and replaces it by an integer, the result of evaluating the expression. Before we do the sum,

we first set the constant `LEFTTABIT` to be (say) 20 points, to move the box off-centre, and note that boxes are usually drawn with their bottom-left corner as the reference point: this means that we must jump to a point half a box-width to the left of, and half a box-height below, the chosen centre point of the box.¹ All of this is rather a mouthful. However, we can now set the current point to its starting position:

```
JUMP(eval((WIDTH-BOXHEIGHT)/2-LEFTTABIT),
      eval((HEIGHT-BOXWIDTH)/2))
```

Now the dashed box. MLLP includes a three-parameter macro which draws a dashed box of size `$1` by `$2`, with the (optional) `$3` centered within it. This operation does not affect the current position. We can now write

```
DASHBOX(BOXHEIGHT,BOXWIDTH,
        '\begin{tabular}{|c|} \hline Here \\
        we go \\ again \\ \hline
        \end{tabular}')
```

demonstrating in rather a flashy way that the third parameter of `DASHBOX`, the object to be centered within it, can be a complicated `LATEX` object. This is not exclusive to `DASHBOX`: the other macros in the set can also have complicated picture objects as parameters. Notice that to be on the safe side the parameter is wrapped in paired left and right single-quotes: this protects it from any unwanted processing by `m4`.

We now wish (say) to draw the cluster of arrows, and the associated strings, under the box. First we move the current point from the bottom-left corner of the box: in doing so we use another macro from MLLP, `HALF`, whose effect is obvious. Once we've arrived, we want to remember this position because it will be the base for three arrows, so we shall use the MLLP macro `SET` to hold the position.

```
RIGHT(HALF(BOXWIDTH))
DOWN(SEPARATION)
SET('arrowbase1')
```

The string `arrowbase1` is the name of an MLLP position variable, as described above. It can be any identifier which won't interfere with `LATEX` or `m4`. When acting as the parameter to `SET`, it needs to be in paired left and right single-quotes: this is for reasons internal to `m4`.

Now we draw an arrow and the string at the end of it. MLLP includes a three-parameter macro `ARROW`, which is just a packaging of Lampert's `vector`. The first two parameters give the slope and the third the length, just as described by Lampert

¹ Alternatively, one could make the centre of the box the reference point: but if you work it through this doesn't simplify things.

([3], page 106). The arrow is drawn from the current point and the current point is moved to the head of the arrow. There are two variants, ARROWUP and ARROWDOWN, which move the current point slightly away from the end of the arrow, either up or down: the length of the move is given by SEPARATION. The string at the end of the arrow is written using PUT, which is just a packaging of Lamport's put. The first argument is the string to be written. The (optional) second argument gives the relative position of the string with respect to the current point. The default is to centre the string round the current point, horizontally and vertically, but this can be changed by using the second parameter. Just as in Lamport, \$2 can be 0, 1 or 2 of the letters t, b, l or r. These determine where the current point is with reference to the text. For example, t1 means that the current point is at the top left of the text. PUT does not move the current point. So:

```
ARROWDOWN(1,-1,XARROWLEN)
PUT(Henry,t1)
```

The remaining two arrows in the cluster can be drawn easily once we note that JUMP will accept a position variable as its (single) argument. This of course resets the current point to the position stored in the position variable. Off we go:

```
JUMP(arrowbase1)
ARROWDOWN(-1,-1,XARROWLEN)
PUT(William,tr)
JUMP(arrowbase1)
ARROWDOWN(0,-1,XARROWLEN)
PUT(Freddy,t)
```

Drawing the top set of arrows doesn't require any new techniques: we move to the top of the box, establish a new arrow-base and draw the cluster.

```
JUMP(arrowbase1)
UP(eval(BOXWIDTH+2*SEPARATION))
SET('arrowbase2')
ARROWUP(1,1,XARROWLEN)
PUT(Rudolf,b1)
JUMP(arrowbase2)
ARROWUP(-1,1,XARROWLEN)
PUT(Wilhelm,br)
JUMP(arrowbase2)
ARROWUP(0,1,XARROWLEN)
PUT(Marta,b)
```

