

TEX is alive and well. Oh what a TRIP he has just taken.

Don Knuth

as of 3:07am PST, 6 October 1989

TUGBOAT

COMMUNICATIONS OF THE TEX USERS GROUP

EDITOR BARBARA BEETON

VOLUME 10, NUMBER 3

PROVIDENCE

RHODE ISLAND

NOVEMBER 1989

U.S.A.

TUGboat

During 1990, the communications of the TeX Users Group will be published in four issues. One issue will consist primarily of the Proceedings of the Annual Meeting.

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

Submissions to TUGboat are for the most part reproduced with minimal editing, and any questions regarding content or accuracy should be directed to the authors, with an information copy to the Editor.

Submitting Items for Publication

The deadline for submitting items for Vol. 11, No. 1, is January 15, 1989; the issue will be mailed in April. (Deadlines for future issues are listed in the Calendar, page 429.)

Please note that the 1990 Annual Meeting will be held in June. Any material that should be published in TUGboat prior to the meeting must be submitted for issue no. 1.

Manuscripts should be submitted to a member of the TUGboat Editorial Committee. 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, in care of the TUG office.

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 L^AT_EX, will be sent on request; please specify which is preferred. For instructions, write or call Karen Butler at the TUG office.

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

TUGboat Advertising and Mailing Lists

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

TUGboat Editorial Committee

Barbara Beeton, *Editor*

Ron Whitney, *Production Assistant*

Helmut Jürgensen, *Associate Editor, Software*

Georgia K.M. Tobin, *Associate Editor, Font Forum*

Don Hosek, *Associate Editor, Output Devices*

Jackie Damrau, *Associate Editor, L^AT_EX*

Alan Hoenig and Mitch Pfeffer, *Associate Editors, Typesetting on Personal Computers*

See page 309 for addresses.

Other TUG Publications

TUG publishes the series *TeXniques*, in which have appeared 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, contact Karen Butler at 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.

AMS-TEX is a trademark of the American Mathematical Society.

APS μ5 is a trademark of Autologic, Inc.

METAFONT is a trademark of Addison-Wesley Inc.

PC TeX is a registered trademark of Personal TeX, Inc.

PostScript is a trademark of Adobe Systems, Inc.

TeX is a trademark of the American Mathematical Society.

UNIX is a trademark of AT&T Bell Laboratories.

General Delivery

From the President

Bart Childs

The 10th Meeting was a fantastic success. At least I thought it was. Everybody seemed to have a lots of fun, the sessions were well attended, and the weather was beautiful, as expected.

I was particularly pleased that we had METAFONT and WEB sessions that were well attended. There were also papers about the use of *TEX* and *LATEX* in real world environments. This demonstrates that our membership is widely based and not just a bunch of hackers.

We started on a Sunday afternoon with the METAFONT session. We had feared that the early sessions might have no more than 20% or 30% attendance, but more than a hundred participants showed up. In fact, not only was attendance high at all the sessions, but also the audience generally had lots of questions and/or constructive criticism. The authors and audience are both to be congratulated.

Jan Michael Rynning presented a proposal for some changes to *TEX* to handle problems that Nordic users, with their national keyboards and alphabets with accented letters, found in their everyday work. Our Grand Wizard considered them for a day and announced that he really must make some changes to *TEX* so that it will handle languages other than English in a natural way. These changes include making *TEX* 8-bit, adding a \language primitive, allowing hyphenation of words from more than one language in a paragraph, and an extended ligature mechanism that will introduce another primitive or so. The new features will affect TANGLE, *TFtoPL*, and *PLtoTF*, in addition to *TEX* and METAFONT. After some testing, Don plans to release version 3.0. More on this will be released soon.

We have some changing of the guard. The nominating committee convinced Nelson Beebe and Cal Jackson to be candidates for President and Secretary, respectively. I am pleased that these two fine gentlemen who have participated for so long were willing to accept the nominations and were elected. They take office on January 1 and will attend the meeting of the Finance Committee in late October.

I wish to thank Alan Hoenig for his yeoman service as Secretary. I also wish to thank Sam

Whidden for his years of service as Treasurer (David Ness was elected to that office last year) and, more recently, as one of the at-large representatives of the Board of Directors to the Finance Committee. The Board selected Alan to replace Sam on the Finance Committee. I think this was appropriate and will ensure that Alan's considerable talents will continue be available to our fine organization. To round out the Finance Committee, I was appointed to a special one-year term as the Past President.

There was considerable discussion by the Board concerning the directions in which the organization should evolve, the appropriate membership and size of the Board, and how the Finance Committee should meet the changing needs of our membership and yet provide stability. The By-Laws Committee (Allen Dyer, Barbara Beeton, and Lynne Price) is still in existence and many of us feel such questions should be addressed in a new set of By-Laws that incorporates many of these thoughts. Sam Whidden proposed and the Board agreed that we should have a long-range planning committee. I will appoint this committee after consultation and agreement with Nelson. Sam's reasoning was eloquently stated. He pointed out the significant changes that TUG has gone through and gave ample evidence that we can expect more in the immediate future. By the time you are reading this, the committee will be at work. You can contact me or the TUG office to obtain a copy of our charge and how to give input.

We were honored to have the elected leaders of the five formally organized European *TEX* user communities at the conference. Their exact titles vary. All were invited to be present at the meetings of the Board of Directors. I had already appointed Joachim Lammarsch of DANTE (*TUGboat* 10, no. 2, page 287) to the Board of Directors to fill an allocated position. A motion from the Finance Committee was introduced to appoint the other leaders as Vice Presidents for their respective regions. Awarding this title to Joachim as well, we now have the following new regional Vice Presidents:

1. Malcolm Clark, Vice President for the U.K.
Malcolm was already on the Board, serving as European Coordinator.
2. Bernard Gaulle, Vice President for France.
3. Roswitha Graham, Vice President for the Nordic Users.
4. Kees van der Laan, Vice President for the Netherlands.
5. Joachim Lammarsch, Vice President for Germany.

The Board discussed whether Board membership should be automatic for any users group and decided that these should be handled on a case by case basis. (I really think that came about because I pointed out that TeXas used to be a country!) Regardless, we welcome these new Board members. I had the opportunity to work with most of them again in September in Karlsruhe and they are good members!

The conference was particularly frustrating to the Board of Directors, as they spent more than each lunch hour trying to conduct the business of TUG. It is my hope that next year we can do that on the day before the meeting and maybe the last day at lunch. The Board deserves the chance to meet with the membership and *vice versa*.

We had a wonderful birthday party. (I must admit that I had the best seat in the house.) As a small token of our appreciation we gave Don a print, *Man in Blue*; a compact disk player; and some compact disks. QMS/Imagen is giving Don a laser printer which he said will make volume 4 appear four months earlier. Don and his lovely Jill cut our birthday cake.

We tried to find a CD of the two pieces of music that are referred to in *The TeXbook*. When we could not find Dvořák's *Legends*, we settled on a CD with the *New World Symphony* and *Slavonic Dances*. Don said they were Dvořák's second and third best works. Alan Hoenig said that it would be appropriate to give *Showboat* by Jerome Kern. I received a thank you note from Don in which he admitted that it was a week or two before he made the connection on *kern*. He thought we had selected it simply for the beautiful music. I noticed that some members gave Don CDs as a token of their appreciation. Good work!

The print was discovered by Cal Jackson. It has a printer sitting in front of a press somewhat like the one on the cover of *TUGboat*. It is a limited edition print of a painting by Ron Adams of Santa Fe.

A large number of the early contributors to the TeX project were in attendance and recognized. A complete list of these will appear in a future *TUGboat* or a special commemorative edition of the program. Sam Whidden showed a home movie of an early AMS visit to the TeX project. Highlights included Barbara Beeton in a bathing suit, Don carving the TeX logo in a table, and a younger view of a lot of early workers.

Ray Goucher took it upon himself to name yours truly the *Lion of TUG* or was it TeX? Oh well; in any case, a lion suit suddenly appeared for me. It was really a nice suit, except:

1. my \hfuzz value was too small;
2. I appeared as an overfull hbox; and
3. I appeared as an overfull vbox.

It had a big button on it that said "Kiss me it is my birthday." I think that I did get a kiss from each and every lady present, so it was a pretty good evening. I have been asked by several if I got to keep the suit. The answer is no, it was a rented suit. I kept the button!

Thanks to Ray, Karen, Charlotte, and the entire TUG staff for putting on such a wonderful conference and party!

◊ Bart Childs
 Dept of Computer Science
 Texas A&M University
 College Station, TX 77843-3112
 bart@ccsun.tamu.edu

Message from the New President

Nelson H. F. Beebe

I am pleased to accept the post of President of the TeX Users Group for the next two years, and hope to follow the goals and directions of my distinguished predecessors. I am delighted to have Calvin Jackson as our new Secretary, and am equally pleased that our outgoing President, Bart Childs, and Secretary, Alan Hoenig, will continue to offer us their valuable experience.

I have always enjoyed attending the TUG meetings and getting to know many of you. My first encounter with TeX was a talk that Don Knuth presented at Xerox PARC laboratories in the summer of 1978 or 1979; it was entitled *The Errors of TeX*. At our Tenth Anniversary meeting at Stanford in August, he presented an update of that talk, and I understand that in ten years' time, we may have the final story. A short version of his talk will of course appear in the Conference Proceedings (*TUGboat* 10 #4); earlier versions were presented in Stanford Computer Science Department Report STAN-CS-88-1223 (September 1988) and in Software—Practice and Experience, Volume 18, July 1989, pp. 607–686.

Several new issues confront us for the coming year.

First, `score.stanford.edu`, the venerable DEC-20, on which `TeX` master archives have resided for the past decade, is to be retired. We are looking for a new home for these archives. It is necessary that the replacement host be accessible from all major networks, and I hope that we can establish a mechanism that will permit automatic updating to at least major redistribution sites in other countries, to `TeX` site coordinators, and to commercial vendors. The persistent problems of Bitnet mail corruption, and Usenet loss or truncation, will need to be dealt with to make this successful. My own experience with my DVI driver family distribution, and regular problem reports to the `TeXhax` bulletin board of long-since-fixed `TeXware` and macro file bugs, provides evidence that we need to improve things so that end users can obtain the latest versions without long delays. If we were a commercial undertaking, we would offer software maintenance contracts, but as a non-profit organization, we have to manage with volunteer, part-time, labor. The `netlib` service at Argonne National Laboratory for the automated electronic distribution of software provides at least one model we can follow. Similarly, I hope that we can establish a centralized electronic mail routing system for TUG members, similar to the `na-net` mail drop used by the numerical analysis community.

Second, the thorny legal issue of trademarks has cropped up this year. `TeX`, `METAFONT`, `AMS-TeX`, and `LATeX` are all trademarks, but most other programs, and file extensions, in the `TeX` repertoire are not. DVI is now a registered trademark of Intel Corporation, for Digital Video Interface. We are now investigating what legal steps need to be taken to prevent our loss of use of such names.

Third, we now have at least five formal (international) user groups for British, Dutch, French, German, and Nordic `TeX` users, and we need to decide how these are related to TUG, which is a worldwide organization; our membership comes from at least 44 countries. The national (or language) groups are important and healthy signs of the wide utility of `TeX`. I had the great pleasure of attending the GUTenberg conference in Paris in May 1989, which attracted about 150 people. National groups offer `TeX` users the opportunity to talk about `TeX` in their native languages, and address problems relevant to native language typesetting in a coordinated fashion. I find it noteworthy that the July 1989 issue of `TUGboat` carried an interesting article on typesetting Vietnamese, and the August conference had a talk on typesetting of Thai. Certain problems posed by non-English language sup-

port led Don Knuth to announce in August that a Version 3.0 of `TeX` will be produced to address issues of eight-bit character sets, and multi-lingual hyphenation. Many suggestions for other changes were raised by the audience, but he wants to make this a limited modification so he can return to finishing his book series, *The Art of Computer Programming*.

Fourth, we need to expand our financial base, and also to include more users of `TeX` in TUG. From sales volumes of books and commercial software, we expect there are now several tens of thousands of users of `TeX`, yet only about 3,500 of us have joined TUG. Our financial advisors are urging us to increase our bank balance to about one year's revenues, following the practice of many other, similar, non-profit scientific organizations. To do this, we need to increase membership, and we need to increase sales of `TeX`-related products from TUG.

Fifth, the burden of editing `TeXhax`, and coordinating the UNIX `TeX` distribution, has become a significant drain on the resources of the University of Washington. TUG is helping with the support of `TeXhax`, but we are told that the increasing use of (free) network retrieval of software is reducing revenues from tape distributions that are necessary for support of the distribution. Here, I think, we must remember the maxim of the Free Software Foundation, that *free* does not mean without monetary cost, but instead, freely available and redistributable. I feel strongly that there is a place for both public-domain and commercial `TeX` implementations, but if we are to be able to continue to make the public-domain versions available, mechanisms for adequate support of that effort need to be found.

Sixth, at the August TUG meeting, Leslie Lamport met with a group of interested people to discuss the future of `LATeX`. He has insufficient time to begin a redesign of `LATeX` to address some of the problems that are now evident. Frank Mittelbach and Rainer Schöpf from Mainz seem to be the ones who have done most in this area, and I suggest that `LATeX` is of sufficient importance to us that we need to consider supporting such an effort. The wish list for `LATeX` includes a style-writer's guide, removal of the distinction between fragile and robust arguments, better control over placement of floats, and support for multi-page tabular environments, multi-column output, mixed single- and multi-column output, paper sizes other than the standard American A format (8.5in × 11in), and non-English sectional labelling. There are surely others, and it would therefore be helpful to gather them in one place; I hereby volunteer to coordinate such a collection.

Seventh, and last, the literate programming paradigm that is a critical part of the TeX and METAFONT projects is spreading. Two noteworthy efforts have just appeared. E. Wayne Sewell's new book

```
@Book{Sewell:web,
  author = "E. Wayne Sewell",
  title = "Weaving a Program:
            Literate Programming
            in {WEB}",
  publisher = "Van Nostrand Reinhold",
  year = "1989",
  ISBN = "0-442-31946-0",
}
```

on WEB is the first textbook on the subject for Pascal, C, and Modula-2 programming. Norman Ramsey's SpiderWEB is described in the Literate Programming section of Communications of the ACM, Volume 32, September 1989, pp. 1051-1055; it is a generalization of the original WEB system, starting from Silvio Levy's CWEB, that abstracts the input parsing and output formatting into grammars that are processed automatically into program code for versions of TANGLE and WEAVE. This has made it possible to support WEB-style literate programming in several languages (C, AWK, Ada, SSL, and others) without having to expend great effort in re-implementing TANGLE and WEAVE from scratch.

Finally, let me urge you to send me your ideas and comments, and complaints and criticisms. Because of my DVI driver activities, my telephone and mail volumes are already substantial; mail contacts are generally preferable to telephone ones, because they give me more freedom in scheduling responses.

◊ Nelson H. F. Beebe
 Department of Physics
 201 North Physics Building
 University of Utah
 Salt Lake City, UT 84112
 beebe@plot79.utah.edu

Editorial Comments

Barbara Beeton

TUG's great Tenth Anniversary Bash at Stanford has come and gone. The meeting was certainly the best-attended and busiest ever, and I think the program was among the best too. Since the Proceedings are being published this year as an issue of *TUGboat*, I will confine my remarks to happenings that occurred "off the record".

These are the most important developments, as I see them.

- Don Knuth's announcement of TeX 3.0, with significant additions that will make it more flexible for use with languages other than English (that is, with languages containing what we English speakers think of as "accented letters");
- Leslie Lamport's decision to turn over the maintenance and development of LATEX to a committee;
- appointment of the elected heads of five European national and regional groups to the Board as Vice Presidents representing their respective areas, as a visible step toward "internationalization";
- decommissioning of the Score computer, home to the TeX Project since the beginning, and transfer of the Stanford TeX files to a new location.

The keynote address by Don Knuth on "The Errors of TeX" confirmed something that I'd felt for a long time, namely that a single vision of what a program should be can lead to a more uniform "standard" than any amount of cooperation by a committee. Read about it in *TUGboat* 10#4, coming soon to your mailbox.

The equipment changes at the Stanford Computer Science Department, as well as Don's desire to concentrate on other projects (mainly the *Art of Computer Programming*), will have a significant impact on the way the "first copy" of TeX is distributed. Knuth has by now moved almost entirely off the venerable Sail computer (a DEC-10) and onto a new Sun (Unix) workstation at home. This workstation will not be connected to any network, both for protection against virus attacks and to insulate Don against interruptions (and overflowing disks) by electronic mail. His efficient secretary will continue to accumulate and distribute mail for him, and responses won't be electronic, or instantaneous. Other channels for reporting bugs will be established; details were not available at press time.

The *T_EX* files from the Score computer have been moved to a new Unix machine, labrea (`labrea.Stanford.edu`), and Don will periodically transport new versions of various files from his workstation to labrea on tape. As of press time, the files for *T_EX* 3.0 had not yet been moved to labrea, but Don produced an article describing the details of the new implementation (see page 325). The errata and changes will be mailed with the Proceedings issue, coming soon. And other developments will be reported in *TeXhax* and in the next regular issue of *TUGboat*.

In addition to increased international representation on the Board, there have been formal and informal discussions on how *TUGboat* might be made more accessible to readers whose native language is not English. A consensus has been reached that some material of interest primarily to non-English-speaking regional groups will be published in the appropriate language. In addition, abstracts of technical articles will be published in other major languages. You will see the first instances of both kinds of material later in this issue. In order to simplify the task of preparing the non-English abstracts, authors of technical articles are now asked to include an abstract. (Other suggestions to authors are presented in an article by Ron Whitney and me in the Macros column, p. 378; the user interface is new, many of the macros are new, and I have nothing but good things to say about Ron's cheerful assistance with trying to meet *TUGboat* deadlines. Thanks, Ron! The new macros will be shipped off to the various archives as soon as this issue is "put to bed".)

In the last issue, I presented an algorithm for determining *TUGboat* editorial deadlines. I have been reminded that Labor Day is not an international holiday. Oops! Please amend the deadline for issue #4 to be the 2nd Tuesday after the 1st Monday in September.

Another facet of supporting an international user population is to collect hyphenation patterns for "all possible languages". Michael Ferguson has volunteered to build such a collection, and anyone with information on hyphenation patterns is invited to get in touch with Michael at

`mike@inrs-telecom.uquebec.ca`

Several important posts have now changed hands. TUG's new officers have already been announced. Nelson Beebe and Cal Jackson, President and Secretary, respectively, have been active supporters of *T_EX* since almost the beginning, and I am looking forward to working with them in a more

official capacity. The chairmanship of the Output Device Driver Standards Committee has changed hands too; Don Hosek has taken over from Robert McGaffey. Thanks to Robert for getting the committee started; Don — when are you going to send me a report to publish? And by no means least important, the future maintenance and development of *L_AT_EX* has been entrusted to the competent hands of Frank Mittelbach, whose name should certainly be familiar by now to *TUGboat* readers (Frank's article on "An environment for multicolumn output", page 407, was the basis for his citation as this year's Donald E. Knuth Scholar); congratulations, good luck, and we'll all be sending you our favorite suggestions.

While I was at Stanford, I paid a visit to Adobe headquarters to find out about the status of the Bigelow & Holmes Lucida math symbol fonts. The font people at Adobe are understandably interested in having these fonts in good working order when they are released, and pleaded, along with more pressing items on their schedule, a lack of experience with *T_EX* and a lack of knowledge of math that prevented them from completing the thorough testing the fonts deserve. With Adobe's concurrence, I have suggested the names of several mathematicians who are experienced *T_EX* users and who have said they would be interesting in beta testing the Lucida symbols. I have also sent in some suggestions for modifications based on font dumps I was given during my visit. I hope to have more news by next issue.

Several weeks later, ignoring the *TUGboat* authors' deadline, I attended Euro_T*E*X89 in Karlsruhe. Although some of the same faces were there that had been at Stanford, there were many that were new to me, and I was delighted to meet more *T_EX*ers who had been just names. Anne Brüggemann-Klein and Rainer Rupprecht were splendid hosts, and everything was very well organized. The technical program was varied and interesting (the program is listed on p. 436), and a unique social event was held on Tuesday evening — a visit to a research vineyard followed by an excellent buffet and wine-tasting at the Weingut Theobald Pfaffmann in Landau-Nussdorf. For one who knows about German wines mainly through what is imported to the U.S., I was most pleasantly surprised by the selection of red wines produced in that region and presented to us at the tasting along with the more familiar whites. A sidelight of the daily routine was

BACK\ LASH

"The unofficial quasi-daily organ of the TeX89 Conference, '*All the news printed to fit.*'" by Peter Flynn conspiring with Malcolm Clark. Good show!

Next year there will be two meetings sponsored by TUG: the annual meeting at TeXas A & M and a meeting in Europe at the University College, Cork; see the Calendar and Calls for Papers for the important dates. It's time to start planning now.

The First Dutch TeX Days

Victor Eijkhout and Nico Poppelier

1 Introduction

The Dutch TeX Users Group, in Dutch 'Nederlandstalige TeX Gebruikersgroep' or *NTG*, was started about one year ago in Groningen. At the meeting in the autumn of 1988 it was decided to have a first presentation to 'the outside world' in the summer of 1989. Three people from the University of Utrecht volunteered to organize these first 'Dutch TeX days' and suggested June 29 and 30, in the Utrecht University computer centre.

After some thought the organizers decided upon a two-part programme: the first day would consist of two courses — one for beginners and one for experienced users — and the second day would consist of various presentations.

2 First day: L^AT_EX courses

The *NTG* members who were invited to teach the courses — Jan Luyten (Computer Centre, University of Groningen) and one of the authors (NP) — suggested that, since in the Netherlands L^AT_EX is used more than plain TeX, the courses on the first day should be L^AT_EX courses. Therefore the organization settled upon a beginners' course which would focus on presenting the possibilities of L^AT_EX by means of examples that the students could try out themselves, and an expert-level course which would focus on L^AT_EX's document styles.

The two groups that gathered on June 29, twenty-seven people in the beginners' course and nineteen in the document style course, were very enthusiastic. The beginners' group, which consisted of secretaries, employees of Elsevier Science Publishers and various other people, even insisted on getting back to their PCs half an hour earlier after lunch.

During lunch and the breaks for coffee and tea there was also enough time for discussions and problem solving. At the end of the day organizers, teachers and students decided that it had been a successful day: plans were made to improve the course material, to add a course for advanced L^AT_EX users and to teach these courses again next year.

3 Second day: presentations

On June 30 some sixty people gathered to attend the opening address by Kees van der Laan, chairman of the *NTG*, an invited talk by Malcolm Clark, *TUG*'s European coordinator, and ten contributed talks which were given in parallel sessions. As 'guests of honour' Joachim Lammarsch and Luzia Dietsche, chairman and secretary of *Dante*, the German TeX users group, were present.

3.1 Opening talk — Kees van der Laan

In the opening talk, Kees van der Laan, chairman of the *NTG*, summarized the short history of the *NTG* and presented the list of working groups of the *NTG*. Most of these working groups have been rather active in the short period since the *NTG* started and a lot has been achieved already:

- material for introductory meetings and courses has been developed
- there is a *Bitnet* discussion list, called *tex-nl* (which is now an *open* list) where questions are usually answered within a few hours by some of *NTG*'s more experienced members
- since mid-summer, there is also a file server where hyphenation patterns and style files can be found, and where more material will be added in the near future.

Kees van der Laan also mentioned that there are now firm contacts with the Dutch *SGML* users group, and that most likely next year's presentation of the *NTG* will be a *joint* one of the Dutch *SGML* users group and the *NTG*.

He concluded by saying that the *NTG*, a young and energetic users group, will try to establish contacts with other user groups in Europe, such as the groups in Germany, France and the Northern-European countries.

3.2 TeX in Europe & TeX in the future — Malcolm Clark

Malcolm Clark, *TUG*'s European coordinator, spoke about the importance of organization in the world of TeX and, especially, the need for European TeX users to organize and to present themselves. He stressed the point that, if European users feel that

TUG should pay more attention to what is happening here, they should become *TUG* members.

He also tried to answer the question 'Why does *TeX* get bad publicity, or none at all?'. His answer was that the *TeX* users just *do not* write often enough that *TeX* is a marvelous and powerful document production system, which features things that other programs are just beginning to show—and guess where they got the ideas from? He added that *TeX* is a *de facto* standard, a fact that is too often ignored.

3.3 From SGML to *TeX* and vice versa— Jos Warmer

At the invitation of the *NTG* working group on the *TeX-SGML* relation, Jos Warmer, who has been involved in an Amsterdam project for the implementation of an *SGML* parser, gave an introduction to *SGML*, briefly explaining document type definitions and the mechanism of tags and end tags. As an example he showed the title page of his lecture in both *LATEX* and *SGML* form. He then explained that an *SGML* parser primarily tests for whether the *SGML* document conforms syntactically to the document type definition. However, as a further elaboration of the earlier example showed, a simple substitution mechanism seems almost to be able to perform a translation from *SGML* to both *TeX* and *Troff*.

Jos Warmer concluded by indicating some of the problems involved in the translation from *SGML* to *TeX* and vice versa, where in his opinion the reverse operation is the more difficult one. He mentioned the problem of *TeX*'s special characters (which necessitate insertion of backslashes), the need to specify a syntax for mathematical formulae in *SGML* for translation to *Eqn* or *TeX*, and the fact that the choice of back-end (for instance *LATEX*) already influences the structure of the document type structure. He suggested that the translation to *TeX* may very well need to be context-sensitive because of such idiosyncrasies as the gobbling up of spaces after *TeX* commands.

3.4 DTP versus *TeX*—Ad Emmen

The speaker, who works in the Amsterdam universities' computer centre and has ample experience with both *desktop-publishing* programs and *TeX*, talked about the general features of *desktop publishing* programs and the relative simplicity with which you can integrate text with graphics. He then discussed the differences between *dtp* systems and *TeX/LATEX*, and their respective advantages and disadvantages. Ad Emmen concluded by giving examples of cases

where *TeX/LATEX* is to be preferred and cases where he considered *dtp* systems to be a better tool.

3.5 Metafont—Walter Jaffe

Some time ago Walter Jaffe finished a very nice-looking Hebrew font, suitable for reproducing texts from, e.g., the Jewish Bible and Talmud, since it contains the characters for the consonants as well as for the vowels.

He started his presentation with a short introduction to *METAFONT*, the nature of the language, and the process of designing letters in *METAFONT*. He then presented his work on Hebrew characters.

3.6 *TeX*'s hyphenation algorithm— Gerard Kuiken

Gerard Kuiken, author of a 5000-item list of hyphenation patterns for the Dutch language—generated by hand!—discussed in detail the line breaking algorithm of *TeX* and the computation of demerits therein. He then explained the hyphenation algorithm and the compact storage that *TeX* uses for both these patterns and the hyphenation exceptions.

The last part of his presentation focussed on the particular difficulties of hyphenating the Dutch language. In Dutch, words are on average longer than in English, and situations abound where a hyphenation may be morphologically correct but semantically unfortunate, the classic example being the compound word *bet-o-ver-groot-moe-der* ('great-grandmother') where the first part of the compound *betoever* means 'bewitch' and hyphenating after *beto* suggests a similar continuation.

The speaker treated in detail the example *ver-kleden* ('change dress') versus *kerk-leden* ('church members'), which is treated correctly by his patterns, but on which even a recently compiled set of patterns, derived from a list of 350 000 hyphenated words, fails.

3.7 Simple *TeX*—Andries Lenstra

While the Dutch *TeX* days had in general a bias towards *LATEX*, Andries Lenstra's lecture was solely concerned with plain *TeX*. However, he started out by criticising the way plain *TeX* is often used, namely as a deluxe typewriter, with people interspersing the input with skips and explicit font changes. Like the speaker before him, Auke van der Groot, who talked about a macro package for user manuals, Andries Lenstra stressed the need to separate the logical structure of the document from its visual structure: an author need only be concerned with getting his message across, and must leave typography to people qualified in that respect. If an

author is also the macro writer—as in the case of this speaker—he must try to keep as many degrees of freedom open as is possible, and confine the actual visual commands to a section containing what Andries Lenstra called ‘the tuning knobs’.

He illustrated this by means of his macro package ‘**simple.tex**’—of less than 10k size—which he developed when writing the course material for a statistics class he teaches. He treated in some detail the way his macros cope with the problems of optional arguments, when, for instance, only in the case of two consecutive non-empty arguments a separating space needs to be inserted.

3.8 Transparencies with LATEX— Kees van der Laan

Kees van der Laan, who is involved in teaching various courses and is also a member of the *NTG* working group on course material, presented a small package, in the form of a LATEX document style, for developing transparencies to be used in teaching courses. In this lecture Kees van der Laan explained how to use the macro package and discussed the philosophy behind the transparency document style, which is of course based on the notion of separating logical structure from visual structure.

3.9 TEX for the Dutch language— Victor Eijkhout

This lecture, an official presentation of any of the working groups of the *NTG*, consisted of an inventory of the collective efforts of three members of working group number 13 and a project not related to any of the *NTG* working groups. This latter item concerned the public availability of a new list of hyphenation patterns for the Dutch language, which was compiled using *PATGEN* on a list of 350 000 hyphenated words. Even this list, however, is not quite up to the problems associated with compound words in Dutch. An extra problem is that its size requires recompilation of both INITEX and TEX.

Victor Eijkhout then presented three LATEX style options that remedy some acute problems: first of all the option *dutch* replaces in all four standard LATEX styles the definitions containing English words such as ‘Contents’, by parametrized definitions and sets the parameters initially to Dutch values, but English settings are included as well. The style options *a4*—not the one by John Pavel—and *sober* then set the page size to European standards and reduce white spaces and font sizes in the standard LATEX styles.

In the third part of his lecture, the speaker discussed why the layout of, for instance, the *article*

style of LATEX is unacceptable for Dutch use, and he presented an *artikel* style—currently under development—that is compatible to *article*, but which has a different layout. His conclusion was that the flexibility of LATEX is in part illusory: in order to achieve certain effects it has turned out to be necessary to rewrite certain macros contained in *latex.tex*, including them in the new style file.

The speaker concluded his talk by mentioning that all of the above items are freely available on the recently installed Dutch *tex-nl* fileserver, and gave the appropriate commands for retrieval.

4 Other topics

Apart from the parts of the programme we wrote about in some detail, there were also presentations of AmigaTEX, various applications of LATEX, and the use of TEX in preparing company manuals.

During the entire day there were also various vendor presentations in the computer centre by, e.g., the local *Atari* dealer and *Hewlett Packard*.

In the computer centre there was a stand with reports and various other publications on TEX, LATEX and SGML, such as a Dutch book on LATEX, ‘Publiceren met LATEX’ (‘Publishing with LATEX’), written by *NTG* members from Groningen, and a report written by *NTG* members and LATEX users from Utrecht and Amsterdam, discussing LATEX and the differences between LATEX and desktop-publishing systems.

◊ Victor Eijkhout
Department of Mathematics
University of Nijmegen
Toernooiveld 5
6525 ED Nijmegen,
The Netherlands
Bitnet: U641000@HNYKUN11

◊ Nico Poppelier
University of Utrecht
Department of Physics
Postbus 80.000
3508 TA Utrecht
The Netherlands
Bitnet: Poppelier@HUTRUU51

Answers to TEXtests

Bart Childs

Following are the questions and answers to the TEXtests shown in the last issue of *TUGboat* (pp. 157-159). We hope to build a list of questions and answers, and we solicit help from all our readers.

The TEXtest — Level One

1. One of the visible ASCII characters is used as TEX's escape character. It is the _____ symbol and its name is _____.

- \ and "backslash".

2. In the following TEX code fragment:

1. ...
2. last line of a paragraph.
3. \parskip=6pt

4. First line of a new paragraph
what horizontal and vertical spaces will
be between "... a paragraph." and "First
line...?"

- End of sentence space only. The setting of a new paragraph parameter does not also signal the end of a paragraph.

3. What TEX control sequence is the equivalent of a blank line?

- The \par command for ending a paragraph.

4. How do you cause the TEX program to execute and process the file "testfile.tex" on your system?

- Depending upon the system it will probably be

tex testfile

or

tex testfile.tex

or a command to invoke an environment and select the appropriate option from a menu or click on the appropriate icon.

5. When TEX has finished processing **testfile.tex**, how can you get another look at the error messages (with more detail) without running TEX again?

- Look at **testfile.log**.

6. The code fragment the \TEX program produces output "the TEXprogram" which is obviously missing a space after the TEX logo. Give two or more ways to correct this.

- Follow \TEX with a tilde, control space, left-right brace followed by a space, left brace-space-right brace, or ...

7. What is the name of TEX's monospaced font and what control sequence is used to access it?

- Typewriter and \tt.

8. What is the typographer's name for straight lines?

- rules.

9. How do you end the indentation from the \narrower instruction?

- Group the \narrower command and text in braces. Be sure to end the paragraph before ending the group.

10. How should you end the current paragraph before ending the \narrower mode?

- It is ended by a blank line, \par, or a \vskip command that terminates a paragraph.

11. What is the indentation of the following paragraph and why?

1. {\narrower\narrower
2. first line of a paragraph.
3. ...
4. last line of a paragraph.
5. }\par

- The paragraph will use the text width outside the \narrower mode, because the \par follows the end of the group.

12. Consider the following code fragment:

1. \parindent=0.5in
- 2.
3. A first paragraph ...
4. \parindent1.0in
- 5.
6. A second paragraph ...
7. \bye

How much will each of the paragraphs be indented? first _____ second _____

- The first will be indented 0.5 inch and the second 1.0 inch.

13. How do you specify an italic correction?

- It is denoted by \it.

14. What does an italic correction do?

- It is used to make sure that tall italic characters do not lean into tall upright

*characters. With: tall boys, and without:
tall boys.*

15. Is the space in:

`\centerline {Centered}`

necessary ___, optional ___, or in error ___?

- *Optional; spaces after alphabetic control sequence names are gobbled, and a brace may delimit such a control sequence.*

16. What will TeX output from the following code fragment?

`\centerline Center This!`

- *It will center the 'C' and start a new paragraph with "enter This!"*

17. Describe the output of this code fragment?

`\bf{this is bold text}...`

- *This will cause everything to be bold from now on until another font is specified. The obvious intent could have been achieved by placing the opening brace before the \bf. The braces, as they are, are wasted.*

18. What is a widow?

- *It is the last line of a paragraph which appears at the top of a page. An orphan (also called a club line) is the first line of a paragraph that appears alone at the bottom of a page.*

19. How do you place the page number flush right in a running head?

- *Define the \headline to have \hfill\folio in it.*

20. How do you keep the left margin fixed and move the right margin to the left by 0.5in?

- *By increasing the \rightskip by .5in or decreasing the \hsize by .5in.*

A new TeX user has decided to create some macros. The following definitions are OK or BAD! Mark each of these OK or BAD, and indicate what is wrong with the BAD ones. Assume the plainest of TeXs.

21. `\def\A1{...}`

- *BAD; names are all alphabetic or 1 character.*

22. `\def\A-OK{...}`

- *BAD; ditto.*

23. `\def\Test{...}`

- *OK to mix cases.*

24. `\def\\{\dots}`

- *OK to use the escape character, too.*

25. `\def\10{...}`

- *BAD; because if not alpha, 1 character.*

There are ten visible ASCII characters that TeX has reserved for special uses. For example, the dollar sign is used to toggle mathematics mode. List the other nine and their use as illustrated.

26. \$ *toggle math mode*

27. _____

28. _____

29. _____

30. _____

31. _____

32. _____

33. _____

34. _____

35. _____

- \$ *toggle math mode*

- # *for arguments and parameters*

- & *tab character*

- ~ *tie or hard space*

- { *begin group*

- } *end group*

- \ *the escape character*

- % *begin comment*

- ^ *superscript*

- - *subscript*

The TeXtest — Level Two

1. What mode is TeX in when building a paragraph?

- *Horizontal.*

2. How do you end a \topinsert?

- *The \endinsert ends \topinsert, \midinsert, and \pageinserts.*

3. If TeX hyphenates a word badly, how do you fix it?

- *If the word is used frequently, you should use \hyphenation. If it is rarely used you can use a discretionary hyphenation, \-, when using the word. If the word should not be hyphenated, put it in an \hbox.*

4. What happens to a paragraph that has both normal indentation and a \hangindent specification?

- The first line will be indented an amount of `\parindent` and the other lines numbered \geq the value of `\hangafter` will be indented by the amount of `\hangindent`.
5. What will `\line{A Short Line}` look like in a normal page?
- It will be spread out to the entire width of the page and will give you an underfull `\hbox` message unless you have a small value of `\hsize`.
6. Why won't a field like `{\hfil x \hfil}` be centered in a `\settabs` environment?
- An `\hfil` is furnished by `\settabs` and glue is additive. Thus, there will be twice as much space following as there is preceding.
7. How can you get a black square, like , in the middle of a line of text?
- Use a vrule; this one is 10pt by 10pt with zero depth.
8. Consider the code fragment:
1. ...
 2. `\eject\vskip2in`
 3. How now brown cow...
- Where is the "How now brown cow" placed relative to the top margin of the page?
- It immediately follows the top margin, since vertical glue is gobbled up as the first item on a page.
9. How do you move a `\vbox` to the right one inch?
- `\overright1in`; the `\raise` and `\lower` commands are used to move boxes in the vertical direction.
10. How do you reduce or prevent widows?
- Raising the `\widowpenalty` will reduce the possibility. If raised high enough (10000) it will prevent them. The `\clubpenalty` is similar for orphans. Using the `\filbreak` macro between paragraphs will make breaks there more attractive.
11. What is a penalty?
- A penalty is the "cost" assigned by TeX to breaking a line or a page at a particular place. Each possible break has a penalty assigned to it depending upon some complicated rules.
12. What happens when you forget to end a `\footnote`?
- The text following keeps being taken into memory. TeX is trying to build a single box of all of it. Thus, you will often get the message that it has exceeded its memory capacity.
13. What happens when you try to end a document without a proper end to an insert?
- You get an error message that you cannot end a document while in internal vertical mode.
14. What happens if you have a blank line in display math mode?
- You get a message that `\par` is not allowed in math mode.
15. How is the `\tabskip` parameter used?
- It is used to put white space between columns of an alignment.
16. How are the `\lineskip` and `\lineskiplimit` parameters used?
- If the depth of one "line" (i.e. box) plus the height of the following "line" is $> \baselineskip - \lineskiplimit$, then glue with value `\lineskip` is inserted between the lines rather than the glue needed to achieve `\baselineskip`. This often happens when boxes which themselves contain more than one line of text adjoin one another.
17. What actions should you consider to correct the conditions that caused the warning `Overfull hbox`?
- Change the wording, add discretionary hyphens, use `\slash` for slashed acronyms, or adjust `\hfuzz` and/or `\tolerance`.
18. What actions should you consider to correct the conditions that caused the warning:
`Underfull \vbox has occurred`
`while \output is active`
- The `\raggedbottom` command or `\vfil` will often do enough. If you want to fill the page without the `\raggedbottom` appearance, you can put extra stretch in `\parskip`, `\baselineskip`, `\abovedisplayskip`, and `\belowdisplayskip`, (and of course, `\abovedisplayshortskip` and `\belowdisplayshortskip`.)
19. Show how to assign the control sequence `\8` to a new font "cmss8" that has not already been defined in plain TeX.
- `\font\8=cmss8`.

20. What is the meaning of #1 in a macro definition?
 - It represents the first argument.
21. Arguments to macros may be *delimited* or *undelimited*. Describe how *TEX* determines arguments in the two cases. How do users notice the difference?
 - In the case of an undelimited argument, *TEX* reads the next token or the next group of tokens with balanced braces. For delimited arguments, *TEX* reads tokens until it encounters the next occurrence of the delimiting string on the same grouping level at which it started the search. Thus, undelimited arguments that are not a single character or control sequence must be enclosed in braces. Delimited arguments will have a specific character string that denotes their end. Semicolons, commas, new lines, etc. are often used as delimiters.

The *TEX*test — Level Three

1. How is *TEX*'s escape character determined?
 - It is the \ by default. It can be any character whose \catcode is set to zero.
2. What is the name of the control sequence that can be used to accomplish the function of the left brace, {, inside a macro that will not have its matching right brace, }?
 - \bgroup can act as { and \egroup can act as }. \begingroup and \endgroup are similar and can be used for localization, but they cannot be used to delimit box contents.
3. What are the names of the parameters that specify the amount of glue above and below display math?
 - The \abovedisplayskip and \belowdisplayskip parameters have descriptive names as do \abovedisplayshortskip and \belowdisplayshortskip.
4. *TEX* treats several consecutive spaces as one. Thus the usual practice of keyboarding 2 spaces after periods does not insert extra space at the end of a sentence. How does *TEX* know to put more than an ordinary interword space at the ends of sentences?

- *TEX* looks at the current value of \spacefactor whenever a space is read. If this value is 1000, ordinary interword space is inserted, but if the \spacefactor is ≠ 1000, *TEX* adjusts space accordingly. As each character is set, its \sfcode resets the \spacefactor if the \sfcode is non-zero. The \sfcode of a period, for example, is 3000. When a space follows a period, this generally causes *TEX* to insert extra space. However, *TEX* disallows consecutive settings of the \spacefactor to jump from below 1000 to above 1000 (or vice-versa). It sets the \spacefactor to 1000 when the second character of such a pair is encountered. This can show different effects depending upon the stream of \sfcodes and \spacefactors devoured by *TEX*. For example, this is the means by which *TEX* "knows" what to do with space after initials.
- 5. Are the leading spaces in \halign entries significant?
 - No.
- 6. Are the trailing spaces in \halign entries significant?
 - Yes, they are significant up to the ampersand or \cr.
- 7. Are the leading and trailing spaces in \settabs entries significant?
 - Neither are significant.
- 8. How serious will underfull vbox badness 3412 be?
 - Not very serious.
- 9. Build a macro called \xx that has one parameter delimited by a semicolon. The macro is to center its one parameter and set it in bold.
 - \def\xx#1;{\centerline{\bf#1}}
- 10. What happens when


```
\def\p#1{\it\centerline#1}
```

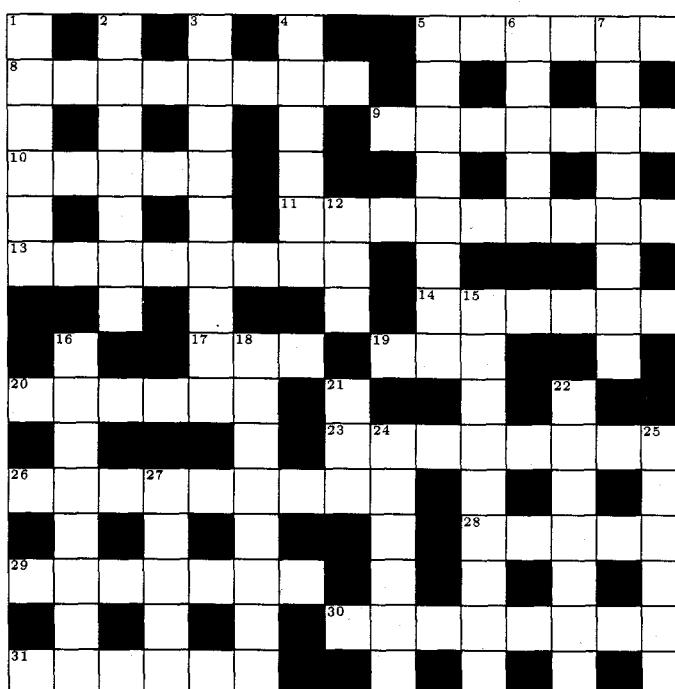
 is called with \p01234?
 - It changes the default font to italics. It then centers the 0. The rest of the digits are indented to start a new paragraph. In some cases *TEX* expects a numeric field and it will eat digits until it finds a non-digit.
- 11. The following code fragment is an exercise in being careful:

1. `\newcount\cntr`
 2. `\advance\cntr1\the\cntr`
- What is the resulting value of the counter?
- *The answer is 101. Before the addition (to zero) is done, `\the\cntr` is evaluated as 0. Then, the three digits in a row make up the addend. Put in a couple of spaces and see some different results!*
12. How can you zero all the dimensions of `\box0`?
 - *You can set the height, depth, and width to zero or you can `\setbox0=\hbox{}`.*
 13. What information is in a `tfm` file?
 - *The `tfm` file contains the *TEX* Font Metrics. This is the height, depth, and width of each character. Also included is information about which character pairs are kerned, ligature substitution, italic corrections, etc. *TEX* uses only this information about fonts.*
 14. What information is stored in a `pk` or `gf` file?
 - *This file contains actual picture or pixel information about the font. This file is necessary for most of the programs that put the ink on the paper. They are not needed on systems where the font that *TEX* is using is already in the printer.*
 15. How are `\vtop` and `\vbox` similar? How do they differ?
 - *They both create a vertical box. `\vtop` takes the height of its first line and grows in depth while `\vbox` grows in height and has the depth of its last line.*
 16. A two column macro package works by gathering enough information for both columns before invoking the `\output` routine. What is the name of the *TEX* primitive that is probably used to determine each column?
 - *The `\vsplit` primitive allows the creator of an output routine to control the breaking of pages (and columns).*
 17. What does `\futurelet` allow you to do?
 - *`\futurelet` allows a user to replicate a token just ahead of the next token to be read in the input stream. The next token can then test the replication to determine what will be coming next and hence to take the appropriate action.*
 18. What *TEX* commands would you have to use to automatically build an index and/or table of contents to a separate file, and print it in the output?
 - *The commands would certainly include `\openout`, `\write`, and `\closeout`. After it is written and closed, then you would open the file to read, or use `\input`.*
 19. What happens when you underestimate the number of lines in a `\parshape` command?
 - *All subsequent lines take the shape of the last line of the definition.*
 20. Under what conditions can you use the built-in fonts of an arbitrary printer?
 - *The fonts must be baseline fonts and the `tfm` file must be available to *TEX*. You can build a `tfm` file by measuring, say, 10 or 20 of each character, editing a `pl` (property list) file, and running `pltotf` to make a `tfm` file. You should expect to need to enter some kerning information after a few tests.*
 21. What characteristics should you look for on a page of output to try to determine if the page was prepared using *PageMaker*, *troff*, ..., or *TEX*.
 - **TEX* is most easily identified by the quality of mathematics. Usually the other systems will have limited fonts, poor hyphenation, no ligatures, and inconsistent or no kerning.*
 22. What element(s) of *TEX* is (are) case insensitive?
 - *The dimensions are case insensitive, `IN` \equiv `in` ... In some systems file names will also be case insensitive.*

◊ Bart Childs
 Dept of Computer Science
 Texas A&M University
 College Station, TX 77843-3112
 bart@cssun.tamu.edu

No. 1**by LogoTeXnes**

Please note that the somewhat insulting clues for certain personages are purely a compiler's artifice!
 (One American spelling, and certain proper nouns are not in *Chambers*.)