Given that our aim here is not to produce a reference manual for MLLP, or anything like it, it will perhaps be enough to leave the reader to infer from the code the properties of the remaining macros to be used, LINE, VLINE, OVAL and CIRCLE, given the information that OVAL and CIRCLE are drawn centred on the current point. We first move round to

the right-hand side of the box, then draw the rest of the picture:

```
JUMP(arrowbase2)
RIGHT(eval(BOXHEIGHT/2+SEPARATION))
DOWN(eval(BOXWIDTH/2+SEPARATION))
LINE(4,1,eval(XARROWLEN*2))
%
UP(HALF(OVALWIDTH))
OVAL(OVALHEIGHT,OVALWIDTH,Oval shape)
DOWN(HALF(OVALWIDTH))
%
VLINE(-XARROWLEN)
DOWN(HALF(CIRCLEDIAM))
CIRCLE(CIRCLEDIAM,A circle)
\end{picture}
```

7 Conclusion

This note has attempted to identify a number of factors which make L^AT_EX picture-drawing a frustrating and error-prone business, and to suggest a simple approach which ameliorates those difficulties, and which can be implemented without much difficulty. An example has been presented: the code of this example is, I believe, strikingly easier to understand and to change than the original L^AT_EX code. Practical experience with a number of drawings has reinforced the belief that the approach presented here is simple and effective.

No claim is made that the implementation of these ideas in m4 is particularly elegant, or that the MLLP set of macros (which is larger than that shown above) is optimal or complete. I have, however, found it to be effective for my purposes. I should be very grateful for suggestions or comments on these points.

References

- [1] Brian W Kernighan and P J Plauger. *Software Tools*. Addison-Wesley, Reading, Mass, 1976.
- [2] Brian W Kernighan and Dennis M Ritchie. The m4 macro processor. Technical report, Bell Laboratories, Murray Hill, New Jersey, 1977.
- [3] Leslie Lamport. *L^AT_EX: a document preparation system*. Addison-Wesley, Reading, Mass, 1986.

◊ Richard Bland
Computing Science and
Mathematics
University of Stirling
Stirling FK9 4LA
Scotland

Calendar

1993

- Dec 2 **TeX-Stammtisch** at the Universität Bremen, Germany. For information, contact Martin Schröder (115d@alf.zfn.uni-bremen.de; telephone 0421/628813).
- Dec 20 **TeX-Stammtisch** in Bonn, Germany. For information, contact Herbert Framke (Herbert_Framke@BN.MAUS.DE; telephone 02241 400018).
- Dec 21 **TeX-Stammtisch** in Duisburg, Germany. For information, contact Friedhelm Sowa (tex@ze8.rz.uni-duesseldorf.de; telephone 0211/311 3913).
- Dec 22 **TeX-Stammtisch**, Hamburg, Germany. For information, contact Reinhard Zierke (zierke@informatik.uni-hamburg.de; telephone (040) 54715-295).

1994

- Jan 6–9 Linguistic Society of America, Annual Meeting, Sheraton Boston Hotel, Boston, Massachusetts. (**TeX** and linguistics poster session, Friday, January 7.) For information, contact the LSA office, Washington, DC (202 834 1714, zzlsa@gallua.bitnet).
- Feb 1 **TUG Annual Meeting**, titles and outlines due. Send proposals to tug94@tug.org.

TUG Courses, Santa Barbara, California
(For information, contact john@tug.org.)

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

- Feb 15 **TUGboat Volume 15, 2nd regular issue:**
Deadline for receipt of technical manuscripts.
- Feb 16–18 DANTE'94, 18th general meeting, Münster, Germany. For information, contact Wolfgang Kaspar (kaspar@dmswwu1a.uni-muenster.de).
- 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.
- Apr 11–15 Four conferences, Darmstadt, Germany:
 - EP94, Electronic Publishing, 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
- Spring NTG 13th Meeting, “(IA)**TeX**, METAFONT, and tools education”, Groningen, at RUG. For information, contact Gerard van Nes (vannes@ecn.nl).
- May 23 **TUGboat Volume 15, 2nd regular issue:**
Mailing date (tentative).
- Jul 6–8 C.N.E.D. 94: 3^{ème} Colloque National sur l'Ecrit et le Document, Rouen, France. For information, contact Jacques Labiche (labiche@la3i.univ-rouen.fr).

- 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.)
- Jul 31–
Aug 4 **TUG 15th Annual Meeting,**
Santa Barbara, California.
For information, contact
Debbie Ceder (tug94@tug.org).
- Aug 17 **TUGboat Volume 14,**
3rd regular issue:
Deadline for receipt of *technical* manuscripts (tentative).
- Sep 14 **TUGboat Volume 14,**
3rd regular issue:
Deadline for receipt of news items, reports (tentative).
- Nov 23 **TUGboat Volume 14,**
3rd regular issue:
Mailing date (tentative).

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.

Production Notes

Barbara Beeton

Input and input processing

Electronic input for articles in this issue was received by e-mail and on diskette, and was also retrieved from remote sites by anonymous ftp.

In addition to text and various code files processable directly by TeX, the input to this issue includes several encapsulated PostScript files. More than 60 files were required to generate the final copy; over 60 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 *TUGboat* 14, no. 2, p. 100. The TUG office will provide copies of the macros on diskette to authors who have no electronic access.

Almost 75% of the articles in this issue are in LATEX, accounting for about 85% of the pages.

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.

The following articles were prepared using the plain-based *tugboat.sty*:

- the NTS update, Philip Taylor, page 381
- Two extensions to GNU Emacs, Thomas Becker, page 382
- Icons for TeX and METAFONT, Donald Knuth, page 387
- A tough table, Kevin Carmody, page 420
- the TUG calendar, page 438.
- these Production notes.
- “Coming next issue”.

The index to the article by Nelson Beebe (page 395) required processing with Makeindex. A 1991 version of this program failed miserably under both VMS and UNIX, first looping and then terminating with a segmentation fault/invalid access. Thanks to George Greenwade a more recent version was found on CTAN and installed under the pressure of the deadline; this version did work properly. *bibclean*, the package described in this article, will be available in the archives, as will the article; anyone intending to TeX the article is advised to make sure that their copy of Makeindex is up to date.

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.

Coming Next Issue

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. [Delayed for technical reasons.]

Slanted lines with controlled thickness

David Salomon describes a method that makes it possible to typeset slanted lines of any thickness by typesetting a rule, shifting it in the desired direction, and repeating the process a number of times.

New techniques in METAFONT

Certain geometrical problems that arise very often in glyph design are not directly solvable by METAFONT's plain macros. Yannis Haralambous presents two such problems and solutions for them, along with a discussion of an approach that, although geometrically correct, does not work in real-world METAFONT practice and should be avoided.

Book reviews

Writing new books about TeX and related subjects has become a cottage industry. Reviews of the following are expected:

- Stephan von Bechtolsheim, *TeX in Practice*
- Malcolm Clark, *A Plain TeX Primer*
- George Grätzer, *Math into TeX*
- and possibly others ...

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



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

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

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

Index of Advertisers

- 447 American Mathematical Society
- 447 ArborText
- Cover 3 Blue Sky Research
- 446 Ed Baker Technical Services
- 448, 449 Kinch Computer Company
- 440 Micro Programs, Inc.
- 450 Y&Y

TEX USERS GROUP

Complete and return this form with payment to:

TEX Users Group
Membership Department
P.O. Box 869
Santa Barbara, CA 93102 USA
Telephone: (805) 963-1338
FAX: (805) 963-8358
Email: tug@tug.org

Membership is effective from January 1 to December 31 and includes subscriptions to *TUGboat*, *The Communications of the TEX Users Group* and the TUG newsletter, *TEX and TUG NEWS*. Members who join after January 1 will receive all issues published that calendar year.

For more information ...

Whether or not you join TUG now, feel free to return this form to request more information. Be sure to include your name and address in the spaces provided to the right.

Check all items you wish to receive below:

- Institutional membership information
- Course and meeting information
- Advertising rates
- Products/publications catalogue
- Public domain software catalogue

Individual Membership Application

Name _____

Institutional affiliation, if any _____

Position _____

Address (business or home (circle one)) _____

City _____ Province/State _____

Country _____ Postal Code _____

Telephone _____ FAX _____

Email address _____

I am also a member of the following other TEX organizations:

Specific applications or reasons for interest in TEX:

There are two types of TUG members: regular members, who pay annual dues of \$60; and full-time student members, whose annual dues are \$30. Students must include verification of student status with their applications.