**ACROSS**

- 5** Sextet bring you in to play very loudly underscore introduction (6)
- 8** A metallic element media disturbed in halls of learning (8)
- 9** (see 20)
- 10** Letter from Greek mountain area (5)
- 11** Girl with sex appeal in Middle Eastern source of silvery metal (9)
- 13** Senate decrees North will be made to give a public exhibition (8)
- 14** Sorrowful wishes don't begin to describe white herons (6)
- 17** Finally pairs up with 28! (3)
- 19** Sounds like a new companion for Nat (3)
- 20 & 9** Eastern quarter-backs provide optional introduction (6,7)
- 23** A body-builder produces heavenly body! (8)
- 26** Without this character's help, there's no escape! (9)
- 28** Make a start on fencing in the environment? (5)
- 29** Note taken from short time to finish? No, but that from which it's taken (7)
- 30** Scour and beat after concerning boxes which are too high or wide (8)
- 31** Scots island points after Imperial measures (6)

DOWN

- 1** Measures of capacity formerly unused by 27 (6)
- 2** Too bad French brought up this man's lack of multilingual support (7)
- 3** 1,864,679,811 of the smallest units 27 abbreviated a measurer (9)
- 4** Galley filled with two lines of Roman types? Rim involved six-footer in its construction (6)
- 5** Shun kern construction: it's contracted (8)
- 6** This man spreads TeX's secrets like his atomic namesake! (5)
- 7** Tiny diet indisposed personality (8)
- 12** Very refined language used by world's first programmer? (3)
- 15** Moveable type, inventor gent moved around eastern burg (9)
- 16** One mixed up about insignificant Shakespearean person with another one, yields formula to express chemical action (8)
- 18** Fresh verses: two or more finish the paragraph (8)
- 21** Talk a lot of hot air! (3)
- 22** Chewing sort of sugar cane could lead to tender oral tissue, we hear (7)
- 24** Woman greeting a speck in paper (6)
- 25 & 27** Hunt odd lank confused professor (6,5)

Software

The New Versions of **TEX** and **METAFON**T

Donald E. Knuth

For more than five years I held firm to my conviction that a stable system was far better than a system that continues to evolve. But during the TUG meeting at Stanford in August, 1989, I was persuaded to make one last set of changes, in order to bring **TEX** and **METAFON**T to a state of completion consistent with their overall philosophy and goals.

The main reason for the changes was the fact that I had guessed wrong about 7-bit character sets versus 8-bit character sets. I believed that standard text input would continue indefinitely to be confined to at most 128 characters, since I did not think a keyboard with 256 different outputs would be especially efficient. Needless to say, I was proved wrong, especially by developments in Europe and Asia. As soon as I realized that a text formatting program with 7-bit input would rapidly begin to seem as archaic as the 6-bit systems we once had, I knew that a fundamental revision was necessary.

But the 7-bit assumption pervaded everything, so I needed to take the programs apart and redo them thoroughly in 8-bit style. This put **TEX** onto the operating table and under the knife for the first time since 1984, and I had a final opportunity to include a few new features that had occurred to me or been suggested by users since then.

The new extensions are entirely upward compatible with previous versions of **TEX** and **METAFON**T (with a few small exceptions mentioned below). This means that error-free inputs to the old **TEX** and **METAFON**T will still be error-free inputs to the new systems, and they will still produce the same outputs.

However, anybody who dares to use the new extensions will be unable to get the desired results from old versions of **TEX** and **METAFON**T. I am therefore asking the **TEX** community to update all copies of the old versions as soon as possible. Let us root out and destroy the obsolete 7-bit systems, even though we were able to do many fine things with them.

In this note I'll discuss the changes, one by one; then I'll describe the exceptions to upward compatibility.

1. The character set

Up to 256 distinct characters are now allowed in input files. The codes that were formerly limited to the range 0..127 are now in the range 0..255. All characters are alike; you are free to use any character for any purpose in **TEX**, assigning appropriate values to its `\catcode`, `\mathcode`, `\lccode`, `\uccode`, `\sfcode`, and `\delcode`. Plain **TEX** initializes these code values for characters above 127 just as it initializes the codes for ordinary punctuation characters like '!.

There's a new convention for inputting an arbitrary 8-bit character to **TEX** when you can't necessarily type it: The four consecutive characters $\sim\alpha\beta$, where α and β are any of the "lowercase hexadecimal digits" 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, or f, are treated by **TEX** on input as if they were a single character with specified code digits. For example, $\sim\sim80$ gives character code 128; the entire character set is available from $\sim\sim00$ to $\sim\simff$. The old convention discussed in Appendix C, under which character 0 was $\sim\sim0$, character 1 (control-A) was $\sim\simA$, ..., and character 127 was $\sim\sim?$, still works for the first 128 character codes, except that the character following $\sim\sim$ should not be a lowercase hexadecimal digit when the immediately following character is another such digit.

The existence of 8-bit characters has less effect in **METAFON**T than in **TEX**, because **METAFON**T's character classes are built in to each installation. The normal set of 95 printing characters described on page 51 of *The METAFON book* can be supplemented by extended characters as discussed on page 282, but this is rarely done because it leads to problems of portability. **METAFON**T's `char` operator is now redefined to operate modulo 256 instead of modulo 128.

2. Hyphenation tables

Up to 256 distinct sets of rules for hyphenation are now allowed in **TEX**. There's a new integer parameter called `\language`, whose current value specifies the hyphenation convention in force. If `\language` is negative or greater than 255, **TEX** acts as if `\language = 0`.

When you list hyphenation exceptions with **TEX**'s `\hyphenation` primitive, those exceptions apply to the current language only. Similarly, the `\patterns` primitive tells **TEX** to remember new hyphenation patterns for the current language; this operation is allowed only in the special "initialization" program called **INITEX**. Hyphenation exceptions can be added at any time, but new

patterns cannot be added after a paragraph has been typeset.

When TeX reads the text of a paragraph, it automatically inserts "whatsit nodes" into the horizontal list for that paragraph whenever a character comes from a different \language than its predecessor. In that way TeX can tell what hyphenation rules to use on each word of the paragraph even if you switch frequently back and forth among many different languages.

The special whatsit nodes are inserted automatically in unrestricted horizontal mode (i.e., when you are creating a paragraph, but not when you are specifying the contents of an hbox). You can insert a special whatsit yourself in restricted horizontal mode by saying \language{number}. This is needed only if you are doing something tricky, like unboxing some contribution to a paragraph.

3. Hyphenated fragment control

TeX has new parameters \lefthyphenmin and \righthyphenmin, which specify the smallest word fragments that will appear at the beginning or end of a word that has been hyphenated. Previously the values \lefthyphenmin=2 and \righthyphenmin=3 were hard-wired into TeX and impossible to change. Now plain TeX format supplies the old values, which are still recommended for most American publications; but you can get more hyphens by decreasing these parameters, and you can get fewer hyphens by increasing them. If the sum of \lefthyphenmin and \righthyphenmin is 63 or more, all hyphenation is suppressed. (You can also suppress hyphenation by using a font with \hyphenchar=-1, or by switching to a \language that has no hyphenation patterns or exceptions.)

4. Smarter ligatures

Now here's the most radical change. Previous versions of TeX had only one kind of ligature, in which two characters like 'f' and 'i' were changed into a single character like 'fi' when they appeared consecutively. The new TeX understands much more complex constructions by which, for example, we could change an 'i' following 'f' to a dotless 'i' while the 'f' remains unchanged: 'fi'.

As before, you get ligatures only if they have been provided in the font you are using. So let's look at the new features of METAFONT by which enhanced ligatures can be created. A METAFONT programmer can specify a "ligature/kerning program" for any character of the font being created. If, for example, the 'fi' combination appears in font

position 12, the replacement of 'f' and 'i' by 'fi' is specified by including the statement

"i" =: 12

in the ligature/kerning program for "f"; this is METAFONT's present convention.

The new ligatures allow you to retain one or both of the original characters while inserting a new one. Instead of =: you can also write |=: if you wish to retain the left character, or =:| if you wish to retain the right character, or |=:| if you want to keep them both. For example, if the dotless i appears in font position 16, you can get the behavior mentioned above by having

"i" |=: 16

in f's program.

There also are four additional operators

|=:>, =:|>, |=:|>, |=:|>>,

where each > tells TeX to shift its focus one position to the right. For example, if f and i had been replaced by f and dotless i as above, TeX would begin again to execute f's ligature/kern program, possibly inserting a kern before the dotless i, or possibly changing the f to an entirely different character, etc. But if the instruction had been

"i" |=:> 16

instead, TeX would turn immediately to the ligature/kern program for characters following character 16 (the dotless i); no further change would be made between f and i even if the font had something specified there.

5. Boundary ligatures

Every consecutive string of 'characters' read by TeX in horizontal mode (after macro expansion) can be called a 'word'. (Technically we consider a 'character' in this definition to be either a character whose \catcode is a letter or otherchar, or a control sequence that has been \let equal to such a character, or a control sequence that has been defined by \chardef, or the construction \char{number}.) The new TeX now imagines that there is an invisible "left boundary character" just before every such word, and an invisible "right boundary character" just after it. These boundary characters take effect if the font designer has specified ligatures and/or kerning between them and the adjacent letters. Thus, the first or last character of a word can now be made to change its shape automatically.

A ligature/kern program for the left boundary character is specified within METAFONT by using

the special label `||:` in a `ligtable` command. A ligature or kern with the right boundary character is specified by assigning a value to the new internal METAFONT parameter `boundarychar`, and by specifying a ligature or kern with respect to this character. The `boundarychar` may or may not exist as a real character in the font.

For example, suppose we want to change the first letter of a word from ‘F’ to ‘ff’ if we are doing some olde English. The METAFONT font designer could then say

```
ligitable ||: "F" |:= 11
```

if character 11 is the ‘ff’. The same ligtable instruction should appear in the programs for characters like (and ‘ and “ and - that can precede strings of letters; then ‘Bassington-French’ will yield ‘Bassington-ffrench’.

If the ‘s’ of our font is the pre-19th century s that looks like a mutilated ‘f’, and if we have a modern ‘s’ in position 128, we can convert the final s’s as Ben Franklin did by introducing ligature instructions such as

```
boundarychar := 255;
ligitable "s": 255 =:| 128,
". ." =:| 128,
", , " =:| 128,
")" =:| 128,
," , " =:| 128,
```

and so on. (A true oldstyle font would also have ligatures for ss and si and sl and ssi and ssl and st; it would be fun to create a Computer Modern Oldstyle.)

The implicit left boundary character is omitted by TeX if you say `\noboundary` just before the word; the implicit right boundary is omitted if you say `\noboundary` just after it.

6. More compact ligatures. Two or more ligtables can now share common code. To do this in METAFONT, you say ‘`skipto <n>`’ at the end of one `ligtable` command, then you say ‘`<n>::`’ within another. Such local labels can be reused; e.g., you can say `skipto 1` again after `1::` has appeared, and this skips to the *next* appearance of `1::`. There are 256 local labels, numbered 0 to 255. Restriction: At most 128 ligature or kern commands can intervene between a `skipto` and its matching label.

The TFM file format has been upwardly extended to allow more than 32,500 ligature/kern commands per font. (Previously there was an effective limit of 256.)

7. Better looking sloppiness

There is now a better way to avoid overfull boxes, for people who don’t want to look at their documents to fix unfeasible line breaks manually. Previously people tried to do this by setting `\tolerance=10000`, but the result was terrible because TeX would tend to consolidate all the badness in one truly horrible line. (TeX considers all badness ≥ 10000 to be infinitely bad, and all these infinities are equal.)

The new feature is a dimension parameter called `\emergencystretch`. If `\emergencystretch` is positive and if TeX has been unable to typeset a paragraph without exceeding the given tolerances, another pass over the paragraph is made in which TeX pretends that additional stretchability equal to `\emergencystretch` is present in every line. The effect of this is to scale down all the badnesses into a range where previously infinite cases become finite; TeX will find an optimum solution to the scaled-down problem, and this will be about as good as possible in a practical sense. (The extra stretching is not really present; therefore underfull boxes will be reported in warning messages unless `\hbadness` is increased.)

8. Looking at badness

TeX has a new internal integer parameter called `\badness` that records the badness of the box it has most recently constructed. If that box was overfull, `\badness` will be 1000000; otherwise `\badness` will be between 0 and 10000.

9. Looking at the line number

TeX also has a new internal integer parameter called `\inputlineno`, which contains the number of the line that TeX would show on an error message if an error occurred now. (This parameter and `\badness` are “read only” in the same way as `\lastpenalty`: You can use them in the context of a `\{number\}`, e.g., by saying ‘`\ifnum\inputlineno>\badness ... \fi`’ or ‘`\the\inputlineno`’, but you cannot set them to new values.)

10. Not looking at error context

There’s a new integer parameter called `\errorcontextlines` that specifies the maximum number of two-line pairs of context displayed with TeX’s error messages (in addition to the top and bottom lines, which always appear). Plain TeX now sets `\errorcontextlines=5`, but higher level format packages might prefer `\errorcontextlines=1` or even `\errorcontextlines=0`. In the latter case,

an error that previously involved three or more pairs of context would now appear as follows:

```
! Error.
(somewhere) The \top
line
...
1.123 \The
bottom line.
```

(If `\errorcontextlines<0` you wouldn't even see the '...' here.)

11. Output recycling

One more new integer parameter completes the set. If `\holdinginserts>0` when `TEX` is putting the current page into `\box255` for the `\output` routine, `TEX` will not move anything from insertion nodes into the corresponding boxes; all insertion nodes will stay in place. Designers of output routines can use this when they want to put the contents of box 255 back into the current page to be re-broken (because they might want to change `\vsize` or something).

12. Exceptions to upward compatibility

The new features of `TEX` and `METAFONT` imply that a few things work differently than before. I will try to list all such cases here (except when the previous behavior was erroneous due to a bug in `TEX` or `METAFONT`). I don't know of any cases where users will actually be affected, because all of these exceptions are pretty esoteric.

- `TEX` used to convert the character strings `^0, ^1, ..., ^9, ^a, ^b, ^c, ^d, ^e, ^f` into the respective single characters `p, q, ..., y, !, ", #, $, %, &`. It will no longer do this if the following character is one of the characters `0123456789abcdef`.

- `TEX` used to insert no character at the end of an input line if `\endlinechar>127`. It will now insert a character unless `\endlinechar>255`. (As previously, `\endlinechar<0` suppresses the end-of-line character. This character is normally 13 = ASCII control-M = carriage return.)

- Some diagnostic messages from `TEX` used to have the notation `["80] ... ["FF]` when referring to characters 128...255 (for example when displaying the contents of an overfull box involving fonts that include such characters). The notation `^80 ... ^ff` is now used instead.

- The expressions `char128` and `char0` used to be equivalent in `METAFONT`; now `char` is defined modulo 256 instead. Hence `char-1 = char255`, etc.

- `INITEX` used to forget all previous hyphenation patterns each time you specified `\patterns`. Now all hyphenation pattern specifications are cumulative, and you are not permitted to use `\patterns` after a paragraph has been hyphenated by `INITEX`.

- `TEX` used to act a bit differently when you tried to typeset missing characters of a font. A missing character is now considered to be a word boundary, so you will get slightly more diagnostic output when `\tracingcommands>0`.

- `TEX` and `METAFONT` will report different statistics at the end of a run because they now have a different number of primitives.

- Programs that use the string pool feature of `TANGLE` will no longer run without changes, because the new `TANGLE` starts numbering multicharacter strings at 256 instead of 128.

- `INITEX` programs must now set `\lefthyphenmin=2` and `\righthyphenmin=3` in order to reproduce their previous behavior.

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

PublC METAFONT Available

Editor's note: Klaus Thull announces that, as of 6 October, PublC `METAFONT` is available. PublC `METAFONT` compiles with Turbo Pascal v.4 or 5 and has passed the `trap` test. As with its companion, PublC `TEX` (see *TUGboat* 10#1, pp. 15-22), this program has virtual memory (and is also somewhat slow).

Work is going on at sites other than Klaus' for improving performance and video, as has been the case with PublC `TEX`. Distribution is now being handled by DANTE, the German speaking `TEX` users association. The changefile for version 0 of PublC `METAFONT` is available at

Bitnet: `listserv@dhdurz1`

A .dvi File Processing Program

Stephan v. Bechtolsheim

Introduction. This article discusses .dvi file processing programs (or DFPs for short). I will discuss such programs in general and the DFP (*dvi2dvi*) which I developed in particular. I will show that there is a variety of problems which can be solved with a DFP in a very elegant way. In particular these are: the *insertion of change bars* into a document, the *insertions of rules underlining text*, the *overlay of .dvi files*, and the extraction of *positioning information* from a .dvi file. And to prove my claims, I include a figure generated with the help of *dvi2dvi*.

What is a DFP?

A DFP is a program whose input consists of one or more .dvi files. The DFP processes these .dvi files either to generate a new .dvi file or to extract information (in particular, positioning information), or both.

The important observations in this context are the following:

1. One way to look at DFPs is to regard them as a *device independent way to extend drivers*. DFPs may evaluate \special commands (in other words commands written to a .dvi file by TeX using \special). DFPs offer the advantage that no driver needs to be modified to evaluate these \special commands (in particular, not every driver used at a specific site must be extended). Instead, a DFP is invoked performing the requested functions. The DFP may remove \special commands within a .dvi file, outputting a new .dvi file modified as per these \specials.

A DFP may be regarded, therefore, as a filter program that is positioned between the TeX program and the driver program used to print a document.

2. I will also show applications of DFPs which actually generate textual information, in many cases the TeX source for another TeX execution.

One case which falls in this category is where a DFP extracts positioning information from a .dvi file. There is no straightforward way in TeX to determine at the time a page is written to a .dvi file at what position a certain item will be printed. It is fairly straightforward, though, to let a DFP do such positioning computations and to let it write the position of

an item marked by a \special command to a text file.

Existing DFPs

It should be mentioned here that three separate .dvi file processors are already available in the TeX community. dviselect (Chris Torek) is well known and is a program which allows the selection of specific pages from a .dvi files (my DFP contains this functionality). There is also dvipaste, described in *TUGboat* 10#2, p. 164 (my DFP contains three different types of overlays, which is what I called it). Finally there is ivd2dvi [TeXhax digest, vol 89, no. 25] (I will add the functionality of this DFP later to my DFP). I do claim that *dvi2dvi* is by far the most powerful and versatile among all currently existing DFPs.

As noted before, my DFP is called *dvi2dvi* and therefore any \special command to be recognized by *dvi2dvi* must start with "dvi2dvi:".

The remainder of this article will discuss some of the applications mentioned above and will also show an example.

.dvi File Overlays

dvi2dvi defines three types of overlay:

1. *Selective overlay.* In this case the master input .dvi file contains \special commands which instruct *dvi2dvi* to print another .dvi file's page on top of the current page of the master input .dvi file. Parameters of this \special include offset values to be applied (so the page which is being "pulled in" can be positioned properly), the file name of the .dvi file to be pulled in and the page number of the page from the pulled in .dvi file which should be selected by *dvi2dvi*.

I used this feature of *dvi2dvi* writing my book "TeX in Practice." In this book I described output routines and I wanted to include sample output generated by these output routines. I left an empty page in the main text of my book, and then used *dvi2dvi* to "paste in" the output contained in a separate example .dvi file.

2. *Every page overlay.* In this application the assumption is that there is one .dvi file with exactly one page. This .dvi file may print the version number and the date of the draft of some document on top of the page. This one page .dvi file is then overlaid on top of every page of the master input .dvi file.

3. *Parallel overlay.* In this case *two .dvi* files which have identical page numbering are printed on top of each other (page *i* of the first and the second *.dvi* file overlaid from page *i* of the output *.dvi* file). Later you will see an application of this type of overlay (I used it originally to insert change bars, where the change bars were contained in the second *.dvi* file and the main document in the first *.dvi* file, but I found a more elegant way of solving the change bar problem to be discussed below).

Extracting Positioning Information

The very first version of *dvi2dvi* was a modified driver. I removed all the code which generated output for the original output device, and then I replaced the reading of pixel files by reading in *.tfm* files (a DFP needs to be able to keep track of the current position). I then added the feature of actually writing an output *.dvi* file (at this stage of course an exact copy of the original input *.dvi* file).

I then added a command line option to *dvi2dvi* which allows me to specify a text file to which positional information may be written. To be more precise: if the input *.dvi* file contains a *\special* command like

```
\special{dvi2dvi: position XXX}
```

then *dvi2dvi* writes a line to the positioning text file which consists of a macro call to macro *\XXX* with the current page number and the current position (horizontal and vertical coordinates) as parameters.

Let me now discuss an application. When composing the index for a document it is very useful to list all index terms out in the margins of a document. I did so by adding an additional parameter (an index term) to the positioning *\special* command which would become an additional parameter for *\XXX*. The position text file (generated from the main *.dvi* file) is created by *dvi2dvi*, and this file is fed to *TEX* simply for the purpose of writing all index terms out into the margins of the document (this *.dvi* file contains no other text). The resulting *.dvi* file and the *.dvi* file of the main document are then merged together using the *parallel overlay* (discussed previously) in another *dvi2dvi* execution. Note that, by omitting this step, the main document *without* the index terms printed in the margins can be generated, and there is no need to process the main document by *TEX* again.

Inserting Change Bars Into a Document

Let me now address the issue of change bars. I use *dvi2dvi* to insert the change bars directly. In other words it is *dvi2dvi* which pastes in the change bar rules. The user triggers the insertion of the change bars using macros *\ChangeBarOn* and *\ChangeBarOff*. Preceding use of these macros, a *\ChangeBarAdvice* macro call advises *dvi2dvi* of, for instance, the thickness of the change bar rules to be inserted. This call is also used to instruct *dvi2dvi* about the horizontal placement of the change bars. As you can see in the example figure on the opposite page, change bars can be placed to the left and to the right of the pages of a document (pages with odd page numbers of a double sided document appear on the right hand side and change bars are typically inserted into the outside margins; any other horizontal position could have been chosen by the user).

Macros *\ChangeBarPush* and *\ChangeBarPop* are used to turn off and back on a potentially existing change bar around floating bodies (figures and tables) or other insertions. Within those entities a change bar can be inserted if this is so desired. In other words *dvi2dvi* will automatically interrupt a change bar if the current text marked by a change bar is "interrupted" because a figure is inserted. *dvi2dvi* will also handle change bars correctly which start on one page and end on another page.

Using an algorithm similar to that used in programming the change bar problem, I was able to solve another problem which occurs when typesetting classified documents. In documents with mixed classifications (i.e. documents containing pages with different security classifications), *dvi2dvi* can be used to determine the proper classification of each page and then pages can be marked with their security classification very easily.

Font Underlining and Replacement

One other feature of *dvi2dvi* is the underlining of text. If *dvi2dvi* discovers a font definition in a *.dvi* file where the name of the font starts with *pu-* (which stands for *print* and *underline*) then the output in this font is replaced by the output to its "master font" with an underlining rule added to each character of this font. Note that the master font of, for instance, *pu-cmr10.tfm* is obviously Computer Modern Roman 10 point (*cmr10.tfm*). Note that *pu-cmr10.tfm* and *cmr10.tfm* are actually identical (*pu-cmr10.tfm* is generated by making a copy of *cmr10.tfm*). There is also the possibility to replace

Ok, here is some text. And now it's time for a change bar. So as we go on there will be a time where the change bar is turned off. Which is right here and there. Change bar off. And now let me continue this paragraph. Actually it is time to finish it.

Now let me show that dvi2dvi also handles the following case properly. First of all I will start the change bar right here. In addition to that I will now produce a \topinsert. The vertical size of this \topinsert is such that it will appear on top of the next "mini page." Note that the \topinsert text contains its own change bar.

Well, how about another paragraph. By the way the change bar of the main text is still in effect. And we will leave it in effect a little longer. More text is needed.

Well, how about another paragraph. By the way the change bar of the main

1

the change bar now. The change bar ends at the end of this paragraph (but could end at other positions too, of course).

Now let me show the underlining functions of dvi2dvi. First of all one must declare two fonts (details can be found in the article). Here are the two font declarations:

```
\font\purm = pu-cmr10
\font\urm = u-cmr10
```

Now let me use font \purm: This is some text where the font \purm is used. Now let me use \urm with the identical text: _____ \urm

The font modifications shown here can be applied to any font and are in no way restricted to the font used in this example.

Now let me present another example which shows that the positioning of the rule used to underline any text is arbitrary and under complete user control. All

3

Some text here.
Some "changed" text here.
Some text here.

text is still in effect. And we will leave it in effect a little longer. More text is needed. This is the second silly text paragraph.

Well, how about another paragraph. By the way the change bar of the main text is still in effect. And we will leave it in effect a little longer. More text is needed. This is the third silly text paragraph.

Let me finish the text now and turn off

2

the user has to do is to issue an "advice \special" to dvi2dvi and the position of the underlining rule will be changed. Here is some text using \purm again, but with a different positioning of the underline rule: Here is some more text using \purm, with a differently placed underline this time though.

Well, I may as well also show some underlined italics text to show that the approach works with other fonts too. *This is fun as far as I am concerned.*

4

text in some specific font by underlining rules only (no text). For that purpose use a *.tfm* file like, for instance, *u-cmr10.tfm*. See the figure of this article for an example.

dvi2dvi also supports font emulation where output in one font is replaced by output in a different font when the document is printed. This capability is not shown in this article.

Concluding Remarks

I hope that I was able to demonstrate the usefulness of DFPs in general and *dvi2dvi* in particular. Note that I have *not* discussed all the features of *dvi2dvi*. A 60 page long document describing *dvi2dvi* contains the description of all features plus additional macros which should be useful in applications of *dvi2dvi*.

I hope that I can encourage people to buy my DFP (yes, it costs a little money), and to port it to other operating systems (give me a call in case you are interested). Contact me at the address below and I think we can work something out. *dvi2dvi* is written in "standard C" and runs currently on a SUN running OS 3.5 (BSD 4.2). There should be no problem to port it to other operating systems with a C compiler.

Finally I would like to thank Ron Whitney for his cooperation: he had to transfer the *.dvi* file for this article to my computer to process it by *dvi2dvi* and then back to the AMS's computer for printing, a little additional inconvenience.

◊ Stephan v. Bechtolsheim
2119 Old Oak Drive
W. Lafayette, IN 47906
317-463-0162
svb@cs.purdue.edu

Notes on Russian T_EX

Dimitri Vulis

By combining the new Cyrillic fonts from the University of Washington in Seattle with my hyphenation patterns, I've been able to create a usable Russian-language version of T_EX.

Coding Cyrillic letters

The customary way to represent Russian letters in an ASCII computer is to use 8-bit coding, with capital Russian letters A-Ya in 176-207, followed by lower case a-ya in 208-239. This scheme, commonly known as GOSTCII (pronounced GOST-ski), is formally defined by the standards ISO 8859 part 5 [2] and ECMA 113. I use GOSTCII to code Russian text on my personal computer.¹ I use this coding in my Russian T_EX files, but a convenient way of entering transliterated Russian text using only 7-bit ASCII (unfortunately, different from the elegant AMS scheme that uses ligatures) is also available.

Hyphenation patterns

To create the patterns, I ran PATGEN on a dictionary of over 50,000 fully hyphenated Russian words with inflections. Remarkably, PATGEN found all the good breaks and no bad breaks, outputting 4204 patterns. I keyed in and hyphenated most of the dictionary by hand; some words were supplied by Alexander Samarin, for which I am grateful.

I started by keying in the Russian part of a pocket Russian-French dictionary, hyphenating the words manually. I then ran PATGEN to examine the patterns, and also tried them on Russian texts. I saw that the patterns did not handle inflected words well because I keyed in only the nominative/singular/masculine/infinitive (whichever are applicable) forms. Hence I inflected a number of words representative of different classes, and continued this practice when I added words later.

I also noted that a number of patterns were of the form

⟨vowel⟩ 1 ⟨consonant⟩ ⟨vowel⟩

Rather than seeking words containing all such combinations, I preloaded to PATGEN patterns of the form

¹ This can be achieved with any MS-DOS PC that supports code pages; the required software can be FTPed from SIMTEL20.ARMY.MIL as PD1:<MSDOS.SCREEN>CYRILIC2.ARC.

<vowel> 1 <consonant> <vowel>

for all vowels and all valid consonant-vowel combinations (i.e., no жя) and of the form

<vowel> 1 <consonant> ь <vowel>

for all vowels and all vowels that can follow ь, and also of the form

$\alpha 1 \alpha$ where $\alpha \in \langle \text{letter} \rangle$

for all the letters that can occur twice. In the few cases when these patterns would hyphenate dictionary words incorrectly, they are overridden by the PATGEN-generated patterns.

It was no longer necessary to add to the dictionary the words that exhibited no new patterns, like корова or дорога. I browsed through a number of dictionaries (including Ozhegov, Foreign (to Russian) Words, English-Russian Polytechnic, Mathematical, Geographical, Computer Science, Medical, Obscenities, Thieves' Jargon, and others) looking for words that exhibited new patterns (mostly unusual combinations of consonants) and added them to the dictionary, hyphenating by hand.

The rules of Russian hyphenation were first formulated by Yakov H. Grot [1] and then revised, and made less strict, in 1918 and 1956 [3; 5; 6]. Russians prefer to break their words at syllable boundaries: after a vowel and before a consonant or another vowel, except when this would result in a conspicuous-looking cluster of consonants at the beginning of the next line, as in ка-рман; in such cases they advance the break, taking care not to split certain consonant combinations, e.g., мест-ный. Derivation may take precedence over pronunciation when breaking off a prefix, e.g., вы-рвать, or when the word has been borrowed from a foreign language, e.g., дис-пе-псия. A simplified English rendering of the rules can be found in the US Government Printing Office Style Manual, the Chicago Manual of Style, and the like. The following list of words illustrates their application. (Note that some words have the last 2 letters broken off; TeX won't find these breaks.)

аб-зац-ный аб-сорб-ция адрес-ный айс-берг ал-ло-хон-ный ан-глий-ский ар-хео-пле-рикс арк-функ-ция арт-об-стрел ас-фальт-ный асин-хрон-ный астро-навт ах-нуть без-дна бес-ком-про-мисс-ный блок-схема бой-сказуат-ский борт-про-вод-ни-ца бу-кварь бур-жуаз-ный бух-гал-тер взро-слый ви-део-уси-ли-тель во-жди во-сем-на-дцать воль-фрам вольт-метр вось-ми-раз-ряд-ный впол-ли-сты все-гда все-общ-ность вы-жжен-ный вы-рвать глас-ные го-ло-во-тяп-ство го-мео-морф-ный

го-мо-сек-су-а-ли-сты горш-ко-ый гос-цирк гра-мот-ный гро-мозд-кий гроз-дьях гроc-бух гуа-шью дву-языч-ный ден-знак дер-жать ди-влюсь диа-гно-сти-ка ду-плекс-ный жем-чуж-ный жуж-жать за-вши-вать за-мкну-тый за-мще-вый зав-лаб затх-лый злост-ный зоо-гео-gra-фия из-вест-ный из-да-тельств изо-ане-мо-на изы-скан-ный им-пло-зив-ный им-пульс-ных иму-ще-ство ин-верс-ный ин-декс-ный ин-клю-зив-ный ин-те-граle ин-тер-ак-тив-ный ин-фра-крас-ный ис-клю-чать ис-кро-уло-ви-тель-ный ис-тлеть их-тио-за-вры ка-мен-но-уголь-ный ка-пуст-ный каз-нью ква-дра-тич-ный квинт-эс-сен-ция ки-ло-ватт-метр класс-ный ко-манд-ный кол-хоз ком-пью-тер комс-орг конц-ла-геръ кре-стьян-ский крест-цо-вый кри-пто-си-сте-ма крио-элек-трон-ный кросс-эму-ля-тор ку-плю ку-рьеz-ный культ-про-свет-ра-бот-ник кунст-ка-ме-ра кур-орт ла-текс-ный ланд-шафт-ный ландс-кнх-тах ле-минс-ка-той лег-ко-атле-ти-ка лек-се-ма лин-гви-сти-че-ский ло-га-риф-ми-че-ский лог-ич-еский ло-каль-ный лох-ма-тый луч-ший львом льня-ной ма-ну-скри-пты марк-ист-ский мас-штаб-ный мат-обес-пе-че-ние ме-тео-стан-ция ме-ча меж-атом-ный мест-ный микс-ту-ра мин-здрав мно-го-уголь-ник мо-кий мор-фем-ный на-взрыд на-гра-ждать на-из-уть на-име-но-ван-ный наи-выс-ший не-аде-кват-ный не-ест-ствен-ный не-льзя нем-цах нео-ло-гизм неф-тя-ной но-ябрь-ские обл-ис-пол-ком обо-льстить обо-рвыш общ-ность оглох-ший од-но-днев-ка ок-тябрь-скую орг-вы-во-ды осво-бо-жда-ет оскор-бле-ние осле-пли осу-ще-ствлял-ся от-мстить отра-вле-ние па-лео-био-гео-gra-фия па-три-ар-хат парт-ак-тив парт-съезд паст-би-ще пе-ре-жешь при-льнуть при-мкнув-ший при-со-еди-нить про-бле-мой про-гно-зах про-грамм-ный про-образ про-цесс-ный про-чут проф-вред-ность псев-до-ана-ли-ти-че-ский пя-ти-этаж-ный ра-дио-изо-топ ра-зы-гры-вать рав-ный разъ-езд-ной рай-он-ный рас-чет рас-ши-фро-вать ре-ко-гинос-ци-ров-ка ро-жде-ние ру-блей ряв-кнуть са-мо-очист-ка сак-во-я-жик сап-фири сверх-опе-ра-тив-ный свое-образ-ный се-го-дняш-ний сем-на-дца-тый си-зи-гий-ный си-сте-мо-тех-ни-ка син-хро-сиг-нал скольз-кий смеж-ник смерт-ность со-впа-де-ние со-еди-нить со-лжешь со-мна-бу-ла сов-ин-форм-бю-ро сот-ник соц-культ-быт спец-от-дел

срав-нить сред-ство сте-рео-угол стерж-нем су-ма-сшед-ший та-блич-ный те-ле-съем-ка те-тра-дью тек-сту-аль-ный тер-мо-ядер-ный тер-плю то-жде-ство транс-фи-ни-тую трех-адрес-ный три-фонг уль-тра-основ-ной успеш-ный уст-ный устрои-ств утвер-ждать уточ-нять фак-си-ми-ле фак-тор-ал-ге-бра фото-вспыш-ка фырк-нуть хаус-дор-фо-во хри-пло храст-нуть ци-кло-и-да ци-фро-вых че-рес-чур че-ты-рех-уголь-ник чер-ствый чест-ный чи-слен-ный чист-кой чув-ство шах-тер ши-фров-кой штрих-пунк-тир-ный эв-фе-мизм экзэм-пляр элек-tron

The dictionary contains a large number of:

1. words borrowed from foreign languages (mostly technical and engineering terms); PATGEN is very good at understanding that one should break дур-шлаг, but марш-рут;
2. abbreviations (сложносокращенные слова), e.g., гос-за-каз, парт-учеба, комс-орг. These are not really part of the language, but occur often in newspapers and technical literature. The number of words usually abbreviated for such compounds is finite and the patterns are very good at identifying them. It is possible to construct abbreviations that these patterns will not hyphenate properly; when in doubt, one has to use \- in such words;
3. compound words. The patterns will not split a single vowel off a part of many compound words. Thus, прямоу-гольник, би-олог, нео-бычный, не-офашизм, etc., are suppressed. Such breaks are not strictly illegal, but they don't look good, and a better break is only a letter away. Once again, I took advantage of the fact that the number of words usually used in compounds is manageable small. A sufficient number of examples given to PATGEN eliminates the unwanted breaks in many compound words that are not in the dictionary.

A preliminary version of this work was presented in my M.A. thesis, "An Implementation of Liang's Algorithm for the Russian language", submitted to CCNY in October of 1988.

After the hyphenation patterns were complete, A. Samarin graciously sent me the paper [4] describing a non-TeX algorithm for hyphenating Russian text. Pavlova's algorithm produces the same hyphenations that I produced by hand, except for a few "special" words, like нововведение that it does not handle correctly; it breaks the words containing the letter combinations вн, сн, and x+consonant

differently: ди-вный, ме-стный, ша-хта, which is unusual; and there are other minor differences.

In order to get PATGEN to run under MS-DOS, I had to concoct a very long .CH file, based on the Kellerman and Smith VAX change file. I will be happy to discuss it with anyone trying to get PATGEN to work. In order to trick PATGEN into processing GOSTCII input, I used the following:

```

xord[chr(208)] := "A";
xord[chr(209)] := "B";
xord[chr(210)] := "C";
...
xord[chr(231)] := "X";
xord[chr(232)] := "Y";
xord[chr(233)] := "Z";
xord[chr(234)] := "[";
xord[chr(235)] := "\";
xord[chr(236)] := "]";
xord[chr(237)] := "^";
xord[chr(238)] := "_";
xord[chr(239)] := "'";
...
xchr["A"] := chr(208);
xchr["B"] := chr(209);
xchr["C"] := chr(210);
...
xchr["X"] := chr(231);
xchr["Y"] := chr(232);
xchr["Z"] := chr(233);
xchr["["] := chr(234);
xchr["\"] := chr(235);
xchr["]"] := chr(236);
xchr["^"] := chr(237);
xchr["_"] := chr(238);
xchr["'"] := chr(239);
@z

@x
@d cmin="@@"
@d cmax="Z"
@d edge_of_word="@@"
@y
@d cmin="@@"
@d cmax="`"
@d edge_of_word="@@"
@z

```

Russian TeX

I used with my Russian TeX the Cyrillic fonts kindly mailed to me by Thomas Ridgeway, the director of the Humanities and Arts Computing Center at U. of Washington, Seattle. Their organization is similar to that of the Cyrillic fonts developed

by AMS in MF79. In particular, TeX's ligature mechanism is used to enter certain Russian letters. For example, to type the word жней, one enters its MR transliteration zhnets. If the current font is Latin, the transliteration is printed out; and if the current font is Cyrillic, then the letters zh, taken as a ligature, produce character '031, which is ж in the font, while ts, as a ligature, produce т. If the text is being set in Latin, then it produces its transliteration. A problem arises when a letter combination used for a ligature actually occurs in a word. For example, to enter the word отсеб one has to type ot{\cydot}sev, where {\cydot} has to be defined as kern0pt for Cyrillic, to suppress the ligature, and as \$`cdot\$ for Latin, to transliterate the word as "ot·sev". When hyphenation is desired, the explicit kern interferes with it. Moreover, one of the hyphenation patterns is 2t1s, meaning that entering otsenka for оценка is liable to result in от·сенка being hyphenated. The fault, of course, lies with the transliteration scheme.

Thus, I had to abandon this elegant ligature scheme and to define the following control sequences to enter Russian letters that have no obvious Latin equivalents.

```
\chardef\Zh='021
\chardef\zh='031
\chardef\Ui='022
\chardef\ui='032
\chardef\Kh='110
\chardef\kh='150
\chardef\Ts='103
\chardef\ts='143
\chardef\Ch='121
\chardef\ch='161
\chardef\Sh='130
\chardef\sh='170
\chardef\Shch='127
\chardef\shch='167
\chardef\cDprime='137
\chardef\cdprime='177
\chardef\cPrime='136
\chardef\cprime='176
\chardef\Ee='003
\chardef\ee='013
\chardef\Yu='020
\chardef\yu='030
\chardef\Ya='027
\chardef\ya='037
```

I used PLtoTF and TFtoPL, TeXware programs, to delete all the ligatures in the Cyrillic fonts except those for quotes and dashes. The examples above would be entered as {\zh}ne{\ts}, otsev

and of\ts}enka. When Russian text is being transliterated, the control sequences need to be redefined:

```
\def\Zh{\t{Z}{h}}
\def\zh{\t{z}{h}}
\def\Ui{\u{I}}
\def\uif{\u{i}}
\def\Kh{\t{K}{h}}
\def\kh{\t{k}{h}}
\def\Ts{\t{T}{s}}
\def\ts{\t{t}{s}}
\def\Ch{\t{C}{h}}
\def\ch{\t{c}{h}}
\def\Sh{\t{S}{h}}
\def\sh{\t{s}{h}}
\def\Shch{\t{S}{h}\t{c}{h}}
\def\shch{\t{s}{h}\t{c}{h}}
\def\cprime{$`$\prime$}
\def\cPrime{$`\underline{$`$\prime$}$}
\def\cdprime{$`$\prime$\prime$}
\def\cDprime{$`\underline{$`$\prime$\prime$}$}
\def\Ee{\`{E}}
\def\ee{\`{e}}
\def\Yu{\t{Y}{u}}
\def\yu{\t{y}{u}}
\def\Ya{\t{Y}{a}}
\def\ya{\t{y}{a}}
```

The control sequence {\cydot} is no longer needed and the tie accent indicates when a single Russian letter is transliterated by two Latin ones. The ability to change the transliteration scheme is an additional benefit:

```
\def\Zh{\v{Z}}
\def\zh{\v{z}}
\def\Ui{\J}
\def\ui{\j}
\def\Kh{\Ch}
\def\kh{\ch}
\def\Ts{\C}
\def\ts{\c}
\def\Ch{\v{C}}
\def\ch{\v{c}}
\def\Sh{\v{S}}
\def\sh{\v{s}}
\def\Shch{\v{S}\v{c}}
\def\shch{\v{s}\v{c}}
\def\cprime{\kern0pt`\kern0pt\relax}
\def\cPrime{\kern0pt`\kern0pt\relax}
\def\cdprime{\kern0pt`\kern0pt`\kern0pt\relax}
\def\cDprime{\kern0pt`\kern0pt`\kern0pt\relax}
\def\Ee{\`{E}}
\def\ee{\`{e}}
\def\Yu{\t{J}{u}}
\def\yu{\t{j}{u}}
\def\Ya{\t{J}{a}}
\def\ya{\t{j}{a}}
```

As with the ligature scheme, it is the user's responsibility to switch the meanings of control sequences together with the fonts.

There are 32 letters in the Russian alphabet, and only 26 in the English one. For this reason, before the \patterns command can be used in INITEX, it is necessary to extend the \uccode and \lccode tables for the letters ж, ў, э, ю, й, ъ, and я.

I use a short PASCAL program to translate GOSTCII letters into Latin letters and control sequences before it can be fed to TeX. The proposed version 3.0 of TeX will accept 8-bit input, making this preprocessor unnecessary; and its enhanced handling of ligatures and hyphenation may make it unnecessary to enter as many as 10 characters to produce a single Russian letter, when GOSTCII is not used. For Russian \language, these patterns should work with \left and \righthyphenmin=2.

There is a Bitnet mailing list dedicated to the discussion of the Russian TeX project. To subscribe, send the command

```
SUB RUSTEX-L <your name>
to
Bitnet: LISTSERV@UBVM
To submit an article, mail it to
Bitnet: RUSTEX-L@UBVM
```

(Note that node UBVM on BITNET is the same as node UBVM.CC.BUFFALO.EDU on Internet.)

Acknowledgements

I would like to thank Barbara Beeton for the truly incredible amount of help, support, and warm encouragement, all rendered via Internet; Alexander Samarin for the voluminous advice and support, again rendered via Internet; Donald Knuth, both for his interest in this project, and for creating TeX in the first place. Last but not least, I would like to thank Frank Liang for his doctoral research; without him, we would still be breaking перес-тройка.

Bibliography

- [1] Яков Гrot = Yakov Grot. Русское Правописание: Руководство = Russian Orthography: a Guide.
- [2] International Organization for Standardization, Standard 8859-5: 1989, Information Processing—8-bit Single-Byte Coded Graphic Character Sets—Part 5: Latin/Cyrillic Alphabet.

- [3] В. Ф. Иванова = V. F. Ivanova. Современный русский язык = Modern Russian language. Moscow, Prosveshchenie, 1976.
- [4] Ю. Г. Павлов, В. А. Павлова, А. П. Соколов = Yu. G. Pavlov, V. A. Pavlova, A. P. Sokolov. Алгоритм автоматизированного переноса русских слов = An algorithm for automatic hyphenation of Russian words, Institute of High Energy Physics, Preprint #83-72, Serpukhov, 1983.
- [5] А. И. Кайдалова, И. К. Калинина = A. I. Kaidalova, I. K. Kalinina. Современная Русская Орфография = Modern Russian Orthography, p. 189. Moscow, Vysshaya Shkola, 1973.
- [6] Д. Э. Розентал = D. E. Rozental, ed. Современный Русский Язык = Modern Russian language, v. 5, §100. Moscow, Vysshaya Shkola, 1986.

◊ Dimitri Vulis
Department of Mathematics
Graduate Center
City University of New York
33 West 42nd Street
New York, NY 10036-8099
Bitnet: dlv@cunyvms1

Hyphenation Exception Log

Barbara Beeton

This is the annual update of the list of words that TeX fails to hyphenate properly. The list last appeared in Volume 9, No. 3, starting on page 239. Everything listed there is repeated here. Owing to the length of the list, it has been subdivided into two parts: English words, and names and non-English words that occur in English texts.

This list is specific to the hyphenation patterns that appear in the original *hyphen.tex*, that is, the patterns for American English. In the future, if such information becomes available, exceptions to other patterns will also be listed. See below, "Hyphenation for languages other than English".

In the list below, the first column gives results from TeX's \showhyphens{...}; entries in the

second column are suitable for inclusion in a `\hyphenation{...}` list.

In most instances, inflected forms are not shown for nouns and verbs; note that all forms must be specified in a `\hyphenation{...}` list if they occur in your document.

Thanks to all who have submitted entries to the list. Since some suggestions demonstrated a lack of familiarity with the rules of the hyphenation algorithm, here is a short reminder of the relevant idiosyncrasies. Hyphens will not be inserted before the second letter, nor after the third-from-last letter of a word; thus no word shorter than five letters will be hyphenated. (For the details, see *The T_EXbook*, page 454. For a digression on other views of hyphenation rules, see below under "English hyphenation".) This particular rule is violated in some of the words listed; however, if a word is hyphenated correctly by T_EX except for "missing" hyphens at the beginning or end, it has not been included here.

Some other permissible hyphens have been omitted for reasons of style or clarity. While this is at least partly a matter of personal taste, an author should think of the reader when deciding whether or not to permit just one more break-point in some obscure or confusing word. There really are times when a bit of rewriting is preferable.

One other warning: Some words can be more than one part of speech, depending on context, and have different hyphenations; for example, 'analyses' can be either a verb or a plural noun. If such a word appears in this list, hyphens are shown only for the portions of the word that would be hyphenated the same regardless of usage. These words are marked with a '*'; additional hyphenation points, if needed in your document, should be inserted with discretionary hyphens.

The reference used to check these hyphenations is *Webster's Third New International Dictionary*, Unabridged.

English hyphenation

It has been pointed out to me that the hyphenation rules of British English are based on the etymology of the words being hyphenated as opposed to the "syllabic" principles used in the U.S. Furthermore, in the U.K., it is considered bad style to hyphenate a word after only two letters.

In order to make T_EX defer hyphenation until after three initial letters, some new patterns can

be added, as communicated to me my Donald Knuth:

To suppress hyphenation after two letters, you need new patterns of the form `.ab6` for all pairs of letters `ab` that begin words of English. I think the number of such pairs is well under 200.

Running PATGEN on a British, rather than a U.S., dictionary would probably result in a useful, but smaller, set of patterns, as more ambiguities might be expected in an etymologically-segmented word base. This is just a guess; I would be interested in a report on actual results, if anyone has tried it.

Hyphenation for languages other than English

Patterns now exist for many languages other than English. Michael Ferguson, the father of Multilingual T_EX (also known as MLT_EX), has volunteered to collect these hyphenation patterns for TUG, and I have volunteered him to collect exceptions to those patterns as well. If you have any patterns or exceptions to contribute, please send them to

`mike@inrs-telecom.quebec.ca`

via the Bitnet.

At the top of pattern files, please include full documentation, including the name of the author and an address at which the author or other knowledgeable contact can be reached. For exceptions, please specify the patterns on which they fail, and for each word the "bad" hyphenation and the correct hyphenation; a transcript of a T_EX session with `\showhyphens` is also useful.

I will request a report on the collection for the next issue, and exceptions will be published annually along with the (American) English list.

The List — English words

academy(ies)	acad-e-my(ies)
addable	add-a-ble
ad-di-ble	add-i-ble
adrenaline	adren-a-line
af-terthought	af-ter-thought
agronomist	agron-o-mist
am-phetamine	am-phet-a-mine
anal-yse	an-a-lyse
anal-y-ses	analy-ses*
anomaly(ies)	anom-aly(ies)
an-tideriva-tive	an-ti-deriv-a-tive
anti-nomy(ies)	an-tin-o-my(ies)
antin-u-clear	an-til-u-clear
antin-u-cleon	an-til-u-cle-on
an-tirev-o-lution-ary	an-til-rev-o-lution-ary

apotheoses	apoth-e-o-ses	crosshatch(ed)	cross-hatch(ed)
apotheo-sis	apoth-e-o-sis	dachshund	dachs-hund
ap-pendix	ap-pen-dix	database	data-base
archipelago	arch-i-pel-agو	dat-a-p-ath	data-path
archety-pal	ar-che-typ-al	declarable	de-clar-able
archetyp-i-cal	ar-che-typ-i-cal	defini-tive	de-fin-i-tive
assignable	as-sign-a-ble	delectable	de-lec-ta-ble
as-sig-nor	as-sign-or	democratism	de-moc-ra-tism
as-sis-tantship	as-sist-ant-ship	de-mos	demos
asymp-tomatic	asymp-to-matic	deriva-tive	de-riv-a-tive
asymp-totic	as-ymp-tot-ic	diffraet	dif-fract
asyn-chronous	asyn-chro-nous	di-rer	dire
atheroscle-ro-sis	ath-er-o-scle-ro-sis	di-re-ness	dire-ness
at-mo-sphere	at-mosphere	dis-traugh-tly	dis-traught-ly
at-tributed	at-trib-uted	dis-tribute	dis-trib-ute
at-tributable	at-trib-ut-able	dol-lish	doll-ish
avoirdupois	av-oir-du-pois	drif-tage	drift-age
awo-ken	awok-en	driver(s)	dri-ver(s)
ban-dleader	band-leader	dromedary(ies)	drom-e-dary(ies)
bankrupt(cy)	bank-rupt(-cy)	duopolist	du-op-o-list
ba-ronies	bar-onies	duopoly	du-op-oly
base-li-neskip	\base-line-skip	eco-nomics	eco-nom-ics
bathymetry	ba-thym-e-trу	economist	econ-o-mist
bathyscaphe	bathy-scaphe	elec-trome-chan-i-cal	electro-mechan-i-cal
bea-nies	bean-ies	elec-tromechano-a-cous-tic	electro-mechano-acoustic
be-haviour	be-hav-iour	eli-tist	elit-ist
be-vies	bevies	en-trepreneur(ial)	en-tre-pre-neur(-ial)
bib-li-o-graphis-che	bib-li-o-gra-phi-sche	epinephrine	ep-i-neph-rine
bid-if-fer-en-tial	bi-dif-fer-en-tial	equiv-ari-ant	equi-vari-ant
bil-l-able	bill-able	ethy-lene	eth-y-lene
biomath-e-mat-ics	bio-math-e-mat-ics	ev-er-sible	ever-si-ble
biomedicine	bio-med-i-cine	ev-ert(s,ed,ing)	ever(s-ed,-ing)
biorhythms	bio-rhythms	ex quisite	ex-quisi-te
blan-der	bland-er	ex-traor-di-nary	ex-tra-or-di-nary
blan-d-est	bland-est	fermions	fermi-ons
blin-der	blind-er	flag-el-lum(la)	fla-gel-lum(-la)
blon-des	blondes	flammables	flam-ma-bles
blueprint	blue-print	fledgling	fledg-ling
bornolog-i-cal	bor-no-log-i-cal	flowchart	flow-chart
bo-tulism	bot-u-lism	formidable(y)	for-mi-da-ble(y)
brus-quer	brusquer	forsythia	for-syth-ia
bus-ier	busier	forthright	forth-right
bus-i-est	busiest	freeloader	free-loader
buss-ing	bussing	friendlier	friend-lier
but-ted	butted	frivolous	friv-o-lous
buz-zword	buzz-word	ga-some-ter	gas-om-e-ter
ca-caphony(ies)	ca-caph-o-ny(ies)	geodesic	ge-o-des-ic
cam-er-a-men	cam-era-men	geode-tic	ge-o-det-ic
cartwheel	cart-wheel	ge-o-met-ric	geo-met-ric
catar-rhs	ca-tarrhs	geotropism	ge-ot-ro-pism
catas-trophic	cat-a-stroph-ic	gnomon	gno-mon
catas-troph-i-cally	cat-a-stroph-i-cally	grievance	griev-ance
cauliflower	cau-li-flow-er	grievous(ly)	griev-ous(-ly)
cha-parral	chap-ar-ral	hairstyle	hair-style
chartreuse	char-treuse	hairstylist	hair-styl-ist
cholesteric	cho-les-teric	harbinger	har-bin-ger
cigarette	cig-a-rette	harlequin	har-le-quin
cin-que-foil	cinque-foil	hatcheries	hatch-eries
cognac	co-gnac	hemoglobin	he-mo-glo-bin
cog-nacs	co-gnacs	hemophilia	he-mo-phil-ia
comptroller	comp-trol-ler	hep-atic	he-pat-ic
congress	con-gress	hermaphrodite(ic)	her-maph-ro-dite(-ic)
crankshaft	crank-shaft	heroes	he-roles
crocodile	croc-o-dile	hex-a-dec-i-mal	hexa-dec-i-mal

holon-omy	ho-lo-no-my	mo-noen-er-getic	mono-en-er-getic
ho-mo-th-e-tic	ho-mo-thetic	monopole	mono-pole
horseradish	horse-rad-ish	monopoly	mo-nop-o-ly
hy-potha-la-mus	hy-po-thal-a-mus	monos-pline	mono-spline
ide-als	ideals	monos-trofic	mono-strofic
ideo-graphs	ideo-graphs	mono-tonies	mo-not-o-nies
id-i-o-syn-crasy	idio-syn-crasy	monotonous	mo-not-o-nous
ig-niter	ig-nit-er	mo-ro-nism	mo-ron-ism
ig-n-i-tor	ig-ni-tor	mosquito	mos-qui-to
ig-nores-paces	\ignore-spaces	mu-d-room	mud-room
impedances	im-ped-ances	mul-ti-faceted	mul-ti-fac-eted
in-finitely	in-fin-i-te-ly	mul-ti-pli-ca-ble	mul-ti-plic-able
in-finites-i-mal	in-fin-i-tes-i-mal	mul-tiuser	multi-user (better with explicit hyphen)
in-fras-truc-ture	in-fra-struc-ture	ne-ofields	neo-fields
in-ter-dis-ci-plinary	in-ter-dis-ci-pli-nary	none-mer-gency	non-emer-gency
inu-tile	in-utile	nonequiv-ari-ance	non-equi-vari-ance
inu-til-i-ty	in-util-i-ty	noneu-clidean	non-euclid-ean
ir-re-vo-ca-ble	ir-rev-o-ca-ble	non-i-so-mor-phic	non-iso-mor-phic
itinerary(ies)	itin-er-ary(ies)	nonpseu-do-com-pact	non-pseudo-com-pact
jeremi-ads	je-re-mi-ads	non-s-mooth	non-smooth
keystroke	key-stroke	nore-pinephrine	nor-ep-i-neph-rine
kil-ning	kiln-ing	nutcracker	nut-crack-er
la-ciest	lac-i-est	oer-st-eds	oer-steds
lamentable	lam-en-ta-ble	oligopolist	oli-gop-o-list
land-sca-per	land-scrap-er	oligopoly(ies)	oli-gop-o-ly(ies)
larceny(ist)	lar-ce-ny(-ist)	orangutan	orang-utan
lightweight	light-weight	orthodon-tist	or-tho-don-tist
limousines	lim-ou-sines	orthok-er-a-tol-ogy	or-tho-ker-a-tol-ogy
linebacker	line-backer	or-thoni-tro-toluene	ortho-nitro-toluene (or-tho-ni-tro-tol-u-ene)
lines-pac-ing	\line-spacing	ox-i-dic	ox-id-ic
lithographed	lith-o-graphed	painlessly	pain-less-ly
lithographs	lith-o-graphs	pal-mate	palmate
lobotomy(ize)	lo-bot-omy(-ize)	paradigm	par-a-digm
lo-ges	loges	parabolic	par-a-bol-ic
macroe-co-nomics	macro-eco-nomics	paraboloid	pa-rab-o-loid
malapropism	mal-a-prop-ism	parachute	para-chute
manuscript	man-u-script	paradimethyl-ben-zene	para-di-methyl-benzene (para-di-meth-y-lben-zene)
marginal	mar-gin-al	paraflu-o-ro-toluene	para-fluoro-toluene (para-flu-o-ro-tol-u-ene)
mat-tes	mattes	para-g-ra-pher	para-graph-er
med-i-caid	med-ic-aid	par-ale-gal	para-le-gal
medi-o-crities	medi-oc-ri-ties	para-m-ag-netism	para-mag-net-ism
me-galith	mega-lith	paramedic	para-medic
metabolic	meta-bol-ic	param-ethy-lanisole	para-methyl-anisole (para-meth-yl-an-is-ole)
metabolism	me-tab-o-lism	parametrize	pa-ram-e-trize
met-a-lan-guage	meta-lan-guage	paramil-i-tary	para-mil-i-tary
metropo-lis(es)	me-trop-o-lis(es)	paramount	para-mount
metropol-i-tan	met-ro-pol-i-tan	pathogenic	path-o-gen-ic
mi-croe-co-nomics	micro-eco-nomics	pee-vish(ness)	peev-ish(-ness)
mi-crofiche	mi-cro-fiche	pen>tagon	pen-ta-gon
mil-lage	mill-age	petroleum	pe-tro-le-um
milliliter	mil-li-liter	phe-nomenon	phe-nom-e-non
mimeographed	mimeo-graphed	philatelist	phi-lat-e-list
mimeographs	mimeo-graphs	phos-pho-ric	phos-phor-ic
mimi-cries	mim-ic-ries	pi-cador	pic-a-dor
mi-nis	min-is	pi-ran-has	pi-ra-nhas
min-uter(est)	mi-nut-er(-est)	pla-ca-ble	placa-ble
mis-chieviously	mis-chie-vous-ly	plea-sance	pleas-ance
mis-ers	mi-sers	poltergeist	pol-ter-geist
mis-ogamy	mi-sog-a-my	Polyene	poly-ene
mod-elling	mod-el-ling		
molecule	mol-e-cule		
monar-chs	mon-archs		
mon-eylen-der	money-len-der		
monochrome	mono-chrome		

polyethy-lene	poly-eth-y-lene	robotics	ro-bot-ics
polygamist(s)	po-lyg-a-mist(s)	roundtable	round-table
poly-go-niza-tion	polyg-on-i-za-tion	salesclerk	sales-clerk
polyphonous	po-lyph-o-nous	salesclerks	sales-clerks
polystyrene	poly-styrene	saleswoman(en)	sales-woman(en)
pomegranate	pome-gran-ate	salmonella	sal-mo-nel-la
poroe-las-tic	poro-elas-tic	sarsaparilla	sar-sa-par-il-la
postam-ble	post-am-ble	sauerkraut	sauer-kraut
postscript	post-script	sca-to-log-i-cal	scat-o-log-i-cal
pos-tu-ral	pos-tur-al	schedul-ing	shed-ul-ing
pream-ble	pre-am-ble	schizophrenic	schiz-o-phrenic
preloaded	pre-loaded	schnauzer	schnau-zer
pre-pro-ces-sor	pre-proces-sor	schoolchild(ren)	school-child(-ren)
pre-s-plitt-ing	\pre-split-ting	schoolteacher	school-teacher
priestesses	priest-esses	scy-thing	scyth-ing
pro-ce-du-ral	pro-ce-dur-al	semaphore	sem-a-phore
proc-ess	process	semester	se-mes-ter
procu-rance	pro-cur-ance	semidef-i-nite	semi-def-i-nite
pro-ge-nies	prog-e-nies	semi-ho-mo-th-et-ic	semi-ho-mo-thet-ic
progeny	prog-e-ny	semiskilled	semi-skilled
pro-hibitive(ly)	pro-hib-i-tive(-ly)	seroepi-demi-o-log-i-cal	sero-epi-de-mi-o-log-i-cal
prosci-utto	pro-sci-utto	ser-vomech-a-nism	ser-vo-mech-anism
protestor(s)	pro-test-er(s)	setup	set-up
protestor(s)	protes-tor(s)	severely	se-ver-ely
pro-to-ty-pal	pro-to-typ-al	sha-peable	shape-able
pseu-dod-if-fer-en-tial	pseu-do-dif-fer-en-tial	shoestring	shoe-string
pseud-ofi-nite	pseu-do-fi-nite	sidestep	side-step
pseud-ofinitely	pseu-do-fi-nite-ly	sideswipe	side-swipe
pseud-o-forces	pseu-do-forces	skyscraper	sky-scraper
pseudonym	pseu-do-nym	smokestack	smoke-stack
pseu-doword	pseu-do-word	snorke-l-ing	snor-kel-ing
psychedelic	psy-che-del-ic	solenoид	so-le-noid
psy-chs	psychs	so-lute(s)	solute(s)
pubescence	pu-bes-cence	sovereign	sov-er-eign
quadrat-ics	qua-drat-ics	specious	spe-cious
quadra-ture	quad-ra-ture	spelunker	spe-lunk-er
quadriplegic	quad-ri-pleg-ic	spendthrift	spend-thrift
quain-ter(est)	qua-int-er(est)	spheroid(al)	spher-oid(al)
quasiequiv-a-lence	qua-si-equiv-a-lence	sphin-ges	sphin-ges
quasi-hy-ponor-mal	qua-si-hy-po-nor-mal	spic-i-ly	spic-i-ly
quasir-ad-i-cal	qua-si-rad-i-cal	spinors	spin-ors
quasiresid-ual	qua-si-resid-ual	spokeswoman(en)	spokes-woman(en)
qua-sis-mooth	qua-si-smooth	sportscast	sports-cast
qua-sis-ta-tion-ary	qua-si-sta-tion-ary	sportively	spor-tive-ly
qua-a-si-tri-an-gu-lar	qua-si-tri-an-gu-lar	sportswear	sports-wear
quintessence	quin-tes-sence	sportswriter	sports-writer
quintessen-tial	quin-tes-sen-tial	sprightlier	spright-liер
rab-bitry	rab-bit-ry	squeamish	squea-mish
ra-dio-g-ra-phy	ra-di-og-ra-phy	stan-dalone	stand-alone
raf-f-fish(ly)	raff-ish(-ly)	startling(ly)	star-tling(ly)
ramshackle	ram-shackle	statis-tics	sta-tis-tics
ravenous	rav-en-ous	stealthily	stealth-ily
re-ar-range-ment	re-arrange-ment	steeplechase	steeple-chase
re-ciproci-ties	rec-i-proc-i-ties	stochas-tic	sto-chas-tic
reci-procity	rec-i-proc-i-ty	straight-est	straight-est
rect-an-gle	rec-tan-gle	strangeness	strange-ness
ree-cho	re-echo	stratagem	strat-a-gem
restorable	re-stor-able	stretchier	stretch-i-er
ret-ri-bu-tion(s)	ret-ri-bu-tion(s)	stronghold	strong-hold
retrofit(ted)	retro-fit(-ted)	stupi-der(est)	stu-pid-er(est)
rhinoceros	rhi-noc-er-os	summable	sum-ma-bile
righ-teous(ness)	right-eous(-ness)	su-perego	super-ego
ring-leader	ring-leader	su-pere-gos	super-egos
robot	ro-bot	supremacist	su-prema-cist

surveil-lance	sur-veil-lance	Di-jk-stra	Dijk-stra
swim-m-ingly	swim-ming-ly	dy-namis-che	dy-na-mi-sche
symp-tomatic	symp-to-matic	En-glish	Eng-lisch
syn-chromesh	syn-chro-mesh	Eu-le-rian	Euler-ian
syn-chronous	syn-chro-nous	Febru-ary	Feb-ru-ary
syn-chrotron	syn-chro-tron	Forschungs-in-sti-tut	For-schungs-in-sti-tut
talkative	talk-a-tive	funk-t-sional	funk-tsional
tapestry(ies)	ta-pes-try(ies)	Gaus-sian	Gauss-ian
tarpaulin	tar-pau-lin	Greif-swald	Greifs-wald
tele-g-ra-pher	te-leg-ra-pher	Grothendieck	Grothen-dieck
telekinetic	tele-ki-net-ic	Grundlehren	Grund-leh-ren
teler-obotics	tele-ro-bot-ics	Hamil-to-nian	Hamil-ton-ian
testbed	test-bed	Her-mi-tian	Her-mit-ian
tha-la-mus	thal-a-mus	Jan-u-ary	Jan-u-ary
ther-moe-las-tic	ther-mo-elas-tic	Japanese	Japan-ese
times-tamp	time-stamp	Kadomt-sev	Kad-om-tsev
to-po-graph-i-cal	topo-graph-i-cal	Karl-sruhe	Karls-ruhe
to-ques	toques	Ko-rteweg	Kor-te-weg
traitorous	trai-tor-ous	Leg-en-dre	Le-gendre
transceiver	trans-ceiver	Le-ices-ter	Leices-ter
transgress	trans-gress	Lip-s-chitz(ian)	Lip-schitz(-ian)
transvestite	trans-vest-i-te	Manch-ester	Man-ches-ter
traversable	tra-vers-a-ble	Marko-vian	Mar-kov-ian
traver-sal(s)	tra-ver-sal(s)	Mas-sachusetts	Mass-a-chu-setts
treacheries	treach-eries	Ni-jmegen	Nij-me-gen
treach-ery	treach-ery	Noethe-rian	Noe-ther-ian
troubadour	trou-ba-dour	No-ord-wi-jk-er-hout	Noord-wijker-hout
turnaround	turn-around	Novem-ber	No-vem-ber
ty-pal	typ-al	Poincare	Poin-care
unattached	un-at-tached	Po-ten-tial-gle-ichung	Po-tent-ial-glei-chung
unerringly	un-err-ing-ly	rathskeller	raths-kel-ler
un-friendly(ier)	un-friend-ly(i-er)	Rie-man-nian	Rie-mann-ian
va-guer	vaguer	schot-tis-che	schot-tische
vaudeville	vaude-ville	Schrodinger	Schro-ding-er
vi-cars	vic-ars	Schwarzschild	Schwarz-schild
vil-lai-ness	vil-lain-ess	Septem-ber	Sep-tem-ber
viviparous	vi-vip-a-rous	Stokess-che	Stokes-sche
voiceprint	voice-print	tech-nis-che	tech-ni-sche
vs-space	\vspace	ve-r-all-ge-mein-erte	ver-all-ge-mein-erte
wallflower	wall-flower	Verteilun-gen	Ver-tei-lun-gen
wastew-a-ter	waste-water	Wahrscheinlichkeit-s-the-o-rie	Wahr-schein-lich-keits-the-o-rie
waveg-uide	wave-guide	Winch-ester	Win-ches-ter
we-b-like	web-like	Ying-yong Shuxue Jisuan	Ying-yong Shu-xue Ji-suan
weeknight	week-night		
wheelchair	wheel-chair		
whitesided	white-sided		
whites-pace	white-space		
widespread	wide-spread		
wingspread	wing-spread		
witchcraft	witch-craft		
workhorse	work-horse		
wraparound	wrap-around		
wretched(ly)	wretch-ed(-ly)		
yesteryear	yes-ter-year		

Names and non-English words used in English text

al-ge-brais-che	al-ge-brai-sche
au-toma-tisierte	auto-mati-sier-ter
Be-di-enung	Be-die-nung
Brow-n-ian	Brown-ian
Columbia	Co-lum-bia
Czechoslo-vakia	Czecho-slo-va-kia

Fonts

Erratum:

Typesetting Concrete Mathematics
TUGboat Vol. 10, No. 1

Donald E. Knuth

Editor's note: Some erroneous parameter values for the Concrete fonts were given in the footnote on page 33. The following are the correct values. *beak_darkness* = 4/30 for all Concrete fonts except for *ccslc9*, which has *beak_darkness* = 5/30. Also for *ccslc9*, *fudge* = 1.

The METAFONT sources and the macros for using the Concrete fonts can be found in the *tex-fonts* archive at Clarkson. A list of the files appears on page 200, *TUGboat* Vol. 10, No. 2.

TeX and latin alphabet languages

Yannis Haralambous

(Ἐκείνη) μιλάει τή γλώσσα τῶν Ἐτρούσκων, μιάν ἀγνωστη, παράσενη γλώσσα, πού κανένας δέν τήν ξέρει καὶ κανένας ποτέ δέν θά τήν μιλήσει.

—Κωστούλα Μητροπούλου¹

TeX provides enough accents and special letters to write in (almost) every latin alphabet language. But there are problems; following D. E. Knuth,² "... format designers should provide rules for how to obtain accents and special characters in their particular systems. plain TeX works well enough when accents are infrequent, but (its) conventions are by no means recommended for large-scale applications of TeX to other languages." The best solution to this is to create accented and special letters by METAFONT and to put them in the still-empty 128 boxes of the 256-code fonts.*

* Editor's note: A technique developed long ago by David Fuchs, but never publicized, permits the creation of "virtual fonts" that have instructions for putting pieces together, at the output driver level, from the same or separate fonts, so that detailed character descriptions are not required for such

But to keep the "universal properties" of TeX, this should be done (a) once and for all, and (b) in the most democratic and complete way. It is important that —in an international level— everybody has the same fonts and ligature conventions (to prevent having for example *un TeX français* and *ein deutsches Tex* with no compatibility to each other).

Now the natural question to ask is: is it possible to cover ALL latin alphabet languages? To answer this question I have searched (for some hours) in the library of the Centre Pompidou in Paris and made a list of languages with their respective accented letters and special characters (beside the well-known 26-letter alphabet). I am neither a linguist, nor a geographer, and this makes me pretty sure that many languages and dialects are missing. So I would be very grateful for additional remarks and corrections (just write to *yannis@frccit171.bitnet*). Also I would like to thank my friends Ana Cristina Matos (Portugal), Vladimir Rosenhaus (Estonia, USSR) and Petr Plecháč (Czechoslovakia) for their precious help.

List of languages

1. **Albanian**³: č Č, ё Ѓ
2. **Basque**⁴: ñ Ñ
3. **Breton**⁵: é É, è È, ê Ê, ñ Ñ, ô Ô, ü Ü
4. **Catalan**⁶: à À, ç Ç, é É, è È, í Í, ó Ó, ò Ò, ú Ú, ü Ü
5. **Corsican**⁷: à À, é É, è È, ì Ì, ô Ô, ù Ù, û Ù
6. **Creole**⁸: ä Ä, å Å, ã Ä, é É, ë Ë, ê Ê, ê Ê, è Ë, ï Ì, ñ Ñ, ò Ò, ö Ö, õ Ö, ò Ö, ü Ü, ù Ü, ú Ü, ý Ý
7. **Croatian**²⁸: č Č, č Č, đ Đ, š Š, ž Ž
8. **Czech**: á Á, č Č, đ D, é É, è È, í Í, ñ Ñ, ó Ó, ř Ř, š Š, ť T, ú Ú, ü Ü, ý Ÿ, ž Ž
9. **Danish**⁹: å Å, æ æ, ø Ø
10. **Dutch**¹⁰: é É, è È
11. **English**: none
12. **Esperanto**¹¹: ĉ Ĉ, ĝ Ĝ, ĥ Ĥ, ĵ Ĵ, ķ Ĵ, ū Ū, ū Ū
13. **Estonian**¹²: ä Ä, ö Ö, ö Ö, š Š, ü Ü, ž Ž
14. **Faroese**⁹: á Á, æ æ, đ þ, í Í, ó Ó, ø Ø, ú Ú, ý Ÿ
15. **Finnish**¹³: ä Ä, ö Ö
16. **French**: à À, å Å, ç Ç, é É, è È, ê Ê, ê Ê, ï Ì, ï Ì, ô Ô, œ œ, ù Ù, û Ù, ü Ü (and it seems that in pre-Louis XIV texts¹⁴ there is also the letter ý Ÿ)

"composite" characters. Arrangements are being made to publish the details in a future TUGboat.

17. **Gaelic**¹⁵: à À, é É, è È, ì Ì, ó Ó, ò Ò, ù Ù
 18. **Galician**: á Á, é É, í Í, ñ Ñ, ó Ó, ú Ú
 19. **German**: ä Ä, ö Ö, ß, ü Ü
 20. **Guarani**¹⁶: á Á, ä Ä, é É, è È, í Í, ï Ï, ñ Ñ, ó Ó, ö Ö, ú Ú, ü Ü, ý Ý, ÿ ÿ
 21. **Hungarian**¹⁷: á Á, é É, í Í, ó Ó, ö Ö, ô Õ, ú Ú, ü Ü, ű Ÿ
 22. **Icelandic**⁹: á Á, æ Æ, ð þ, é É, í Í, ó Ó, ö Ö, ú Ú, ý Ý
 23. **Indonesian**¹⁸: none
 24. **Irish**¹⁹: irish language has its own, most beautiful alphabet
 25. **Italian**: à À, é É, è È, í Í, ì Ì, ó Ó, ú Ú, ù Ù
 26. **Kurdish**²⁰: ç Ç, ê Ê, ë Ë, î Ï, ş Ş, û Û, x X
 27. **Latin**: none (sometimes æ Æ is used)
 28. **Latvian**^{12,21}: ä Ä, č Č, ē Ē, ģ Ģ, ī Ī, ķ Ķ, ļ Ľ, ñ Ñ, š Š, ū Ÿ, ž Ž
 29. **Lithuanian**^{12,22}: å Å, č Č, ç Ç, è É, i Í, š Š, y Ý, ū Ÿ, ū Ÿ, ž Ž
 30. **Maltese**²³: c Ĉ, g Ĝ, h H, z Z
 31. **Norwegian**⁹: å Å, æ Æ, ø Ø
 32. **Occitan**²⁴: ç Ç, é É, è È, í Í, ì Ì, ó Ó, ò Ò, ú Ú
 33. **Polish**²⁵: å Å, c Č, e E, l L, n Ñ, ó Ó, s Š, z Ž, ž Ž
 34. **Portuguese**: á Á, à À, â Â, ã Â, ç Ç, é É, ê Ê, í Í, ó Ó, ô Ô, õ Õ, ú Ú, ü Ü
 35. **Qhëshwa**²⁶: á Á, é É, è È, í Í, ñ Ñ, ó Ó, ö Ö, ú Ú, ü Ü
 36. **Romansh?**: á Á, ó Ó, ö Ö, ù Ù, ü Ü...?
 37. **Romanian**²⁷: â Â, ă Â, î Í, ş Š, t T
 38. **Slovakian**: à À, č Č, d D, é É, ê Ê, í Í, l L, ñ Ñ, ó Ó, ô Ô, ť Ť, š Š, t T, ú Ú, ü Ü, ý Ý, ž Ž
 39. **Slovenian**²⁹: č Č, š Š, ž Ž
 40. **Spanish**: á Á, é É, í Í, ñ Ñ, ó Ó, ú Ú, ü Ü
 41. **Swedish**⁹: ä Ä, å Å, ö Ö
 42. **Turkish**³⁰: â Â, ç Ç, ğ Ģ, i I, ì Ì, ö Ö, ş Š, ū Ÿ, ü Ü
 43. **Vietnamese**³¹: à À, ã Â, đ Đ, ê Ê, ô Ô, œ œ, ū Ÿ and the following accents: ' ', ' ', ' ', ' '
 44. **Welsh**³²: â Â, è È, ê Ê, ì Ì, ï Ï, ô Ô, ö Ö, ū Ÿ, w W, ý Ý
 and finally,
MR transliteration of USSR cyrillic: è È, ë Ë, ì Ì for є є, è È, ï й.

Conclusions

There are 190 different symbols on this list, and 8 (æ Æ, ø Ø, œ œ) are already in the CM fonts. So the remaining 182 are far too many to be included in the 128 empty boxes. But, looking closer, we see that for the most important "european" languages (languages of the European Community, central europe socialist countries and Turkey) we only need 112 symbols from this list, and automatically most of the other languages are covered too. So I think the best thing to do is to take these 112 symbols and leave the 16 empty boxes for arbitrary use (also I wait for your comments and suggestions). These 112 symbols are the following:

á Á, à À, â Â, ã Â, å Å, ä Ä, ö Ö
 ç Ç, ê Ê, ë Ë, î Ï, ñ Ñ, ò Ò, ù Ù
 ð Ð, ð þ ð þ
 é É, è È, ê Ê, ë Ë, è È, ê Ê
 ÿ ÿ
 í Í, ì Ì, ì Ì, ï Ï, ï Ï, ì Ì
 l L, l L
 n Ñ, ñ Ñ, ñ Ñ
 ó Ó, ò Ò, ô Ô, ö Ö, õ Õ, õ Õ, œ œ
 ť Ť
 ſ ſ, ſ ſ, ſ ſ
 t T, t T
 ú Ú, ù Ù, û Û, ü Ü, ü Ü, ü Ü, ū Ÿ
 ý Ý, ý Ý
 ž Ž, ž Ž, ž Ž, ž Ž

Which ligatures to choose

Once we have the new font, the problem is to assign the right ligatures, according to the three basic rules:

- compatibility with TeX instructions (for example, ~n is not admissible for ñ, since ~ is active).
- compatibility with 128-character fonts (the example

e'crire de cette manie're
 nai"ve en franc/ais

disobeys this rule; to use it we would have to change the use of ', ', and /, and texts written in "classical" TeX couldn't be typeset anymore).

Taking (a) and (b) into account, the only ASCII characters we can use are <, >, and, by slightly disobeying rule (b), we could (and will have to) take also ", +, and = (this could be done because

‘‘ also gives ”, and \$+\$ and \$=\$ are even better choices for + and =).

(c) readability.

The solution I propose is the following:

- 1) take < and > for the acute and grave accents respectively (they are easy to find on the keyboard, symmetric, and give the direction of the accent).
- 2) take = for the ^ accent; also take =n for ñ.
- 3) take " for " on vowels. Also take "s for ß, following Part¹³³.
- 4) take the plus sign + for both the háček and tilde accents (the portuguese language doesn't have háčeks and czech has no tildes, so there is no essential risk of confusion). So for example +a means á, and +z, ž (this is why we took =n for ñ: it's the only letter that takes both háček and tilde accents).
- 5) take > also for the “french-style” cedilla on c, s, t and == for the “polish-style” hook on a and e (a is a very wicked letter; it takes almost every existing diacritical mark).
- 6) since <o, <u give ó, ú, it's reasonable to put <<o, <<u for ö, ü.
- 7) take " before d, l, t for ð, ï, ð; take <>o and <>u for ø, ù (o is at least as wicked as a).
- 8) the rest of the distribution is purely random.

You can see the whole of it in the following table:

\langle	followed by	$\begin{cases} \text{a A} \\ \text{c C} \\ \text{e E} \\ \text{i I} \\ \text{n N} \\ \text{o O} \\ \text{s S} \\ \text{u U} \\ \text{y Y} \\ \text{z Z} \end{cases}$	gives	$\begin{cases} \acute{\text{a}} \text{ Á} \\ \acute{\text{c}} \text{ Č} \\ \acute{\text{e}} \text{ É} \\ \acute{\text{i}} \text{ Í} \\ \acute{\text{n}} \text{ Ñ} \\ \acute{\text{o}} \text{ Ó} \\ \acute{\text{s}} \text{ Š} \\ \acute{\text{u}} \text{ Ú} \\ \acute{\text{y}} \text{ Ÿ} \\ \acute{\text{z}} \text{ Ž} \end{cases}$
$>$	followed by	$\begin{cases} \text{a A} \\ \text{c C} \\ \text{e E} \\ \text{i I} \\ \text{o O} \\ \text{s S} \\ \text{t T} \\ \text{u U} \end{cases}$	gives	$\begin{cases} \grave{\text{a}} \text{ Á} \\ \grave{\text{c}} \text{ Č} \\ \grave{\text{e}} \text{ É} \\ \grave{\text{i}} \text{ Í} \\ \grave{\text{o}} \text{ Ó} \\ \grave{\text{s}} \text{ Š} \\ \grave{\text{t}} \text{ Ð} \\ \grave{\text{u}} \text{ Ú} \end{cases}$

=	followed by	$\begin{cases} \text{a A} \\ \text{e E} \\ \text{i I} \\ \text{n N} \\ \text{o O} \\ \text{u U} \end{cases}$	gives	$\begin{cases} \hat{\text{a}} \text{ Ä} \\ \hat{\text{e}} \text{ È} \\ \hat{\text{i}} \text{ Ï} \\ \hat{\text{n}} \text{ Ñ} \\ \hat{\text{o}} \text{ Ö} \\ \hat{\text{u}} \text{ Ü} \end{cases}$
---	-------------	--	-------	--

"	followed by	$\begin{cases} \text{a A} \\ \text{d D} \\ \text{e E} \\ \text{i I} \\ \text{l L} \\ \text{o O} \\ \text{s} \\ \text{t T} \\ \text{u U} \\ \text{y Y} \\ \text{z Z} \end{cases}$	gives	$\begin{cases} \ddot{\text{a}} \text{ Ä} \\ \ddot{\text{d}} \text{ Ð} \\ \ddot{\text{e}} \text{ È} \\ \ddot{\text{i}} \text{ Ï} \\ \ddot{\text{l}} \text{ Í} \\ \ddot{\text{o}} \text{ Ö} \\ \ddot{\text{s}} \text{ ß} \\ \ddot{\text{t}} \text{ T} \\ \ddot{\text{u}} \text{ Ü} \\ \ddot{\text{y}} \text{ ÿ} \\ \ddot{\text{z}} \text{ Ž} \end{cases}$
---	-------------	--	-------	---

+	followed by	$\begin{cases} \text{a A} \\ \text{c C} \\ \text{d D} \\ \text{e E} \\ \text{g G} \\ \text{i I} \\ \text{l L} \\ \text{n N} \\ \text{o O} \\ \text{r R} \\ \text{s S} \\ \text{z Z} \end{cases}$	gives	$\begin{cases} \check{\text{a}} \text{ Ā} \\ \check{\text{c}} \text{ Č} \\ \check{\text{d}} \text{ Ð} \\ \check{\text{e}} \text{ Ě} \\ \check{\text{g}} \text{ Ĝ} \\ \check{\text{i}} \text{ Ĩ} \\ \check{\text{l}} \text{ Ľ} \\ \check{\text{n}} \text{ Ķ} \\ \check{\text{o}} \text{ Ķ} \\ \check{\text{r}} \text{ Ě} \\ \check{\text{s}} \text{ Š} \\ \check{\text{z}} \text{ Ž} \end{cases}$
---	-------------	--	-------	--

==	followed by	$\begin{cases} \text{a A} \\ \text{e E} \end{cases}$	gives	$\begin{cases} \text{a A} \\ \text{e E} \end{cases}$
----	-------------	--	-------	--

++	followed by	$\begin{cases} \text{a A} \\ \text{D} \\ \text{o O} \end{cases}$	gives	$\begin{cases} \text{æ} \text{ È} \\ \text{þ} \text{ Ø} \end{cases}$
----	-------------	--	-------	--

<<	followed by	$\begin{cases} \text{a A} \\ \text{o O} \\ \text{u U} \end{cases}$	gives	$\begin{cases} \check{\text{a}} \text{ Á} \\ \check{\text{o}} \text{ Ö} \\ \check{\text{u}} \text{ Ü} \end{cases}$
----	-------------	--	-------	--

<>	followed by	$\begin{cases} \text{o O} \\ \text{u U} \end{cases}$	gives	$\begin{cases} \text{o} \text{ O} \\ \text{u} \text{ U} \end{cases}$
----	-------------	--	-------	--

'''	followed by	$\begin{cases} \text{a A} \\ \text{i I} \\ \text{o O} \\ \text{u U} \end{cases}$	gives	$\begin{cases} \text{å} \text{ Å} \\ \text{i} \text{ Í} \\ \text{œ} \text{ œ} \\ \text{ü} \text{ Ü} \end{cases}$
-----	-------------	--	-------	--

An example

By typing

Zde se v+semo+zne sna+z*i* m+e p+remluvit,
abych z""ustal je+st+e n+ekolik m+es<ic""u
a napsal je+st+e jednu operu. Hay""ir!
""I>s "oyle de+gil. B"uy"u+g"u
k"u>c"u+g"une takilmay""i pek severdi.
Ce f=ut d'ores et d<ej>a une id<ee
d<eg<en<er<ee et ambig"ue.

instead of

Zde se v\v semo\v zn\v e sna\v z\'\i\ m\v
e p\v reluvit, abych z{\accent'027u}stal
je\v st\v e n\v ekolik m\v es\'\i
c{\accent'027u} a napsal je\v st\v e jednu
operu. Hay\i r! \D I\c s \"oyle de\u
gil. B"uy"\u\u g"\u k"\u\c c"\u\u g"\une
takilmay\i\ pek severdi. Ce f\^ut d'ores
et d'\ej\^a une id\^ee d\^eg\^en\^er\^ee
et ambig"\ue.

(which looks better?) you get

Zde se všemožně snaží mě přeluvit, abych zůstal
ještě několik měsíců a napsal ještě jednu operu.
Hayır! İş öyle değil. Büyügü küçüğüne takilmayı
pek severdi. Ce fût d'ores et déjà une idée dégénérée
et ambiguë.

References

1. K. Mhtropo'ulou, *Za'ar* 19, K'edroc, *l'Aj'hna* 1980.
2. D. E. Knuth, *The TeXbook*, Addison-Wesley, 1983.
3. H. Boissin, *Grammaire de l'Albanais moderne*, chez l'Auteur, Paris 1975.
4. X. Kintara, *Diccionario vasco-español*, Hiztegia bi mila, Bilbo 1984.
5. E. Breiz, *Dictionnaire Breton*, Garnier, Paris 1986.
6. B. Margarit, *Gramatica Catalana*, Gredos, Madrid, 1962.
7. P. Ciavatti (Ed.), *Dizziunariu Corsu-Francese*, Albiana, Levie 1985.
8. *this alphabet comes from a project of R. Germain, cf. R. Germain, Grammaire créole, l'Harmattan, Paris 1980.*
9. M. O'C. Walshe, *Introduction to the Scandinavian Languages*, Andre Deutsch, London 1965.
10. F. van Passel, *Le Néerlandais pour tous*, Presses Pocket, Paris 1983.
11. L. L. Zamenhof, *Fundamento de Esperanto*, Esperantoj Francaj Eldonoj, Marmande 1963.
12. *communicated to me by V. Rosenhaus.*
13. H. Fromm & M. Sadeniemi, *Finnisches Elementarbuch*, Carl Winter Universitätsverlag, Heidelberg 1956.
14. *as for example in the lovely text:*
Le dit du Saige trop le feiz
Favorable, bien n'en puis mais
Qui dit: "Esjoys toy, mon filz,
En ton adolescence"; mais

*Ailleurs sert bien d'ung autre mes,
Car "Jeunesse et adolescence",
C'est son parler, ne moins ne mais,
"Ne sont qu'abus et ignorance".*

taken from F. Villon (1431-1463?), *Le Grand Testament*, dtv 2078, München 1982.

15. G. Calder, *A Gaelic Grammar*, Gairn Publications, Glasgow 1972.
16. B. M. Lliteras, A. P. Peñasco & L. F. Maluquer, *El Guarani a su Alcance*, Loyola, Asuncion 1976.
17. G. Darai, *Universal-Wörterbuch Ungarisch*, Langenscheidt, Berlin 1988.
18. A. L. N. Kramer, *Kamus Inggeris Ketjil*, Ch. E. Tuttle, Tokyo 1975.
19. Rev. P. S. Dinneen, *Irish-English Dictionary*, The educational company of Ireland, Dublin 1927.
20. E. D. B. Khan & R. Lesot, *Grammaire Kurde*, Librairie d'Amérique et d'Orient, Paris 1970.
21. A. Гутмане, *Русско-Латышский Разговорник, Лиесма*, Рига 1975.
22. F. Bielenstein, *Lettische Grammatik*, Zentralantiquariat der DDR, Leipzig 1972.
23. S. Paterson, *Guide de voyage: Malte*, Berlitz, Lausanne 1987.
24. R. Barthe, *Lexique Français-Occitan*, Collection des Amis de la Langue d'Oc, Paris 1973.
25. L. Bielas, *Słownik minimum Polsko-Francuski*, Wiedza Powszechna, Warszawa 1987.
26. J. Lara, *Diccionario Qhëshwa-Castellano*, Los Amigos del Libro, La Paz 1978.
27. N. P. Condeescu, *Dicționar Francez-Român*, Editura Științifică, București 1967.
28. V. Javarek & M. Sudijć, *Serbo Croat*, Teach Yourself Books, London 1972.
29. F. Tomšić, *Slovensko-Nemški Slovar*, Državna Založba Slovenije, Ljubljana 1973.
30. H. Wendt, *Praktisches Lehrbuch Türkisch*, Langenscheidt, Berlin 1979.
31. N. D. Hoa, *Vietnamese-English Dictionary*, Ch. E. Tuttle, Tokyo 1966.
32. S. J. Williams, *A Welsh Grammar*, University of Wales Press, Cardiff 1980.
33. H. Partl, German *TEX*, *TUGboat* 9 (1988) 70-72.

◊ Yannis Haralambous
U.F.R. de Mathématiques
Université de Lille-Flandres-Artois
59655 Villeneuve d'Ascq Cedex
France
Bitnet: yannis@frccit171

Fonts and PostScript*

Mike Parker

Historical Background

Our view of fonts, of the way that they fit into publishing systems and of the ways that we use them, is shaped by the characteristics of the vanishing systems that have been in use for the past century. Digital publishing systems differ profoundly, offering possibilities that are only starting to be appreciated. The wide distribution of the page description language PostScript and the promise of the recently announced Microsoft/Apple PDL offer a standard for font distribution and use on which this realization could turn. To understand the promise of digital technology and the new PDLs we must examine the accepted perceptions of fonts, which have been largely shaped by vanishing technologies.

We normally think of a font as a fixed design. Twenty-five years ago we would have envisioned it as one size; now we typically envision a series of sizes, but in either case we think of it as a design with weight and proportions rigidly established by the creator or supplier.

Five centuries ago, before the introduction of Gutenberg's technology, this rigidity would have appeared very strange to the scribe, free to modify the proportions and weight of his hand as the work required. The fixed form of Claude Garamont's type was required by the rigid limits of Gutenberg's metalsmith technology, limits that existed one way or another for five centuries through the ages of mechanical and photocomposing equipment until the recent appearance of the digital pixel field. Donald Knuth's METAFONT, Peter Karow's Ikarus system (now appearing on the Macintosh and PC) reintroduce the concept of the font as a fluid series of hundreds of designs available to the user, varying in weight and proportion but held together by the principles common to the series as laid down by the font designer. URW is working out serif and sanserif typographic series where a full set of hundreds of changes is rung on the principles of a single design. The supply of ever growing numbers of the implied members of each type family is perhaps the most

obvious change in the typographic world over the last thirty years.

Patterns of distribution have also been changing. A century ago, in the world of handset type, fonts from any typefoundry could be freely mixed; the only difficulty was the varying standards for type sizes between foundries—a difficulty that all but vanished with the establishment of the standard Didot and then Pica point systems. The engineering specifications of handset type can be seen as the first great page description language.

The introduction of mechanical composing machines at the end of the last century limited the user to the typefaces supplied by the manufacturer of that equipment (plus in some cases auxiliary companies). Several limited type libraries, each available on one company's composing machines, replaced the single great handset library available to everybody in common form from hundreds of foundries worldwide. This division was caused by the mechanical differences between composing machines, each of which required fonts to be manufactured in a different form. Different font libraries continued to be required through the age of photocomposing machines, and can only be reunited with the advent of digital composition and page description languages like PostScript.

During this century of competition between composing machine manufacturers, the leaders have sought to gain and maintain advantage by the introduction of new and better typefaces peculiar to one line of equipment. They have sought intellectual property protection for typefaces to reinforce their advantage. Lawmakers worldwide were faced with the choice between granting the providers of composing machines with rights to limit distribution of typefaces or denying rights in order to encourage broad distribution of typefaces. Until recently, with few exceptions, law has decided worldwide in favor of broad distribution. As the advent of the common pixel field and page description languages has opened typographic supply channels, effective protection for type designs has begun to appear.

We frequently hear from European companies that European law has granted protection unavailable in the U.S. The real difference between Europe and the U.S. has centered in a tendency for greater cross licensing and cooperation among the older companies in Europe. In the U.S., companies like Photon, Autologic and Compugraphic were formed to exploit the invention of photocomposition machines. Typically they equipped their machines with large libraries centering on unauthorized copies of their competitors' typefaces, fonts that had become

* Editor's note: This is an update of an article originally appearing in PostScript Language Journal, vol. 2, no. 2, pp. 24-28. The TUGboat editors kindly thank the author and PLJ for permission to reprint the article here.

popular in the market place and were seen as necessary to achieve sales. Over the last forty years, these unauthorized copies have been, and continue to be, distributed throughout America, Europe and the world with impunity. Usually the unauthorized copies are called by an alternative name—although even this is not always so. The older European companies have been unable to block sales of their new competitors' equipment by forbidding unauthorized copying of their typefaces. However, new law in Germany, France and England is starting to change the picture.

Property Rights

Three forms of protection can be considered for typefaces—patent and copyright for the design itself, and trademark for the name.

Typefaces have been registered under design patent in the U.S. The protection offered has proved uninteresting, being largely limited to ineffective protection of commercially uninteresting typefaces. In Europe, limited protection has been available under minor sections of patent law, particularly in Germany. The period is limited to fifteen years.

There are many who believe that typefaces are an art form best protected under copyright. Protection under copyright law has not been granted to typefaces anywhere until the mid-nineteen-seventies, when activity generated by the 1973 Vienna treaty began to compel new legal analysis.

The difficulties of protecting type designs cause serious concern at A Typ I, the International Typographic Association, headquartered in Switzerland. Until the mid 1970s, A Typ I required its member companies to abide by the A Typ I Moral Code, which forbade any member to copy typefaces cut by another. Linotype caused Berthold and A Typ I serious problems by refusing to license Helvetica to Berthold for text use (after having granted Berthold a license for display use). Berthold was faced with the choice of losing sales on their new text machines, or of resigning from A Typ I. Legal opinion warned A Typ I that the requirements of the A Typ I Moral Code exceeded the requirements of the law—and indicated possible consequences. A Typ I altered the Moral Code to allow members who were refused a license by another member to re-apply after the typeface became fifteen years old (a period based on European minimal patent rights possibly applying to the typeface), and if refused, copy the design and call it by another name (due to the long period of protection possibly offered to the original name under trademark law).

Faced with inadequacy of existing law, A Typ I persuaded WIPO, the World Intellectual Property Organization in Geneva, to sponsor the Vienna Treaty for the Protection of Typefaces. The treaty was executed in June 1973. The Vienna Treaty requires nations who ratify the treaty to offer at least the protection defined in the treaty, including a minimum period of fifteen years protection. The treaty permits protection under full patent or copyright, although special legislation country by country, a slow procedure, was envisioned as the most likely method. Protection in each country was to be granted to typefaces designed after the date on which protection was legislated. Retroactive protection was not envisioned. Twelve nations signed the treaty. The U.S. was not one of the signatories.

In the mid-seventies, probably influenced by the Vienna Treaty activity, a Frankfurt court decided that the typeface Futura created by the German designer Paul Renner in 1927 was a work of art protected under copyright, and that the heirs of Paul Renner were entitled to royalties. The decision implied that major designs by other German type designers working in Germany would be similarly protected by German copyright. In the early nineteen-eighties Germany passed special legislation protecting new typeface designs, and ratified the Vienna Treaty.