Please indicate the type of membership for which you are applying:

- Regular at \$60 Full-time student at \$30

Amount enclosed for 1994 membership: \$ _____

Check/money order payable to TEX Users Group enclosed
(checks in US dollars must be drawn on a US bank; checks in other currencies are acceptable, drawn on an appropriate bank)

Bank transfer:

TEX Users Group, Account #1558-816,
Santa Barbara Bank and Trust, 20 East Carrillo Street,
Santa Barbara, CA 93101 USA

your bank _____

ref # _____

Charge to MasterCard/VISA

Card # _____ Exp. date _____

Signature _____

TEX USERS GROUP

Complete and return this form with payment to:

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

Membership is effective from January 1 to December 31. Members who join after January 1 will receive all issues of *TUGboat* and *TEX and TUG NEWS* published that calendar year.

For more information ...

Correspondence

TEX Users Group
P.O. Box 869
Santa Barbara, CA 93102
USA

Telephone: (805) 963-1338
FAX: (805) 963-8358
Email: tug@tug.org

Whether or not you join TUG now, feel free to return this form to request more information.

Check all items you wish to receive below:

- Course and meeting information
- Advertising rates
- Products/publications catalogue
- Public domain software catalogue
- More information on TEX

Institutional Membership Application

Institution or Organization _____

Principal contact _____

Address _____

City _____ Province/State _____

Country _____ Postal Code _____

Daytime telephone _____ FAX _____

Email address _____

Each Institutional Membership entitles the institution to:

- designate a number of individuals to have full status as TUG individual members;
- take advantage of reduced rates for TUG meetings and courses for *all* staff members;
- be acknowledged in every issue of *TUGboat* published during the membership year.

Educational institutions receive a \$100 discount in the membership fee. The three basic categories of Institutional Membership each include a certain number of individual memberships. Additional individual memberships may be obtained at the rates indicated. Fees are as follows:

Category	Rate (educ. / non-educ.)	Add'l mem.
A (includes 7 memberships)	\$ 540 / \$ 640	\$50 ea.
B (includes 12 memberships)	\$ 815 / \$ 915	\$50 ea.
C (includes 30 memberships)	\$1710 / \$1810	\$40 ea.

Please indicate the type of membership for which you are applying:

Category _____ + _____ additional individual memberships

Amount enclosed for 1994 membership: \$ _____

Check/money order payable to TEX Users Group enclosed
(payment in US dollars must be drawn on a US bank; payment in other currencies is acceptable, drawn on an appropriate bank)

Bank transfer: your bank _____
ref # _____

TEX Users Group, Account #1558-816,
Santa Barbara Bank and Trust, 20 East Carrillo Street,
Santa Barbara, CA 93101 USA

Charge to MasterCard/VISA

Card # _____ Exp. date _____

Signature _____

Please attach a list of individuals whom you wish to designate as TUG individual members. Minimally, we require names and addresses so that TUG publications may be sent directly to these individuals, but we would also appreciate receiving the supplemental information regarding phone numbers, email addresses, and TEX interests as requested on the TUG Individual Membership Application form. For this purpose, the latter application form may be photocopied and mailed with this form.

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

TypeTeX 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

IAC_OD

A LATEX Source Code Development System

«**LaTeX-Commands**»

«**Array-&-Tabular**»
 «**Boxes**»
 «**Cross-References**»
 «**Definitions**»
 «**Document-&-Page-Styles**»
 «**Environments**»
 «**Figures-&-Tables**»
 «**File-Input-Output**»
 «**Fonts**»
 «**footnotes**»
 «**Index-&-Glossary**»
 «**Length-Space-Rules**»
 «**Line-Para-&-Page-Control**»
 «**Lists**»
 «**Math-Tools**»
 «**Misc**»
 «**Numbering**»
 «**Pictures**»
 «**Preamble**»
 «**Sectioning**»
 «**Sym**»
 «**Index-&-Glossary**»
 «**Tab**» «**makeindex**»
 «**makeglossary**»
 «**index**»
 «**indexentry**»
 «**indexspace**»
 «**glossary**»
 «**glossaryentry**»
BACKUP TO PREVIOUS SYMBOL

- Pop-Up LATEX Command Menu
- Pop-Up LATEX Hyper-Help
- OS/2 and DOS Compatible
- 10 Editing Windows
- Versatile Macro Capability
- Unlimited Undo
- Regular Expression Search
- Journaling w/playback

\$65.00

Includes 2nd Day Priority Mail Shipping
for U.S. customers. Shipping not included
for international orders. However, interna-
tional orders gladly expedited via Air
or Express Mail.