France followed with similar legislation, and ratified.

In November 1988, Parliament in England passed an Act which stated that typefaces designed by English designers working in England were covered under copyright, with certain limitations. Users of typefaces (typesetters, printers, publishers) were held harmless. Creators of illegal copies in England or importers of illegal copies are to be held liable. Protection is to be limited to twenty-five years. Status of typefaces that are at present covered by copyright is not clear to me. This legislation took effect in August 1989, with ratification of the Vienna Treaty to follow. Reciprocal rights will be granted to countries offering similar rights; at present this would be limited to Germany, possibly France.

Dominions like Australia and Canada normally follow Britain's lead in international law. When five nations have ratified, the Vienna Treaty will become international law, with a common repository for new type designs in Geneva.

In the U.S., the Typeface Design Coalition and the Font Software Association, stimulated by this activity, plan to reopen the question of protection for digital fonts under U.S. copyright. A bill

for protection of industrial designs at present in committee in Congress contains wording that would offer ten years of non-retroactive protection for original type designs.

A Typ I has now replaced 'fifteen years' in the Moral Code with 'appropriate period', as the period of protection lengthens.

Growing protection, centering in Europe, will encourage purchase of each series from the original manufacturer wherever open systems are found.

Typeface names have been broadly claimed and indeed registered as trademarks through most of this century. However, in spite of broad and frequent claims, in twenty years I have been unable to find case law anywhere in the world. Internationally a true trademark must be an adjective modifying a generic noun that defines the product. No generic descriptions of typefaces are in general use. The only description of the face normally available is the name claimed as a trademark. This situation threatens validity of typeface names as trademarks to the point where suits are seldom pursued; somehow a settlement is reached. While the reality of typographic names as trademarks may be in question, the source of the reputation conferred by licensing is real, and keeps the practice alive.

Fonts for PostScript Engines

The publishing world that buys digital typefaces consists of three main groups:

- the Professional publishing elite growing out of the traditional publishing world;
- the Aware people, desktop publishers who used to buy typeset work from the professionals but now buy digital tools instead to gain control, beat deadlines and save money; and
- the Unaware who were turned off by the expense and delays involved in working with the professionals; accepting lesser quality, they bought typewriters, then word processors, and would benefit today from better typographic imagery, if only they knew it.

The page description language PostScript ties the first two together and with the Microsoft/Apple PDL may prove powerful in the third.

PostScript was initially envisioned as a means of bringing into being the Aware world we now know as Desktop Publishing, with a link to high quality Professional service centers in the professional world. While the Professionals initially regarded PostScript and the Desktop world with

suspicion, this view is rapidly changing, with typesetter manufacturers competing to adopt PostScript as the standard for typesetting centers serving the Desktop world with high resolution pages, using typefaces on Adobe metrics.

The typographic lesson to be drawn from PostScript: the part of the system used by the specifier controls the rest. Traditionally the designer saw proofs from the typesetter—so the screen and proof printer had to match the typesetter in metrics and, as far as possible, in appearance. Now that so many designers have LaserWriters, the typesetter must match the LaserWriter. Adobe has the world by the proofer.

Adobe chose to follow two font paths in releasing the Adobe PostScript implementation. The first comprises a limited group of fonts supplied from Adobe and its chosen suppliers in encrypted format which gave them a favored position at the heart of the system. These Type 1 fonts carry encoded hints that stretch and fit the outlines to give good results on laser printers and, in Display PostScript and the Adobe Type Manager, on screens.

The second group of fonts can be supplied by anyone. Type 3 [*sic*] fonts can be downloaded into PostScript systems, but suffer disadvantages in the number that can be used at one time, in speed, and above all in quality on laser printers and screens.

Initially Adobe offered Type 1 fonts from Linotype for Linotype faces, from artwork supplied by URW for ITC faces, and for faces of their own design. This limited library has been offered in the fashion typical of composing machine manufacturers over the last century. Instead of the number of sources being limited by the requirements of equipment, the concealment of the hints and encryption of the fonts have limited the supply of Type 1 PostScript fonts to sources licensed by Adobe.

In response to recent pressures for open systems, Adobe partially opened their system at the March Seybold. They announced the licensing of their font technology to Monotype, Compugraphic and Varityper to prepare libraries in hinted PostScript Type 1 format, with further discussions continuing with other major suppliers.

Adobe went the rest of the way at the September Seybold. In response to the Microsoft/Apple announcement of their new page description language, upwardly compatible from PostScript, with wholly open font technology, John Warnock announced the publication of Adobe's font standards for all to use. Adobe's tools for manufacturing PostScript fonts are still to be sold.

The library offered by Apple on the LaserWriter, driven by the Macintosh, consists of hinted Adobe Type 1 fonts. This equipment has been the leading choice of the Aware segment of the market, selling into Desktop Publishing, with over 200,000 now in use. The effect can most clearly be seen in the adoption of Adobe metrics, or character widths, as a standard across the Aware industry. One of two packages is found at every installation: either the original basic LaserWriter set of thirteen fonts, or the larger LaserWriter Plus set of thirty-five. The widths of the characters in these thirteen or thirty-five fonts are all but required of every manufacturer of applications packages who wishes to sell products into Desktop Publishing.

Because Desktop Publishing installations are using this limited LaserWriter library on this single set of widths, anyone wishing to offer high resolution output must also offer the same fonts on the same metrics if the high resolution page from the service center is to match the page created on the LaserWriter. Instead of the proofer having to match the high resolution imagesetter, the ubiquitous proofer now requires all imagesetters to conform, a measure of the power of PostScript as a standard.

As others beside Adobe bring PostScript systems to market, what is the current outlook for fonts? The need of the PostScript and clone manufacturers for hinted outline-to-bitmap algorithms was initially met by the licensing of Bitstream's Fontware and Compugraphic's Intellifont. Licensees of both these systems were limited to fonts provided by the supplier.

Pressure for open font systems led URW to release Nimbus R, Folio to release TypeScaler and The Company to release Nimbus Q as open systems. Bitstream has announced that outside manufacturers will be permitted to provide fonts for Fontware under appropriate conditions, and Compugraphic has announced that they will similarly open Intellifont.

Folio has been bought by Sun Microsystems and is concentrating on the creation of a display PostScript for Sun. They have no library of their own, but offer the type source of your choice with the Folio TypeScaler algorithm; Monotype, Linotype, Berthold, Bigelow & Holmes have agreed to produce fonts to the F-3 format. They offer an encrypted format, preferred by major foundries as well as an open format that anyone may use. They provide the type source with TypeMaker, an automatic algorithm running on a Sun workstation that transfers the font to their proprietary form

of general conics and automatically adds hints. TypeScaler requires the general conics format.

The Company offers the URW library for Nimbus Q in the standard PostScript Bezier format recommended for PostScript. The Company publishes their format, and also offers to sell all major foundries their PC-based hint insertion software for use and distribution down to the end-user level.

All of the six hint-based systems, the Adobe PostScript implementation, Fontware, Intellifont, TypeScaler, Nimbus R and Nimbus Q, offer comparable quality. Conographics offers a fast hintless algorithm whose quality is said to fall somewhat short of the others.

Berthold and Bitstream share a belief that the real market for a large variety of fonts lies in the high resolution market. They have stated that for this work the laserprinter will be used only as a proofer, with rough, unhinted quality adequate for proofing work that will later be typeset on high resolution systems. Berthold makes a package of 400 Berthold fonts available in unhinted Bezier form for this market. Bitstream goes one step further: having obtained Adobe's encryption they offer a growing library in hintless (at present) Type 1 format. RIPS has also announced fonts in Type 1 format, with more expected to follow.

We have seen four new page description languages emerge in competition with the PostScript rasterizer:

The closest is the version of Display PostScript announced by Sun. The PDL appears to follow Adobe—but the spline used to describe the outlines of the characters in the fonts is the General Conics format provided by Sun's subsidiary, Folio. Sun is expected to offer fonts from all suppliers who subscribe to Folio's F-3 format, encrypted or open, Linotype, Monotype, Berthold, ITC, and Bigelow & Holmes to date. Sun will be under increasing pressure to adopt the Apple Royal or Adobe PostScript rasterizer.

Hewlett-Packard continues to plan PCL-5, a mature version of the original H-P PCL, with font scaling based upon Compugraphic's Intellifont, which to date has been limited to a Line and Arc format. H-P is expected to distribute CG fonts and offer access to CG typesetters. CG has promised to open the format to other font suppliers. Hewlett-Packard is pressed to join the Apple Royal font format.

The most interesting announcement came from Microsoft and Apple at the September Seybold conference. In the spring, Microsoft had announced a new page description language upwardly compatible

from PostScript, while Apple announced their new font scaling technology, called both System 7 and Royal. System 7 can be used to equal the quality and power of present rasterizers at a basic level, but promises advances in screen quality, non-linear font scaling and innovative forms of font manipulation at the expense of complex font preparation. The new PDL promises to run existing PostScript files while adding as yet undefined advances and refinements.

IBM must decide whether to cement their alliance with Next and Adobe by favoring Display PostScript and the Adobe Type Manager, or whether to favor Microsoft and Royal.

The new combination is promised for release in 1990 to the Apple and Microsoft world. The price will be low, the market measured in tens of millions of installations. All font suppliers have been invited to provide fonts in the new public format.

A couple of Seybolds ago, Steve Jobs opened the seminar with four slides:

- A big old \$20,000 Wang system from several years before.
- A \$200 floppy disc, successor to number one.
- Contemporary desktop publishing technology, Macintosh, LaserWriter and Linotronic L-300, a \$50,000 package.
- A \$500 floppy disk containing everything necessary for future desktop publishing.

The Microsoft/Apple PDL could be that reality.

One year ago each RIP maker found an attractive font supplier and established a monogamous relationship—Adobe/Linotype, Hewlett-Packard/Compugraphic, Phoenix/Bitstream, etc. We are witnessing the breakdown of these relationships as, led by Microsoft and Apple, each PDL and RIP maker pursues fonts from all major font suppliers and each major font supplier pursues entry to all major RIPS and PDLs.

Meanwhile, specialist type designers who once concentrated on analog niches (special designs for magazines, ad agencies, corporations, technical composition, non-romans, etc., conventionally produced as 2" film strips) now buy tooling to speed their production of digital outlines to be distributed across the PostScript world and the budding new PDLs. We expect them to be the typographic phenomenon of the early nineties.

Close behind them are the end-users with specialized needs of their own, who with new tools and open systems could conveniently digitize, hint and slip a logotype or a group of special characters into a system in a matter of hours.

As for fonts and PDLs, to quote Jonathan Seybold, "We think that the ultimate answer will have to lie in the domain of marketing, not law. Good designs that are aimed at the mass market—priced low, packaged conveniently, distributed widely and compatible with lots of screens and printers—will generate more revenue than equally good expensive, encrypted, narrowly marketed fonts that only work with one brand of printer."

Where are the PostScript and Royal font markets going?

1. PostScript and the forthcoming Microsoft/Apple PDL offer the page description languages of choice, capable of providing a unifying link between Desktop and Professional publishing, now that fonts from the necessary manufacturers can be offered in convenient form across systems and typesetters open to all players.

2. Closed font policies have favored a few centralized font manufacturers. The movement of Professional digital font creation and manipulation tools down the market is starting to create many desktop digital typefoundries offering ever greater variety of fonts, down to the personalized level. Open-font PDLs and rasterizers will support and benefit this growth.

3. Growing protection of original type designs can be expected to encourage supply of fonts from many original sources across open markets centering on open, commonly available page description languages.

The need is for imaginatively designed and competently executed series of typefaces. The names of suppliers of good fonts are the real trademarks: ITC, Linotype, Monotype, Berthold, Bitstream, URW, The Font Bureau, etc. The real commodity is the creativity (now under growing protection), the competence and the reputation of the supplier. Truly open PDLs offer a worldwide opportunity for establishing and furthering the position of those suppliers and their reputations, each supplying their own original contributions across the market to a broad range of output devices. Niche players will come into being, specializing in fonts of a given kind. Font publishers are being formed to distribute them to users. Increasingly fonts will be designed by typographers and graphic designers for the world to use.

◊ Mike Parker
The Company
World Trade Center
Suite 400
Boston, MA 02210-2050

A chess font for **TEX**

Jan Eric Larsson

Computer chess is a fascinating subject, and can, as is seen from this article, spawn quite unexpected projects. I have been working on chess programs for several years, but this year I decided that the time was ripe for a report. Then I found out that there was no good **TEX** font for producing a chess diagram. After some hesitation, I decided to make one myself.* Producing bitmaps for the chess pieces seemed to be quite straight-forward, as I could use the SunView iconedit and the Touchup raster editor **pktopx** manual told me the format for **pxl** files. With the help of this knowledge, I wrote a small C program that could read a number of icon files, and produce a **pxl** file from those bitmaps. I also wrote my own **p1** file and used **pltotf** to make a suitable **tfm** file. Using icons was easy, but it means the font is like a string bikini, i.e. one size fits all. The 64×64 bit squares on the chess board are 5.4187 mm high when output by a 300 dpi laser writer. Ideally, one would of course want to have **METAFONT** code, but that remains to be written.

The chess font contains 32 characters. There are 6 types of pieces, and each may be white or black. An empty field is also needed. All these 13 patterns must be available with a light and a dark background. This makes up for 26 of the characters. The remaining 6 are smaller (32×32 bit) pieces, for use in "figurine" notation. In Swedish (and in German, by the way), the initials of the piece names are B, S, L, T, D, and K, while in English they are P, Kt or N, B, R, Q, and K. In the figurine notation, small chess pieces are used instead, and thus, this notation is easily understood by all.

Character 0 is the empty white field, and character 1 is the empty black. Characters 2 to 5 are the white pawn on white field, white pawn on black, black pawn on white, and black pawn on black, respectively. In like manner, the knight, bishop, rook, queen, and king use characters 6 to 25. The characters 26 to 31 are figurines for pawn, knight, bishop, rook, queen, and king. The pawn is not written out in figurine notation, but it is there if you should want it. It looks like \hat{a} . Now let us look at an example of the use of both figurine notation and the diagram font.

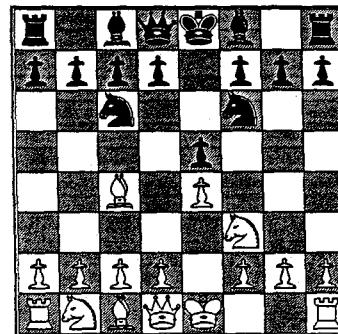
* Editor's note: Zalman Rubinstein and his co-workers have since produced **METAFONT** chess fonts. One appears in **TUGboat** 10#2, pp.170-172; the other is illustrated in this issue of **TUGboat**, p. 386.

von Holzhausen — Tarrasch

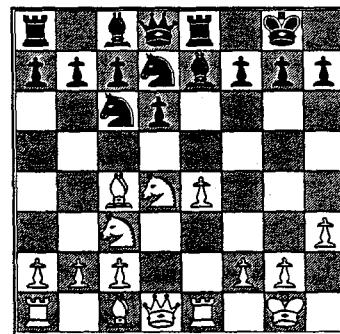
Frankfurt 1912

Two Knights Defense

1. e4 e5 2. $\hat{Q}f3$ $\hat{Q}c6$ 3. $\hat{A}c4$ $\hat{Q}f6$



4. d4 exd4 6. $\hat{Q}xd4$ $\hat{A}e7$ 8. h3 $\hat{E}e8$
5. 0-0 d6 7. $\hat{Q}c3$ 0-0 9. $\hat{E}c1$ $\hat{Q}d7?$



10. $\hat{A}xf7!$ $\hat{Q}xf7$ 11. $\hat{A}e6!$ Resigns.

The continuation could be

11. — $\hat{Q}xe6$ 12. $\hat{W}d5\ddagger$ $\hat{Q}f6$ 13. $\hat{W}f5\ddagger$

I would like to thank Mats Lilja, who made the figurines with Touchup. I am also grateful to Leif Andersson for giving me inspiring advice about **TEX** fonts and helping me to find the appropriate manuals.

◇ Jan Eric Larsson M.Sc. Lic.Tech. B.A.
Department of Automatic Control
Lund Institute of Technology
Box 118, S-221 00
LUND, Sweden
Phone: +46 46 108795
Usenet: JanEric@Control.LTH.Se

The Road to Ethiopic **TEX**

Abass Andulem

In the olden days, when an Ethiopian scribe wrote a manuscript, the scribe used *bērāna* (made up of a goat or a sheep skin) to write on, a scribe's pen or *bērē* (made up of a reed), and ink (made up of different substances)[1]. After he/she gathered of all necessary materials, it was the scribe's responsibility to write and typeset a whole manuscript. The number of characters in Ethiopic alphabet was not a critical problem for the scribe although the entire process of writing a manuscript was time consuming and cumbersome.

When a modern printing press was established in Ethiopia at the end of the nineteenth century, new methods of printing books became easy, reliable, and fast. Old manuscripts that had been used in churches and in some other institutions were gradually replaced by newly printed books. Despite numerous technological advances in typesetting and in the printing press, the large number of Ethiopic characters is still a fundamental problem. There are 231 letters in the Ethiopic alphabet excluding special characters, numbers, and punctuation marks. As a result of that, the Ethiopic typewriter is extremely difficult to use compared with the English typewriter. Furthermore, this problem is exacerbated in computer programming where hardware restrictions are common in many applications.

There are very few application systems known to the author that are designed with the Ethiopic alphabet. Some of the output of these word processors, as observed from different advertisements, leaves much to be desired. Since there hasn't been published literature on any of these application systems, not much can be said about them.

The idea of designing a word processor that can produce high quality Ethiopic documents led us to begin a project, called "ETHTEX". This project was initiated in the Fall of 1987 during a discussion with Dr. Brian Bourgeois at the University of Houston-Downtown. At that particular time, there was not a clear plan how this project would be carried out. However, Dr. Victor Espino's introduction of the **TEX** typesetting system in our university and Dr. Bourgeois' suggestion that the **TEX** typesetting and **METAFONT** system could be helpful to develop the system solved the fundamental problem.

The purpose of this article is to introduce the Ethiopic font that is designed with the **METAFONT** system, to discuss the effort that has continued to expand **TEX**'s application into Ethiopic languages,

and to raise some of the critical problems that have been encountered along the way.

Ethiopic Alphabet

According to the inscription on the monolith of Aksum, at Tigre province in Northern Ethiopia, Ethiopic script has been known for at least 2000 years[2]. The Ethiopic script has undergone several changes through the years. In particular, the introduction of a modern printing press in the country transferred the hand-lettering to printing which added a new style to the Ethiopic script. The usage of the Ethiopic alphabet has much in common with that of English. In the Ethiopic language, letters are written from left to right, words are separated by two dots (sometimes square dots or a blank), and sentences are separated by four dots (sometimes square dots). Basically, there are seven vowel sounds (*ä*, *u*, *i*, *a*, *e*, *ee*, *o*), representations of which are appended to each consonant letter. Therefore, unlike in English, the vowels are not independent of the consonant letters in written Ethiopic. For example, '*w*', represents the consonant 's' sound. The sounds *sä*, *su*, *si*, *sa*, *se*, *see*, *so* are represented respectively by *w*, *u*, *ü*, *ü*, *ü*, *ü*, and *ü*. If we want the word **TEX** in Ethiopic alphabet, we write it as "*tħ*" in which the vowels sounds are expressed by the two consonant letters. The font table shows the 33 consonant letters families in which each family consists of seven characters.

Today, the dominant language Amharic, the official language of Ethiopia, uses Ethiopic script. The Amharic alphabet is made up of 268 characters consisting of 33 consonants each with 7 vowel sounds and of 37 rarely used additional characters. This number excludes numerals and punctuation marks. Presently, both Ethiopic numbers and Arabic numerals are used in Amharic languages for various applications.

Designing Ethiopic Script

Designing the Ethiopic font with **METAFONT** has been the primary goal in this project. Learning the language, using it to design the font, and becoming accustomed to the **METAFONT** system, were the day-to-day experiences. One of the biggest problems in this process was that of figuring out the so called "standard Ethiopic script" that would be used as a model. There are various types which are used in typesetting, but the distinction can be classified neither as stylistic nor as traditional. Presumably, this problem can be overcome in the future.

Ethiopic alphabet and its phonetic representation

ሀ	[hä]	ሁ	[hu]	ሃ	[hi]	ሃ	[ha]	ሃ	[he]	ሀ	[hee]	ሁ	[ho]
ለ	[lä]	ሉ	[lu]	ሉ	[li]	ሉ	[la]	ሉ	[le]	ሉ	[lee]	ሉ	[lo]
ወ	[hä]	ወ	[hu]	ወ	[hi]	ወ	[ha]	ወ	[he]	ወ	[hee]	ወ	[ho]
ሙ	[mä]	ሙ	[mu]	ሙ	[mi]	ሙ	[ma]	ሙ	[me]	ሙ	[mee]	ሙ	[mo]
ሙ	[sä]	ሙ	[su]	ሙ	[si]	ሙ	[sa]	ሙ	[se]	ሙ	[see]	ሙ	[so]
ሩ	[rä]	ሩ	[ru]	ሩ	[ri]	ሩ	[ra]	ሩ	[re]	ሩ	[ree]	ሩ	[ro]
ሰ	[sä]	ሰ	[su]	ሰ	[si]	ሰ	[sa]	ሰ	[se]	ሰ	[see]	ሰ	[so]
ሻ	[shä]	ሻ	[shu]	ሻ	[shi]	ሻ	[sha]	ሻ	[she]	ሻ	[shee]	ሻ	[sho]
ቁ	[kä]	ቁ	[ku]	ቁ	[ki]	ቁ	[ka]	ቁ	[ke]	ቁ	[kee]	ቁ	[ko]
በ	[bä]	በ	[bu]	በ	[bi]	በ	[ba]	በ	[be]	በ	[bee]	በ	[bo]
ተ	[tä]	ተ	[tu]	ተ	[ti]	ተ	[ta]	ተ	[te]	ተ	[tee]	ተ	[to]
ቁ	[chä]	ቁ	[chu]	ቁ	[chi]	ቁ	[cha]	ቁ	[che]	ቁ	[chee]	ቁ	[cho]
ተ	[hä]	ተ	[hu]	ተ	[hi]	ተ	[ha]	ተ	[he]	ተ	[hee]	ተ	[ho]
ኅ	[nä]	ኅ	[nu]	ኅ	[ni]	ኅ	[na]	ኅ	[ne]	ኅ	[nee]	ኅ	[no]
ኅ	[gnä]	ኅ	[gnu]	ኅ	[gni]	ኅ	[gna]	ኅ	[gne]	ኅ	[gnee]	ኅ	[gno]
አ	[ä]	አ	[u]	አ	[i]	አ	[a]	አ	[e]	አ	[ee]	አ	[o]
ከ	[kä]	ከ	[ku]	ከ	[ki]	ከ	[ka]	ከ	[ke]	ከ	[kee]	ከ	[ko]
ኩ	[khä]	ኩ	[khu]	ኩ	[khi]	ኩ	[kha]	ኩ	[khe]	ኩ	[khee]	ኩ	[kho]
ወ	[wä]	ወ	[wu]	ወ	[wi]	ወ	[wa]	ወ	[we]	ወ	[wee]	ወ	[wo]
ወ	[ä]	ወ	[ú]	ወ	[í]	ወ	[á]	ወ	[é]	ወ	[ée]	ወ	[ó]
ዘ	[zä]	ዘ	[zu]	ዘ	[zi]	ዘ	[za]	ዘ	[ze]	ዘ	[zee]	ዘ	[zo]
ዴ	[jä]	ዴ	[ju]	ዴ	[ji]	ዴ	[ja]	ዴ	[je]	ዴ	[jee]	ዴ	[jo]
የ	[yä]	የ	[yu]	የ	[yi]	የ	[ya]	የ	[ye]	የ	[yee]	የ	[yo]
ደ	[dä]	ደ	[du]	ደ	[di]	ደ	[da]	ደ	[de]	ደ	[dee]	ደ	[do]
ጭ	[djä]	ጭ	[dju]	ጭ	[dji]	ጭ	[dja]	ጭ	[dje]	ጭ	[djee]	ጭ	[djo]
ገ	[gä]	ገ	[gu]	ገ	[gi]	ገ	[ga]	ገ	[ge]	ገ	[gee]	ገ	[go]
ሙ	[dchä]	ሙ	[dchu]	ሙ	[dchi]	ሙ	[dcha]	ሙ	[dche]	ሙ	[dchee]	ሙ	[dcho]
ሙ	[tä]	ሙ	[tu]	ሙ	[ti]	ሙ	[ta]	ሙ	[te]	ሙ	[tee]	ሙ	[to]
ሸ	[pä]	ሸ	[pu]	ሸ	[pi]	ሸ	[pa]	ሸ	[pe]	ሸ	[pee]	ሸ	[po]
ሶ	[tsä]	ሶ	[tsu]	ሶ	[tsi]	ሶ	[tsa]	ሶ	[tse]	ሶ	[tsee]	ሶ	[tso]
ሸ	[tsä]	ሸ	[tsu]	ሸ	[tsi]	ሸ	[tsa]	ሸ	[tse]	ሸ	[tsee]	ሸ	[tso]
ሮ	[fä]	ሮ	[fu]	ሮ	[fi]	ሮ	[fa]	ሮ	[fe]	ሮ	[fee]	ሮ	[fo]
ጥ	[pä]	ጥ	[pu]	ጥ	[pi]	ጥ	[pa]	ጥ	[pe]	ጥ	[pee]	ጥ	[po]

The font table shows the Ethiopic alphabet, which was designed with **METAFONT**, and its phonetic representation. The Computer Modern family of fonts served as a raw model almost in every aspect. Fortunately, with slight modification, the CM parameter and driver files (of course under different names) filled a gap that would have required intensive work in defining and generating the font. The font has distinctive characteristics compared with the 'traditional' one. The height of all characters, except the families of "፳, ፻, ፻, ፻", is the same. Full calligraphic effects are not added because the 'usual types' don't feature identical calligraphic patterns.

Typesetting Ethiopic Alphabet

One of the serious challenges remaining in this project is that of incorporating the Ethiopic alphabet with the **TeX** typesetting system. Now, the testing process is underway, but some problems remain unsolved.

- a. Overall, there are more than 256 characters in the Ethiopic alphabet. Both **METAFONT** and **TeX** can handle 256 characters without problem, but hardware restrictions and the methods of usage require that the number of characters must be adjusted with the keyboard[3].
- b. Let's assume that problem (a) is solved using "ligtable" and "control sequences". Now, for a person who cannot speak and write English, but who wants to use the Ethiopic alphabet to typeset documents, there has to be an "Ethiopic editor". Thus, an interface between the **TeX** system and the Ethiopian language must be devised for non-English speaking users.

These are the main problems that require extensive effort in order to make the **EHTeX** project fruitful. Hopefully, in the near future, these problems will be solved.

In the Fall of 1987, we were talking 'how we should start the project', but now we are talking 'how we should solve the remaining task', so that the project would be fruitful. The **EHTeX** project was done partially as senior project at University of Houston-Downtown. Above all, it wouldn't have been possible to come this far without a profound commitment and genuine participation of Dr. Brian Bourgeois and Dr. Victor Espino, the faculty members of Department of Applied Mathematical Sciences at University of Houston-Downtown.

Bibliography

- [1] E. A. Wallis Budge, KT, A HISTORY OF ETHIOPIA, Anthropological, 1966, page 557-560.

- [2] E. Ullendorff, THE ETHIOPIANS, Oxford University Press, 1973, page 126-130.
- [3] Donald E. Knuth, The **TeX**book, Addison-Wesley, 1986 page 43-49.
- [4] Donald E. Knuth, The **METAFONT**book, Addison-Wesley, 1986.

Typesetting Modern Greek with 128 Character Codes

Yannis Haralambous and Klaus Thull

"Ved dette været,
når det regner
så snør det"

— Frøydis Frosk

In european scripts where diacritical marks are common, there are (at least) two reasons to avoid **TeX**'s accent mechanism in favor of many accented characters.

One is the possible misplacement of accents by dvitype's rounding algorithm; the second is lack or invalidity of hyphenation. For example, large portions of german text may be unhyphenatable, and, given the german inclination to long words, may not be in shape to be typeset at all. Thus, in Europe, the obvious thing to do is: let **METAFONT** put the accents onto the letters, then access these characters via **TeX**'s ligature mechanism.

Accordingly, the greek fonts created by Silvio Levy¹ have 256 characters each, and are a fine tool to typeset greek texts, ancient as well as modern, except those containing the most recent unique accent "·" (see below). But alas, there is the commercial world, whose device drivers just cannot do 256-code fonts (even .pxl-fonts were seen on the "Big-Tech" sales exhibition in West Berlin last winter). The free drivers are in better shape generally, but often the commercial ones cannot be disposed of in a hurry. So we decided to reduce these fonts to 128 characters. We kept only the ones strictly necessary for writing modern greek without misusing the \accent primitive. At the same time, we constructed some new fonts, which we describe below.

The Reduced Greek Fonts

In modern post-war greek, the use of the grave accent ``'' ($\beta\alpha\rho\varepsilon\iota\alpha$) progressively faded, so that only two accents and the breathings were left (this was the kind of greek the first author learned at school). So the first reduction we did on Levy's fonts was to omit all grave accents. Secondly, we made σ a free character again, so that in the transliteration one has to type s for σ , and c for ς . Thirdly, we omitted the iota subscribed letters. All these, however, can still be accessed by macros if so wished.

Let's recall now the procedure, most of which is due to Levy: to typeset greek, you get into "Greek mode" by typing `\begingreek`. Similarly, you get out by typing `\endgreek`, but if you have to do this often, it is better to type `\greekdelims` at the beginning of your file. In that case $\$$ is used to enter and leave greekmode, and `\mathit{math}` takes the former meaning of $\$$ (do not forget to type "`_`"). The transliteration code is the following:

```
 $\alpha \beta \gamma \delta \epsilon \zeta \eta \theta \iota \chi \lambda \mu \nu \xi \circ \pi \rho \sigma \varsigma \tau \upsilon \varphi \chi \psi \omega$ 
 $a b g d e z h j i k l m n x o p r s c t u f q y w$ 
```

To get an acute ($\acute{\alpha}$), or circumflex ($\pi\acute{e}r\iota\sigma\pi\omega\mu\acute{e}\neta$) accent you type ' (single quote), or ~ (tilde) resp. in front of the vowel. To get a rough ($\grave{\alpha}$) or smooth ($\grave{\psi}\lambda\grave{\eta}$) breathing you type < or >, resp., in front of the vowel or the accent, if there is one. A diaeresis ($\grave{\delta}\iota\alpha\lambda\upsilon\tau\iota\kappa\acute{\alpha}$) is represented by " (double quote), and for a diaeresis with acute accent, just type " (double quote, single quote). To get a vowel with a iota subscript ($\grave{\nu}\pi\acute{o}\gamma\grave{\rho}\alpha\mu\acute{e}\neta$) you have to use the macro `\I{...}`.

If you need one of the omitted accents or combination of accents, you can get it by a macro: for example

- \grave{e} by `\grave{e}`,
- \grave{a} by `\grave{a}`,
- $\grave{\eta}$ by `\grave{\eta}`,
- \grave{i} by `\grave{i}`,
- \grave{o} by `\grave{o}`,
- $\grave{\omega}$ by `\grave{\omega}`,
- $\grave{\upsilon}$ by `\grave{\upsilon}`,
- $\grave{\rho}$ by `\grave{\rho}`,
- $\grave{\chi}$ by `\grave{\chi}`.

Finally, you have access to the following punctuation marks:

```
 $\cdot , . : ! ; ' " "$ 
 $\cdot , ; : ! ? ' " (( ))$ 
```

as well as the ten digits, parentheses, hyphen, en-dash, slash, percent sign, asterisk, plus and equal signs.

Some New Fonts

To write mathematics in greek one also needs slanted letters (for statements of theorems, according to `amsppt` style) and small capital letters (for titles and references). We have constructed these fonts, in the same reduced way, so that, together with the reduced Levy fonts, we obtain a complete family of greek fonts, namely regular, boldface, slanted, and small caps. We have called these fonts `rgrrrg`, `rgrbf`, `rgrsl`, `rgrsc`. Inside of greek mode you just write `\bf`, `\sl`, `\smc` as usual and `\rg` (instead of `\rm`) to get the regular greek font.

Here is an example of an alleged mathematical text, complete with translation:

1.1.4. ΘΕΩΡΗΜΑ. Γιά κάθε θετικό όκεραιο n , υπάρχει μία τριάδα μή μηδενικών όκεραιών (x, y, z) , τέτοια

ὅτε

$$x^n + y^n = z^n.$$

ΑΠΟΔΕΙΞΗ ΤΟΥ 1.1.4. Γιά $n = 2$, φτάνει νά πάρουμε $x = 3, y = 4, z = 5$. Γιά $n > 2$, όφήνουμε τήν όποδειξη στόν άναγνώστη σάν ασκηση. **Q.E.D.**

1.1.4. THEOREM. For each positive integer n , there exists a triple of non-zero integers (x, y, z) such that

$$x^n + y^n = z^n.$$

PROOF OF 1.1.4. For $n = 2$, we find $x = 3, y = 4, z = 5$. For $n > 2$, the proof left to the reader as an exercise. **q.e.d.**

FONTS FOR ONE-ACCENT GREEK

Some years after the re-establishment of democracy in Greece in 1974, a new system of accentuation has been introduced, omitting completely breathings and subscript iota, and simplifying the two remaining accents into one "universal accent," ' ($\tau\o\iota\kappa\acute{\alpha}$). This system is currently taught at school and it seems that any official document (including written examinations in some high schools) written in the old fashion multi-accent system is considered invalid (!).

So we thought that perhaps people would like to write in the old system and have a font to print the same text in fully official one-accent greek. We created fonts analogous to the reduced regular, boldface, and slanted which we have denoted by the prefix "m" (for $\mu\o\iota\o\tau\o\iota\kappa\acute{\alpha}$): `mrgrrrg`, `mrgrbf`, `mrgrs1`. Note that the small capitals font `rgrsc`

doesn't have any accents at all, and so may be used in any accent system.

These new fonts are designed to work with the same input as the old accent system. The printed text will follow the current grammar² (at least concerning the accent), with one exception: monosyllables (like articles, prepositions and other auxiliary words) don't take any accent at all. To solve this problem we are working at a Pascal word processor program, based on Fred M. Liang's packed trie device, which will, once given the list of the accented monosyllables, recognize them, and replace them by non-accented words. According to the dictionary of H. Mihiotis³, there are 284 such words, to which we must add many new and foreign words.

Of course, you can write your text in one-accent greek right away (unfortunately there is no "magic" macro to transform it back into multi-accent greek ...). With these new fonts you will get a nice symmetric "universal" accent instead of an acute or a circumflex.

To write in one-accent greek you get into "Greek one-accent mode", by typing \begin{greek}. If you are in greek multi-accent mode already, you must use the macro \monotoniko. There is also the converse macro \polotoniko. So if you want to obtain

O Ηράκλειος ἐλεγε «τὰ πάντα φεῦ»
και εἰχε δίκιο! ...

you type

```
\begin{greek} O Hr'akleitoc 'elege
\polotoniko
((t\grave{a} p'anta \rhorough{e~i})
\monotoniko
kai e'iqe d'ikio!...\end{greek}
```

The Greek Numeral Symbols

The so-called Ionian⁴ system of numeration (~fifth century BC) consisted of the following numerals:

A	B	Γ	Δ	E	F	Z	H	Θ
1	2	3	4	5	6	7	8	9
I	K	Λ	M	N	Ξ	Ο	Π	Ψ
10	20	30	40	50	60	70	80	90
P	Σ	T	Υ	Φ	X	Ψ	Ω	Δ
100	200	300	400	500	600	700	800	900

The letters F, Ψ, Δ are called digamma, qoppa, sanpi. They belong to an older alphabet. Later on, as lowercase letters were introduced and as the need for higher numbers grew, the numerals became:

α'	β'	γ'	δ'	ε'	Ϛ'	ζ'	η'	θ'
1	2	3	4	5	6	7	8	9

ι'	χ'	λ'	μ'	ν'	ξ'	ο'	π'	γ'
10	20	30	40	50	60	70	80	90
ρ'	σ'	τ'	υ'	φ'	χ'	ψ'	ω'	ϗ'
100	200	300	400	500	600	700	800	900
α	β	γ	δ	ε	Ϛ	ζ		
1000	2000	3000	4000	5000	6000	7000		
		η		θ		M		
				8000	9000	10000		

So, for example, the date February 16th, 1989 would be written ΙϚ' Φεβρουαρίου ,αλπθ and the following equality holds:

$$\sigma\zeta' + \psi\pi\theta' = \lambda\vartheta\zeta'.$$

Notice that there is no zero. Zero is, and has always been, the cardinal of the empty set which in Ancient Greece was not considered an entity in its own right.

To express numbers greater than 10,000 there were many ways. One of them was to use 10,000 as a base: thus, for example, 67,536,753 (= $6753 \cdot 10,000 + 6753$) was written M,Ϛψνγ:,Ϛψνγ.

EXERCISE: If -γωνο means "-gon", which of the following polygons can be constructed by rule and compasses?

Ϛ'-γωνο, λϙ'-γωνο, δτξθ-γωνο,
ακδ-γωνο, Με'-εφλζ-γωνο, λϙϙ'-γωνο.

Let's return now to TeX: you can obtain these symbols by the following macros: \digamma for ϖ, \vardigamma for ρ, \Digamma for F, \qoppa for Ψ, \Qoppa for Π, \sanpi for ς, and \Sanpi for Δ. To get the tick marks which distinguish units and thousands, you can use \overstroke after the numeral, or \understroke in front of the numeral.

Symbols for cypriotic greek

The official language of Cyprus is greek. It is also the language used in the mass-media and at school. But the language actually spoken is a dialect, derived from byzantine greek (and as it seems, far more faithful to ancient greek than the one spoken in Greece). Some literature has been written in the dialect, and since there are phonemes not available in the greek alphabet, cypriotic writers use several conventions of new symbols to express them.

In the convention we followed,⁵ the symbols ς, Ž stand for the sound "sh" (like "shower" in english, or "шашка" in russian), ζ, Ζ stand for "j" (like "jazz" in english or "джунгли" in russian), ψ, Ψ stand for a ϕ followed by a ς (like "пшеницза"

in russian) and ξ , Ξ stand for a χ followed by a $\check{\sigma}$ (like in hindi “ksetriya”).

You can get these symbols by the macros `\ssh`, `\SSH`, `\dz`, `\DZ`, `\psh`, `\PSH`, `\ksh`, `\KSH`. Here is an example of a small text using these symbols:

Δκιαλέξες τούς ξερόροτσους, δεντρόν μου, νά ριζώσεις,
νά φάεις τήν ζωούλαν σου, ποττέ σδν θά στερκώσεις.
Τζί εν κανεὶ πον'ξεροκαγιάς τζί, εν ἔσει στάξην χώμα
νά σόσιεπαστοῦν οἱ ρίζες σου, πλαστήκαν τζί αλλ' ἀκόμα.
Οι σπαλαθικίες τζάι οι βατιές πᾶσιν νά σέ τυλίξουν,
ἐπκιάσαν σε πού τόν λαψύν, σφίγγουν σε νά σέ πνίξουν.
Τζί άν πλέσει μιά σταξιά νερόν, ἐννά τήν πίννουν ἄλλοι
τζί ἐννά σκεντζέρεις τζειαχαμάι, ἐναν ξεροδρουμπάλιν.
Ἐθθα ξορτώσεις νά σόσιαστεις ποττέ τοῦ ἥιου ἀμμάτι,
γιατ' ἐννά σέ χασκιάζουσιν οι σπαλαθικίες τζί οι βάτοι.
Τζί ἐσούνη, κακορίζικον, ἐννά δουλεύκεις γρόνους
τζάι νά γυρεύκεις 'ποσπασίαν πού τούς κακούς γειτόνους.
Τζί οι κόποι εἰς τ' ἀνάθιθεμαν. Μάγκουμου μέν προκάλμεις...
τζί ἐσούνη τό χαΐριν μου, φτωχόν μου, ἐννα κάμεις.

From Ilias Georgiou's "Geloklamani"

On hyphenation

There is still no greek hyphenation list, so one has to use hyphenation from other languages. We have compared on an ordinary text, the standard english (Liang), the german (Schwarz) and portuguese (Rezende)⁶ hyphenation patterns. The results were surprising: on 267 possible hyphens, these three patterns missed 199 (!!), 141 and 149 resp, found 46, 115, 111 correct ones and 22, 11, 7 bad ones (the portuguese mistakes were less embarrassing than the german ones). So, for a temporary substitute, we would choose either the german or the portuguese patterns.

And since you will be forced to make corrections by hand, here are the complete actual rules of greek hyphenation:

Let c_1, c_2, \dots, c_n be consonants ($n \geq 2$) and v_1, v_2, v_3, v_4 vowels. Then we have

RULE 1. The combination $v_1c_1v_2$ is separated as
 $v_1 - c_1v_2$ (ex. $\pi\alpha-\alpha-x\alpha-\lambda\tilde{\omega}$)

RULE 2. The combination $v_1c_1 \dots c_nv_2$ is separated as $v_1 - c_1 \dots c_nv_2$ if there is a greek word starting

Rule 3. The combinations of vowels v_1 , v_2 , v_3 , v_4 and consonants c_1 , c_2 , c_3 , c_4 are to be generated by the following rules:

RULE 3. The combinations of vowels v_1v_2 , $v_1v_2v_3$ or $v_1v_2v_3v_4$ are not to be separated if they are pronounced as one phoneme ($\ddot{\alpha}\eta\delta\circ\nu$ but $\ddot{\alpha}\circ\acute{\eta}\tau\circ\eta\circ\tau\circ\varsigma$, $\pi\circ\acute{o}$ but $\pi\circ\acute{u}\circ o$).

S. Levy made in his fonts separate characters of all possible accented letters, to prevent problems

of hyphenation (TEX doesn't yet hyphenate words with accents). The only exceptions he made, were the two combinations ``, '' which occur only on monosyllables.

In our case, to be able to reduce the fonts, we were forced to make accents also of ̄, ̄, ̄, ̄, (and macros of ̄, ̄, ̄, ̄, ̄, ̄, ̄, ̄, ̄ as already mentioned). This of course adds problems to hyphenation. Nevertheless, ̄, ̄ occur on one- and 2-syllable words only, ̄, ̄ occur rarely and depends on the kind of language one is writing (to find it, you have to go back to older versions of καθαρεύουσα as for example in the following lovely text:

Ἐνῷ ἐσπέραν τινὰ ἔξαντλήσας τὰ μυρολόγια του
ἐκομάτῳ δὲ Φρουμέντιος ἐπὶ τῆς ἄμμου τῆς παραλίας,
καταβὰς ἐξ οὐρανῶν ὁ ἀπόστολος ἐκεῖνος τῶν Σαξό-
νων ἤνοιξε διὰ μαχαίρας τὰ στήθη τοῦ κοιμωμένου,
εἰσήγαγε τοὺς Ἱεροὺς δακτύλους του εἰς τὴν ὄπην καὶ
ἔξαγὼν τὴν καρδίαν ἐβύθισεν αὐτήν εἰς λάκκον πλήρη
ῦδατος, ὅπερ ἥγιασεν προηγουμένως. Ἡ φλέγουσα
ἐκείνη καρδία ἔφριξεν εἰς τὸ ῦδωρ ὡς σμαρὶς ἐντὸς
τοῦ τηγανίου, ἀφοῦ δὲ ἐκρύωσεν, ἔθεσε πάλιν αὐτήν
δὲ ἄγιος εἰς τὸν τόπον της καὶ κλείσας τὴν πληρήν
ἐπέστρεψεν εἰς τὸν ίδικόν του.

Ἐτυχέ ποτε, ἀναγνῶστά μου, νὰ ἀποκοινηθῆς μὲ
ἀνυπόφορον βῆχα, κοιμώμενος νὰ ἰδρώσῃς καὶ ἔξυ-
πνήσας νὰ εὑρεθῆς λατρευμένος; Ἀγνοῶν δτι εἶσαι
καλὰ ἀνοίγεις μηχανικῶς τὸ στόμα, ἵνα πληρώσῃς εἰς
τὸν ἐπικατάρατον βῆχα τὸν συνήθη φόρον. Ἄλλὰ
πόσην αἰσθάνεσαι χαράν, μὴ εύρισκων εἰς τὸν λά-
ρυγγα τὸ ὄχληρόν θηρίον! Οὕτω ὅμα ήνοιξε καὶ ὁ
Φρουριμέντιος τοὺς ὄφθαλμούς, ἡτοιμάσθη νὰ προσφέ-
ρῃ εἰς τὴν ἀχάριστον Ἰωάνναν τὴν συνήθη δακρύων
σπονδήν, ἀλλὰ παρὰ πᾶσαν προσδοκίαν οἱ ὄφθαλμοι
του εὑρέθησαν ξηροὶ καὶ νὰ προγευματίσῃ μᾶλλον ἢ
νὰ κλαύσῃ ἡσθάνετο ὅρεξιν μετὰ πολυήμερον νηστεί-
αν ὁ καλὸς Βενεδεκτῖνος.

From Emmanouil Boëdis' "The Popess Johanna (1866)"

Note that in one-accent greek, all accented letters are represented by separate characters in the code table, so that no hyphenation problem arises.

Samples, Tables, and Remarks

The font rgrrg10

Καθόταν μπροστά της καὶ τὴν κύπταζε. Νοιώθοντας μιὰ ἀπέραντη εὐχαρίστηση νά τὴν βλέπει ἔτσι μπροστά του καὶ ἔνα ἀπέραντο ἀνικανοποίητο πού δέν μποροῦσε νά τὴν τραβήξει στὴν ἀγκαλιά του καὶ νά τὴν φιλήσει ἐκεῖ στὸν λαιμό πού τὴν εἶχε φιλήσει τὴν μία καὶ μοναδική φορά καὶ εἶχε νοιώσει μέσα του τὴν πιό

Layout for fonts rgrrg, rgrbf, rgrsl

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	-	à	à	à	à	ă	ă	ń	"0x
'01x	ń	ń	ń	ń	ń	ó	ó	ó	
'02x	ő	ő	ő	í	í	í	í	í	"1x
'03x	í	ú	ú	ú	ú	ü	ü	é	
'04x	é	!	"	"	é	%	ě	'	"2x
'05x	()	*	+	,	-	.	/	
'06x	0	1	2	3	4	5	6	7	"3x
'07x	8	9	:	.	'	=	'	;	
'10x	ˇ	A	B	ó	Δ	E	Φ	Γ	"4x
'11x	H	I	Θ	K	Λ	M	N	O	
'12x	Π	X	P	Σ	T	Υ	ˇ	Ω	"5x
'13x	Ξ	Ψ	Z	ö	ˇ	ö	ˇ	ó	
'14x	ð	α	β	ς	δ	ε	φ	γ	"6x
'15x	η	ι	θ	χ	λ	μ	ν	ο	
'16x	π	χ	ρ	σ	τ	υ	ü	ω	"7x
'17x	ξ	ψ	ζ	«	„	»	—	ě	
	"8	"9	"A	"B	"C	"D	"E	"F	

Όμορφη στιγμή τοῦ καλοκαιριοῦ κι ὅτι τό καλοκαίρι αὐτό, φτάνοντας στό ἀποκορύφωμά του, εἶχε κιόλας περάσει. Ἀλλά δέν εἶχε ἀκόμα περάσει τελείως, γιατὶ ἔκεινη βρισκόταν τώρα μπροστά του. Καὶ δέν μποροῦσε νά κάνει τίποτα.

The font rgrbf10

Καθόταν μπροστά της καὶ τήν κύτταζε. Νοιώθοντας μιά ἀπέραντη εὐχαρίστηση νά τήν βλέπει ἔτσι μπροστά του καὶ ἔνα ἀπέραντο ἀνικανοποίητο πού δέν μποροῦσε νά τήν τραβήξει στήν ἀγκαλιά του καὶ νά τήν φιλήσει ἔκει στόν λαιμό πού τήν εἶχε φιλήσει τήν μία καὶ μοναδική φορά καὶ εἶχε νοιώσει μέσα του τήν πιό Όμορφη στιγμή τοῦ καλοκαιριοῦ κι ὅτι τό καλοκαίρι αὐτό, φτάνοντας στό ἀποκορύφωμά του, εἶχε κιόλας περάσει. Ἀλλά δέν εἶχε ἀκόμα περάσει τελείως, γιατὶ ἔκεινη βρισκόταν τώρα μπροστά του. Καὶ δέν μποροῦσε νά κάνει τίποτα.

The font rgrsl10

Καθόταν μπροστά της καὶ τήν κύτταζε. Νοιώθοντας μιά ἀπέραντη εὐχαρίστηση νά τήν βλέπει ἔτσι μπροστά του καὶ ἔνα ἀπέραντο ἀνικανοποίητο πού δέν μποροῦσε νά τήν τραβήξει στήν ἀγκαλιά του καὶ νά τήν φιλήσει ἔκει στόν λαιμό πού τήν εἶχε φιλήσει τήν μία καὶ μοναδική φορά καὶ εἶχε νοιώσει μέσα του τήν πιό

όμορφη στιγμή τοῦ καλοκαιριοῦ κι ὅτι τό καλοκαίρι αὐτό, φτάνοντας στό ἀποκορύφωμά του, εἶχε κιόλας περάσει. Ἀλλά δέν εἶχε ἀκόμα περάσει τελείως, γιατὶ ἔκεινη βρισκόταν τώρα μπροστά του. Καὶ δέν μποροῦσε νά κάνει τίποτα.

The font mrgrrg10

Καθόταν μπροστά της καὶ τήν κύτταζε. Νοιώθοντας μια απέραντη ευχαρίστηση να τήν βλέπει ἔτσι μπροστά του καὶ ἔνα απέραντο ανικανοποίητο που δέν μπορούσε να τήν τραβήξει στήν αγκαλιά του καὶ να τήν φιλήσει εκεί στόν λαιμό που τήν εἶχε φιλήσει τήν μία καὶ μοναδική φορά καὶ εἶχε νοιώσει μέσα του τήν πιό Όμορφη στιγμή

The font mrgrbf10

Καθόταν μπροστά της καὶ τήν κύτταζε. Νοιώθοντας μια απέραντη ευχαρίστηση να τήν βλέπει ἔτσι μπροστά του καὶ ἔνα απέραντο ανικανοποίητο που δέν μπορούσε να τήν τραβήξει στήν αγκαλιά του καὶ να τήν φιλήσει εκεί στόν λαιμό που τήν εἶχε φιλήσει τήν μία καὶ μοναδική φορά καὶ εἶχε νοιώσει μέσα του τήν πιό Όμορφη στιγμή

Layout for fonts mrgrrg, mrgrbf, mrgrs1

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	-	α	α	α	α	ă	ă	ń	"0x
'01x	ń	ń	ń	ń	ń	ω	ω	ó	
'02x	ő	ő	ő	í	í	í	í	í	"1x
'03x	í	ú	ú	ú	ú	ü	ü	é	
'04x	é	!	"	"	é	%	ě	'	"2x
'05x	()	*	+	,	-	.	/	
'06x	0	1	2	3	4	5	6	7	"3x
'07x	8	9	:	.		=		;	
'10x	·	A	B	ó	Δ	E	Φ	Γ	"4x
'11x	H	I	Θ	K	Λ	M	N	O	
'12x	Π	X	P	Σ	T	Υ	·	Ω	"5x
'13x	Ξ	Ψ	Z	ó	·	ó	·	o	
'14x	o	α	β	ς	δ	ε	φ	γ	"6x
'15x	η	ι	θ	χ	λ	μ	ν	ο	
'16x	π	χ	ρ	σ	τ	υ	ü	ω	"7x
'17x	ξ	ψ	ζ	«	„	»	·	ě	
	"8	"9	"A	"B	"C	"D	"E	"F	

Layout for font rgrsc

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	-	'	"	*	*	"	"	-	"0x
'01x	þ	þ	ȝ	F	F	,	â	â	"1x
'02x	ȝ	ȝ				'			
'03x						,			
'04x	!	"	"			%			"2x
'05x	()	*	+	,	-	.	/	
'06x	0	1	2	3	4	5	6	7	"3x
'07x	8	9	:	.		=		;	
'10x		A	B		Δ	E	Φ	Γ	"4x
'11x	H	I	Θ	K	Λ	M	N	O	
'12x	Π	X	P	Σ	T	Υ		Ω	"5x
'13x	Ξ	Ψ	Z						
'14x		A	B	Σ	Δ	E	Φ	Γ	"6x
'15x	H	I	Θ	K	Λ	M	N	O	
'16x	Π	X	P	Σ	T	Υ		Ω	"7x
'17x	Ξ	Ψ	Z	«	.	»		⌚	
	"8	"9	"A	"B	"C	"D	"E	"F	

του καλοκαιριού κι ότι το καλοκαιρι αυτό, φτάνοντας στο αποκορύφωμά του, είχε κιόλας περάσει. Αλλά δεν είχε ακόμα περάσει τελείως, γιατί εκείνη βρισκόταν τώρα μπροστά του. Και δεν μπορούσε να κάνει τίποτα.

The font mrgrs110

Καθόταν μπροστά της και την κύτταζε. Νοιώθοντας μια απέραντη ευχαριστηση να την βλέπει έτσι μπροστά του και ένα απέραντο ανικανοποίητο που δεν μπορούσε να την τραβήξει στην αγκαλιά του και να την φιλήσει εκεί στον λαμβό που την είχε φιλήσει την μία και μοναδική φορά και είχε νοιώσει μέσα του την πιο δμορφη στιγμή του καλοκαιριού κι ότι το καλοκαιρι αυτό, φτάνοντας στο αποκορύφωμά του, είχε κιόλας περάσει. Αλλά δεν είχε ακόμα περάσει τελείως, γιατί εκείνη βρισκόταν τώρα μπροστά του. Και δεν μπορούσε να κάνει τίποτα.

We conclude with the following remark: people writing french, czech, turkish or other languages with many diacritical marks complain that there is no space left in Computer Modern to incorporate already-accented letters. The solution (in the case of French) that Désarménien⁷ proposed, was to replace greek uppercase letters by the french é, è, ê, ô, ï, â, û, à, ù. But then the question is: where to put the greek uppercase letters, which are necessary for mathematical formulas. We answer: if

you have the greek rgr family of fonts, you already have all kinds of greek uppercase letters. Just take them from there! Of course, math families must be restructured in that case since math family 7 cannot be used for those letters anymore. As Gariepy⁸ pointed out already, the inconvenience with this solution is that for every language with accents you will need another cm family of fonts. That's why we still believe that the best once and for all solution would be to be able to work with fonts of 256 characters.

References

1. S.LEVI: Using Greek Fonts with TEX, *TUGboat*, 9 (1988) 20-24
2. M.ΤΡΙΑΝΤΑΦΥΛΛΙΔΗ: Νεοελληνική Γραμματική, Οργανισμός Έκδοσης Σχολικών Βιβλίων, Αθήνα 1982
3. Χ.ΜΗΧΙΩΤΗ: Νεώτατον Λεξικόν τῆς Νεοελληνικῆς Γλώσσης, Έκδόσεις Κασταλία, Αθήναι 1972
4. C.B.BOYER: A History of Mathematics, J. Wiley & Sons, New York 1968
5. Η.ΓΕΩΡΓΙΟΥ: Γελόχλαμάν, Σειρά Κυπριακῆς Λαϊκῆς Ποίησης Υπουργείου παιδείας 4, Λευκωσία 1980
6. UNIX TEX distribution tape, Seattle 1988
7. J.DÉSARMÉNIEN: How to run TEX in a French environment: hyphenation, fonts, typography, *TUGboat*, 5 (1984) 91-102
8. A.GARIEPY: French in TEX, *TUGboat*, 9 (1988) 65-69

- ◊ Yannis Haralambous
U.F.R. de Mathématiques
Université de Lille-Flandres-Artois
59655 Villeneuve d'Ascq Cedex
France
Bitnet: yannis@frccitl71
- ◊ Klaus Thull
Freie Universität Berlin
Usenet: thull@fubinf

Erratum:

Chess Printing via METAFONT and TEX
TUGboat Vol. 10, No. 2

Zalman Rubinstein

Editor's note: Within the METAFONT code for the pawn on p. 171, {dir 135} should be replaced by {dir 315}.

Resources

Users' Guide to LaTeX-help

Max Hailperin

All sites with L^AT_EX should have one or more L^AT_EX experts to help users. Those experts communicate with each other about difficult problems through various forums, including the TeXhax mailing list.

Lately, many sites have installed L^AT_EX without having, acquiring, or developing a L^AT_EX expert. Many simple L^AT_EX questions from those sites have been posted directly to TeXhax, clogging it and prompting redundant replies.

Therefore, a number of TeXhax subscribers have formed a volunteer L^AT_EX question answering corps. L^AT_EX users with questions should take the following steps:

numbered

- Read the manual very carefully, including a careful check of the index. Most questions are answered there.
- Check whether anyone locally can answer your question. Consider not only paid systems staff but also more experienced users. Similarly, if you paid a commercial company good money for L^AT_EX, you should demand customer support from them—after all L^AT_EX is available for free.
- See if you can work it out yourself, and in the process build L^AT_EX expertise, by use of careful test cases, tracing mode, examining the L^AT_EX source files, etc. Don't go crazy if you're a non-programmer, but give it a shot.
- If all of the above fail, don't send mail to TeXhax. Instead, send mail to

`LaTeX-help@sumex-aim.Stanford.EDU`.

Your mail will automatically be forwarded to a member of the volunteer corps, in a round-robin rotation. You should hear back shortly, either with a solution to your problem, a request for additional information, or the remark that it exceeded the volunteer's abilities and has been forwarded to other experts, including further volunteers and the TeXhax mailing list. If you don't hear anything after waiting a reasonable period, write to

`LaTeX-help-coordinator@`

`sumex-aim.Stanford.EDU`

with as much information about your original

mailing as you have, and the coordinator will try to track down how it got lost.

Please do not abuse this service. We volunteers have lots of work of our own to do, and will not continue volunteering if the burden is excessive. Make sure you try steps 1-3 before step 4, and always be eager to help others locally who are a step behind you. Also, join TUG (the TeX Users Group) if you haven't and avail yourself of their classes and publications to develop in-house L^AT_EX expertise.

If you have any questions or comments on this, please write to

`LaTeX-help-coordinator@`

`sumex-aim.Stanford.EDU`

not directly to the current person holding that position, as it may change.

Editor's note: Additional volunteers for this project are welcome. Anyone who wishes to lend a hand should get in touch with the coordinator.

◊ Max Hailperin
Stanford Knowledge Systems
Laboratory
701 Welch Rd., Bldg. C
Palo Alto, CA 94304
`mxh@sumex-aim.Stanford.EDU`

TeX-Ed

Yin Kean

At the TeX User's Group 10th Anniversary Conference (September 1989), many attendees expressed similar concerns on the courses that are currently being offered by the TeX User's Group. In general, it was believed that the TeX user community is evolving and that training materials and courses need to be developed for the changing nature of the TeX user community. As a result, several proposals were drawn-up and discussed at the conference. To continue discussing and refining the proposals, TeX-Ed, a listserv mailing list has been created at the University of Illinois at Chicago. TeX-Ed will also serve as a forum on all educational aspects of TeX in general, and teaching materials such as templates, macros, etc. will also be obtainable from TeX-Ed archives. The teaching materials may

eventually be of interest to TeX User's Group, and will be removed, in which case, from the archive and be handled by them.

To Subscribe or Signoff TeX-Ed

TeX-Ed is accessible to all through Bitnet or Internet via a LISTSERV machine. To add your name to a list, IBM/VM users should send an interactive message as follows:

```
TELL LISTSERV AT UICVM SUB TeX-Ed <your name>
```

For example,

```
TELL LISTSERV AT UICVM SUB TeX-Ed John Doe
```

LISTSERV picks up your userid and node name automatically.

VAX/VMS sites running JNET version 2 can send an interactive message such as:

```
SEND LISTSERV@UICVM SUB TeX-Ed <your name>
```

Users with other environments can contact their local user services group for assistance. Users without interactive messaging capability can send "Command jobs" to LISTSERV via RFC822, PROFS, or IBM NOTE formatted mail.

To remove your name from any list substitute SIGNOFF for SUB as:

```
SEND LISTSERV@UICVM SIGNOFF TeX-Ed
```

Submitting the Archives

The archives is maintained by the Computer Center at the University of Illinois at Chicago. The archives consist of course materials which any individual or groups have developed and find effective for the teaching of TeX. The material submitted must include the author's name, the person to contact for more information if different than the author, a description of the environment under which the program or file operates, date of the current version, and a comment if further development is expected. Send the file for the teaching archive to

U46491@UICVM

or

U46491@uicvm.cc.uic.edu

to be reviewed and installed.

Retrieving the Archives

Information in the archives can be retrieved as individual files describing single programs. If you have access to BITNET through a VM system you can communicate with the LISTSERV virtual machine by using the command

```
tell listserv at uicvm get [<filename> <filetype>]
```

where "<filename> <filetype>" will be replaced by the filename and filetype of the file in which you are interested.

If you are on the Internet, send a standard note addressed to

```
listserv@uicvm.cc.uic.edu
```

The note should contain just the appropriate LISTSERV commands, such as

```
get [<filename1> <filetype1>]
get [<filename2> <filetype2>]
```

substituting the names of the files of interest.

Individual files can be retrieved once you know the name of the appropriate file. For example,

```
tell listserv at uicvm get filename filetype
```

would retrieve the file FILENAME FILETYPE.

Other useful LISTSERV commands

The LISTSERV software allows you to request to be informed of any updates to certain files by means of the AFD ("Automatic File Distribution") and FUI ("File Update Information") commands. LISTSERV will send you information about its own commands if you request it. You can get this information by using the commands like:

```
info ?
info refcard
get listafd memo
```

These commands are used as the message of "tell listserv at uicvm" or the contents of standard notes sent to the LISTSERV machine.

Anonymous FTP

The archive files are also available via anonymous FTP at UICVM. After logging on to userid

ANONYMOUS@UICVM.CC.UIC.EDU

issue the command

CD TeX-Ed.191

to access the TeX-Ed directory.

◊ Yin Kean
 University of Illinois at Chicago
 Computer Center, Mail code 135
 Box 6998
 Chicago, IL 60680
U46491@UICVM.uic.edu

Contents of the Clarkson Archive Server as of 22 September 1989

Michael DeCorte

Due to the size of the archive, this listing will only contain updates to the archive. The first issue of the year contains the complete list of files.

A few changes have been made. One, you can now get files via tape for a small charge. Two, I am now working in Philadelphia but will continue to maintain the Archive Server. Special thanks to Rob Logan for helping to maintain the Archive.

As always, submissions are encouraged. If you do submit a file please include at the top of the file: your name; your email address; your real address; the date. Also please make certain that there are no lines in the file longer than 80 characters as some mailers will truncate them. Mail should be sent to

`mrd@sun.soe.clarkson.edu`
`archive-management@sun.soe.clarkson.edu`

For Internet users: how to ftp

An example session is shown below. Users should realize that ftp syntax varies from host to host. Your syntax may be different. The syntax presented here is that of Unix ftp. Comments are in parentheses. The exact example is for retrieving files from the **L^AT_EX Archive**; the syntax is similar for the other archives only the directories differ. The directory for each archive is given in its description.

Non-Internet users: how to retrieve by mail

To retrieve files or help documentation, send mail to `archiver-server@sun.soe.clarkson.edu` with the body of the mail message containing the command `help` or `index` or `send` and the command `path`. The `send` command must be followed by the name of the archive and then the files you want. The `path` command must be followed by a path from Clarkson to you in domain style format. This means

that Clarkson can only send mail to Internet, Bitnet and registered UUCP sites. Therefore `host!user` is guaranteed to bounce but `user@host.UUCP` will work. For example this user should send

```
To: archive-server@sun.soe.clarkson.edu
Subject:
path user@host.UUCP
send latex-style README Index
send latex-style resume.sty
```

Unfortunately it is not at this time possible for mail users to request files larger than 100k. They are only available via ftp or tape.

Traffic on the network servers and gateways has been very high recently, and in order to provide improved service, there have been some volunteers to maintain local "slave" repositories of the L^AT_EX style collection. There is usually a geographic or network restriction requested, since the idea is to cut down traffic, not add to it. The following areas will be covered by the volunteers listed.

- Bitnet users: Texas A&M maintains a list-server and file-server which is already handling (with `TEX-L`) much of the Bitnet distribution of `TEXhax`. An inquiry via listserv will retrieve a list of all `TEX`-related files:
`tell listserv at tamvml get tex filelist`
- UK users: Aston University maintains a `TEX` archive covering all aspects of `TEX`, `LATEX`, `METAFONT`, and ancillary software. `UKTEX` (like `TEXhax`) digests are distributed from Aston. For users with Colour book software FTP access is available; for all users mail access is available. Send enquires in the first instance to `info-tex@uk.ac.aston` (via internet use `pabbott@nss.cs.ucl.ac.uk`).
- Italian users: Marisa Luvisetto and Max Calvani maintain a SPAN/DECNET repository. They have software for redistribution

Sample FTP session for Internet users

```
% ftp sun.soe.clarkson.edu          (a.k.a. 128.153.12.3)
...
user: anonymous                      (general blurb)
password: <any non-null string>
ftp> cd pub/latex-style             (where the files are)
ftp> ls                             (to see what is there)
...
ftp> get Index                       (lots of output)
...
ftp> quit                           (more blurb)
```

like the latex-style collection, Beebe's driver family, **TExhax**, **TEXMAG**, **UKTeX** magazines, **dvitovdu**, **psprint**, **texsis**. For more info on what is available and how to get it, please send a mail message to **39947::luvisetto** or **39003::fisica**. American users can also contact Ed Bell at **7388::bell**. Max Calvani's internet address is **fisica@astrpd.infn.it**.

- Canadian users: A shadow copy of the **LATeX Style Archive** is kept on **neat.ai.utoronto.ca**, and is updated automatically from the master source. It can be accessed via anonymous FTP (128.100.1.65). Mail access is also possible by mailing to **info@ai.utoronto.ca** or **utai!info**. For more details about mail access, send a message to that address with a message body that reads

```
request: info
topic: help
request: latex-style
topic: info
```

Additional volunteers for slave depositories should contact Michael.

Tape Distribution by USMail

To obtain archives on tape via US-mail send

- Rob Logan
ERC
Clarkson University
Potsdam NY 13676

a self addressed stamped tape (8mm, 0.25 inch, 0.5 inch) with a check for \$20 made to Clarkson University and a list of the archives that you want (eg latex-style and texhax). You can ask for as many archives as will fit on the tape, but you may not ask for individual files. The tape will be written in Unix tar format. Unless you specify otherwise, the tape will be written at the highest possible density (0.5 inch at 6250 BPI, 8mm at 2.3 gigabytes per tape, 0.25 inch at 60 megabytes per tape). If you do not live in the US, we will provide postage if you send a self-addressed tape without stamps and a check for \$40 instead of \$20.

NOTE: if the tape is not self addressed we will keep the tape and use it for backups.

For your information, money is used to pay a student to copy the tapes, and any money left over is put into an account from which we will eventually buy a disk drive dedicated to the archive server. Contributions are strongly encouraged and tax deductible.

Distribution for IBM PC and clone users

There are two sources.

- David W. Hopper
446 Main Street
Toronto, Ontario
Canada M4C 4Y2

has **LATeX** style files only. David has been in a state of flux for a little while and would like to apologize for any delays. If you have not received requested files from him you should get in contact with him. Send:

1. Either one 1.44 MB 3.5 inch diskette, one 1.2 MB diskette or four 360 KB diskettes, blank and formatted;
2. Indication of the format required;
3. A self-addressed mailer; and
4. A \$5.00 donation per set of files, to cover postage and equipment wear & tear. (If you live outside North America, airmail delivery will probably require more postage. You should probably contact David for details.)
5. No phone calls or personal visits please.

- Jon Radel
P. O. Box 2276
Reston, VA 22090

has **LATeX** style files and other material including **TeX**. For a list of what is available and other information send a SASE.

AMSTeX Sources

This contains the **TeX** source needed to build *AMSTeX*. It is a duplicate directory of **tex.amstex** on Score. Files are located in **pub/amstex** for ftp users. Mail users should request files from the **amstex** archive.

AMSTeX Style

This contains style files specific to *AMSTeX* users. Files are located in **pub/amstex-style** for ftp users. Mail users should request files from the **amstex-style** archive.

BIBTeX Sources

This contains the **BIBTeX** style files and the **WEB** files need to build **BIBTeX**. It is a duplicate directory of **tex.bibtex** on Score. Files are located in **pub/bibtex** for ftp users. Mail users should request files from the **bibtex** archive.

BIB_TE_X Style

This contains files that are specific to version 0.99 of BIB_TE_X. Many of these files are to be used with files in the L^AT_EX Collection. Files are located in pub/bibtex-style for ftp users. Mail users should request files from the bibtex-style archive.

BIB_TE_X 0.98 Style

This contains files that are specific to version 0.98 of BIB_TE_X. Many of these files are to be used with files in the L^AT_EX Collection. Files are located in pub/bibtex-style-0.98 for ftp users. Mail users should request files from the bibtex-style-0.98 archive.

CM Fonts

This contains the METAFONT files needed to build the CM fonts. It is a duplicate directory of tex.cm on Score. Files are located in pub/cm-fonts for ftp users. Mail users should request files from the cm-fonts archive.

DVI Driver Standards

This contains digests from the DVI Driver standards committee and articles about DVI standards here. Files are located in pub/dvi-standard for ftp users. Mail users should request files from the dvi-standard archive. Digests are named driver.YY.MM.V where YY is the year of the issue, MM is the month and V is the volume.

L^AT_EX Sources

This contains the T_EX files needed to build L^AT_EX. It is a duplicate directory of tex.latex on Score. Files are located in pub/lamport for ftp users. Mail users should request files from the lamport archive.

METAFONT Sources

This contains the WEB files needed to build METAFONT. It is a duplicate directory of tex.mf on Score. Files are located in pub/mf for ftp users. Mail users should request files from the mf archive.

T_EX Documentation

This contains documentation on T_EX. It is a duplicate directory of tex.doc on Score. Files are located in pub/tex-doc for ftp users. Mail users should request files from the tex-doc archive.

T_EX Inputs

This contains the T_EX files needed to build plain T_EX. It is a duplicate directory of tex.inputs on Score. Files are located in pub/tex-inputs for ftp users. Mail users should request files from the tex-inputs archive.

T_EX Sources

This contains the WEB files needed to build T_EX. It is a duplicate directory of tex.web on Score. Files are located in pub/tex-source for ftp users. Mail users should request files from the tex-source archive.

T_EX Fonts

This contains the METAFONT files for user contributed fonts. Files are located in pub/tex-fonts for ftp users. Mail users should request files from the tex-fonts archive.

T_EX Programs

This contains programs that are of general interest to T_EX users in general. Files are located in pub/tex-programs for ftp users. Mail users should request files from the tex-programs archive.

T_EX Style

This contains style files for plain T_EX. Files are located in pub/tex-style for ftp users. Mail users should request files from the tex-style archive.

T_EX Tests

This contains the files needed to test T_EX using the triptest. It is a duplicate directory of tex.tests on Score. Files are located in pub/tex-tests for ftp users. Mail users should request files from the tex-tests archive.

T_EXhax Digests

This contains all of the back issues of T_EXhax. Files are located in pub/texhax for ftp users. Mail users should request files from the texhax archive. Files are named texhax.YY.NNN where YY is the year of the issue and NNN is the issue number.

T_EXMAG Digests

This contains all of the back issues of T_EXMAG. Files are located in pub/texmag for ftp users. Mail users should request files from the texmag archive. Files are named texmag.V.NN where V is the volume number and NN is the issue number.

Transfig Sources

This contains the C source for Transfig, a program that converts Fig output to other forms such as *PCTEX*. Files are located in pub/transfig for ftp users. Mail users should request files from the transfig archive.

TUGboat Files

This contains files related to *TUGboat*. It is a duplicate directory of *tex.tugboat* on Score. Files are located in pub/tugboat for ftp users. Mail users should request files from the *tugboat* archive.

UK_{TEX} Digests

This contains all of the back issues of UK_{TEX}. Files are located in pub/uktex for ftp users. Mail users should request files from the *uktex* archive. Files are named *uktex.YY.NNN* where YY is the year of the issue and NNN is the issue number.

L_AT_EX Style

This contains files that are specific to L_AT_EX. Most of these are style files but some of them are programs. Some of the files have support BIB_{TEX} style files that are in the **BIB_{TEX} Collection** or the **BIB_{TEX} 0.98 Collection**. Files are located in pub/latex-style for ftp users. Mail users should request files from the *latex-style* archive.

boxedminipage.sty

puts a box round a minipage

bsf.sty Provide access to bold san serif fonts in L_AT_EX

deproc.sty DECUS proceedings style

farticle.sty French version of article.sty

frenchponct.sty
 french punctuation

ltugboat.sty for articles to *TUGboat*

texnames.sty define a couple more T_EX names

usenix.sty for Usenix conference proceedings

- ◊ Michael DeCorte
2300 Naudain Street H
Philadelphia, PA 19146
Bitnet: mrd@clutx

Coming in January from the American Mathematical Society

Upgrades and new features for a number of the American Mathematical Society's T_EX-related products will be available from AMS in January. The most important of these are:

- a style file for L_AT_EX incorporating the mathematical typesetting features of *AMS-TEX*;
- Version 2.0 of *AMS-TEX*;
- an update and expansion of the current AMSFonts collection.

AMS-TEX, created by Michael Spivak with the sponsorship of the AMS, consists of T_EX macros that facilitate the typesetting of complex mathematical formulas and displays. Version 2.0 was prepared by Spivak and AMS staff members and incorporates a number of improvements and changes suggested by users of *AMS-TEX*. These include improved error and help messages; elimination of all known bugs; refinements to macros and other changes to conserve memory space; simplified access to fonts in addition to those defined in plain T_EX. Some optional formatting features, such as the ability to use running heads, have been incorporated in the preprint style. Finally, the *AMS-TEX* macros, which are defined in the file *amstex.tex*, have been fully documented by Spivak in a separate file, *amstex.doc*, which is organized in parallel with *amstex.tex*.

Users of the old Version 1.1 of *AMS-TEX* can obtain the new version from the AMS free of charge. The upgrade package will include an explanation of the differences between the old and new versions and a summary of changes to *The Joy of T_EX*. Those who have been using the AMSFonts package and wish to upgrade to *AMS-TEX* 2.0 must also upgrade the fonts (see below).

The AMS has also been working on making *AMS-TEX* and L_AT_EX more compatible. As part of this effort, the AMS has sponsored the development of several document styles for L_AT_EX. The first is an *AMS-TEX* style option that allows L_AT_EX users to utilize many of the *AMS-TEX* mathematical macros within existing L_AT_EX document styles. Two document styles, *amsbook* and *amsart*, utilize the *AMS-TEX* style option and provide formats suitable for a typical book or for an article that might appear in either a journal or a collection. If an author uses one of these document styles to prepare a manuscript, then that L_AT_EX manuscript can be submitted to AMS publications, whereas

only *AMS-TeX* manuscripts were acceptable before. (The *AMS-TeX* document style option for *LATEX* is different from *LATEX*, a macro package Spivak is developing to incorporate some *LATEX* features into *AMS-TeX*.) *TeX* vendors have agreed to include the files which implement the *AMS-TeX* document styles and style option in all *LATEX* packages they sell in the future, and to make this upgrade available to all current *LATEX* users. In addition, the files will be available free of charge from the AMS and will reside in the public domain archives.

The third item is an update and expansion of the current *AMSFONTs* collection. The new Cyrillic fonts from the University of Washington (the article by Dimitri Vulis, p. 332, of this issue of *TUGboat* uses these fonts, although with a ligature scheme different than that distributed by the AMS) are included in the new package. Also included are an extended set of Euler fonts (see Knuth's "Typeetting Concrete Mathematics," *TUGboat* 10#1, pp. 31-35) and many new sizes (smaller than 10 point) of previously available Computer Modern fonts. There will be two implementations of the new collection—one for PCs and mainframes, and one for use with *Textures* on the Macintosh. All the standard *TeX* magnifications will be available for both implementations. Anyone who has the original *AMSFONTs* package can receive from the AMS, free of charge, the upgraded fonts and instructions for their use. The new fonts can be used either with or without *AMS-TeX*. However, they were created in conjunction with *AMS-TeX* Version 2.0, which in turn contains improvements that cannot be used without the new fonts; the new fonts are also not compatible with earlier versions of *AMS-TeX*.

Anyone wishing more information on *AMS-TeX* Version 2.0 or on the new *AMSFONTs* collection should contact

Customer Services Department
American Mathematical Society
P. O. Box 6248
Providence, RI 02940
800-321-4AMS or 401-455-4000
cust-serv@math.AMS.com

If requesting an upgrade of either item, please specify either the PC or the Macintosh implementation. Unless otherwise specified, the PC implementation of the upgrades will be supplied on 5.25" high-density diskettes; the Macintosh implementation will be supplied on double-sided, double-density 3.5" diskettes. If making a request by electronic mail, please provide a full postal address, as the

AMS-TeX files and the *AMSFONTs* collection will not be shipped electronically.

There are two ways users may obtain the *AMS-TeX* style files for *LATEX*. First, users of electronic mail can receive these files electronically, by sending a request to the Society on Internet, at the address

ams-latex@math.AMS.com

Second, users may request the style files on IBM or Macintosh diskettes by contacting

Rosanne Granatiero
Publications Division

at the Society's address given above.

TEX EURO

Joachim Lammarsch

To communicate with other *TeX* users there are a lot of possibilities:

- *TUGboat*
- *TeXhax*
- *UKTeX*
- *TeXmag*

These are all digests and the *UKTeX* is the only one which appears weekly (*TUGboat* three or four times a year, *TeXhax* in uncertain periods and *TeXmag* sporadically).

Three further lists are used in Europe (EARN):

- *GUT*
- *TEX_D-L*
- *TEX_D-PC*

These are lists, which distribute each note (mail) separately. The languages which are used, are French and German. That's all I know.

In my opinion no real European list exists distributing each note (mail) separately and at once. Therefore I am trying to start a list for use in all of Europe. The languages can vary (not only English, French or German). To my mind people should use the language which is necessary to solve a given problem. For example, a Spanish *TeX* user having a question concerning only Spanish affairs would only be of interest for Spanish people. Then he can (should) use his own language. On the other hand, he can use the English language hoping for more help (I suppose that English is the most

commonly understood language by **TEX** users in Europe).

So we'll be able to use English as the only language, but considering that Europe is a multilingual continent, we should also use the other ones if it will be useful.

I have named the list

TEX-EURO

and it is installed at the

LISTSERV at DHDURZ1

For subscribing you should send the command

SUB TEX-EURO (*your name*)

to your nearest listserver or to **LISTSERV@DHDURZ1**.

The list is intended for use in Europe, but all **TEX** users outside are invited to join in, if they are interested.

I hope that **TEX-EURO** is a possibility for all the **TEX** people in Europe and all the European national groups to get in touch for better cooperation.

◊ Joachim Lammarsch
Research center
Universität Heidelberg
Im Neuenheimer Feld 293
6900 Heidelberg
Federal republic of Germany
Bitnet: **RZ92@DHDURZ1**

GUTenberg will distribute MLTEX

Bernard Gaulle

Nous savons fort bien à GUTenberg que les francophones n'utilisent pratiquement plus que **MLTEX** et que ceux qui n'y sont pas encore passés souhaitent le faire. Malheureusement **MLTEX** fait l'objet d'une licence. Par ailleurs sa distribution en dehors du monde PC (version commercialisée par Personal **TEX**, inc.) n'est pas très conviviale. Il en résulte que différents freins réduisent sa diffusion. Conscient de l'intérêt de **MLTEX** dans le monde francophone et de ces divers problèmes, GUTenberg a souhaité depuis le début, qu'il soit mis dans le domaine public. Cela n'étant malheureusement plus possible *GUTenberg a malgré tout obtenu de pouvoir distribuer gracieusement **MLTEX** à ses adhérents*. Celà concerne les multiples versions UNIX ainsi que les versions VMS, MVS, NOS/VE et CMS. Ces derniers systèmes ne concernent que les centres de

calcul donc relativement peu de sites. Par contre les versions UNIX et VMS concernent une très grande population. Les options de redistribution que GUTenberg prendra ne seront donc probablement pas les mêmes dans les deux cas. Mais il est probable que nous ferons tout le nécessaire pour que nos adhérents puissent obtenir une version directement installable sous UNIX et VMS, avec en plus tout l'attirail d'outils et de pilotes souhaitable (du genre de la distribution des disquettes AT dites 'GUT89'). Nous vous tiendrons au courant de l'évolution de cette distribution dans les *Cahiers GUTenberg* et sur la messagerie GUTenberg (liste GUT sur **LISTSERV@FRULM11**).

Nous remercions Michael FERGUSON pour son **TEX** multilingue et pour l'intérêt qu'il porte à GUTenberg.

◊ Bernard Gaulle
91403 Orsay Cedex, France
Bitnet: **UCIR001@FRORS31**

A **TEX** Mailbox in Germany

Garry Glendown

In Germany, we don't have a lot of good BBS-Systems running. And as far as I know, none that has **any** support for **TEX**. So after getting a modem, I thought about starting one. Well, it is running now!

The Insider consists of two main parts: the Amiga part (which probably won't be of any interest for many of you out there), and the **TEX** part. Both include message and file libraries for general use.

Using the Insider. To get into this BBS, call the number: 06621/77923 in West Germany at 300 to 2400 Baud, F8N1. Login with your name and place where you live. You will then receive a short explanation of the features of **The Insider** (currently only in German. Anybody out there who wants to call from the States?). After that, add yourself to the user list by selecting 'J'. If I am there, I will give you access to the **TEX** boards right away, otherwise you'll be added during the next 24 hours.

Features of the Insider. In the moment, I can offer the following features:

- private and public messages;

- a file library;
- access to the TeXhax-list (all 1989 issues online; '87 and '88 upon request);
- any files of the Unix-Tex distribution tape may be ordered (sorry, I can't afford filling my harddisk with 30 mbs of Tex). This will take something from 1 to 3 workdays, as I get them from our Apollo WS30;
- forwarding of questions to TeXhax, TUG or any other list/person accessible through Bitnet.

At the moment the box will only be online from 21:00 through 00:30 as I don't have enough users to "stay open" longer. This might change if there is enough demand for it. (If I'm there and have time to spare, I'll start it up when you ask for it.)

◊ Garry Glendown
 Box R, APO NY 09141
 or
 Güldene Kammer 35
 6430 Bad Hersfeld
 FRG
 Phone: 0662177923
 Bitnet: Garry@DGIHRZ01

Towards a Complete and Comfortable TeX System for the Atari ST

Stefan Lindner and Lutz Birkhahn

Once upon a time. The Atari shareware TeX story began in August 1985 when we came upon the magazine *Forschung; Mitteilungen der DFG* dating from April 1983. In an article of only 1 page, Prof. Dr. H. Werner and Dr. Paul Janßen reported on a new typesetting program called TeX. In order to make its abilities clear, they showed a small formula in source code and its printout on a 9 pin writer. Although the print's quality was not very good we were enthusiastic. As we could not get more information at that time we forgot TeX for a while. In Autumn 1986 *The Beauty of Fractals* was published and after we had translated a few chapters (an excellent method to study a document in depth) we felt the need of an appropriate text system with the help of which we could typeset the formulas in the right way. And, what was more, the program should be appropriate for future theses

and other works. Now we remembered the article on TeX.

The planning. Another search for literature in spring 1987 brought us to the series of books *Computers & Typesetting*. At that time we knew neither that TeX, METAFONT and everything belonging to them could also be bought on tape, nor that there were already implementations on PCs. After so many pages of source code had lost their horror (thanks to Lutz Birkhahn's art of persuading), we began to look for a programming language. The Pascal compilers which were available on the Atari ST were not very encouraging as they had great difficulties with 32 bit integers and arrays bigger than 32 kb. Anyhow, we had to type in the programs so that we could translate them into another language. Fortunately, all programs were written in a subset of Pascal which saved us from difficulties with VAR parameters and pointers while translating them. At that time, a number of C compilers were available which seemed to be much better than the existing Pascal compilers. 32 bit arithmetic and big arrays were never a problem for C and, what is more, C was also appropriate because of its preprocessor concept which is similar to the WEB macros. At first it might seem a bit foolish to type more than 1000 pages from books, but even if we had had the sources on magnetic tape or disc we would have first needed to adapt the program TANGLE to the ST and then to write a Pascal to C translator.

The realisation. We had found the programming language, and, precisely on April 1, 1987, we started to transform it. Stefan Lindner dealt with TeX, Lutz Birkhahn with METAFONT, and this partition also shows the preferences of each. We needed one month for typing and transforming, then both programs were completely translated into C. This was only a fraction of the time which we would have needed for adapting TANGLE and writing a Pascal to C translator.

Our first attempt to compile TeX ended quite soon with a compiler bug causing the computer to hang up. The following months passed with looking for typing mistakes and compiler errors. The last typo was found in December 1987, but even then the compiler did not want to translate the program in the right way. In January 1988 Stefan Lindner changed the compiler and the new one soon gave birth to the first running version, but it was very big and very slow. But even in this version were some errors. It was not until June 18, 1988 that the TRIP test was passed successfully. In July and August, TeX was adapted to Borland's new Turbo

C Compiler which rewarded us with a very small and fast program. On August 31, 1988 the TRIP test with version 2.93 of *TeX* and the new compiler was passed without any error. In this form, the implementation was fully working.

The development of METAFONT was more or less the same. In December 1987 we were able to admire the first simple graphic produced by METAFONT, but it had to be drawn with the help of a file dump utility and a pencil as we had no printer drivers. But this little achievement encouraged us and so we could go on looking for mistakes. In April 1988 the TRAP test was passed successfully, whereupon we began immediately with an implementation of the output in GEM windows. In August of the same year we also finished the transformation to Turbo C.

The DVI drivers. Our next task was to write DVI drivers for the most current output devices. For us, it was most important to write a driver which could be easily adapted to any other output device providing only bit image graphics. Stefan Lindner did the main part of this work. We created a modular driver which at first prepares one page in memory and then sends it as a bit image to the output device. There is no other possibility with pin writers, but with laser printers allowing font downloading, it is not very efficient. We accepted this deficiency deliberately and our decision later proved to have been right as we were able to produce a prototype for a new printer within a few hours. During the first months of 1989 we went over the driver again very carefully, and increased its reliability in operation considerably. Within only two hours' time, we also made a workable previewer from the driver which at the moment can only be handled by keyboard. The drivers work with the packed (PK) format. Of course we also translated the program GFtoPK into C and delivered it together with METAFONT.

An integrated system. *TeX* and METAFONT are at first command line oriented programs which seem to be a little antiquated on the Atari ST with its modern GEM user interface (a window oriented operating system), and, in this form, they discourage many potential users. METAFONT particularly lives in the shadow as many users are already challenged by *TeX* and do not want to occupy their minds with another complex program. Also the additional step GFtoPK and the use of enlargements and mode_defs contribute to that.

Klaus Heidrich and Reinhard Maluschka also saw these problems and so they developed a GEM

oriented shell (see *TUGboat* 9#3, p.238, "Software-Ergonomics on the ST"). In autumn 1988 we got into contact with them for the first time, and, soon after, we met again for a working weekend. Together with these two authors we decided to continue our work on these programs to harmonize them.

The shell makes the use of *TeX* and METAFONT mere child's play. We will mention only a few of its many features here. After having chosen by mouse a text which is to be worked on, one enters a cycle. At first the editor is started and the text is loaded. You may specify the proper format file in a comment line within the first few lines of the document, e.g. "% macropackage=plain", which will be recognized by the shell. After editing and storing the text, *TeX* is started automatically. If *TeX* finds an error one can return to the editor by typing 'e' whereby the file actually in use is loaded and the editor automatically jumps into that line where the error occurred. But this can only work if the editor which is used allows filename and line number as command line parameters. After correcting the error and leaving the editor, *TeX* is started again. If the *TeX* run this time is finished without mistakes, the previewer is invoked automatically afterwards and the first page is shown on the screen. If there are some fonts missing for the previewer it will show that by special notice. All missing fonts are reported into a batch file for METAFONT and can be produced by only one keyhit (alternatively also with the mouse). This contains not only the choice of the proper mode_def and the size, but also the automatic call of GFtoPK.

The number of necessary passes for LATEX is determined automatically. If MakeIndex or BibTEX are available, the programs are also started automatically at the end of a LATEX run, but only if there have been any changes in the index or aux file. The average users do not get in contact with the underlying programs and are able to concentrate fully on their texts.

In order to install the whole package on hard disk or floppy disk, all input and output files may be scattered over different drives, partitions and subdirectories. A whole list of paths is allowed which is searched following the given order. The installation on hard disk is done automatically by a specially prepared program. Installing the programs on floppy drives is possible but does not allow comfortable working.

Previous to the real input of *TeX*, there can be a transformation of single characters into whole strings. In this way it is possible to prepare texts in a more or less readable form with a text editor

which knows special characters with an ASCII code higher than 127 and is able to display them. Let's suppose the editor is able to show the character which has an ASCII code of 200. Then we will find in the file for translation the line

```
'\200' = "$\alpha$"
```

and all displayed as will be replaced by \$\alpha\$ and will be processed in this form by TeX. In this way, one will roughly know what the result with TeX will be looking like while preparing the text.

The size of the *mem* array is at maximum 65535 words. TeX first of all gives away the memory for all the other static arrays. The remaining memory is then available for the *mem* array whose size will only be determined during runtime. We have also done an extended version of TeX providing enough memory for applications like PICTeX. The size of the *mem* array is only limited by the main memory of the Atari, but due to the enlarged array elements the minimum memory requirements are 2 MByte. METAFONT's *mem* array has been fixed at 65535 words. As it is not easy to create VIRTEX with 'preloaded formats' on the Atari ST, the structure of the format file has been modified slightly, and so the loading time of the format file can be neglected. The *interrupt* variable is supported; TeX and METAFONT can be stopped while they are working and, if necessary, be aborted. The proper key for that is, for reasons of ergonomics, always the same in all other programs of the package.

TeX and graphics. A deficiency of TeX is the missing ability of including graphics. Up to now there have been no compulsory standards so that this variant, which will be described here, depends again on the computer. The underlying operating system, GEM, knows 2 standardised graphic formats, one of them object oriented and the other one a bitmap format. The actual inclusion of graphics into the text is to be done by the drivers. TeX only reserves an empty rectangle and writes a \special command to the DVI file. If using an object oriented format, the driver would need a complete Interpreter. First of all there is the problem of how to use the text commands as GEM has its own fonts. We chose the bitmap format which only needed the addition of a single function in the driver's source code. Moreover, this solution was to be only temporary, until a compulsory standard was formed. Another reason is the fact that the object oriented format is so little known and is only supported by a few graphic programs. Unfortunately the long awaited standard has not

yet been completed, but there have been activities lately which let us hope again.

Distribution as shareware. In the beginning our own interest in TeX and METAFONT had been the reason for the implementation on the ST, but later, when the first running versions existed, we wanted to make the programs available to as many users as possible. It was clear from the beginning that the implementations should not be sold to a commercial software vendor but should be quite cheap so that above all other students would be able to buy them. But TeX and METAFONT are two programs which are too complex to be distributed as public domain. They are not to be used only by programmers or mathematicians, but also by all the other owners of a computer. Leslie Lamport created within LATEX an interface between the user and TeX which allows any user who is interested in the subject to write his texts with TeX.

Our purpose was also to spread TeX quite far with the help of our implementation. For this reason we sell the whole package in the form of Shareware. Everybody is allowed to copy and try it. Anyone who thinks it might be useful for his purposes and wants to work with it more often is supposed to pay the sums which are mentioned below and receives in return the most recent version of TeX or METAFONT. The concept of Shareware has many advantages. As it is so cheap, this implementation has become known throughout the German-speaking countries and in recent times also in the rest of Europe.

In return for the money you will not only get the most recent version and the option for a free update, but also telephone support. As students were the main group we wanted to reach, we came into contact with many of them and were able to win some new colleagues who contributed other drivers, MakeIndex, BIBTeX and numerous macro packages.

All these inventions need to be examined, collected and integrated into the package. So our addresses have become the collecting point for all these voluntary works. The authors have joined the Shareware idea and hand their works over to us without charging a fee. It is only possible to actualize and coordinate the programs at any time if they are sold as Shareware (in contrast to full public domain) but this concept has proved to be right by the increasing number of voluntary colleagues. In this place I want to mention Klaus Heidrich, Robert Kiessling, Reinhard Maluschka and Michael Mies who stand for many others. All this is only

possible because Shareware allows one to ask for the complete source code which would be nearly impossible if the programs were sold by a firm. The price is to cover our expenses and not to make any profit. If there is a profit we use it for further development of the implementation.

Our future planning. An object oriented editor for L^AT_EX pictures will soon be finished. A member of the laboratory for information technology in Hannover where the program was developed has kindly allowed us to distribute it; this is another success of the Shareware concept.

The whole purpose of the work is to produce a fully integrated package consisting of T_EX, editor, previewer and drivers. An extended version will support the integration of WEB, CWEB and compilers in the same package. The Atari ST lacks the ability of multitasking, so that still some problems remain to be solved. The editor should run in one screen window from which T_EX can be started. T_EX runs in its own window in the background. Every page which is formatted is shown in a third window by the previewer. In the meantime one can continue writing the text and correct mistakes which are to be seen in the previewer window. If there is enough memory it is even possible to run METAFONT in another window. As already mentioned, much work remains to be done because these things ask too much of the ST's standard functions.

Availability. Both writers of the programs currently study computer science at the University of Erlangen (Germany).

T_EX can be bought for \$35 (60 DM inside Germany) from Stefan Lindner at the address below. The package includes T_EX, INIT_EX, L^AT_EX, plain, the previewer, one printer driver and the GEM shell. Please do not forget to name the output device you use. The extended T_EX version is sold only in combination with the normal T_EX diskettes and needs additional \$15 (30 DM).

METAFONT (including INIMF) plus shell, CMR font sources and GFtoPK can be bought for \$30 (50 DM inside Germany) from Lutz Birkhahn at the address listed below.

For those who want to order both programs it is enough to send their order to one of the given addresses as both authors are in regular contact and will give the order to the one who is responsible for them.

A complete package consisting of both versions of T_EX, METAFONT, MakeIndex, BIBT_EX, a collection of macros, 3 printer drivers of your choice and the source code for the drivers (including WEB

and CWEB) can be bought for \$130 (220 DM inside Germany).

Currently all documentation is in German, but we are planning to translate it into English in the near future.

Editor's note: The authors also supplied the following METAFONT mode_defs for their output devices, along with accompanying commentary.

Some of the mode_defs have a negative blacker value. This may cause problems with some fonts (e.g. cminch), but the result looks much better than with a blacker of 0. For the fonts which cannot be created with negative values, the blacker value has to be set to zero.