30-Day Money Back Guarantee

The LATEX document preparation system has proven itself as a tremendous system for creating technical documents. It is a feature rich system that can produce documents to the highest standards of typography.

Unfortunately, even if you use it everyday, remembering how to use all those features is next to impossible.

The IAC_OD system brings a new, more user friendly, face to the creation of LATEX documents.

A Pop-Up command selector contains *all* LATEX commands. You simply select a command and IAC_OD will prompt you for all options and fill-in information. IAC_OD then inserts the syntactically correct command, or environment, into your document.

When you need help, or examples for reference, place the cursor under any LATEX command and request Hyper-Help. A Pop-Up screen will display proper command syntax and a complete example of its use. Many screens are cross-referenced by Hyper-Links to related commands and examples.

```

tabular
\begin{tabular}{format}
...
Row Material
\end{tabular}
The tabular environment is used for producing ruled tables.  

It can be used in any mode and it processes text in LR mode.

The format scope defines the overall look of the table.  

The following special characters are used to specify format:  

|           Defines a vertical line.  

l,c,r       Determines left, center or right text placement.  

t{text}     Inserts text in every row.  

p{width}   Produce a parbox column of width units wide.  

*(num){fmt} Produce num columns with the same fmt spec.  

Example: {\|(4){c|}} produce a table of 4  

centered columns each bounded by a vrule.

Controlling Row Material:  

\           Separate row elements.  

\\          Defines the row separator (aka. a carriage return).  

\hline      Draws a horizontal line across the full width  

            of the array. May only appear after a \\ or at the  

            end of the first line.  

\cline{i-j}  Draws a horizontal line across columns i  

            through j, inclusive.

See Also:  

Tabular-Example-1  

Tabular

A single item that spans multiple columns is produced with  

the \multicolumn command.

```

EBTS

PO BOX 642L

Norfolk, MA

02056

TEL: 508-528-7728

FAX: 508-520-3272

email: ejb@world.std.com

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.

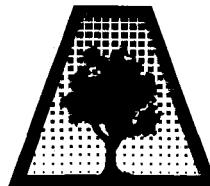
To the Complex:

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!



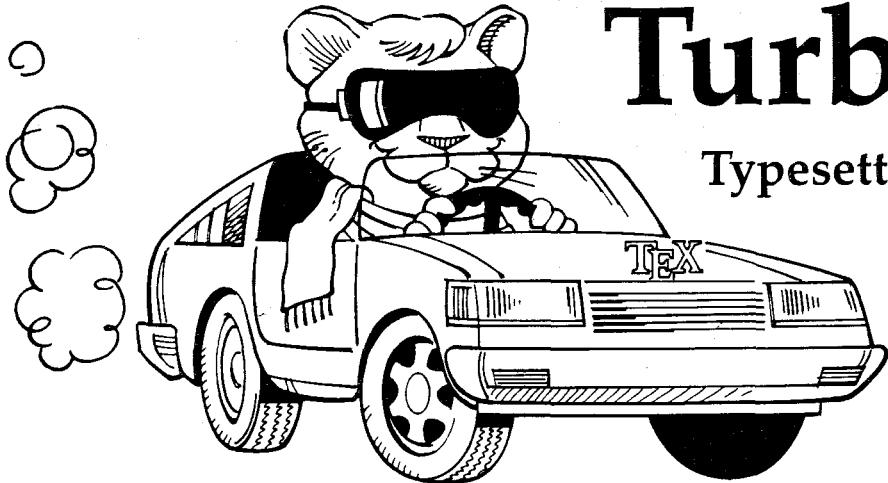
NEW!

- 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!