The Atari ST laser printer SLM804 is a write white engine. You have to use the modified CMBASE according to TUGboat Vol. 8 (1987), No. 1. We have introduced a new boolean write_white_engine in the mode_setup macro, so there is only one CMBASE file which behaves differently according to the setting of this boolean.

```
% stlaser mode: to generate fonts for
% the Atari ST laser printer SLM804
mode_def stlaser =
  pixels_per_inch:=300;
  blacker:=-.25;
  fillin:=.5;
  o_correction:=0;
  write_white_engine:=true;
enddef;

% psix_low mode: to generate fonts for
% the NEC P6 printer (180 dpi)
mode_def psix_low =
  pixels_per_inch:=180;
  blacker:=0.1;
  fillin:=.2;
  o_correction:=.6;
enddef;

% psix_high mode: to generate fonts for
% the NEC P6 printer (360 dpi)
mode_def psix_high =
  pixels_per_inch:=360;
  blacker:=-.75;
  fillin:=.2;
  o_correction:=.75;
enddef;
```

```
% starnl mode: to generate fonts for
% the star NL-10 printer
mode_def starnl =
  pixels_per_inch:=240;
  blacker:=-.6;
  fillin:=0.2;
  o_correction:=.4;
  aspect_ratio:=9/10;
enddef;
```

- ◊ Stefan Lindner
Iltisstraße 3
D-8510 Fürth
Germany
09 11 / 759 18 86
- ◊ Lutz Birkhahn
Fürther Straße 6
D-8501 Cadolzburg 2
Germany
09 103 / 28 86

Editor's note: The authors have asked that any correspondence for them be sent through Klaus Heidrich at

Bitnet: U0275@DGOGWDG5

The last VMS distribution available through Maria Code, assembled by David Fuchs and Eric Berg, is in need of attention: it has slipped well out of date, and there is really no one left at Stanford to update it. There is a need for a public domain distribution with good coverage of the *T_EX* software available for VMS, separate from the commercially supported distributions. Professor Knuth also indicated a personal interest in seeing the Maria Code distribution continue, both because Maria Code has provided a consistent service, and because royalty payments from that distribution continue to help defray costs from the *T_EX* project. It was suggested that the DECUS *T_EX* distribution for VMS, which is actively updated by Ted Nieland, might also be made available through Maria Code. Ted, upon prompting from several sources, has generously agreed to take steps to make this happen. He also mentions that the latest DECUS distribution is August, 1989; notable additions include *T_EX* 2.991, and additional WEB and PostScript software.

Peter Abbott was present to describe the status of the *T_EX* archive at Aston University, which, besides having an imposing selection of software, has knowledgeable staff who appear to have the time to keep it well organized and up to date. I wonder if this might provide another alternative for distribution through Maria Code.

All this activity has spurred me to upgrade my network connections. I am now available on the UUCP network at *uunet!nls!davek* and an Internet address will be forthcoming shortly. There is also a FAX number: 503-228-5662. I expect these to allow me to route information more quickly and to be more effective as site coordinator. There's still the telephone, too—tradition, remember?

- ◊ David Kellerman
Northlake Software
812 SW Washington
Portland, OR 97205
503-228-3383
UUCP: *uunet!nls!davek*

VMS Site Coordinator's Report

David Kellerman

The TUG meeting this summer was a memorable one. I must be a bit of a traditionalist, because it was nice to have it back at Stanford, which still feels like "home" to *T_EX*. The tenth anniversary brought back faces that haven't been around for a while, and I realized that the eight years I've been involved with this group has been long enough for people to age perceptibly—they looked wiser, too.

All this nostalgia was balanced by real excitement in the increasing "internationalization" of *T_EX*. The strong activities of the European user groups and noticeable meeting attendance from outside the U.S. were topped off by the promise of *T_EX* 3.0. It looks like these changes will put other Roman languages on an equal footing with English. And they're certainly going to keep people busy rewriting device drivers, constructing new fonts, and reworking hyphenation tables.

Several important things came out of the VMS "birds of a feather" session.

Typesetting on Personal Computers

TEX-PostScript output on non-PostScript devices

Alan Hoenig and Mitch Pfeffer

PostScript technology provides an alternative way to place typographic elements on a page. Newcomers to PostScript might feel that it and TEX are direct competitors, but they have little territory in common. For several reasons, you might want to translate a TEX dvi file into PostScript format, and several DVI-to-PostScript device drivers (for the PC) exist for this purpose. (PostScript drivers also exist for the Amiga and, of course, for the Macintosh.)

There are advantages to a PostScript way of life. Hundreds of PostScript fonts exist, and several hundred service bureaus exist to render your PostScript files on their phototypesetters to generate true typeset-quality output. (A directory listing the more than 650 such bureaus in the United States is available from Ms. Jean Miller, Electronic Publishing & Printing, Dept. SBD, 29 N. Wacker Drive, Chicago, IL 60606, for \$7.50. Request the *2nd Annual Service Bureau Directory*.) Yet this convenience has a price; PostScript-compatible equipment costs substantially more than non-PostScript equipment. (I also believe that PostScript service bureaus charge more than do TEX output services.) A PostScript Apple LaserWriter printer costs several times the non-PostScript Hewlett-Packard LaserJet II printer.

Several software bridges exist for generating PostScript output on a LaserJet (and selected dot matrix printers), and I wondered how useful they would be for doing TEX work. As it turns out, these programs are effective, but are very slow. Depending on your tolerance for pokey output, these programs may elicit rapture from those who crave font variety (but only modified rapture, as the PostScript fonts don't do math without substantial additional work).

Bitmapped versus Outline

It's important to understand the difference between *outline* fonts and *bitmapped* fonts. Normal PostScript fonts are outline fonts — PostScript specifications determine the outline of each character

which PostScript can scale to the point size you request.* A PostScript laser printer interprets the outline on the fly as it processes a document and scales the outlines up or down to get different point sizes.

Each character in a bitmapped font contains the specific pattern of pixels required to generate that character at that size at a particular resolution. Computer Modern fonts are bitmapped, and the ramified directory structure on a PC to contain them all is necessary for keeping track of files for generating fonts at different design sizes and different magnifications. When used in TEX documents, bitmapped fonts can be previewed, whereas outline fonts cannot!

PostScript Interpreters

You may choose between two programs that take the PostScript outline descriptions and transform them to bitmapped descriptions with which the LaserJet (or various other printers) feel comfortable. (Remember, though, you won't be able to preview.) These are:

1. *GoScript*, by LaserGo Inc., 9235 Trade Place, Suite A, San Diego, CA 92126; (800) 451-0088. List price is \$195, and it requires 640K RAM, 1MB hard disk space, DOS3.0 or greater; not copy protected. (One or more megabytes of EMS memory is recommended.)
2. *Freedom of Press*, by Custom Applications Inc., 5 Middlesex Technology Center, Billerica, MA 01821; (508) 667-8585. The list price is \$495, and this program requires 535K RAM, 4MB hard disk space, DOS2.1 or greater, and 512K or more of EMS memory. (The company recommends a numeric coprocessor if you plan to do extensive graphic work.)

Note Freedom of Press's special needs — it requires EMS (expanded) memory. (Be aware, though, that several utilities exist which will make your extended memory act as if it were expanded memory. This strategy works fine.)

Both programs work essentially the same way. They are easy to install. At the DOS prompt, enter the program name followed by any nonstandard PostScript fonts, followed by the name of your file. The name of your file must be complete; don't neglect the file extension if it exists! All interpreters, so far as I could tell, adequately rendered PostScript

* Editor's note: This is roughly equivalent to TEX magnification, whereas the Computer Modern fonts have different shapes for different sizes.

This is Gill Sans. This is Gill Sans Bold.

This is Computer Modern Roman.

This is Gill Sans. This is Gill Sans Bold.

This is Computer Modern Roman.

Fig. 1. Gill Sans at 30pt, together with *cmr10*. GoScript (top), Freedom of Press (bottom).

graphics on the LaserJet, the printer I used in these experiments. (A nonstandard font is any font not normally resident in a PostScript printer.)

Both programs are slow. I prepared a test document consisting of the abstract to Don Knuth's article "Mathematical Typography" and set entirely in 12 point Bodoni (a PostScript font). *GoScript* required about four minutes, while *Freedom of Press* took 38 minutes!

Print quality is clearly superior with *Freedom*. In large letters, curves were properly sinuous and bitmapped characters in the document appeared as they should. Large curves came out vaguely polygonal with *GoScript*, and bitmapped characters appeared much heavier and muddier than they should have been. (See Figure 1.) But *GoScript* is cheaper, faster, and does not require special EMS memory. If you use your laser printer strictly as a proofing device, and plan to print your files on a phototypesetter, you may find *GoScript* output acceptable. The customer service representatives of both companies get high marks for courtesy and helpfulness.

Of course, to use these programs with *TeX*, you need to translate your *TeX* output to PostScript form. DVI-to-PostScript converters are available from several sources. Nelson Beebe has prepared a public domain version (available from several sources; Jon Radel is one such. For information on Jon's offerings, send a SASE to him at POB 2276, Reston, VA 22090, USA). Commercial versions can be furnished by ArborText and by Personal *TeX*. For this article, I used the Personal *TeX* *ptips* converter.

Conversion to PostScript form adds another step in the *TeX* life cycle. Rather than run the *dvi* file through the device driver, you must first convert the *dvi* file through the DVI-PostScript converter to create a *ps* file. It is this *ps* file that gets printed,

either directly on a PostScript printer or indirectly, if you use one of these interpreters to print the file onto a LaserJet.

But note that before you may use your DVI-to-PostScript converter, you will probably have to make an entry into some auxiliary file. For *ptips*, this file is *font.sub*. PostScript fonts typically own lengthy names, such as *MGillSans-Bold*. Because of the DOS limitation on the length of file names, you cannot name this file with the same name. The outline file for this file might be called *gilb.psf*, and the auxiliary file contains the data which matches the file *gilb* to the font *MGillSans-Bold*. (Check your documentation carefully.)

Calling PostScript Fonts in Your *TeX* Document

You set your font calls for PostScript fonts almost the way you do for Computer Modern fonts. The *tfm* file must be with your other *tfm* files, and the outline font file must be somewhere where the interpreter knows to find it. It's a good idea to keep all outline font files together in one hard-disk directory. In your source file, you might have a *TeX* statement like *\font\gilbf=gilb* at 30 pt, and thereafter you switch to this font with the command *\gilbf*. (Don't forget also the change to the font substitution file so the DVI-to-PostScript converter can make the equivalence between the font file and the font name.)

Both interpreters boast that they can duplicate some subset or superset of the standard complement of resident fonts in a true PostScript laser printer. These lookalike fonts are inaccessible to *TeX* users, as they do not come with font metric information. If you want a PostScript Times Roman (one of the standard resident fonts), you will have to purchase it separately. Both converter programs know where

to find your bitmapped fonts, but you will have to tell them where the outline files reside. Your GoScript command line might look like this:

```
gs \psfonts\bodoni.pfa myfile.ps
```

assuming the Bodoni face is contained in the file *bodoni.pfa* in the directory *\psfonts*. Each font needs a separate mention on the command line and there is no way to continue an invocation. Therefore, you won't be able to include too many PostScript fonts in a file you interpret with GoScript. Freedom of Press requires the same mention in its command line, but it's smart enough to understand DOS wild cards. This is acceptable:

```
fp! \psfonts\*.pfa myfile.ps
```

Bitmapped Fonts in PostScript Output

DVI-to-PostScript converters make it easy to combine bitmapped fonts, such as Computer Modern or Bitstream fonts (rendered with Fontware software), with PostScript outline fonts — the *ps* file that your DVI-to-PostScript creates explicitly contains the instructions for each bit-mapped character. Too many bit-mapped characters can really swell your *ps* file. When you use PostScript interpreters to render your file, you normally have to tell these programs where to find information about each non-standard *outline* font (that is, those fonts that are not normally resident in a PostScript printer). Interpreters render bit-mapped files properly without needing to be told where to find them.

If you use bit-mapped fonts, your PostScript file is no longer device independent. You doubtless have fonts on your computer appropriate to your output device. I use a laser printer, so my Computer Modern font files are appropriate to 300 dpi printers. But these are not right for your service bureau, whose phototypesetters are capable of much greater resolution. Before you ship your file to a service bureau, re-compile it making sure that *TeX* uses fonts appropriate to the resolution of the phototypesetter. See figure 2 to see what happens when a *ps* file in which are embedded 300 dpi fonts is printed on a 600 dpi device (Varityper VT600).

Phototypeset Output

In theory, your *ps* file is ready for rendering on a phototypesetter, but in fact there are some things to watch out for. Foremost, make sure that your service bureau has the same fonts you used, and that they are from the same digital foundry! But another problem has its roots in the fact that PostScript, like *TeX*, is a macro language. Unlike

This is a test of 12 point Dutch (Times Roman) rendered by a Varityper VT-600 typesetter with a resolution of 600dpi. This font, however, has been created at 300dpi. A B C
D E F G H I J K L M N O P Q R S T U V W
X Y Z a b c d e f g h i j k l m n o p q r s t u v
w x y z 1 2 3 4 5 6 7 8 9 0

Fig. 2. Varityper PostScript output combined with *cmt10*.

TeX, PostScript users do not share a generally agreed upon standard such as *plain.tex*, so many DVI-to-PostScript converters create and use their own macro definitions for the *ps* file. You will need to convey this file to the service bureau as well, for otherwise those macros will be meaningless to the PostScript interpreter. For example, the ArborText driver relies on a file called *psload.ps* whereas Personal *TeX*'s *ptips* seems to incorporate an equivalent body of macros into your *ps* file, so you needn't worry about porting anything additional to the service bureau.

Everyone knows that one of PostScript's strong points is graphics. The PostScript language includes a great facility for generating graphic images. Although this facility is greater than anything *TeX* can do, it is (in my opinion) much less sophisticated than METAFONT. Nevertheless, should you desire to include a graphic image into a *TeX* document, you can do so with your DVI-PostScript driver. With *ptips*, for example, you create the image and store it in a separate file. When preparing your source file, leave a hook for this file by including the command *\special{:(filename)}*. It's the job of *ptips* to incorporate the graphic into the document. Bear in mind that *\special* leaves no extra space for the graphic, so if (for example) the graphic is 8 centimeters tall, you should precede your *\special* command with a *\vskip 8 cm*, or something similar.

- ◊ Alan Hoenig
17 Bay Avenue
Huntington, NY 11743
516-385-0736
- ◊ Mitch Pfeffer
Suite 90
148 Harbor View South
Lawrence, NY 11559
516-239-4110

Controlling AmigaTeX from CygnusEd

Garry Glendown

One of the many fine features that Tom Rokicki has added to AmigaTeX is the possibility of controlling the main TeX commands via an ARexx interface. So why not use that?

All you need is a version of AmigaTeX, Cygnus-Ed Professional, AmigaREXX, and lots of memory (1.5 Mb is enough). From CLI, start CED, preview, and finally TeX. As the first two detach totally, you won't need any RUNs.

Now enter one of the two listings below. But before you do, check for the version of your AmigaTeX release. If you have anything since 2.9m, use the first listing. Before that, use the second one. Store it somewhere on your hard disk or disk drive. Now install it as a REXX command using the "SPECIAL.Dos/ARexx interface.install Dos/ARexx command" function.

When using any TeX file, you may now press the function key you've assigned to that program, and CED will run TeX. If there are any errors, it will try to jump to the position returned by AmigaTeX. And, if you forgot the '\end' or '\bye' at the end of your file, '\end' will automatically be inserted at the end for you.

Bugs. I've had some problems with the old AmigaTeX versions, as they would sometimes do weird things when being called with ARexx. But with the new one (2.9m) and the new REXX program, everything has gone fine the last couple months. So if you still have the old AmigaTeX version, contact your dealer and get the new one (especially with the AmigaTeX-version with IFF-support being out...)!

Disclaimer. I'm not bad in C, 68000 Assembler and TeX, but I'm terrible in REXX. I was happy when the old version finally worked, and was as disappointed when it didn't work with 2.9m. And again, I was happy when the new version worked with 2.9m. I didn't care too much about how the code looked, but I'll accept all hints and remarks about it with a thankful smile. Just forward 'em to the address at the end of the listings.

```
* TeXify.ced - Tells TeX about the new file
* and compiles it
* for AmigaTeX Version >= 2.9m
* (C) 1989 by G.Glendown
*
* This is the version for newly distributed
* version of AmigaTeX. The new command
```

* 'NextPrompt' helped reduce the code quite
* a bit, but I'm still not too happy with
* it. If you have any trouble or find bugs,
* send a note to:

Garry@DGIHRZ01.Bitnet

```
options failat 5
options results
address 'rexx_ced'
status 21
srcfile=result
sfxlgt=length(srcfile)-lastpos('.',srcfile)
sfx=upper(right(srcfile,sfxlgt))
if sfx='TEX' then do /* does the filename
have a '.tex' ? */
okay1 'Sorry, not a TeX-Sourcefile!'
exit
end
status 18 /* get number of changes since
last save */
chng=result
if chng=0 then do /* if chng=0 then save */
'save'
end
file=left(srcfile,lastpos('.',srcfile)-1)
address 'AmigaTeX'
'ToFront'
'TeXify' file
'NextPrompt' /* I've had problems with
one NextPrompt, */
'NextPrompt' /* so I use two... */
'Prompt'
prompt = getclip('AmigaTeX.Prompt')
'ToBack'
if prompt='**' then do
/* compiling finished */
address 'rexx_ced'
exit
end
if prompt='*' then do /* no \end */
'TeXify \end'
address 'rexx_ced'
'JumpTo' 99999 99999
'text' '0A'X
'text' '\end' '0A'X
address 'AmigaTeX' 'Abort'
address 'rexx_ced'
okay1 '\end inserted!'
exit
end
'Abort'
'ErrorLoc'
error=getclip('AmigaTeX.ErrorLoc')
```

```

'ToFront'      /* Get the CLI-Screen to
                 the front */
'ToBack'       /* and put the CLI with
                 TeX to the back again */
address 'rexx_ced'
okay1 'Guess what! I found an error!'
'expand view'
say error
parse var error name line pos
'Jumpto' line pos+1

For releases of AmigaTeX before 2.9m:

/***** TeXify.ced - Tells TeX about the new file
*           and compiles it
*           for AmigaTeX w/o new REXX-Commands
* (C) 1989 by G.Glendown
*
* I'm not too good at REXX yet, so this
* program is far off being perfect. I've had
* some problems with deadlocks which lead me if prompt='*' then do /* no \end */
* to put that many 'Prompt'-commands in.
* I've reduced the deadlocks to 'once in a
* while', but they still occur. If you find
* out where the problem is, please tell me:
* Garry@DGIHRZ01.Bitnet
*****/



options failat 5
options results
address 'rexx_ced'
status 21
srcfile=result
sfxlgt=length(srcfile)-lastpos('..',srcfile)
sfx=upper(right(srcfile,sfxlgt))
if sfx='TEX' then do /* does the filename
                     have a '.tex' ? */
    okay1 'Sorry, not a TeX-Sourcefile!'
    exit
    end
status 18 /* get number of changes since
            last save */
chng=result
if chng'=0 then do /* if chng'=0 then save */
    'save'
    end
file=left(srcfile,lastpos('..',srcfile)-1)
address 'AmigaTeX'
prompt=''
do until prompt='**'
    'Prompt'
    prompt = getclip('AmigaTeX.Prompt')
end

prompt=''
'TeXify' file
do until hash(prompt)=0
    'Prompt'
    prompt = getclip('AmigaTeX.Prompt')
end

prompt=''
do until hash(prompt)~=0
    'Prompt'
    prompt = getclip('AmigaTeX.Prompt')
end

prompt=''
'Prompt'
prompt = getclip('AmigaTeX.Prompt')
'ToBack'
if prompt='**' then do
    /* compiling finished */
    address 'rexx_ced'
    exit
    end
if prompt='*' then do /* no \end */
    'TeXify \end'
    address 'rexx_ced'
    'Jumpto' 99999 99999
    'text' '0A'X
    'text' '\end' '0A'X
    address 'AmigaTeX' 'Abort'
    address 'rexx_ced'
    okay1 '\end inserted!'
    exit
    end
'Abort'
'ErrorLoc'
error=getclip('AmigaTeX.ErrorLoc')
address 'rexx_ced'
okay1 'Guess what! I found an error!'
'expand view'
say error
parse var error name line pos
'Jumpto' line pos+1

```

◇ Garry Glendown
 Box R, APO NY 09141
 or
 Güldene Kammer 35
 6430 Bad Hersfeld
 FRG
 Tel.: 0662177923
 Bitnet: Garry@DGIHRZ01

Macros

TUGboat Authors' Guide

Ron Whitney and Barbara Beeton

With this article we hope to fill a lacuna (some might say "void") whose existence we have been attributing to the usual factors: tight schedules, alternative priorities and warty *T_EX* code. We now feel the macros in use for *TUGboat* have stabilized to the extent that documentation and suggestions for authors will remain fairly constant, and we hope this article can serve as a reasonable guide to preparation of manuscripts for *TUGboat*. Authors who have used the *TUGboat* macros before will note several changes (including more modern names for the style files). Suggestions and comments are quite welcome at the addresses listed below.

TUGboat was originally typeset with a package based only on *plain*. Later, as demand for style files follows wherever *L^AT_EX*-devotees wander, a *TUGboat* variant of the *L^AT_EX* *article* style was also created. The two macro sets yield much the same output, differing in certain ways for input. Below we make comment on various aspects of the *TUGboat* package, first for the *plain*-based macros, then for *L^AT_EX*. The macro sets share the file *tugboat.com*, and users of either style should read the section entitled "Common Abbreviations and Utilities". We conclude with some general suggestions to help make the lives of those on the receiving end of (any kind of) electronic copy a little easier.

The plain-based macros: *tugboat.sty*

The macros are contained in two files, *tugboat.sty* and *tugboat.com*.

General description of tags. We attempt whenever possible to tag the various elements of *TUGboat* articles in a "generic" way, modified in some respects by convenience. Authors and editors, of course, need tools to shape their articles to the form they desire, but we also wish to encourage a tagging style which is appropriate for electronic interchange. It seems unfair to expect much thought from authors concerning the markup of their information if we only provide a bag of widgets and do-hickies to hack and pound an article together. The tags whose use we encourage are the higher-level tags that mark the logical document structure. Below

these are formatting macros that we recognize may be essential for certain applications. Both sorts of tags are described in the following article.

Generally, to "mark up" the data *<foo>*, a tag *\xxx* will precede *<foo>* and *\endxxx* will follow (thus: *\xxx <foo>\endxxx*). We use the *{...}* form to delimit arguments of lower-level formatting macros. Optional commands follow tags and are enclosed in *[...]*, à la *L^AT_EX*. Several options may be enclosed within one set of square brackets, or each option may be enclosed in its own set of brackets. These "options" are actually just *T_EX* commands, and it is always possible to insert raw *T_EX* code as an option. Such practice violates truly generic markup, but it is *helpful* and at least confines The Raw and Dirty to a smaller area.

Perhaps a little more detail is of use to some readers here. Upon encountering a tag, the general operational scheme of the macros is as follows:

```

<read tag>
\begin{group}
<set defaults>
\the\every...
<read options>
<branch to appropriate action,
    using "argument" as necessary>
<cleanup>
\end{group}

```

The scheme shows that code inserted as an option is localized and that it may be used to override certain defaults and to guide branching. Things are not always simple, however. Sometimes parameters are set after a branch is taken (e.g. the macros might only call *\raggedright* after determining whether the mode is "*inline*" or "*display*"), and, despite localization, parameter setting might affect the current paragraph if a branch has yet to be taken. This is *not* to say the macros don't work, but rather that those authors who venture beyond the documented regions of the macros should do so with their eyes open.

For convenience, we also allow the * as a delimiter for the higher-level tags; thus we could use either

```
\title \TUG\ Authors' Guide \endtitle
```

or

```
\title * \TUG\ Authors' Guide *
```

to indicate the title of this paper. To typeset a * within text delimited by *, the plain control sequence *\ast* has been extended to give * in text and the usual * in math.

This markup scheme may suffer at the hands of *T_EX*'s parsing mechanism when tagged data is

nested. In these cases, one may group `\{...\}` embedded data so that TeX knows to proceed to the next `\end...` or `*`.

In the cases where we show extra spaces and carriage returns around arguments in this article, those (discretionary) spaces are accommodated in the macros. Thus, for example, when the argument to `\title` above is typeset, `\ignorespaces` and `\unskip` surround it and the extra spaces have no untoward effect. Spaces are also gobbled between options.

Outer form. At the outermost level, a source file will have the form (using the `*...*` delimiters):

```
\input tugboat.sty
<perhaps additional macros for article>
\title * <title> *
\author * <author> *
\address * <address> *
\netaddress * <network address> *

\article
:
<body of article>
:
\makesignature
\endarticle
```

Data preceding `\article` is saved and typeset when `\article` is encountered. Each author should have his/her own

```
\author ...
\address ...
\netaddress ...
```

block, and the macros will do their best to combine the information properly in the appropriate places. Explicit linebreaks can be achieved within any of these items via `\``. Title and authors are, of course, set at the beginning of an article; the address information is listed separately in a "signature" near the end of an article, and is present for the convenience of those who might photocopy excerpts from an issue of *TUGboat*. `\makesignature` does the typesetting work. Generally authors are listed separately in the signature. In cases where authors and addresses are to be combined, one may use `\signature{...}` and `\signaturemark` with some or all of

```
\theauthor {{author number}}
\theaddress {{author number}}
\thenetaddress {{author number}}
```

to get the desired result. For example, for an article with

```
\author * Ray Goucher *
\address * \TUG *
\netaddress *TUG@Math.AMS.com*
```

```
\author * Karen Butler *
\address * \TUG *
\netaddress *TUG@Math.AMS.com*
```

we could say

```
\signature {
  \signaturemark
  \theauthor1 and \theauthor2 \\
  \theaddress1 \\
  \thenetaddress1}
```

to obtain the signature

```
◊ Ray Goucher and Karen Butler
  TeX Users Group
  TUG@Math.AMS.com
```

Use of at least `\thenetaddress` is recommended for this just so that the network address gets formatted properly. The optional command `[\\network{...}]` will introduce the network address with a network name, so

```
\netaddress [\network{Internet}]
  * TUGboat@Math.AMS.com *
```

produces

Internet: TUGboat@Math.AMS.com

`\endarticle` marks the end of input and is defined as `\vfil\end` for most uses. We redefine it as `\endinput` to assemble streams of articles in *TUGboat*.

Section Heads. Heads of sections, subsections, etc. are introduced with `\head`, `\subhead`, etc., respectively. The underlying macros all use `\head`, so `\endhead` is the long-form ending for all these tags. For example, the first two heads of this article could have been keyed as

```
\head The \plain-based macros:
  {\tt tugboat.sty} \endhead
```

and

```
\subhead General description of
tags \endhead
```

In *TUGboat* style, the paragraph following a first-level head is not indented. This is achieved by a look-ahead mechanism which gobbles `\pars` and calls `\noindent`. Actually all of the `\...head` tags gobble pars and spaces after their occurrence. This allows one to enter a blank line in the source

file between head and text, such practice being a visual aid to your friendly *TUGboat* editors (if not to you). Be careful of that `\noindent` after a first-level head: you will be in horizontal mode after the `\head *...*`, so spaces which *appear* innocuous, may not be so.

Lists. Lists are everywhere, of course, and a simple list hierarchy can transform a one-dimensional typesetting problem into something nasty. All of which is to say, we are certainly not done with this area of tagging, but here are the available macros.

Not surprisingly, `\list` marks the beginning of a list. A list can be itemized, wherein each item is tagged with `\item`, or unitemized wherein items are delimited by `^M` (the end of your input line). The itemized style is the default and `[\unitemized]` will get the other. Tags for the items default to the `\bullet` (= •), but can be changed by feeding an argument to `\tag{...}`. The `[\tag{...}]` option used with `\list` assigns the tag for each item of the entire list, while `[\tag{...}]` used with `\item` changes only the tag for that item. The obvious dynamical tags are available with options

```
\numbered
\romannumeraled
\lettered    (lowercase)
\Lettered   (uppercase)
```

Lists can be set in several columns by setting `\cols=...`. The columns are aligned on their top baselines and the user must break the columns with `\colsep`. Thus,

```
\list[\unitemized\numbered][\cols=2]
Fourscore
and seven
years ago
our fathers
\colsep
brought forth
on this
continent
\endlist
```

yields

- | | |
|----------------|------------------|
| 1. Fourscore | 5. brought forth |
| 2. and seven | 6. on this |
| 3. years ago | 7. continent |
| 4. our fathers | |

`\everylist` is a token register which is scanned at the beginning of each list after the default parameters are set and before options are read. If you want all your lists numbered, for example, you might insert

```
\everylist{\numbered}
```

at the top of your file rather than giving an option to each list.

Implementation of sublists is under construction.

Verbatim Modes. There are several variations on this theme. In each case, text is printed in a typewriter font and (almost) all input characters produce the glyph in the font position of their character-code (i.e. you get what you type, no escaping it). In addition to the long form

```
\verbatim...\endverbatim
```

the `|` character can be used to enter and leave verbatim mode, acting as a toggle much as the `$` does with math. `|...|` produces inline verbatim text, while `||...||` displays its output. `\verbatim` itself defaults to display form, but `\verbatim[\inline]` and its contraction `\verbinline` (both terminated by `\endverbatim`) produce the inline form. `^M` yields a space inline, and a new paragraph in display. Generally, for snippets of text we use the `|...|` form, and for longer items the

```
\verbatim
```

```
:
```

```
\endverbatim
```

form (although `||...||` is a good way to display a single line of code).

In addition to formatting text between `\verbatim` and `\endverbatim`, `\verbatim` may read and write data from and to files. We find this variant useful in (*almost*) guaranteeing consonance between macros in use and their published listings.

```
\verbatim[\inputfromfile{foo.inp}]
```

```
:
```

```
\endverbatim
```

will incorporate the contents of file `foo.inp` in the listing before the text between `\verbatim` and `\endverbatim`. The shortened form

```
\verbfile{foo.inp}\endverbatim
```

accomplishes the above in the case that the text is empty. While the code around the data, `foo.inp`, above looks excessively long, do remember the implementation uses the basic `\verbatim` macro, so options can also be read after the filename. For example,

```
\verbfile{foo.inp}[\numbered]
\endverbatim
```

would number the lines of the listing.

We often rearrange code supplied to us so that it fits in the narrow measure of *TUGboat*'s two-column format, and we sometimes make corrections to macro sets (you thought you were perfect!). Since errors can (and do—we aren't perfect either) creep in with these modifications, we use the above technique to maintain consistency between the listing published in *TUGboat* and the underlying macros used for examples.

To write out information, use

```
\verbatim[\outputtofile{foo.out}]
:
\endverbatim
```

An added bonus here is that characters which get internalized as moribund "letters" or "others" in the process of listing them, can return revitalized for perhaps their real use when written out to another file and read in again. The example above involving Ray and Karen was coded as

```
... to get the desired result. For
example, for an article with
\verbatim[\outputtofile{ray.vbm}]
\author * Ray Goucher *
:
\endverbatim
we could say
\verbatim[\outputtofile{sig.vbm}]
\signature {
  \signaturemark
  \theauthor1 and \theauthor2 \\
  \theaddress1 \\
  \thenetaddress1}
\endverbatim
to obtain the signature
\begingroup
\authornumber=0
\input ray.vbm
\input sig.vbm
\makesignature
\endgroup
```

This is perhaps not the most edifying example, but you get the gist. (We localize the process of storing and retrieving these authors and addresses so as not to clobber our own.) We would encourage our authors to use these mechanisms for connecting verbatim text to external files for the sake of maintaining consistency between active code and its documentation.

`\verbatim` scans to `\endverbatim` (a 12-token sequence since the `\` is of type 'other' after `\verbatim` gets going). Only this sequence of characters

will interrupt the scan. On the other hand, `\|` and `\||` scan to the next `|` and `||`, respectively. Needless to say, one should use forms of `\verbatim` to set text which contains `|` (and `|` or `||`) to set text containing `\endverbatim` if you are writing an article like this one). Both the `|` and `\verbatim` tags scan ahead for the usual `[` to check for options. In those rare cases when the `[` is really supposed to be the first character of the verbatim text, use the option `[\lastoption]` to stop option parsing. For example, to show

```
[\lastoption]
```

we keyed

```
[\lastoption][\lastoption]
```

There are situations where one wants to typeset most things verbatim, but "escape" to format something exceptional. For example, the insertions of metacode given in the listings above require some access to the italic font. By giving the option `[\makeescape!]` to `\verbatim`, the `!` is made an escape character in that block. Thus,

```
\verbatim[\makeescape!]
```

```
:
...!it...
:
\endverbatim
```

really calls the italic font in the middle of the listing (one might also want to use `\makebgroup` and `\makeegroup` in the options to define characters to localize this call; see p. 384). Situations will dictate preferences for what character may be used as an escape (we use the `|`, `!`, and `/` in this article). There is also a means of changing the setup of every occurrence of verbatim mode. The contents of token register `\everyverbatim` is scanned after the defaults of verbatim mode have been set. In this article, for example, we have made `<` active and defined it in such a way that `<...>` typesets as metacode. Since `\verbatim` ordinarily changes `<` to type 'other' on startup, we key

```
\everyverbatim{\enablemetacode}
```

at the beginning of the file to have the proper adjustment made whenever verbatim is started.

When "escaping" within a verbatim block, one should be aware that spaces and carriage returns are *active* and hence not gobbled as usual. Using the `!` as the active character, one might key

```
\verbatim[\makeescape\!]
:
:
!vskip .5!baselineskip
:
:
\endverbatim
```

to get an extra half line of space in the middle of the listing. The space and carriage return on this line, however, cause problems. The space expands to `\ifvmode\indent\fi\space` and TeX will not like the `\indent` after `\vskip`. The `^M` expands to `\leavevmode\endgraf`, and therefore puts an extra line into the listing. The solutions, in this case, are to drop the space and to use `!ignoreendline` (which just gobbles the `^M`), but one should be aware, generally, that some thought may be required in these situations.

The option `[\numbered]` causes the lines of a verbatim listing to be numbered, while `[\ruled]` places rules around the whole thing:

-
1. `<code>`
 2. `<more code>`
 3. `<yet more code>`
 4. `...`

The option `[\continuenumbers]` picks up the numbering where it last left off.

5. `<more>`
6. `<and more>`
7. `...`

The code underlying `\verbatim` in display style implements each line as a paragraph and places math-display-size whitespace above and below the verbatim section. Page and column breaks *are* permitted within these listings. To prohibit breaks at specific points or globally, one must insert penalties or redefine `^M` to insert `\nobreak` in the vertical list at the end of each "paragraph" (i.e. line). We should also note that the bottom of such a verbatim listing is implemented so that ensuing text may or may not start a new paragraph depending on whether an intervening blank line (or `\par`) is or is not present.

Figures and Page Layout. Figures are keyed as

```
\figure
<vertical mode material>
\endfigure
```

These are generally implemented as single-column floating top-insertions, but the options `[\mid]` and `[\bot]` can change specific items to be mid- or

bottom-insertions, respectively. Here we recommend that the long-form terminator be used (*not* the `*...*` form). One can think of the information "passed" as being "long" in the sense of possibly containing paragraphs, this being a mnemonic device only. The primary reason for the recommendation is that one is (in some sense, maybe) more likely to encounter a rogue `*` in longer text than in shorter text and hence more likely to encounter a surprising result due to a macro stopping short at the wrong `*`.

Perhaps here is a natural place to mention also that these macros sometimes read their arguments and then act, and sometimes act on the fly, not actually storing an argument as a string of tokens at all. `\title`, for example, is in the former category, while `\figure` is in the latter. Reasons may vary for the choice in methods. Storing a string of tokens as an argument does not allow re-interpretation of the category codes of the underlying character string. Thus, storing the "argument" of `\figure` all at once might misinterpret some characters which should appear as verbatim text. For this reason we set figures as we go and just close off the box with `\endfigure`. On the other hand, using information in multiple situations (e.g. titles and running heads) requires storing the information as a token string, not as a typeset list.

When text delimited by `*...*` is read as an argument, the `*`s are dropped by the parsing process. When the text is handled on the fly, the first `*` is gobbled and the second is made active to perform whatever action is necessary at the close of the macro. When possible, we prefer to operate on the fly since nested tags are handled properly in that case and no memory is consumed to store arguments. Examination of `tugboat.sty` will show which case applies in a given situation, but this general knowledge may help when trying to debug those situations in which an unexpected `*` has disrupted things.

A primitive `\caption{...}` option is available to `\ulap` (i.e. lap upward) its argument into the figure space, but formatting of the caption is left to the user. For example, the code:

```
\figure[\top]
  [\caption{\centerline{Odd Fig.^1}}]
  \vbox to 5pc{}
\endfigure
```

produces the figure at the top of this column or the next.

Figures spanning columns at the top and bottom of a page are currently supported only on the first page of an article, but we expect they will soon be allowed on any page (a general rewrite of

```
\caption{...}
\twocolumnfigure
```

and, of course, | and ||.

Odd Fig. 1

the output routine is in progress). \twocolumnfigure (terminated by \endfigure) starts up such a figure and currently *must* occur before any material has been typeset on the first page (i.e. *before* \article).

Macros \onecol, \twocol, and \threecol provide one-, two-, and three-column layouts, but these cannot currently be intermixed on a page. We hope to provide automatic column-balancing and convenient switching between one- and two-column format within a year. \newpage in each format is defined to fill and eject enough columns to get to the next page. \newcol is just \par\vfill\eject.

Command List Summary. Tags are listed in the order discussed. Options are listed under tags.

```
\title
\author
\address
\netaddress
\network
\signature
\article
\makesignature
\endarticle
\head
\subhead
\subsubhead
\list
  \numbered
  \romannumerated
  \lettered
  \Lettered
  \ruled
  \tag{...}
\item
  \tag{...}
\everylist
\verbatim
  \numbered
  \ruled
  \inputfromfile{...}
  \outputtofile{...}
\verbinline
\verbfile
\figure
  \mid
  \bot
```

The L^AT_EX macros: ltugboat.sty

ltugboat.sty is the primary macro file, tugboat.com a collection of items common to both L^AT_EX and plain input. Articles will have the form:

```
\documentstyle[ltugboat]{article}
<perhaps additional macros for article>
\title {\langle title\rangle}
\author{\langle author\rangle}
\address{\langle address\rangle}
\netaddress{\langle netaddress\rangle}

\begin{document}
\maketitle
:
:
\body{of article}
:
:
\makesignature
\end{document}
```

This is the usual form for L^AT_EX documents, of course, except that now each author will have his/her own

```
\author{...}
\address{...}
\netaddress{...}
```

block. As with the plain style, the author and address macros will store their information for later display. See the discussion of \address, \netaddress and \makesignature on page 379 to understand more. Linebreaks within \title, \author, and \...address are specified with \\.

We refer the user to the L^AT_EX manual for description of section heads, verbatim mode, insertions, and movement between one- and two-column format. The style of printed output has, of course, been somewhat modified to fit *TUGboat* style. ltugboat.sty might be of some use to others wishing to modify the article option in this direction.

Common Abbreviations and Utilities

Definitions of a number of commonly used abbreviations such as \MF and \BibTeX are contained in tugboat.com. Please use these whenever possible rather than creating your own. We will add to the list as necessary.

Several other constructions that we have found useful for both plain- and L^AT_EX-style input have been incorporated in *tugboat.com*. Various **laps* (*\ulap*, *\dlap*, *\xlap*, *\ylap*, *\zlap*) and **smashes* provide means of setting type which "laps" into neighboring regions. *\dash* and *\Dash* are en- and em-dashes that break properly. *\slash* is a breakable slash. The macro

```
\makestrut [ascender dimen];
            [descender dimen]
```

allows *ad hoc* construction of struts.

\makeatletter \catcode the @ for internal control-sequences. There are also more general functions

```
\makeescape
\makebgroup
\makeegroup
\makeletter
\makeother
\makeactive
```

that change the category of a given character into the type mentioned at the end of the macro name. For example, *\makeactive!* changes the category of the ! to 13. We have given many other examples of these in this article. Readers may look at the end of *tugboat.com* after the *\endinput* statement to see further documentation on the contents of the file.

Issue Makeup. Constructing an entire issue of *TUGboat* requires use of a few features that authors may notice when articles are returned for proofing. *\xref{to}* allows for symbolic cross-referencing, but is enabled only late in the production process. The distribution version of *tugboat.com* defines *\xref{to}* so that "???" is typeset whenever it is called. Not to worry.

We also put notes into the file since there are many things to remember, and these appear as *\TB{remark}{...}*. Authors can look for such things, if they are interested.

General Coding Suggestions

Probably 90% of the code we receive is easily handled, and for this we are most appreciative. We do have suggestions of a general nature that authors should keep in mind as they create articles for transmission here or anywhere else.

Those who create code find it much easier to read and understand their own code than do others who read the "finished" product. In fact, some people seem to forget that the electronic file will be viewed (in fact, studied) in addition to the

printed copy. Documentation and uniform habits of presentation always help. Blank lines are easier to digest by eye than *\pars*. Tables and display math can often be keyed in such a way that rows and columns are clear in the source file on a display screen as well as in print. Explanations or warnings of tricky code can be *very* helpful. Authors should place font and macro definitions in one location at the beginning of an article whenever possible.

Authors should anticipate that articles will undergo some transformation, and that positioning of some elements may change simply because articles are *run together* in *TUGboat*. Decisions on line-breaks, pagebreaks, figure and table placement are generally made after the text is deemed correct. We avoid inserting "hard" line- and page-breaks whenever possible, and will not do so, in any case, until the last minute. We also use floating insertions for figure and table placement when we first receive an article. It is easier for us to work with a clean file containing some bad breaks, overfull boxes or other unsightliness, than it is to handle a document containing *ad hoc* code dedicated to a beauteous (albeit narrowly specific) result.

When authors proof their articles in formats other than that of *TUGboat* (for example), they should expect that *TUGboat's* changes in pagewidth and pagedepth may drastically alter text layout. Paragraphs are rebroken automatically when *\hsize* and *\vsize* change, but *\centerline* does not break, and we often see tables and math displays which are rigidly laid out. When possible, authors might use paragraphing techniques instead of calls to, say, *\centerline* (Beeton will be writing up her lectures on paragraphing techniques for a future issue of *TUGboat*), and they should try to code tables in such a way that widths of columns can be changed easily. Generally, authors should attempt to anticipate the work that might be necessary if requests for other reasonable formats of their texts are made. In the case of *TUGboat*, we make a strong effort to stuff macro listings and tables into the two-column format. Since these types of items are not generally susceptible to automatic line-breaking, we give thanks to stuffings made by authors ahead of time. In this context, we recommend the use of *\verbfile{...}* (see p. 380) to maintain consistency between documentation and reality.

Specifically in the domain of T_EX macros, we find that many authors throw in unnecessary % characters to end code lines. Except in cases where the ^^M means something other than end-of-line,

linebreaks can reliably be placed after control words and numerical assignments. We have seen *TeX*'s buffer size exceeded when % was placed after every line.

A wider perspective in the matter of naming macros can prevent problems that occur when definitions are overwritten as articles are run together. The names of control sequences used in plain, *LATEX*, and *AMS-TeX* are documented and authors should avoid using them for other purposes. It is also wise to avoid commonly used names such as \temp, \result, \l, and \mac in presenting code that might be cribbed by other users. The frequently used technique of temporarily \catcodeing a character to be a letter (e.g. the @) provides a good method of hiding control sequences so that they will not be clobbered later. Readers are in need of small macros to do little tricks, and they often try suggestions brought forth in *TUGboat*. A little extra effort in making these macros consistent with the macros in wide distribution and in making them robust will be much appreciated.

Electronic Documentation and Submission Procedure

In addition to *tugboat.sty*, *ltugboat.sty*, and *tugboat.com*, a copy of this article, *tugguide.tex*, will be available at most *TeX* archives, including those at Clarkson and Aston.

Please address all electronic correspondence to the *TUGboat* maildrop:

TUGboat@Math.AMS.com

Mail to either of our personal addresses is liable to go unseen if vacation or illness intervenes. We also request that you supply an abstract of any expository article. This will be used as the basis for translation of abstracts to languages other than that in which the article is published.

- ◊ Ron Whitney
TeX Users Group
P. O. Box 9506
Providence, RI 02940-9506
TUGboat@Math.AMS.com
- ◊ Barbara Beeton
American Mathematical Society
P. O. Box 6248
Providence, RI 02940-6248
TUGboat@Math.AMS.com

Round boxes for plain *TeX*

Garry Glendown

Doing presentation sheets, I stumbled over a small thing I had been missing for quite a while: boxes. Well, normal boxes are boring, so I thought about doing boxes with round corners.

To do that, I took a look at the circle fonts used for the *LATEX* pictures. They would work out fine. But, despite of all my *TeX* knowledge and the information from *The TeXbook*, it didn't work. Either the boxes would look like this:



or like this:



or some other, not very encouraging, way. After some hours (I think it was about 2 1/2 or so) I finally solved the problem as found in the listing below.

The problem is the strange (at least for normal usage) way the circle font has the width and reference point set. The width is exactly twice as big as the quarter circle, and the reference point of the right two quarters is far beyond the character. So, in order to get the right positioning of the characters, the boxes have to be much wider in the inside than they are on the outside.

Using RBox. To use the RBox-Macro, there are two simple forms: \roundBox and \RoundBox. Both get two parameters: the size of the box as a percent of \hsize, and the text. When calling \roundBox, you will get a box with a border .4pt thick; \RoundBox will result in one with a .8pt border.

If you type

```
\hbox{%
\roundBox{.4}{This is}%
\RoundBox{.4}{a Test}%
}
```

it will result in:

This is

a Test

In addition to these two 'interface'-macros, you may use the internal function called \RBox. The syntax is the following:

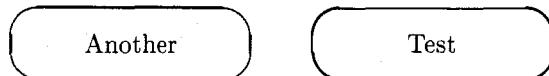
```
\RBox{{total width}}{}  
{}{{text}}
```

When using the \RBox-Command, make sure that you define the font '\cf' to be either `circle10` (.4pt thick) or `circlew10` (.8pt thick).

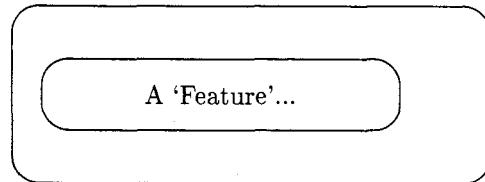
You may use \RBox for some strange effects, like:



This was done by setting the thickness of the normal lines (parameter #3) to 0pt. Or you can do things like this:



Undesirable Features. In the moment, I only know of one little problem: the outside size of the round box is always smaller than the width it is told. So, when putting one box into another, it will look like this:



But I don't see any way out of that problem. If somebody out there has a solution to that problem then please tell me ...

```
%%%%%%%%%%%%%% gdround.mac  
  
% \copyright 1989 by Garry Glendown  
% Box R, APO NY 09141 (U.S. Mail)  
% or G\"uldene Kammer 35,  
% 6430 Bad Hersfeld FRG  
% Tel.: (FRG) 06621/77923  
% eMail: Garry@DGIHRZ01.BITNET  
%  
% \roundBox{}{Text} thin frame (.4pt)  
% \RoundBox{}{Text} thick frame (.8pt)  
% Example:  
  
% internal procedure:  
% \RBox#1#2#3#4 where  
% #1 = {outside width}  
% #2 = {inside(text) width}  
% #3 = {width of frame}  
% #4 = {Text}
```

```
% make sure you define the font  
% \cf right when using \rbox!  
%  
% quarter circles:  
\def\lu{{\cf\char19}} % left upper qtr  
\def\ru{{\cf\char16}} % right upper qtr  
\def\ll{{\cf\char18}} % left lower qtr  
\def\rl{{\cf\char17}} % right lower qtr  
%  
\newdimen\wtemp  
\newdimen\ww % width of words  
\newdimen\wc % width of corner  
\newdimen\wo % width of corner + white  
\newdimen\wl % width of leaderbox  
%  
\def\RBox#1#2#3#4{  
  \setbox0=\hbox{\lu}%  
  \wc=\wd0  
  \setbox0=\hbox{  
    \vrule width#3%  
    \hbox to #1{  
      \hfil%  
      {#4}\hfil%  
    }%  
    \vrule width#3  
  }%  
  \ww=\wd0  
  \wl=\ww  
  \advance\wl by-#3  
  \advance\wl by-\wc  
  \wo=\wc  
  \divide\wc by 2  
  \vbox{  
    \offinterlineskip%  
    \hbox{  
      \hbox to\wc{\lu\hss}%  
      \hbox to\wl{  
        \leaders  
        \hrule height#3  
        \hfil  
      }%  
      \hbox to\wc{\hfil}%  
      \hbox to\wo{\hss\ru}%  
    }  
  }\box0  
  \hbox{  
    \hbox to\wc{\ll\hss}%  
    \hbox to\wl{  
      \leaders  
      \hrule height#3  
      \hfil  
    }%  
    \hbox to\wc{\hfil}%  
    \hbox to\wo{\hss\rl}%
```

```

}%
}

} % end \def\RBox

\def\roundBox#1#2{%
  \wtemp=#1\hsize
  \advance\wtemp by-.05\hsize
  \font\cf=circle10
  \RBox{#1\hsize}{\wtemp}{.4pt}{#2}%
}

\def\RoundBox#1#2{%
  \wtemp=#1\hsize
  \advance\wtemp by-.05\hsize
  \font\cf=circlew10
  \RBox{#1\hsize}{\wtemp}{.8pt}{#2}%
}

%
% End of RoundBox
%

```

- Garry Glendown
Box R, APO NY 09141
or
Güldene Kammer 35
6430 Bad Hersfeld
FRG
Phone: 0662177923
Bitnet: Garry@DGIHRZ01

Printing Annotated Chess Literature in Natural Notation

Zalman Rubinstein

There have been several recent attempts to apply the **TeX** and **METAFONT** computer languages to design a chess literature printing package. Appelt [1] suggested a design for printing chessboards, and this author [2] described a simple **METAFONT** chess font. Based on these results the present note introduces a full **TeX** macro which approximates the actual requirements of chess printing, namely:

1. Chess moves are printed in the source file in their natural appearance (as, e.g., Pe2-e4, Nc3-e4) without recourse to control sequence notation (see [1]).
2. Chess moves normally include annotations or comments as to the adjudged value of the move, such as "h7-h6?" for a questionable move, "Bd3-c2!" for a good move. The length of these annotations varies and can include quite a number of symbols, such as +, -, ++, ±, ≠ and others.
3. Chess literature can start a game from its natural starting position or from a ready setup position as necessary.
4. The printed form of a move may vary from the natural appearance of the move. For example, Pe2-e4, moving a pawn from square e2 to square e4, is usually denoted by e2-e4 or even by e4 (we shall comment on this at the end of this note).

The **TeX** macros to which I refer satisfy the above requirements by using in various ways the category management ability of the **TeX** language so that the letters K, Q, R, B, N and P denoting the chess pieces play the role of different control sequences at different times. In addition, the end of line character and the paragraph control sequence are redefined for various purposes. The two basic sequences of the macros are:

(a)
\ClearBoard
\White Kg1 Qd1 ...
\Black Kg8 Qd8 ...
>ShowBoard (optional)
<text> ...

and

(b)
\BlacksMove \movecounter8
\StartPlay
Ph7-h6?

```

Nc3-e4  Ne7xd5
:
Bd3-c2!
⟨empty line⟩
>ShowBoard (optional)
⟨text⟩ ...

```

The general notational structure of a chess move is
 $\langle piece \rangle \langle position \rangle \langle move \rangle \langle position \rangle \langle additional\ info \rangle$
where

$\langle piece \rangle$ can be any of the chess pieces K, Q, R, B, N, P;
 $\langle position \rangle$ is a pair consisting of one from each of (abcdefg) and (12345678) giving the Cartesian coordinates of a chess square;
 $\langle move \rangle$ is 'x' or '-';
 $\langle additional\ info \rangle$ can be any of the optional characters 'Eqrbn!+?' or ⟨text⟩. The letter 'E' denotes the *en passant* move and the letters 'qrnb' denote the four possible crowning pieces.

Two control sequences are also included for convenience in setting up chess problems and playing full games. These are:

```

\AuthorYear {A. B. Author} 1989
\StartPosition to set up a starting chess
game board.

```

The macro \AuthorYear puts a heading on chess problems indicating name of author and year of first publication. The macro \StartPosition just sets up the chess pieces to begin a game.

Before concluding with a full chess game example, it should be noted that, in order to make the approximation to chess literature practice complete, one would desire to abbreviate chess moves to their destination squares only such as: Pe4 or e4 instead of Pe2-e4 or e2-e4, etc. This seems to require major work, including dealing with certain ambiguous situations in which more than one piece can attain a certain destination, and is not implemented here. Of course, as noted in [1], one would also like to have a *TEX* checking apparatus to verify the validity and the rule compatibility of all moves made.

An example

```

\input chess.mac % (the chess macro file)
\fnt\bigrbf=cmbx10 scaled 1200
\centerline{\bigrbf Spassky-Bronstein 1960}
\medskip
\centerline{King's Gambit.
Game won beauty prize.}

```

```

\medskip
After the moves: 1. e4 e5 2. f4 ef 3. Kf3
d5 4. ed Bd6 5. Nc3 Ne7 6. d4 0-0 7. Bd3 Nd7
8. 0-0.
\smallskip
The position was:

\ClearBoard
\White Kg1 Qd1 Ra1 Rf1 Bc1 Bd3 Nc3 Nf3 %
Pa2 Pb2 Pc2 Pd4 Pd5 Pg2 Ph2
\Black Kg8 Qd8 Ra8 Rf8 Bc8 Bd6 Nd7 Ne7 %
Pa7 Pb7 Pd7 Pf4 Pf7 Pg7 Ph7
>ShowBoard
\bigsip
Here Black committed an error
\movecounter8 \BlacksMove
\StartPlay
Ph7-h6?
Nc3-e4  Ne7xd5
Pc2-c4  Nd5-e3
Bc1xe3  Pf4xe3
Pc4-c5  Bd6-e7
Bd3-c2!

>ShowBoard
\bigsip
A very important move enables White to
launch an attack on Black's King.
\StartPlay
Rf8-e8
Qd1-d3  Pe3-e2
Ne4-d6!? Nd7-f8?

A decisive mistake allowing a beautiful
combination. Bxd6 was necessary.
\StartPlay
Nd6xf7! Pe2xf1q+

```

```

>ShowBoard
\StartPlay
Ra1xf1 Bc8-f5
Qd3xf5 Qd8-d7
Qf5-f4 Be7-f6
Nf3-e5 Qd7-e7
Bc2-b3 Bf6xe5
Nf7xe5+ Kg8-h7
Qf4-e4+ \resigns

>ShowBoard
\bye

```

The resulting printout is on the following page:

Spassky-Bronshtein 1960

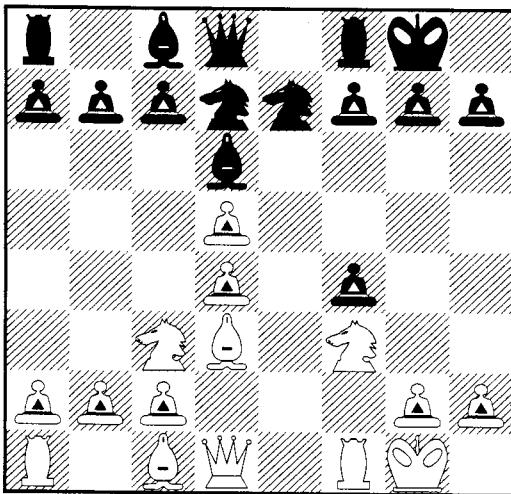
King's Gambit. Game won beauty prize.

After the moves: 1. e4 e5 2. f4 ef 3. Kf3 d5 4. ed Bd6 5. Nc3 Ne7 6. d4 0-0 7. Bd3 Nd7 8. 0-0.

The position was:

White: Kg1 Qd1 Ra1 Rf1 Bc1 Bd3 Nc3 Nf3 Pa2 Pb2 Pc2 Pd4 Pd5 Pg2 Ph2

Black: Kg8 Qd8 Ra8 Rf8 Bc8 Bd6 Nd7 Ne7 Pa7 Pb7 Pc7 Pf4 Pf7 Pg7 Ph7



Here Black committed an error

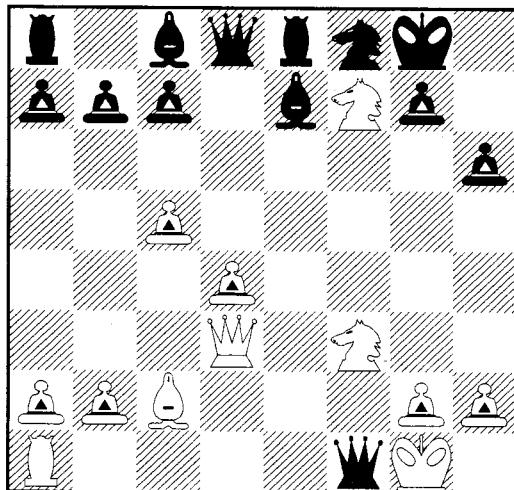
- | | |
|-------------|--------|
| 9. Nc3-e4 | h7-h6? |
| 10. c2-c4 | Ne7xd5 |
| 11. Bc1xe3 | Nd5-e3 |
| 12. c4-c5 | f4xe3 |
| 13. Bd3-c2! | Bd6-e7 |

A very important move enables White to launch an attack on Black's King.

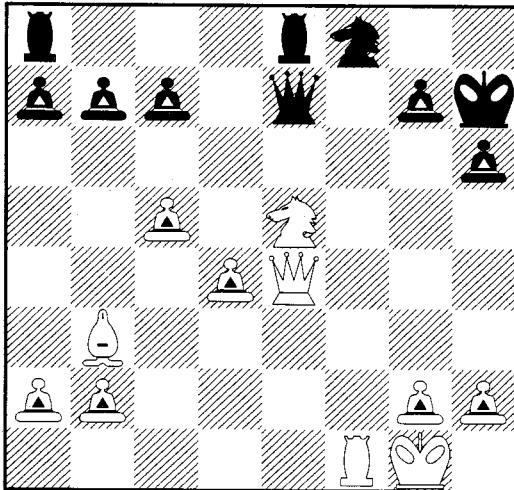
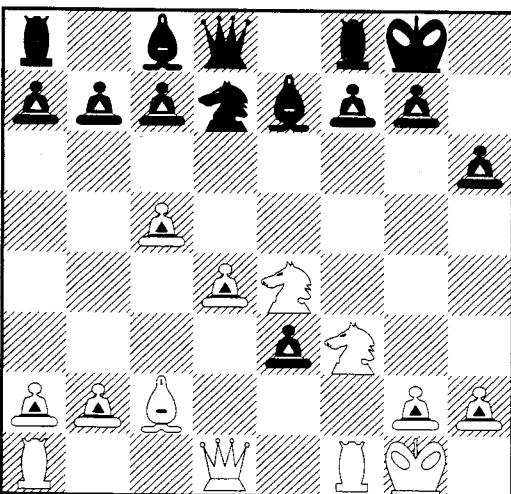
- | | |
|--------------|---------|
| 14. Qd1-d3 | Rf8-e8 |
| 15. Ne4-d6!? | e3-e2 |
| | Nd7-f8? |

A decisive mistake allowing a beautiful combination. Bxd6 was necessary.

- | | |
|-------------|---------|
| 16. Nd6xf7! | e2xf1q+ |
|-------------|---------|



- | | |
|-------------|---------|
| 17. Ra1xf1 | Bc8-f5 |
| 18. Qd3xf5 | Qd8-d7 |
| 19. Qf5-f4 | Be7-f6 |
| 20. Nf3-e5 | Qd7-e7 |
| 21. Bc2-b3 | Bf6xe5 |
| 22. Nf7xe5+ | Kg8-h7 |
| 23. Qf4-e4+ | resigns |



References

- [1] Appelt, W. (1988): Typesetting Chess. *TUGboat* 9#3, pp. 284–287.
- [2] Rubinstein, Z. (1989): Chess Printing via METAFONT and TEX. *TUGboat* 10#2, pp. 170–172.

◊ Zalman Rubinstein
 University of Haifa
 Department of Mathematics and
 Computer Science
 Mount Carmel
 Haifa 31999 Israel
 Bitnet: `rsma407@haifaum`

Editor's note: This article uses a new METAFONT chess font produced by Prof. Rubinstein and his colleagues.

Bibliographic Citations; or Variations on the Old Shell Game

Lincoln K. Durst

This is the first of several tutorials designed to introduce users to some of the subtler parts of TEX, to show how to construct tools to make TEX do things you might like to have it do for you, and to encourage you to take off on your own with the construction of other tools you would find useful. These pieces are no substitute for reading *The TEXbook*; in fact they may be considered successful if they get you to study parts of some danger zones you may have been reluctant to wander into before.

We describe ways plain TEX may be used to perform various clerical functions, useful for authors of papers or books who choose to do their own TEX coding as they create the "manuscript". There exist excellent, finely-tuned, and versatile systems ready to use "off-the-shelf" made by Michael Spivak (*AMS-TEX*) and Leslie Lamport (*LATEX*) which do some of the kind of things we shall be discussing (as well as much more). Newcomers to TEX may find parts of *AMS-TEX* and *LATEX* code hard reading, especially if they try to make changes in order to adapt them for their own needs. Our task is not to reinvent the "wheel"; rather it is to explore ideas that may help users understand how some parts of such "wheels" might work. Here we confine

our attention to plain techniques which are easily modifiable and can be adapted or improved by users to address situations of special interest to them. The code printed here is given in fragments to illustrate underlying ideas one or a few at a time.

In the first of these columns we consider the question of constructing bibliographies and lists of references in mathematical or other articles or books. The objective is to make TEX do as much of the "clerical" work as possible (or reasonable). In particular, the numbering of items will be automated so that, as revisions are made and material is changed, interpolated, deleted, or shifted around, the citations will be adjusted properly when the text is composed.

There are at least three forms for lists of items cited. Chemists and physicists frequently list items in the order cited, as do historians and others, using superior figures in the text in order to refer to them. In these cases, the lists may appear either as endnotes or as a "list of references cited." In mathematical articles and books, on the other hand, references and bibliographies most commonly are listed in alphabetical order by authors' names, and occasionally in chronological order by date of publication. Mathematicians tend to put citations in the text within square brackets [as parenthetical remarks, like this one], treating them as asides to the reader. A bibliography, in contrast to a list of references, may include items not actually cited. See, for example, *Concrete mathematics* by Graham, Knuth, and Patashnik (Addison-Wesley, 1989).

There are some curious, if not notorious, examples in which items are listed in an apparently random order. See, for example, *Mathematics Magazine*, 61#5 (December 1988), pages 275–281. This interesting article by Ivan Niven is about what it takes to win at twenty questions when the person giving the answers is allowed to lie. (THEOREM: *One lie is worth five extra questions.*) The bibliography (mislabelled "References") surely deserves an award for innovation.

The Chicago Manual of Style, "thirteenth edition" (University of Chicago Press, 1982), contains, in chapters 15–17, exhaustive discussions of endnotes, bibliographies, etc., and serves as a source of information on the kind of results desired, as well as suggestions for avoiding many problems, some of which no longer exist, especially for users of TEX.

The construction of FIGURE 1 provides an example of one way the desired results may be obtained using `plain.tex`, with the numbering of sections, displays, references, etc., done automatically. Prior to running off final copy, the macros

1. Fermat numbers

\sect.Fermat.

Fermat considered numbers of the form $2^{2^n} + 1$, which are now known as the *Fermat numbers*, F_n , and he may or may not have asserted [3, pp. 23ff] that he had proved they are primes for all natural numbers n . Subsequently Euler found that the sixth Fermat number,

$$F_5 = 2^{32} + 1 = 42949\,67297,$$

is a multiple of the prime 641. (Early results of this kind will be found in Dickson's history [2, volume i] and more recent results in a book by Brillhart, *et al*, published last year [1].)

Euler's result for F_5 follows from the elementary facts given in displays 1.1 and 1.2:

$$641 = 5 \cdot 2^7 + 1 = 2^4 + 5^4 \quad (1.1) \quad \backslash \text{disp.Powers.}$$

hence

$$5 \cdot 2^7 \equiv -1, \quad 5^4 2^{28} \equiv 1, \quad 5^4 \equiv -2^4, \quad 2^4 2^{28} \equiv -1 \pmod{641}. \quad (1.2) \quad \backslash \text{disp.Congruences.}$$

I learned this arithmetic trick from Olaf Neumann of Friedrich Schiller Universität, Jena, D.D.R.; he did not tell me who invented it. LKD

2. References

\sect.Refs.

- 1 Brillhart, John; Lehmer, D. H.; Selfridge, J. L.; Tuckerman, Bryant; Wagstaff, S. S., Jr. *Factorizations of $b^n \pm 1$, $b = 2, 3, 5, 6, 7, 10, 11, 12$ up to high powers*, American Mathematical Society, Providence (Contemporary mathematics 22, second edition), 1988. *\ref.brillhartFOB.*
- 2 Dickson, Leonard Eugene. *History of the theory of numbers*, three volumes, Carnegie Institution of Washington, Washington, D. C. (Publication number 256), 1918, 1920, 1923. (Reprinted by Hafner and Chelsea.) *\ref.dicksonHTN.*
- 3 Edwards, Harold M. *Fermat's last theorem*, Springer-Verlag, New York, Heidelberg, Berlin (Graduate texts in mathematics 50), 1977. *\ref.edwardsFLT.*

FIGURE 1.

used may be printed in the margin, as shown, to facilitate making cross references during revision. In this installment we describe how the second section and the first paragraph of the first section were composed in a single pass. In the next installment, we describe how the ideas used here can be supplemented by others to handle forward references to displays, exercises, theorems, sections, chapters, etc., without having to typeset the text twice.

First we construct a separate file containing definitions for items to appear in the bibliography; call it *bibliog.fil*:

```
%%% bibliog.fil %%%
\def\dicksonHTN{...}
\def\edwardsFLT{...}
\def\brillhartFOB{...}
...
\endinput
```

The dots in the definitions represent the text to appear when the bibliography is printed (author name[s], title, publisher, date, etc.). This file could

contain other items as well as those required for this occasion, including others related to the current topic (though not actually cited) and any others the author may anticipate wanting to cite in this work or in others on related subjects. Those which prove to be unnecessary can be dropped at the last minute from files to be constructed from this one. The order in which the definitions appear in this file is irrelevant.

We consider first lists of references printed in alphabetical order by author names. If there are not very many items to be cited, it would be easy to construct by hand another file from *bibliog.fil*, say *bibliog.ord*, which contains the lines:

```
%%% bibliog.ord %%%
\bibmac{\brillhartFOB}
\bibmac{\dicksonHTN}
\bibmac{\edwardsFLT}
\endinput
```

sorted into the order in which items are to be printed in the bibliography. For larger cases, this

process can be automated in part in a variety of ways: One may take advantage of keyboard macros or write a program in a high-level language to extract the necessary parts of `bibliog.fil` and perform the required sort. (`TeX` can be made to do part of this work, as indicated below.) At or near the beginning of the file containing the text to be composed, include the line `\input biblio.prc`, which reads in the following file:

```
%%% biblio.prc %%%
\newcount\bib \bib=0
\def\bibmac#1{\advance\bib by 1
  \expandafter
  \xdef\csname #1\endcsname{\the\bib}}
\input bibliog.ord
\def\ref.#1.{\bf\csname#1\endcsname}
\endinput
```

What happens when this is `TeX`ed? First a counter, `\bib`, is allocated and initialized. Then, as `bibliog.ord` is read in, new definitions for macros named `\brillhartFOB`, etc., are constructed using `\csname` [see *The TeXbook*, page 40]. These new definitions assign the numbers to be printed in the text at places the bibliographic items are cited, so that, at least temporarily, we have what amounts to `\def\brillhartFOB{1}` and, therefore, `\ref.brillhartFOB.` is just `{\bf 1}`, etc. The citation itself is made in the text by writing, for example,

```
... he may or may not have asserted
[\ref.edwardsFLT., pp.^23ff]...
```

So far we have had two quite different definitions for the macros `\brillhartFOB`, etc. (first those in `bibliog.fil`, which we haven't really used yet, and now the new ones just constructed). Before we are finished we shall see several definitions for the macro `\bibmac` and other treatments of the bibliographic definitions, some of which will appear in the closing moves of this shell game.

At the place in the text file where the bibliography is to appear, insert the line `\input biblio.set`, which reads in the following file:

```
%%% biblio.set %%%
\def\bib#1#2\endbib{...}
\bib=0
\def\bibmac#1{\advance\bib by 1
  \bib{\bf\the\bib}%
  {\csname#1\endcsname}\endbib}
\input bibliog.fil
\input bibliog.ord
\endinput
```

The first line here contains the definition which specifies the shape of the paragraphs in the list of references (e.g., hanging indentation), type size, leading, parskip, etc. Next we reset the counter `\bib` and change the definition of `\bibmac` so

that, when `bibliog.ord` is read in again, `\bibmac` actually typesets the bibliography.

For a list of references printed in order of citation, we construct a file to play the rôle of `bibliog.ord` as the text file is being processed by `TeX`, and we shall require a revised version of `biblio.set`.

Here we must construct a new file, to replace the file `bibliog.ord` used in the previous example, which contains the bibliographic macros listed in the order of their first appearance in the text.

Instead of using `biblio.prc`, we replace it by `citation.prc`:

```
%%% citation.prc %%%
\newcount\bib \bib=0
\newcount\Bib \Bib=0
\newwrite\bibliolist
\immediate\openout\bibliolist=citation.ord
\def\bibmac#1{\advance\Bib by 1
  \expandafter\def\csname
  #1\endcsname{\the\bib}}
\input bibliog.ord
\def\ref.#1.{\expandafter
  \ifnum\csname#1\endcsname=\the\bib
  \ifnum\the\bib<\the\Bib % not done yet
    \advance\bib by 1%
    \immediate\write\bibliolist
    {\noexpand\bibmac{#1}}%
    \expandafter\edef\csname
    #1\endcsname{\the\bib}%
  \fi\fi
  {\bf\csname#1\endcsname}}}
\endinput
```

As in `biblio.prc`, we begin by allocating a counter, `\bib`, and set it to 0. This time we allocate another counter as well, `\Bib`, which will count for us the number of items in the list of references and, in addition, we allocate a file into which we shall write things and then open it with the name `citation.ord`. [For information on reading and writing files using `TeX`, see *The TeXbook*, pages 217–218, 226–228.] Next we redefine `\bibmac` so that it sets every one of the bibliographic macros equal to `\the\bib` (the value in the counter `\bib`) when `bibliog.ord` is read in, which is what happens next. (Instead of `bibliog.ord` one could use the file `bibliog.uns` described below, since neither the order of its lines nor whether it contains items which will not be cited are relevant in this case.) The first subtlety here is in the definition of `\bibmac`, which uses a plain `\def` for the new definitions of `\brillhartFOB`, etc., which are first defined all to be 0. Use of `\def` here, instead of `\edef`, means that each time a reference to one of these macros is encountered and `\bib` is advanced, the value assigned to *each* of the macros is increased by one. What follows next is a procedure that will fix the value of the current argument of `\ref.#1.`, while

the others will still be permitted to grow. The definition of `\ref.#1.` does this by using `\csname` again, but this time with an `\edef` instead of `\def`.

If we introduce an `\if`-switch we can combine the two ways for handling references when it is time to print out the list. If the following code is tucked away somewhere near the beginning of things

```
\newif\ifOrdCited \OrdCitedfalse
\def\RefsInOrderCited{\OrdCitedtrue}
```

we can adopt a more general form of `biblio.set`:

```
%%% biblio.set %%
\bib=0
\def\bibmac#1{\advance\bib by 1
  \bibl{\bf the\bib}%
  {\csname#1\endcsname}\endbibl}
\input bibliog.fil
\ifOrdCited
  \immediate\write\bibliolist
    {\string\endinput}
  \immediate\closeout\bibliolist
  \input citation.ord
\else
  \input bibliog.ord
\fi
\endinput
```

Thus, one may insert the lines

```
[%] \RefsInOrderCited
\ifOrdCited\input citation.ppr
\else\input biblio.ppr\fi
```

following the definition of `\RefsInOrderCited` and either include or not include a percent sign before the first of these three lines to obtain the references in alphabetical order or in the order of first citation, respectively. Here we have a good example of the value of computers and software such as `TEX`: Authors shouldn't have to fuss over questions at this level of detail, they have more important things to attend to when preparing their ideas for publication. On the whole, it's up to the designers of books and journals to determine the order in which references should be listed, not authors, nor even—for that matter—editors.

There is a practical problem which deserves attention. Suppose typographical errors exist in arguments of `\ref.#1.`; how can they be caught? For this we could exploit a trick discussed by Stephan v. Bechtolsheim in a Tutorial in *TUGboat*, volume 10, number 2 (July 1989), page 205. The idea is that, unless `\csname #1\endcsname` has previously been defined, its value is `\relax`. Using this fact we can test the argument of `\ref.#1.` to see if `\csname #1\endcsname` actually was defined by `\bibmac`; if the test fails, we can arrange to have `TEX` make a fuss that is not liable to be overlooked. Readers may wish to try their hand at constructing such code. (See the Note at the end of this article.)

At the beginning, we claimed that a short list of references cited could easily be handled manually, as far as constructing the file `bibliog.ord`, given the file `bibliog.fil`. In the general case, there are three or four steps in this task, two of which can be automated more or less satisfactorily. The original file `bibliog.fil` can be converted into one whose entries look like those in `bibliog.ord` and then sorted. If there are entries to be eliminated (items neither cited nor to be printed in the "bibliography"), some judgment should be exercised about when and where to do this. Further judgment may be required to compensate for inadequacies in the choice of strings to name the bibliographic items and in the sorting process employed. The ideal procedure will surely result from a trade-off between manual effort and how much fancy automation one is willing to concoct.

One step is easy, fiendishly simple, using `TEX` code suggested by Ron Whitney for the purpose:

```
%%% bibmac.tex %%
\newwrite\outfile
\immediate\openout\outfile=bibliog.uns
\def\gobble#1{} % TeXbook, p 308, ex 7.10
\def\dropslash{\expandafter\gobble\string}
\def\makebibmac#1#2{\immediate\write\outfile
  {\string\bibmac{\dropslash#1}}}
\let\def=\makebibmac
\input bibliog.fil
\immediate\write\outfile{\string\endinput}
\closeout\outfile
\bye
```

This makes a nice quiz with which to end. Notice that `\gobble` eliminates the following token only, while `\makebibmac` eliminates everything that follows *except for* the very next token. The first argument of `\makebibmac` is therefore the macro being defined in `bibliog.fil`, so `\dropslash` nibbles off the backslash (the first token of `\string#1`) and the rest of it is written into the file `bibliog.uns` wrapped up inside `\bibmac`, which is just what we want.

Next we have to sort the file `bibliog.uns`. If you work with UNIX or MS-DOS, use the command

```
p:sort <bibliog.uns>bibliog.ord
```

Here `p:` represents the path to the file `sort.exe`. If you don't have UNIX or MS-DOS, look up sorting in your system manual, or write a program in some high level language to do the job for you.

The next question is how near are we to our destination at this point in our trip from `bibliog.fil` to `bibliog.ord`? Part of the answer depends on how clever you were naming the bibliographic control sequences. Several options are at hand: Be *very* clever choosing the names; exert much effort devising clever sorting algorithms; spend a little

effort studying the results so far and rearrange by hand any items not yet in proper order. Sooner or later, automated activity must end, and some other kind of thought is indicated.

If you prefer the references listed in chronological order, rather than alphabetical order, you might use macro names like \BJIIbrillhart, etc., substituting A, B, ... I, J for the digits 0, 1, ... 8, 9 in dates of publication (with more such "digits" at the end to cope with authors having more than one item per year). Then the same sorting process may be used to make the first (rough) sort of *biblog.uns* as before.

Some authors, especially historians, favor endnotes that are much more extensive than mere bibliographical citations; for endnotes of this kind, some of which may consist of several paragraphs (and may contain cross references to one another), the scheme described above is quite inappropriate. Such endnotes are typographically equivalent to solutions for exercises. How to handle solutions for exercises and discursive endnotes are topics for a later tutorial in this series.

In the next episode, code will be described to produce cross references and marginal notes. In particular, we shall give another version of *biblio.set* containing provision for displaying the marginal notes shown in FIGURE 1.

Note. A disk (5.25 DSDD) containing source text for FIGURE 1 and the code files used to produce FIGURE 1 is available for MS-DOS users who are members of the T_EX Users Group. In addition, code is included for trapping typographical errors in the bibliographic citations as well as identifying (for the case of alphabetical order) bibliographic items not actually cited. The disk also contains source text, including the code samples displayed, for draft versions of other tutorials in the pipeline for this series. Send \$6 (which includes a royalty for the T_EX Users Group) to the address below. Outside North America, add \$2 for air postage.

It is a pleasure to acknowledge the generous help and encouragement of Barbara Beeton and Ron Whitney, without which these ideas would not have been developed.

◊ Lincoln K. Durst
46 Walnut Road
Barrington, RI 02806

Macros for Indexing and Table-of-Contents Preparation

David Salomon

Introduction

Two macros are presented and described in detail. The first is very useful for the preparation of an index; the second prepares a table of contents (toc). It should be noted that L_AT_EX has macros for similar purposes. Ours, however, are different. Our index macro can produce both silent and non-silent index items, whereas L_AT_EX's only generates silent ones. Our toc macros can easily be modified by the user to specify any format for the table of contents.

Another important aspect of the macros is that they are described in detail, thereby illustrating the concept of a multi-pass T_EX job and the use of several advanced T_EX features, such as active characters, file input/output, \edef, \futurelet, and \expandafter.

The Index Macro

A good index is important when writing a textbook. So much so that Knuth, on several occasions (see reference 1 pp. 423–425, and reference 6), said that he does not believe in completely automating the preparation of an index, and he always puts the final touches on his indexes by hand. As a result, "*his books tend to be delayed, but the indexes tend to be pretty good.*"

Index preparation by computer is not a simple problem. References 1–2 discuss certain features that a good index should have, and how to incorporate them in T_EX. The macro described here is relatively simple (even though some readers may not think so) and implements only one advanced index feature namely, *silent* index entries. However, as an example of a T_EX macro it is very interesting because it illustrates the use of the features mentioned above.

The macro accepts an index item and writes it on a file, for the future generation of an index. Its main feature is the use of *optional parameters*. The macro accepts either one, two, or three parameters, of which only one is mandatory. The main parameter should be delimited, as usual, by braces, and the optional ones, by square brackets '[' ']'. The macro writes all its parameters on the index file, as one string. However, only one parameter, the main one, is typeset. The optional parameters are treated as silent index items, items that should appear in the index but not in the text itself. A good example is a sentence such as:

The late Dr.~Mad used to say: ‘‘Computers are good, only people are bad.’’

When writing such a sentence, the author might want to generate the three index items:

```
Mad Nick, 1923--1987
Quotations---Computers
Mad---quotations
```

This is why a good index macro should support silent parameters. We selected the question mark ‘?’ as the name of our macro (see below), so the sentence above should be typed:

```
The late Dr.~?{Mad}[Nick, 1923--1987] used
to say?[Mad---Quotations]{:}
‘?[Quotations---}{Computers} are good,
only people are bad.’'
```

If the optional parameters are used, one of them should precede, and the other one follow, the main parameter. All the parameters are written on the index file—with spaces separating—as one string, followed by the page number. It should be noted that many L^AT_EX macros support optional parameters.

Examples of the use of the index macro are:

```
?{...}
? [...] {...}
? [...] [...]
? [...] {...} [...]
? [...] {}
? {} [...]
```

Note that the main parameter, in braces, should always be present; even if it is empty, as in the last two examples. This happens when the entire index item should be silent, as in ?[Mad---Quotations]{} above.

The main problems in writing this macro are:

1. The macro name should be as short as possible—since it is going to be used a lot—and we have selected the question mark ‘?’ as the name of the index macro. Short macro names consist of a backslash followed by one character. However, it is possible to declare a character as a macro name—by declaring it an *active character*—and then the ‘\’ is not necessary. The character ‘?’ is thus declared active by \catcode‘?=\\active. Since we still want to be able to typeset a question mark, we define a control sequence \? as the ASCII code of ‘?’ by \chardef\?=\\?. There is, of course, nothing special about the question mark. Any other character can be used as the name of the index macro. Reference 1 (p. 423) uses the circumflex ‘^’, but, since the circumflex is also used

in math mode for a superscript, care should be taken not to mix the two uses.

2. The macro should be able to take 1, 2, or 3 parameters. This is achieved by writing several macros that examine the next character in the text and, if it is a ‘[’, treat it as the start of another parameter, collect the rest of that parameter, and save it (it is saved in a macro called \save). The saved text is later written on the index file, together with the rest of the string. This part of the macro uses the \futurelet control sequence, and is described below.

3. Several index items may be declared on a single page, and their optional parameters should all be saved, as described above. Since our macro always saves text in the same place (in macro \save), we should write the saved text onto the index file *immediately*. This is usually accomplished by \immediate\write, which writes the saved string on the index file immediately, so the next string can be saved in the same place.

In our case, however, the actual writing on the index file must be deferred, since we want to include the page number with each index item, and this number is only known in the output routine. Our macro should, therefore, use \write instead of \immediate\write. The problem is that, by the time we get to the output routine, several strings, from several index items, may have to be saved. We should, therefore, make sure that macro \save is emptied, and its contents written somewhere, before we use it again, for the next item.

This problem is solved by using a combination of \write and \expandafter. The \expandafter makes sure that the saved text is expanded into the \write immediately. The \write itself, however, is executed later, in the output routine.

Listing of the macro

Here is a complete listing of the macro:

```
1. \newwrite\inx
2. \immediate\openout\inx=\jobname.idx
3. \def\wrx{\write\inx}
4. \def\space{ }
5. \chardef\?=\\? % Define \? as a cs whose
6. %      value is the ASCII code of ?.
7. \catcode‘?=\\active % Now change the
8. %      cat. code of ‘?’ to 13.
9. \def?\{\futurelet\new\macA\} % This is the
10. %      new definition of ‘?’.
11. \def\macA{\ifx\new[\let\next=\caseA
12. \else\let\next=\caseB\fi\next}
13. \def\caseA[#1]#2{\def\save{#1 #2}}
```

```

14. \futurelet\new\macB
15. \def\caseB#1{#1\def\save{#1}
16. \futurelet\new\macB}
17. \def\macB{\ifx\new[\let\next=\caseC
18. \else\let\next=\caseAB\fi\next}
19. \def\caseC[#1]{%
20. \wrx\expandafter{\save \space#1---page
21. \folio;}}
22. \def\caseAB{%
23. \wrx\expandafter{\save---page \folio;}}

```

Lines 1–2 are concerned with the declaration, naming, and opening of the output file. The file gets the internal name `\inx` and the external name `\jobname.idx` (`\jobname` is a control sequence whose value is the name of the source file being processed by TeX). Lines 5–8 define the control sequence `\?` and declare ‘`?`’ as an active character. Lines 9–18 are concerned with the optional parameters (problem 2 above), and lines 3–4, 19–23 deal with problem 3 above.

Line 9 is the definition of macro ‘`?`’. It uses the `\futurelet` control sequence, which will be briefly explained. When TeX sees the following:

```
\futurelet <token1><token2><token3>
it:
```

1. Lets `<token1>` be = the (future) `<token3>`.
2. Proceeds as usual, scanning and executing `<token2>`, `<token3>`, ...

Reference 4 is a tutorial on `\futurelet`, describing an example similar to ours.

In the case of macro ‘`?`’, on line 9, `<token1>` is `\new` (the name of a control sequence, initially undefined), `<token2>` is the macro `\macA`, and `<token3>` is the first token following `?` in the input stream. Note that, on line 9, when `?` is defined, `<token3>` is unknown. It only becomes known after `?` is expanded. Since `<token3>` is the first token following `?`, it is usually a ‘`\`’ (which starts the main parameter). However, if the first parameter is optional, `<token3>` is a ‘`[`’. As a result, macro `\new` is `\let` to either a ‘`\`’ or a ‘`[`’, and macro `\macA` is expanded.

`\macA`, in turn, examines `\new` and, based on its value, expands either macro `\caseA` or macro `\caseB`. Macro `\caseA` thus handles the case where the first parameter starts with a ‘`\`’ (is silent). The macro therefore assumes that the parameter list has the form `[...]\{...\}` followed, perhaps, by a third, optional, parameter. The macro typesets the second (non-silent) parameter and saves both parameters. It then expands macro `\macB` to find out if the next token is a ‘`[`’ (i.e., if there is a third parameter). Macro `\macB` compares the next token to a ‘`[`’

and, if they are equal, expands `\caseC`; otherwise, `\caseAB` is expanded. Macro `\caseC` thus handles the case where the macro has all three parameters. Using `\expandafter`, it immediately expands the saved material (the first two parameters), adds the third parameter and the page number, and executes a deferred `\write`. Macro `\caseAB` is similar, only it does not add any third parameter.

Macro `\caseB` is expanded if the first parameter does not start with a ‘`[`’. In such a case, the macro may have either one parameter or two (with the second one silent). `\caseB` therefore saves the first parameter and expands `\macB` to determine if there is a second one.

The construct

```
\wrx\expandafter{\save---page \folio;}
```

on lines 20, 23, is a simple application of `\expandafter` to reverse the order of expansion of the two tokens ‘`\`’ and `\save`. When TeX reads the `\write\inx` command, it expects to find a ‘`\`’ followed by the material to be written on the file. TeX is ready to save that material—as a whatsit—on the main vertical list, for a delayed write (see ref. 1, p. 227). However, in our case, TeX finds the `\expandafter` instead of a ‘`\`’. This causes TeX to look at token `\save` first (to expand it), and then to place token ‘`\`’ in front of it. The result is to force TeX to expand `\save` immediately (to empty it), even though the `\write` is done later. Macro `\save` is now ready for the next index item.

Complicated! But this is the nature of advanced stuff. . .

The same method is used in this article later, to solve the same problem when preparing a table of contents.

Tests

To test the index macro, we adopt an idea of Winograd and Paxton (ref. 2), and define a macro, `\setpage`, to simulate a page break, so a small amount of text can be spread over several pages.

```
\def\setpage#1 {\par\vfill\eject\pageno=#1}
```

The text in Figure 1a was used to test the macro. This text generated the index file shown in Figure 1b.

Table-of-Contents Generation

A table of contents should contain an item for each chapter, section, subsection, etc. To generate these items automatically, we need to define macros that will be expanded at the beginning of each chapter, section, etc. Such a macro has two functions—it

```

\setpage 3
Because ?{token matching}[in associative
memory] is so appropriate for an
?{associative memory}[--use for token
matching], we conclude
\setpage 4
Another common mistake is to include
several ?[token---]{tag fields in} a single
token. This habit, seemingly harmless,
can cause conflicts in ?{token
matching}[--conflicts].
\setpage 7
Fig. 12--27b shows how this approach is
implemented. Each iteration increments
the ?[incrementing the token]{label}[in a
data flow computer], thereby generating
?{unique tokens}[--generation of].
\setpage 12
We conclude that ?[token]{matching} must
involve the ?[token matching
by]{iteration number}[and destination],
and a general way of achieving this is
to add a new field, a ?{label}[in data
flow tokens] field, to each token, and to
match tokens by destination, handedness,
{\it and} label.
\setpage 13
A token $A_i$, arriving at the ?{matching
store}[--token arrival at] will match
itself to a $B$ token having the same
label. Such a token is $B_i$ (a $B$ token
that was generated by the same iteration
as $A_i$).
?[matching tokens---various methods]{}

There is another approach to the same
problem. The matching store can be
redesigned such that matching is done
?{token matching}{in order of
seniority.}[A different approach] A token
$A_i$ will always match itself to the
oldest token which has the same
?{destination}[and handedness, used in token
matching] and opposite handedness.
\setpage 16
In our case this would mean that $A_i$
would match itself to $B_i$. In other cases,
however, matching by seniority does not work.

Problem: What could be a good ?{data
structure}[for matching by seniority]
for such a matching store?

```

Fig. 1a

should typeset the name and number of the section, taking care of vertical spacing and pagination—and it should make sure that the proper toc information goes on an auxiliary file, named `\jobname.toc`, for a later preparation of the table of contents. The macros defined here are called `\chapter`, `\section`, `\subsection`. Other macros, such as for a sub-sub-section, can be defined along the same lines.

The `.toc` file should be `\input` in subsequent passes and used to prepare the final table of contents. Our approach to this problem is to write only the basic toc information on the file, then to create the table of contents by the single command `\input\jobname.toc`. This way, the formatting of the table of contents is separated from the problem of preparing the `.toc` file. To change the format of the table of contents, one only needs to change the definitions of the formatting macros, without changing the format of the `.toc` file.

An important feature of this example is that it requires a number of passes over the source file. Traditionally, the pages of a table of contents are numbered separately, using roman numerals. If, however, we want the toc pages to be numbered as part of the document (i.e., to have page numbers 1, 2, ...), we have to perform two passes. In the first pass the `.toc` file is generated and, at the start of the second pass, it is `\input`, and the toc is typeset on the first pages of the document.

A little thinking shows that even two-passes are usually not enough, and a third pass may be necessary. When the toc is typeset, in the second pass, it occupies the first couple of pages, causing the start of the text to be moved from page 1 to an unknown page. The toc just typeset thus has the wrong page numbers. The second pass, however, also creates an auxiliary `.toc` file, this time with the right page numbers. Finally, the third pass typesets a toc with the right page numbers.

Exercise 1: Is a fourth pass ever necessary?

A general problem in any multi-pass job is, how does `TEX` know which pass it is doing? A simple, elegant, solution is to have all the passes identical. This way there is no need to tell `TEX` which pass it is currently working on. Our macros therefore start by trying (on lines 1-2 below) to input a `.toc` file. If such a file exists, it is used, on line 6, to typeset the table of contents, otherwise, a message is sent to the log file, to alert the user. On line 8, the file is closed. The last step is to open a new `.toc` file, on lines 10-11.

The steps described so far are accomplished by:

token matching in associative memory---page 3;
 associative memory ---use for token matching---page 3;
 token--- tag fields in---page 4;
 token matching ---conflicts---page 4;
 incrementing the token label in a data flow computer---page 7;
 unique tokens ---generation of---page 7;
 token matching---page 12;
 token matching by iteration number and destination---page 12;
 label in data flow tokens---page 12;
 matching store ---token arrival at---page 13;
 matching tokens---various methods---page 13;
 token matching in order of seniority. A different approach---page 13;
 destination and handedness, used in token matching---page 13;
 data structure for matching by seniority---page 16;

Fig. 1b

```

1. \newread\toc
2. \immediate\openin\toc=\jobname.toc
3. \ifeof\toc
4. \message{! No file \jobname.toc;}
5. \else
6. \tochead \input\jobname.toc \vfill\eject
7. \fi
8. \immediate\closein\toc
9. %
10. \newwrite\toc
11. \immediate\openout\toc=\jobname.toc

```

The .toc file format

The .toc file described here contains, for each toc item, a simple record with the following fields:

- One of the codes \ch, \se, \sbs, for a chapter, section, and subsection, respectively.
- The chapter (or section) number, followed by a colon ':'.
- The chapter (or section) name, followed by the word '\page'.
- The page number, followed by a '\\'.

A typical, simple .toc file may look like the example below:

```

\ch1:Introduction\page3\
\se1.1:The Use of Tags\page4\
\sbs1.1.1:Incrementing the Tags\page7\
\se1.2:Label of Tokens\page12\
\sbs1.2.1:Seniority Matching\page13\
\se1.3:Summary\page16\

```

Such records are easy to write on the file, and they make it possible to typeset the entire table of contents by the single line

```
\tochead \input\jobname.toc \vfill\eject
```

This is achieved by defining macros \ch, \se, \sbs, to typeset lines in the toc. Macro \ch, for example, typesets a chapter line in the toc. It is expanded automatically during the \input, each time a record starting with a \ch is read off the file. Macros \se, \sbs behave similarly. These macros (plus \tochead, which typesets the heading of the table of contents) are the only ones that typeset the toc and, as a result, the only ones that need to be modified when a different toc format is required. The following are guidelines for writing these macros:

```

\def\tochead{\{typeset a heading for the table of
contents\}}
\def\ch#1:#2\page#3{\{(typeset a line in the
toc, with #1 as the chapter number, #2 as the
chapter name, and #3 as the page number\}}
\def\se#1:#2\page#3{\{(similarly for a section\}}
\def\sbs#1:#2\page#3{\{(similarly for a
subsection\}}

```

Writing the .toc file

At the start of each chapter, the user expands macro \chapter with one argument, the chapter name.

```

1. \newcount\chnum \chnum=0
2. \newcount\snum \newcount\sbsnum
3. \def\chapter#1{\global\advance\chnum 1
4. \global\snum=0 \sbsnum=0
5. \{select a font and typeset \the\chnum and #1\}
6. \edef\save{\string\ch\the\chnum:#1%
7. \string\page\noexpand\folio\string\\}%
8. \write\toc\expandafter{\save}}

```

The macro should typeset the chapter name and take care of vertical spacing and page breaks. Its last step is to store the information necessary

for the toc in a macro called `\save`, and to write `\save` on the .toc file.

At the start of each section or subsection, the user similarly expands `\section` or `\subsection`, which behave similarly to `\chapter`.

```

9. \def\section#1{%
10. \global\advance\snum 1 \sbsnum=0
11. {typeset \the\chnum.\the\snum and #1}
12. \edef\save{%
13.   \string\se\the\chnum.\the\snum:%
14.   #1\string\page\noexpand\folio
15.   \string\}%
16. \write\toc\expandafter{\save}}
17.
18. \def\subsection#1{%
19. \global\advance\sbsnum by 1
20. {typeset \the\chnum.\the\snum.\the\sbsnum
21. and #1}
22. \edef\save{%
23.   \string\sbs
24.   \the\chnum.\the\snum.\the\sbsnum:%
25.   #1\string\page\noexpand\folio
26.   \string\}%
27. \write\toc\expandafter{\save}}

```

The macros have to deal with two related problems; namely the chapter number and the page number on the toc file.

The page number, `\folio`, is not known when the toc record is created. It only becomes known when the output routine is invoked. The `\write` should therefore be delayed. This is a common problem and is solved simply by saying `\write` instead of `\immediate\write`.

The chapter number, `\the\chnum`, on the other hand, is known and should be expanded immediately. If its expansion is delayed to the output routine, the number expanded will be the chapter number in effect during the output routine. The same applies to the section and subsection numbers.

These problems are solved, in macro `\chapter`, on lines 6–8.

Lines 6–7 define macro `\save` with the necessary information for a single toc record. The `\edef` control sequence is used, instead of `\def`, to guarantee that the chapter number, `\the\chnum`, that is expanded inside `\save` will be the one in effect when `\save` is *defined*, not the one when `\save` is expanded.

The `\noexpand\folio`, on the other hand, guarantees that `\folio` will not be expanded when `\save` is defined; instead, it will be expanded when the `\write` is expanded (in the output routine).

The use of `\expandafter` has been explained earlier, in connection with index preparation.