TurboTeX

Typesetting Software

Executables \$150
With Source \$300



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 (ex-

ecutables) 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

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	<small>Bold</small>
Avant Garde	<small>Bold Oblique</small>
Avant Garde	<small>Demibold</small>
Avant Garde	<small>Demibold Oblique</small>
Bookman	<small>Light</small>
Bookman	<small>Light Italic</small>
Bookman	<small>Demibold</small>
Bookman	<small>Demibold Italic</small>
Courier	
Courier	<small>Oblique</small>
Courier	<small>Bold</small>
Courier	<small>Bold Oblique</small>
Helvetica	
Helvetica	<small>Oblique</small>
Helvetica	<small>Bold</small>
Helvetica	<small>Bold Oblique</small>
Helvetica Narrow	
Helvetica Narrow	<small>Oblique</small>
Helvetica Narrow	<small>Bold</small>
Helvetica Narrow	<small>Bold Oblique</small>
Schoolbook	<small>New Century Roman</small>
Schoolbook	<small>New Century Italic</small>
Schoolbook	<small>New Century Bold</small>
Schoolbook	<small>New Century Bold Italic</small>
Palatino	<small>Roman</small>
Palatino	<small>Italic</small>
Palatino	<small>Bold</small>
Palatino	<small>Bold Italic</small>
Times	<small>Roman</small>
Times	<small>Italic</small>
Times	<small>Bold</small>
Times	<small>Bold Italic</small>
Zapf Chancery	<small>Medium Italic</small>
Symbol ΔΦΓΘΛΠΘ	
Zapf Dingbats	

DVIWindo [newtugad.dvi] page: 1

File Preferences Previous! Next! Unmagnify! Magnify! Fonts

Bitmap-free T_EX for Windows

Powerful, fast, flexible T_EX system for Windows

TeX Package

Y&Y T_EX Package includes:

- DVIWindo
- DVIPSONE
- Y&Y big T_EX
- Adobe Type Manager
- Acrobat Reader
- PostScript Type 1 fonts

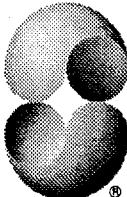
Unique Features

- Uses Type 1 or TrueType fonts
- Provides *partial font downloading*
- Can use *any* Windows printer driver
- Big T_EX runs in Windows or DOS
- Commercial grade, fully hinted fonts
- *Complete* flexibility in font encoding
- Support for EPS and TIFF images

Why Y&Y?

Mature products. Years of experience with Windows, PostScript printers and scalable outline fonts. We *understand* and know how to *avoid* problems with Windows, ATM, 'clone' printers, and problem fonts.

Y&Y — the experts in scalable outline fonts for T_EX



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

DVIWindo and DVIPSONE are trademarks of Y&Y, Inc. Windows is a registered trademark of MicroSoft Co. Adobe Type Manager is a registered trademark of Adobe Systems Inc.

TUGBOAT

Volume 14, Number 4 / December 1993

	369 Addresses
General Delivery	371 Opening words / <i>Christina Thiele</i> T _E X at meetings of other societies; Free-Net; Renew for 1994
	372 Editorial comments / <i>Barbara Beeton</i> Reminder to potential <i>TUGboat</i> authors; Call for volunteers
	372 <i>TUGboat</i> wish list
Dreamboat	374 NEXTEX: A personal view / <i>Malcolm Clark</i>
	381 NTS update / <i>Philip Taylor</i>
Software & Tools	382 Two extensions to GNU Emacs that are useful when editing T _E X documents / <i>Thomas Becker</i>
	387 Icons for T _E X and METAFONT / <i>Donald E. Knuth</i>
	390 bibview: A graphical user interface to BibTeX / <i>Armin Liebl</i>
	395 Bibliography prettyprinting and syntax checking / <i>Nelson Beebe</i>
Graphics	420 A tough table becomes easy with P _I C _T E _X / <i>Kevin Carmody</i>
Book Reviews	421 Book Review: P. W. Abrahams, K. Berry and K. Hargreaves, <i>T_EX per l'impaziente</i> / <i>Claudio Beccari</i>
Hints & Tricks	423 Ten T _E X tricks for the mathematician / <i>Helmer Aslaksen</i>
Macros	424 The bag of tricks / <i>Victor Eijkhout</i>
L_AT_EX	425 The "operational requirement" for support of bibliographies / <i>David Rhead</i>
	433 Relative moves in L _A T _E X pictures / <i>Richard Bland</i>
News & Announcements	438 Calendar
Late-Breaking News	439 Production notes / <i>Barbara Beeton</i>
	440 Coming next issue
TUG Business	441 Institutional members
Forms	443 TUG membership application
Advertisements	442 Index of advertisers
	445 T _E X consulting and production services