Exercise 2: Experiment with lines 6–8 above to find out what happens when the `\edef` is changed to `\def`, when the `\noexpand` is omitted, and when the `\expandafter` is dropped.

Limitations

1. The size of a record on a file is limited by the operating system of the computer. Since each line of the table of contents goes on the file as a record, its size is limited and, as a result, we cannot have chapter or section names which are too long. Note that this limitation has nothing to do with *TEX*.
2. When a large document—such as a book—is typeset, it is common to typeset each chapter individually, creating its own .toc file. In such a case it is possible to create the final table of contents by a special *TEX* job. Each of the individual .toc files is input, and the table of contents is numbered separately, using roman numerals. This only requires two passes, and generates a toc similar to the traditional method.

Exercise 3: Sometimes the book designer wants the chapter numbers in roman numerals, how can this be done?

Acknowledgement

The author would like to thank Ron Whitney for his many important comments and suggestions. They have caused a major revision of this work, and have made it much more useful to the readers.

Answers to exercises

1. Yes, if the text is modified in any of the passes. Even more than four passes may be necessary in such a case.
2. Just do the experiments.
3. Use

```
\uppercase\expandafter
  {\romannumeral\the\chnum}
```

instead of `\the\chnum` on line 3 in macro `\chapter`. This tricky construct is demonstrated in exercise 7.9 of ref. 1.

References

1. Knuth D. E., *The T_EXbook*, Addison-Wesley, Reading MA: 1987.

2. Winograd, T. & B. Paxton, *An Indexing Facility for T_EX*, *TUGboat* 1(1) (Appendix A).
3. Chen, P. & M. A. Harrison, *Index Preparation and Processing*. Soft. Practice & Exp. 18(9), 897-915 (Sept. 1988).
4. Bechtolsheim, S., *A Tutorial on \futurelet*, *TUGboat* 9(3), 276-279, 1988.
5. Bechtolsheim, S., *A Tutorial on \expandafter*, *TUGboat* 9(1), 57-61, 1988.
6. Knuth D. E., *Typesetting Concrete Mathematics*, *TUGboat* 10(1), 31-36, 1989.

◊ David Salomon
 California State University,
 Northridge
 Computer Science Department
 Northridge, CA 91330
bccscdxs@csunb.csun.edu

Query

Editor's note: When answering a query, please send a copy of your answer to the *TUGboat* editor as well as to the author of the query. Answers will be published in the next issue of *TUGboat* following their receipt.

A Scribe-to-T_EX Converter

One of the SEMATECH consortium members donated a software product with lots of documentation. Unfortunately it's marked-up using Scribe. SEMATECH has modified the software product to meet our needs, but the prospect of un-SCRIBING and then T_EX-ing hundreds of large user and system documentation files with a text editor is not attractive. Please let us know if you are aware of any Scribe-to-T_EX translators.

Terry Bush
 SEMATECH
 Montopolis Research Center
 University of Texas
 2706 Montopolis Drive
 Austin, Texas 78741
 (512) 356-3443
terry_bush@sematech.mrc.utexas.edu

LATEX

Towards LATEX 2.10

Frank Mittelbach and Rainer Schöpf

After the TUG meeting at Stanford, Leslie Lamport expressed interest in future developments of LATEX. He and one of the authors (FMi) agreed on a two-stage procedure for this [1, 2]. The first step will be a new style file interface. Therefore we are interested in any style file which implements features that are not provided in the current document styles.

Independently of these efforts we are planning to publish the implementation of a number of enhancements to the current LATEX version:

- A new **verbatim** environment.
 This includes a **\verbatimfile** command to read in a file of verbatim text, and a **comment** environment that discards all text in its body. Other features are: no limitation on the size of the verbatim text, and the possibility of using **verbatim** inside other environments.
- A new version of the **doc**-option.
 One of the most important improvements over the version published in the previous issue of *TUGboat* is the introduction of a check to detect truncations during transmission. We are very interested in hearing about experiences other people have had with this style option. Suggestions for improvements are welcome.
- Enhancements to the new **array** and **tabular** environments published in *TUGboat* 9#3.
 Again, suggestions are welcome.
- The interface between LATEX and the new font selection scheme.
 This interface consists of two parts: one emulates the font selection mechanism of standard LATEX and is ready to use. The second part is made to give full control over the new scheme. However, field tests have shown that the commands we provided for this are not user friendly enough to be released yet.

We are sorry that we have to report a small but very important typo in the article on the new font selection scheme (*TUGboat* 10#2, pp. 222-238).¹ It is very important because it is in the code, namely in the macro **\mathversion** (p. 230): in the first line of the macro definition the primitive **\endcsname** is erroneously spelled "**\endscname**".

¹ Thanks to Sebastian Rahtz for finding this one.

Recently there were some queries in *TeXhax* and *UKTeX* about a *TeX* overflow while using the *doc* option. The source of this is the size of *TeX*'s save stack: usually Don Knuth's original value (600) is used. This works well with plain *TeX*, but it is much too small for *LATEX* documents.² Even size changes in the argument of a \caption command can result in an overflow of the save stack! Therefore we strongly suggest to all *TeX* implementors that they increase this parameter to a value at least as high as 1500.

References

- [1] Frank Mittelbach and Rainer Schöpf, "With *LATEX* into the Nineties." Talk given at the 10th Anniversary TUG meeting, Stanford, August 1989, *TUGboat* 10#4, to appear.
- [2] Rainer Schöpf and Frank Mittelbach, "LATEX limitations and how to get around them." Talk given at the Euro*TeX* conference, Karlsruhe, September 1989, to appear.

- ◊ Frank Mittelbach
Electronic Data Systems
(Deutschland) GmbH
Eisenstraße 56
D-6090 Rüsselsheim
Federal Republic of Germany
Bitnet: pzf5hz@drueds2
- ◊ Rainer Schöpf
Institut für Theoretische Physik
der Universität Heidelberg
Philosophenweg 16
D-6900 Heidelberg
Federal Republic of Germany
Bitnet: BK4@DHDURZ1

² A similar problem arose with *TeX*'s main memory size and hash table size which were increased when *LATEX* was released.

The Development of National *LATEX* Styles

Johannes Braams, Victor Eijkhout, and Nico Poppelier

Abstract

At its autumn 1988 meeting, the Dutch *TeX* users group (NTG) established a working group (number 13) that was to concentrate on the problems involved in the use of *TeX* for Dutch texts. Since then the working group, which includes the authors, has created a number of style options for *LATEX* that remedy some common problems with the non-English use of *LATEX*, and is along the way developing document styles that are compatible with the standard styles, but have a layout that is more palatable for Dutch users. In this article we treat implementation aspects of the styles and style options, and we discuss some matters of layout.

1 The need for national *TeX*

On several occasions it is stressed in both *The TeXbook* [1] and the *LATEX* book [2] that non-English users of *TeX* may have to take steps in order to adapt *TeX* to their native language. For several languages such steps have indeed been taken, for instance for the German language [3]. It was only natural that the Dutch *TeX* users group (NTG) would also initiate an effort in this direction. Thus the active life of working group 13 began somewhere about the beginning of 1989.

As the use of *LATEX* is quite wide-spread in the Netherlands, and because most matters of national standardization can be conveniently handled in the context of document styles — and also because the Dutch language does not have the problems of national characters that are prominent for *TeX* users to the North, South and East of this country — it was decided to focus mainly on the development of national styles and style options for *LATEX*. This article treats some of the problems encountered and the way they were solved.

2 The 'chapter' problem

One of the first problems non-English users of *LATEX* run into, is that of English terms ('Abstract', 'Contents') contained in the document styles. The resourceful user, or the *TeX*nician consulted, will probably take out a text editor and hunt through the style file for the offending string, replacing it by its equivalent in his/her native language. This process will most likely result in new styles, called (for Dutch) *artikel*, *rapport* and so on.

At this point it may occur to the conscientious user that Leslie Lamport must have foreseen this situation, and probably have made provisions for it, so why not see if there is a suggested approach for this. A cursory perusal of the table of contents of the L^AT_EX book will then lead our user to section 5.1.4 'Customizing the document style'. There Lamport takes half a page to explain how the word 'Chapter' is really just the value of a control sequence \@chapapp. This implies that changing this text to 'Hoofdstuk' does not require editing of document styles at all, it merely needs a one-line option file. Charming, one would say. However, immediately after that there is a strange sentence 'You may also want to redefine the \appendix command, replacing Appendix [...]' Curiouser and curiouser! Can't I just redefine some control sequence that yields the word 'Appendix'? Well, as it turns out, 'Chapter' is the *only* text that has been parametrized, the rest is hard-wired into the document styles.

And such is the situation in which working group 13 found itself: we could think of at least three ways of solving the 'Chapter' problem.

- We could make exact copies of the standard styles and all point size options (art10.sty and such), replacing all English text by Dutch text. The disadvantages of this are that (1) supposing a certain installation of T_EX will be used for three languages, then every style file has to be present three times, and (2) if Lamport then finds a bug or decides to issue upgrades of the standard styles, this would require a multitude of changes.
- We might also rewrite the standard styles, parametrizing them, and add option files for the different languages that contain only parameter settings. This solution is subject only to the second objection above.
- In fact there exists an even better solution (due to Piet van Oostrum): if an option file would be able to find out what style is being used, it can replace and parametrize just those commands that contain text, and afterwards set the parameter values to some appropriate language. This approach is probably the most economical one: the original styles are still used, and, if the option file contains parameter settings for a number of languages, this single option file suffices.

The option file dutch, created by one of the authors (JB), implements the third possibility above. It uses for parameter names those suggested by Hubert Partl [3] in the german style. Note however,

that german — an implementation of the second possibility above — does not do the actual redefinitions but merely sets the parameters. Thus it will only function correctly with edited standard styles.

As an example of parametrization of commands consider the following definition, taken from article.sty¹

```
\def\abstract{\if@twocolumn
  \section*{Abstract}
\else \small
  \begin{center}
    \bf Abstract\vspace{-5em}
  \end{center}
  \quotation
\fi}
```

This definition is overridden in dutch.sty by

```
\def\abstract{\if@twocolumn
  \section*{\abstractname}
\else \small
  \begin{center}
    \bf \abstractname\vspace{-0.5em}
  \end{center}
  \quotation
\fi}
```

to which is added a command initializing the language-dependent parameters:

```
\def\captionsdutch{
\def\abstractname{Samenvatting}
.
.
}
```

Some comments about this approach are in order. First of all, the option file dutch.sty really consists of two disparate parts, the redefinitions and the initializations. The redefinitions are only to make up for what we see as a deficiency in the distribution style files. The parameter initializations are then the truly language-dependent part.

Secondly, the option dutch obviously works with the standard document styles, but will collide with some other styles, for instance with new styles to be developed — as we found out to our chagrin. The reason for this is that the same mechanism that repairs the article style, will attempt to repair any style that looks like it. Thus, if a Dutch artikel style has its own ideas about how an abstract should look, it must have a way of protecting itself against dutch's zeal. Such a possibility exists, and it has been incorporated in the new Dutch styles that will

¹ The macros presented here have been simplified to convey the essence of what we are telling. They are not usable in this form.

be discussed below. Other non-standard styles may need to be edited before they can be used with the dutch option, however.

3 Making the layout less 'loud'

To Dutch—and probably some other—eyes, the L^AT_EX styles are a bit 'loud', and therefore two style options have been developed by one of the authors (NP). These work together with the standard styles and make the general layout a bit more compact. A first option is `a4.sty` which sets various parameters in order to accommodate European standard A4 paper. This option started out as a bare union of the `a4` style option of John Pavel and the `a4wide` style option of Jean-François Lamy, but the current version has undergone fine tuning. The second option is called `sober`: it reduces the sizes of fonts in section headings, and it eliminates the white spaces surrounding section headings and items in list structures.

4 Compatible replacement styles for Dutch

Without asserting that there exists such a thing as 'a typically Dutch layout', we can still state that certain aspects of the L^AT_EX styles are less desirable for Dutch documents. Therefore the working group at its first meeting already declared it a goal to develop styles with a Dutch look. With the guidance of a graphical designer [4], and using some books on the subject [5, 6, 7]—including one by a Dutch typographer—two styles have since then reached completion, implemented by one of the authors (VE). One style is compatible with `article`, and one with `report`. By compatibility we mean here that the new style implements the same commands as one of the original styles. Both styles are based on the same graphical design. It is our intention to have further compatible styles available in the near future based on different layouts.

4.1 The flexibility of L^AT_EX

Probably the easiest way to develop a new document style is to start out with an already existing one, and to modify it gradually. In this process one discovers that L^AT_EX has a lot of possibilities for modication built into it. For instance, whether or not the fifth parameter of `\@startsection`—the generic command used in document styles to define section headings—is positive controls the placement of the section heading above or embedded in the text. The absolute value of this parameter is then either

- the vertical distance between section heading and the first line of the text when it is positive, or
- when negative it is the horizontal distance between the run-in heading and the first word of the text.

On the other hand a number of design decisions have been hard-wired into the sectioning commands, and cannot be easily changed, for instance, by means of the parameters of `\@startsection`. Consider as an example the distance between the section number and the heading. This turns out to be exactly '1 em' in the font of the section heading, as can be seen from the following definition—again, this is a rather simplified form of the definition actually appearing in `latex.tex`.

```
\def\@sect#1#2#3#4#5[#7]{#8{%
  \refstepcounter{#1}%
  % #1 is 'subsection' for example
  \edef\@svsec{\csname the#1\endcsname
    \hspace{1em}}%
  \begingroup#6\relax % #6 is the style
  \Changefrom{\hspace{#3}\relax\@svsec}{%
    \interlinepenalty=\@M
    #8\par}%
  % #8 is the heading text
  \endgroup \@xsect{#5}}}
```

Clearly, L^AT_EX makes it easy for the user to determine whether the section number be set using digits or roman numerals or letters: the only action required is redefinition of `\thesection` and so on. Also it is easy for the document-style designer to specify the style of the section heading: this is determined by the sixth parameter of `\@startsection`. It is not easy to change that '1 em', which one might just want to do occasionally.

4.2 Going Dutch

The styles `artikel` and `rapport` embody a number of changes with respect to `article` and `report`. Some of these are fairly trivial, such as the fact that we switch on 'french spacing', and some are more complicated. In the rest of this section we will treat the most significant change, the notion of a 'unit indent', from both a typographical and implementer's point of view.

One of the principles of document design is² that the eye is able to pick up regularities in a page layout, and that their presence is considered posi-

² Or rather: seems to be. We have not encountered explicit statements to this effect, but implicitly it seems to be there.

tive, but that having too much variation is confusing. For instance, the designer we consulted insisted that the white space separating a section heading and the following text should bear some simple relation to the baselineskip, and should not have any stretch.

One point that all our sources seemed to agree on was that the number of ‘implied left margins’ in a document should be as low as possible. By an implied left margin we mean here a non-zero distance from the actual left margin that is taken by more than one item of the document. Examples of implied left margins are

- the paragraph indentation; furthermore
- the left margins of items in an ‘itemize’ or ‘enumerate’ list construct, and
- the left (or right) sides of the numbers and labels in such list constructs, but also
- the left side of the text of a section heading.

In the standard styles of L^AT_EX all of these four distances are independent and are different from one another. In the style we have developed it was decided to strengthen the visual coherence of the layout by taking the same value for each of them whenever possible.

Implementing this idea meant adopting a new dimension \unitindent which first of all unifies the \parindent and the \leftmargini, the indentation of non-embedded lists.

```
\newdimen\unitindent
\fsetbox0=\hbox{\normalsize\rm 2.2.2
  \hskip.5em}
\global\unitindent=\wd0
\parindent=\unitindent
\leftmargini=\unitindent
```

Admittedly this will give a rather large indentation, but this does not seem to be uncommon in contemporary typographical design. In fact, while the L^AT_EX book takes the classical quad — the Dutch term translates to ‘a square of white’ — for the paragraph indent, *The T_EXbook* shows a large indentation which is equal to that for lists on the outer level.

From the computation of the size of the unit indent, the reader may have gathered already that it will also be put to another use: we want the text of section headings — of all numbered sections — to appear at a distance of \unitindent from the left margin. As was indicated above, this requires some modification of the \sect macro. We therefore include in the style file the following redefinition:

```
\def\@sect#1#2#3#4#5#6[#7]{#8{
```

```
\refstepcounter{#1}
  % #1 is ‘section’ for example
\edef\@svsec{\hbox to \unitindent
  {\csname the#1\endcsname \hfil}}
\begingroup #6\relax
  % #6 is the style
\hangfrom{\hskip #3\relax\@svsec}%
  \interlinepenalty=\OM
  \hyphenpenalty=\OM
  \exhyphenpenalty=\OM
  \rightskip=0cm plus 13cm
  #8\par}
  % #8 is the heading text
\endgroup \@xsect{#5}}
```

where the macro \@svsec for the heading now gives a horizontal box of a predetermined width. Note that we have also ensured that the heading is set ragged right. Hyphenation in a heading — even of words with an explicit hyphen — is something so hideous, that we suspect this to be an oversight of Leslie Lamport.

A further unification of implied left margins can be achieved if we look more carefully at embedded lists. In the standard styles an embedded list construct defines two left margins: the left margin of the items, and the left side of the labels. This second margin can be eliminated by making it equal to the text margin of the surrounding list. The placement of such labels is governed by the value of \labelwidth and by the macro \makelabel which is passed by itemize and enumerate to \list. We therefore specify that the label will take the full width of the indentation, for instance

```
\def\@listii{\leftmargin=\leftmarginii
  \labelwidth=\leftmarginii}
```

and redefine \itemize (in the document style file)

```
\def\itemize{\advance\@itemdepth \one
  \edef\@itemitem{\labelitem
    \romannumeral\the\@itemdepth}%
\list{\csname\@itemitem\endcsname}%
  \def\makelabel##1{\hfil##1\hfil}}
```

so that it will make labels that are flush left. The result of these actions can be seen in figures 1, 2, and 3.

4.3 The international feel

It has occurred to us that L^AT_EX users outside the Netherlands may also like the new styles, and may also want to use them. We decided therefore to make the styles in a sense truly compatible to article and report: when used on their own the styles will produce English captions, and using them in combination with dutch or german will alter these. However,

Figure 1:

1 Comparison of styles

1.1 Properties of the distribution styles

There are people who criticize the layout of the L^AT_EX distribution style files.

Main point of contention is usually a perceived lack of unity, in particular the fact that the indentations for

- lists, paragraph indentation, and the implied indentation of
- section labels on levels
 1. for sections,
 2. for subsections,
 3. and subsubsections

are all different.

Also the layout is sometimes considered ‘loud’: fonts in headings tend to be quite big, and there is a lot of white space in the layout. Maybe such things are regionally determined; for use in Dutch, in any case, it was necessary to change them.

Figure 2:

1 Comparison of styles

1.1 Properties of the distribution styles

There are people who criticize the layout of the L^AT_EX distribution style files.

Main point of contention is usually a perceived lack of unity, in particular the fact that the indentations for

- lists, paragraph indentation, and the implied indentation of
- section labels on levels
 1. for sections,
 2. for subsections,
 3. and subsubsections

are all different.

Also the layout is sometimes considered ‘loud’: fonts in headings tend to be quite big, and there is a lot of white space in the layout. Maybe such things are regionally determined; for use in Dutch, in any case, it was necessary to change them.

Figure 3:

1 Comparison of styles

1.1 Properties of the distribution styles

There are people who criticize the layout of the L^AT_EX distribution style files.

Main point of contention is usually a perceived lack of unity, in particular the fact that the indentations for

- lists, paragraph indentation, and the implied indentation of
- section labels on levels
 1. for sections,
 2. for subsections,
 3. and subsubsections

are all different.

Also the layout is sometimes considered ‘loud’: fonts in headings tend to be quite big, and there is a lot of white space in the layout. Maybe such things are regionally determined; for use in Dutch, in any case, it was necessary to change them.

the reader will understand after the above discussion that we achieve this by somewhat more sophisticated means than are employed in the standard styles.

None of the commands in the `artikel` style, for instance, contain actual texts. Instead they contain such commands as `\abstractname`. These commands are initialized at the end of the file to give English texts. Any language option based on the same parameter names, such as `dutch` or `german`, can override these settings for use in other languages. Naturally, the `artikel` style takes precautions to prevent `dutch` from mistaking it for `article`.

On a more philosophical note, we may add that we feel that this situation is how it should have been from the beginning. In fact, Leslie Lamport himself has shown the way to international styles—see the passage quoted above. For some reason however, he stopped short and did not implement this idea.³

5 Conclusion

It has turned out to be possible to adapt L^AT_EX for use with the Dutch language, in such a way that hardly any action on the part of the user is required. With some style options English terms can be replaced, and the typically American layout can be made more acceptable to Dutch eyes. The development of completely new document styles for Dutch, however, leaves the implementer with the feeling that he has had to unearth sections of L^AT_EX that were never meant to be rewritten.

References

- [1] Donald Knuth, *The T_EXbook*, Addison-Wesley, 1986.
- [2] Leslie Lamport, *L^AT_EX, A Document Preparation System*, Addison-Wesley, 1986.
- [3] Hubert Partl, *German T_EX*, TUGboat, vol 9 (1988), no 1, pp. 70–72.
- [4] Inge Eijkhout, Conversations and correspondence, 1989.
- [5] John Miles, *Ontwerpen voor Desktop Publishing, Alles over layout en typografie op de personal computer*, Uitgeverij Mingus, Baarn 1988, Dutch

³ Editor's Note: At the TUG meeting, Leslie Lamport met with a group of users "seriously interested in the future of L^AT_EX" and established a committee, chaired by Frank Mittelbach, to undertake maintenance of the L^AT_EX code and creation of L^AT_EX version 2.10 (and 3.10) (see p. 400 in this issue of *TUGboat*). It is likely that "internationalization" of the L^AT_EX code will be one of the results of this committee's work.

translation of *Design for Desktop Publishing: a guide to layout and typography on the personal computer*, Fraser, London 1987.

- [6] Stanley Morison, *Grondbeginse der Typografie*, Uitgeverij de Buitenkant, Amsterdam 1987, Dutch translation of *First Principles of Typography*.
- [7] Karel Treebus, *Tekstwijzer, een gids voor het grafisch verwerken van tekst*, SDU uitgeverij, The Hague 1988. (A book covering all aspects of typographical design. In Dutch.)

◊ Johannes Braams
PTT Research Neher Laboratories
St. Paulusstraat 4
2264 XZ Leidschendam
The Netherlands
all nets: `JL.Braams@pttrnl.nl`

◊ Victor Eijkhout
Department of Mathematics
University of Nijmegen
Toernooiveld 5
6525 ED Nijmegen
The Netherlands
Bitnet: `U641000@HNYKUN11`

◊ Nico Poppelier
University of Utrecht
Department of Physics
Postbus 80.000
3508 TA Utrecht
The Netherlands
Bitnet: `Poppelier@HUTRUU51`

An environment for multicolumn output^{*†}

Frank Mittelbach

Abstract

This article describes the use and the implementation of the `multicols` environment. This environment allows switching between one and multicolumn format on the same page. Footnotes are handled correctly (for the most part), but will be placed at the bottom of the page and not under each column. L^AT_EX's float mechanism, however, is partly disabled in the current implementation and will be added in a later version. At the moment only floats contributed outside the scope of the environment will find their way into the actual output.

1 Introduction

Switching between two column and one column layout is possible in L^AT_EX, but every use of `\twocolumn` or `\onecolumn` starts a new page. Moreover, the last page of two column output isn't balanced and this often results in an empty, or nearly empty, right column. When I started to write macros for `doc.sty` (see "The

doc-Option", *TUGboat* volume 10 #2, pp. 245–273) I thought that it would be nice to place the index on the same page as the bibliography. And balancing the last page would not only look better, it also would save space; provided of course that it is also possible to start the next article on the same page. Rewriting the index environment was compar-

atively easy, but the next goal, designing an environment which takes care of footnotes, floats etc., was a harder task. It took me a whole weekend¹ to get together the few lines of code below and there is still a good chance that I missed something after all. Try it and, hopefully, enjoy it; and *please* direct bug reports and suggestions back to Mainz.

2 The User Interface

To use the environment one simply says

```
\begin{multicols}{<number>}
  <multicolumn text>
\end{multicols}
```

where `<number>` is the required number of columns and `<multicolumn text>` may contain arbitrary L^AT_EX commands, except that floats and marginpars are not allowed in the current implementation².

As its first action, the `multicols` environment measures the cur-

rent page to determine whether there is enough room for some portion of multicolumn output. This is controlled by the `<dimen>` variable `\premulticols` which can be changed by the user with ordinary L^AT_EX commands. If the space is less than `\premulticols`, a new page is started. Otherwise, a `\vskip` of `\multicolsep` is added.³

When the end of the `multicols` environment is encountered, an analogous mechanism is employed, but now we test

whether there is a space larger than `\postmulticols` available. Again we add `\multicolsep` or start a new page.

It is often convenient to spread some text over all columns, just before the multicolumn output, without any page break in between. To achieve this the `multicols` environment has an optional second argument which can be used for this purpose. For example, the text you are now reading was started with

```
\begin{multicols}{3}
```

^{*} Editor's note: This paper, with slight modification, is the basis for Mr. Mittelbach's citation as the Donald E. Knuth Scholar at the 1989 TUG Meeting.

[†] This file has version number v1.1a, last revised 89/09/20, documentation dated 89/09/20.

¹ I started with the algorithm given in the T_EXbook on page 417. Without this help a weekend would not have been enough.

² This is dictated by lack of time. To implement floats one has to reimplement the whole L^AT_EX output routine.

³ Actually the added space may be less because we use `\addvspace` (see the L^AT_EX manual for further information about this command).

```
[\section{The User
Interface}] ...
```

If such text is unusually long (or short) the value of `\premulticols` might need adjusting to prevent a bad page break. We therefore provide a third argument which can be used to overwrite the default value of `\premulticols` just for this occasion.

Separation of columns with vertical rules is achieved by setting the parameter `\columnseprule` to some positive value. In this article a value of `.4pt` was used.

Since narrow columns tend to need adjustments in interline spacing we also provide a `(skip)` parameter called `\multicolbaselineskip` which is added to the `\baselineskip` parameter inside the `multicols` environment. Please use this parameter with care or leave it alone; it is intended only for style file designers since even small changes might produce totally unexpected changes to your document.

2.1 Balancing Columns

Besides the previously mentioned parameters, some others are provided to influence the layout of the columns generated.

Paragraphing in `TeX` is controlled by several parameters. One of the most important is called `\tolerance`: this controls the allowed ‘looseness’ (i.e. the amount of blank space between words). Its default value is 200 (the `LATEX \fussy`) which is too small for narrow columns. On the other hand the `\sloppy` declaration (which sets `\tolerance` to

$10000 = \infty$) is too large, allowing really bad spacing.⁴

We therefore use a `\multicolstolerance` parameter for the `\tolerance` value inside the `multicols` environment. Its default value is 9999 which is less than infinity but ‘bad’ enough for most paragraphs in a multicolumn environment. Changing its value should be done outside the `multicols` environment. Since `\tolerance` is set to `\multicolstolerance` at the beginning of every `multicol` environment one can locally overwrite this default by assigning `\tolerance=\langle desired\ value \rangle`.

Generation of multicolumn output can be divided into two parts. In the first part we are collecting material for a page, shipping it out, collecting material for the next page, and so on. As a second step, balancing will be done when the end of the `multicols` environment is reached. In the first step `TeX` might consider more material whilst finding the final columns than it actually use when shipping out the page. This might cause a problem if a footnote is encountered in the part of the input considered, but not used, on the current page. In this case the footnote might show up on the current page, while the footnotemark corresponding to this footnote might be set on the next one.⁵ Therefore the `multicols` environment gives a warning message⁶ whenever it is unable to use all the material considered so far.

If you don’t use footnotes too often the chances of something actually going wrong are very slim, but if this happens you can help

`TeX` by using a `\pagebreak` command in the final document. Another way to influence the behavior of `TeX` in this respect is given by the counter variable ‘`collectmore`’. If you use the `\setcounter` declaration to set this counter to `\langle number \rangle`, `TeX` will consider `\langle number \rangle` more (or less) lines before making its final decision. So a value of `-1` may solve all your problems at the cost of slightly less optimal columns.

In the second step (balancing columns) we have other bells and whistles. First of all you can say `\raggedcolumns` if you don’t want the bottom lines to be aligned. The default is `\flushcolumns`, so `TeX` will normally try to make both the top and bottom baselines of all columns align.

Additionally you can set another counter, the ‘`unbalance`’ counter, to some positive `\langle number \rangle`. This will make all but the right-most column `\langle number \rangle` of lines longer than they would normally have been. ‘Lines’ in this context refer to normal text lines (i.e. one `\baselineskip` apart); thus, if your columns contain displays, for example, you may need a higher `\langle number \rangle` to shift something from one column into another.

Unlike ‘`collectmore`’, the ‘`unbalance`’ counter is reset to zero at the end of the environment so it only applies to one `multicols` environment.

The two methods may be combined but I suggest using these features only when fine tuning important publications.

⁴ Look at the next paragraph, it was set with the `\sloppy` declaration.

⁵ The reason behind this behavior is the asynchronous character of the `TeX page-builder`. However, this could be avoided by defining very complicated output routines which don’t use `TeX` primitives like `\insert` but do everything by hand. This is clearly beyond the scope of a weekend problem.

⁶ This message will be generated even if there are no footnotes in this part of the text.

2.2 Tracing the output

To understand the reasoning behind the decisions TEX makes when processing a `multicols` environment, a tracing mechanism is provided. If you set the counter '`tracingmulticols`' to a positive $\langle\text{number}\rangle$ you then will get some tracing information on the terminal and in the transcript file:

$\langle\text{number}\rangle = 1$. TEX will now tell you, whenever it enters or leaves

a `multicols` environment, the number of columns it is working on and its decision about starting a new page before or after the environment.

$\langle\text{number}\rangle = 2$. In this case you also get information from the balancing routine: the heights tried for the left and right-most columns, information about shrinking if the `\raggedcolumns` declara-

tion is in force and the value of the 'unbalance' counter if positive.

$\langle\text{number}\rangle \geq 3$. Setting $\langle\text{number}\rangle$ to such a high value will additionally place an `\hrule` into your output, separating the part of text which had already been considered on the previous page from the rest. Clearly this setting should *not* be used for the final output.

3 The Implementation

We are now switching to two-column output to show the abilities of this environment (and bad layout decisions).

3.1 Starting and Ending the `multicols` Environment

As always we begin by identifying the latest version of this file on the VDU and in the transcript file but we abort if this file was already read in.

```
\@ifundefined{mult@cols}{}{\endinput}
\typeout{Style option: 'multicol'
  \fileversion\space <\filedate> (FMi)}
\typeout{English documentation
  @spaces\@spaces\space<\docdate> (FMi)}
```

As mentioned before, the `multicols` environment has one mandatory argument (the number of columns) and up to two optional ones. We start by reading the number of columns into the `\col@number` register.

```
\def\multicols#1{\col@number#1\relax}
```

If the user forgot the argument, TEX will complain about a missing number at this point. The error recovery mechanism will then use zero, which isn't a good choice in this case. So we should now test whether everything is okay.

```
\ifnum\col@number<\!one
  \warning{Using '\number\col@number'
    columns doesn't seem a good idea.\!J
    I therefore use two columns instead\!J}
  \col@number\two\fi
```

Now we can safely look for the optional arguments.

```
\@ifnextchar[\mult@cols{\mult@cols[]}]
```

The `\mult@cols` macro grabs the first optional argument (if any) and looks for the second one.

```
\def\mult@cols[#1]{\@ifnextchar[%
```

This argument should be a $\langle\text{dimen}\rangle$ denoting the minimum free space needed on the current page to start the environment. If the user didn't supply one, we use `\premulticols` as a default.

```
{\mult@cols{#1}%
 {\mult@cols{#1}[\premulticols]}}
```

After removing all arguments from the input we are able to start with `\mult@cols`. First we look to see if statistics are requested:

```
\def\mult@cols#1[#2]{%
 \ifnum\c@tracingmulticols>\z@
 \typeout{\!J^JStarting multicolumn
        output with \the\col@number
        \space columns:\!J\!fi}
```

Then we measure the current page to see whether a useful portion of the multicolumn environment can be typeset. This routine might start a new page.

```
\enough@room#2%
```

Now we output the first argument and produce vertical space above the columns. (Note that this argument corresponds to the first optional argument of the `multicols` environment.)

```
#1\par\addvspace\multicolsep
```

We start a new grouping level to hide all subsequent changes (done in `\prepare@multicols` for example) and finish by suppressing initial spaces.

```
\begingroup
 \prepare@multicols\ignorespaces}
```

The `\enough@room` macro used above isn't perfect but works reasonably well in this context. We measure the free space on the current page by subtract-

ing `\pagetotal` from `\pagegoal`. This isn't entirely correct since it doesn't take the 'shrinking' (i.e. `\pageshrink`) into account. The 'recent contribution list' might be nonempty so we start with `\par` and an explicit `\penalty`.⁷

```
\def\enough@room{\par \penalty\z@  
  \page@free \pagegoal  
  \advance \page@free -\pagetotal
```

Now we test whether tracing information is required:

```
\ifnum \c@tracingmulticols>\z@  
  \typeout{Current page:}\%  
  \message{\spaces goal height=%  
    \the\pagegoal: used \the\pagetotal  
    \space -> free=\the\page@free}\%  
  \typeout{\spaces needed \the\#1  
    (for \string#1)}\fi
```

Our last action is to force a page break if there isn't enough room left.

```
\ifdim \page@free <\#1\newpage \fi}
```

When preparing for multicolumn output several things must be done. First we remove everything from the 'current page' and save it in the box `\partial@page`.

```
\def\prepare@multicols{  
  \output{\global\setbox\partial@page  
    \vbox{\unvbox\@cclv}}\eject
```

Then we assign new values to `\vbadness`, `\hbadness` and `\tolerance` since it's rather hard for TeX to produce 'good' paragraphs within narrow columns.

```
\vbadness9999 \hbadness5000  
\tolerance\multicoltolerance
```

We also set the register `\doublecol@number` for later use. This register should contain $2 \times \text{\col@number}$.

```
\doublecol@number\col@number  
\multiply\doublecol@number\tw@
```

Additionally, we advance `\baselineskip` by `\multicolbaselineskip` to allow corrections for narrow columns.

```
\advance\baselineskip\multicolbaselineskip
```

The thing to do is to assign a new value to `\vsize`. TeX maintains the free room on the page (i.e. the page height without the space for already contributed floats) in the register `\@colroom`. We must

subtract the height of `\partial@page` to put the actual free room into this variable.

```
\advance\@colroom-\ht\partial@page
```

Since we have to set `\col@umber` columns on one page, each with a height of `\@colroom`, we have to assign `\vsize = \col@umber × \@colroom` in order to collect enough material before entering the `\output` routine again.

```
\vsize\col@umber\@colroom
```

But this might not be enough since we use `\vsplit` later to extract the columns from the gathered material. Therefore we add some 'extra lines,' the number depending on the value of the 'collectmore' counter.

```
\advance\vsize\c@collectmore\baselineskip
```

The `\hsize` of the columns is given by the formula:

$$\frac{\text{\columnwidth} - (\text{\col@umber} - 1) \times \text{\columnsep}}{\text{\col@umber}}$$

This will be achieved with:

```
\hsize\columnwidth \advance\hsize\columnsep  
\advance\hsize-\col@umber\columnsep  
\divide\hsize\col@umber
```

We also set `\linewidth` to `\hsize` but leave `\columnwidth` unchanged. This is inconsistent, but `\columnwidth` is used only by floats (which aren't allowed in their current implementation) and by the `\footnote` macro. Since we want pagewide footnotes⁸ this simple trick saves us from rewriting the `\footnote` macros.

```
\linewidth\hsize
```

Now we switch to a new `\output` routine which will be used to put the gathered column material together.

```
\output{\multi@columnout}\%
```

Finally we handle the footnote insertions. We have to multiply the magnification factor and the extra skip by the number of columns since each footnote reduces the space for every column (remember that we have pagewide footnotes). If, on the other hand, footnotes are typeset at the very end of the document, our scheme still works since `\count\footins` is zero then, so it will not change.

```
\multiply\count\footins\col@umber  
\multiply\skip\footins\col@umber
```

⁷ See the documentation of `\endmulticols` for further details.

⁸ I'm not sure that I really want pagewide footnotes. But balancing of the last page can only be achieved with this approach or with a multi-path algorithm which is complicated and slow. But it's a challenge to everybody to prove me wrong! Another possibility is to reimplement a small part of the `fire-up` procedure in TeX (the program). I think that this is the best solution if you are interested in complex page makeup, but it has the disadvantage that the resulting program cannot be called TeX thereafter.

For the same reason (pagewide footnotes), the `\dimen` register controlling the maximum space used for footnotes isn't changed. Having done this, we must reinsert all the footnotes which are already present (i.e. those encountered when the material saved in `\partial@page` was first processed). This will reduce the free space (i.e. `\pagetotal`) by the appropriate amount since we have changed the magnification factor, etc. above.

```
\reinsert@footnotes}
```

When the end of the `multicol`s environment is sensed we have to balance the gathered material. We end the current paragraph with `\par` but this isn't sufficient since TeX's *page_builder* will not totally empty the contribution list.⁹ Therefore we must also add an explicit `\penalty`. Now the contribution list will be emptied and, if its material doesn't all fit onto the current page then the output routine will be called before we change it.

```
\def\endmulticols{\par\penalty\z@}
```

Now it's safe to change the output routine in order to balance the columns.

```
\output{\balance@columns}\eject
```

The output routine above will take care of the `\vsize` and reinsert the balanced columns, etc. But it can't reinsert the `\footnotes` because we first have to restore the `\footins` parameter since we are returning to one column mode. This will be done in the next line of code; we simply close the group started in `\multicols`.

```
\endgroup \reinsert@footnotes
```

We also set the 'unbalance' counter to its default. This is done globally since L^AT_EX counters are always changed this way.¹⁰

```
\global\c@unbalance\z@
```

We also take a look at the amount of free space on the current page to see if it's time for a page break. The vertical space added thereafter will vanish if `\enough@room` starts a new page.

```
\enough@room\postmulticols
\addvspace\multicolssep
```

If statistics are required we finally report that we have finished everything.

```
\ifnum\c@tracingmulticols>\z@
```

⁹ This once caused a puzzling bug where some of the material was balanced twice, resulting in some overprints. The reason was the `\eject` which was placed at the end of the contribution list. Then the *page_builder* was called (an explicit `\penalty` will empty the contribution list), but the line with the `\eject` didn't fit onto the current page. It was then reconsidered after the output routine had ended, causing a second break after one line.

¹⁰ Actually, we are still in a group started by the `\begin` macro, so `\global` must be used anyway.

```
\typeout{^^JEnding multicolumn
output.^^J^^J}\fi}
```

Let us end this section by allocating all the registers used so far.

```
\newcount\c@unbalance \c@unbalance = 0
\newcount\c@collectmore \c@collectmore = 0
\newcount\c@tracingmulticols
\c@tracingmulticols = 0
\newcount\col@number
\newcount\doublecol@number
\newcount\multicoltolerance
\multicoltolerance = 9999
\newdimen\page@free
\newdimen\premulticols \premulticols = 50pt
\newdimen\postmulticols \postmulticols = 20pt
\newskip\multicolssep
\multicolssep = 12pt plus 4pt minus 3pt
\newskip\multicolbaselineskip
\multicolbaselineskip=0pt
```

We also need a box into which the "current page" can be put.

```
\newbox\partial@page
```

3.2 The output routines

We first start with some simple macros. When typesetting the page we save the columns either in the box registers 0, 2, 4,... (locally) or 1, 3, 5,... (globally). This is PLAIN TeX policy to avoid an overflow of the save stack.

Therefore we define a `\process@cols` macro to help us in using these registers in the output routines below. It has two arguments: the first one is a number; the second one is the processing information. It loops starting with `\count@=1` (`\count@` is a scratch register defined in PLAIN TeX), processes argument #2, adds two to `\count@`, processes argument #2 again, etc. until `\count@` is higher than `\doublecol@number`. It might be easier to understand it through an example, so we first define it and explain its usage afterwards.

```
\def\process@cols#1#2{\count@#1\relax
\loop #2%
\advance\count@\two@
\ifnum\count@<\doublecol@number
\repeat}
```

We now define `\pagesofar` to give an example of the `\process@cols` macro. `\pagesofar` should output everything on the ‘current page’. So we start by unboxing `\partial@page` (i.e. the part above the `multicols` environment). If the `\partial@page` is void (i.e. if the `multicols` environment started on a new page or if we typeset several pages within the `multicols` environment) this will produce nothing.

```
\def\pagesofar{\unvbox\partial@page}
```

Now we output the columns gathered assuming that they are saved in the box registers 2 (left column), 4 (second column), ... However, the last column (i.e. the right-most) should be saved in box register 0.¹¹ First we ensure that the columns have equal width. We use `\process@cols` for this purpose, starting with `\count@ = 0`. Therefore `\count@` loops through 0, 2,... (to `\doublecol@number`).

```
\process@cols\z@\{wd\count@\hsize}\%
```

Now we put all columns together in an `\hbox` of width `\textwidth`

```
\hbox to\textwidth\%
```

separating them with a rule if desired.

```
\process@cols\tw@\{ \box\count@  
    \hss\vrule\@width\columnsep\hss\} \%
```

As you will have noticed, we started with box register 2 (i.e. the left column). So this time `\count@` looped through 2, 4,... Finally we add box 0 and close the `\hbox`.

```
\box\z@\}
```

Before we tackle the bigger output routines we define just one more macro which will help us to find our way through the mysteries later. `\reinsert@footnotes` will do what its name indicates: it reinserts the footnotes present in `\footinbox` so that they will be reprocessed by `TEX`’s *page_builder*.

```
\def\reinsert@footnotes{\ifvoid\footins\else  
    \insert\footins{\unvbox\footins}\fi}
```

Now we can’t postpone the difficulties any longer. The `\multicolumnout` routine will be called in two situations. Either the page is full (i.e. we have collected enough material to generate all the required columns) or a float or marginpar is sensed. In the latter case the `\outputpenalty` is less than -10001 , otherwise the penalty which triggered the output routine is higher. Therefore it’s easy to distinguish both cases: we simply test this register.

```
\def\multi@columnout{\%  
    \ifnum\outputpenalty <-@Mi
```

If this was a float or a marginpar we call `\speci@ls`
`\speci@ls \else`

otherwise we construct the final page. Actually a `\clearpage` will be silently accepted, producing the same effects as a `\newpage`, since we didn’t distinguish between a penalty of -10000 and -10001 (produced by a `\clearpage`). Let us now consider the normal case. We have to `\vsplit` the columns from the accumulated material in box 255. Therefore we first assign appropriate values to `\splittopskip` and `\splitmaxdepth`.

```
\splittopskip\topskip  
 \splitmaxdepth\maxdepth
```

Then we calculate the current column height (in `\dimen@`). Note that the height of `\partial@page` is already subtracted from `\@colroom` so we can use its value as a starter.

```
\dimen@\@colroom
```

But we must also subtract the space occupied by footnotes on the current page. Note that we first have to reset the skip register to its normal value.

```
\divide\skip\footins\col@number  
 \ifvoid\footins\else  
     \advance\dimen@-\skip\footins  
     \advance\dimen@-\ht\footins\ fi
```

Now we are able to `\vsplit` off all but the last column. Recall that these columns should be saved in the box registers 2, 4,...

```
\process@cols\tw@\{ \setbox\count@  
    \vsplit\@ccly to\dimen@\} \%
```

Then the last column follows.

```
\setbox\z@\vsplit\@ccly to\dimen@
```

Having this done we hope that box 255 is emptied. If not, we reinsert its contents.

```
\ifvoid\@ccly\else  
    \unvbox\@ccly  
    \penalty\outputpenalty
```

If the ‘tracingmulticols’ counter is 3 or higher we also add a rule. And in any case we inform the user about our bad luck.

```
\ifnum \c@tracingmulticols>\tw@  
    \hrule\allowbreak\fi  
 \warning{I moved some lines to  
         the next page.\^J  
 \spaces Footnotes on page  
 \thepage\space might be wrong}\fi
```

¹¹ You will see the reason for this numbering when we look at the output routines `\multi@columnout` and `\balance@column`.

With a little more effort we could have done better. If we had, for example, recorded the shrinkage of the material in `\partial@page` it would be now possible to try higher values for `\dimen0` (i.e. the column height) to overcome the problem with the nonempty box 255. But this would make the code even more complex so I skipped it in the current implementation.

Now we use L^AT_EX's standard output mechanism.¹² Admittedly this is a funny way to do it.

```
\setbox\@cclv\vbox{\page@sofar}%
```

The macro `\@makecol` adds all floats assigned for the current page to this page. `\@outputpage` ships out the resulting box. Note that it is just possible that such floats are present even if we do not allow any inside a `multicols` environment.

```
\@makecol\@outputpage
```

Now we reset `\@colroom` to `\@colht` which is L^AT_EX's saved value of `\textheight`.

```
\global\@colroom\@colht
```

Then we process deferred floats waiting for their chance to be placed on the next page.

```
\process@deferreds
```

If the user is interested in statistics we inform him about the amount of space reserved for floats.

```
\ifnum\c@tracingmulticols>0
  \typeout{Colroom: \the\@colht\space
           after float space removed
           = \the\@colroom }\fi
```

Having done all this we must prepare to tackle the next page. Therefore we assign a new value to `\vsize`. New, because `\partial@page` is now empty and `\@colroom` might be reduced by the space reserved for floats.

```
\global\vsize\col@number\@colroom
\global\advance\vsize
  \c@collectmore\baselineskip
```

We also have to readjust the `\footins` skip register.

```
\multiply\skip\footins\col@number\fi}
```

We left out two macros: `\process@deferreds` and `\speci@ls`. If we encounter a float or a marginpar in the current implementation we simply warn the user that this is not allowed. Then we reinsert the page and its footnotes.

```
\def\speci@ls{%
  \typeout{Floats and marginpars not
          allowed inside 'multicols'
          environment!}%
  \unvbox\@cclv\reinsert@footnotes}
```

¹² This will produce a lot of overhead since both output routines are held in memory. The correct solution would be to redesign the whole output routine used in L^AT_EX.

Additionally we empty the `\@currlist` to avoid later error messages when the L^AT_EX output routine is again in force.

```
\gdef\@currlist{}%
```

`\process@deferreds` is a simplified version of L^AT_EX's `\@startpage`. We first call the macro `\@floatplacement` to save the current user parameters in internal registers. Then we start a new group and save the `\@deferlist` temporarily in the macro `\@tempb`.

```
\def\process@deferreds{%
  \@floatplacement
  \begingroup
  \let\@tempb\@deferlist
```

Our next action is to (globally) empty `\@deferlist` and assign a new meaning to `\@elt`. Here `\@scolelt` is a macro that looks at the boxes in a list to decide whether they should be placed on the next page (i.e. on `\@topl` or `\@botl`) or should wait for further processing.

```
\gdef\@deferlist{}%
\let\@elt\@scolelt
```

Now we call `\@tempb` which has the form

```
\@elt<box register>\@elt<box register>...
```

So `\@elt` (i.e. `\@scolelt`) will distribute the boxes to the three lists.

```
\@tempb \endgroup}
```

The `\raggedcolumns` and `\flushcolumns` declarations are defined with the help of a new `\if...` macro.

```
\newif\ifshrnking
```

The actual definitions are simple: we just switch to true or false depending on the desired action. To avoid extra spaces in the output we enclose these changes in `\@bsphack...` `\@esphack`.

```
\def\raggedcolumns{%
  \@bsphack\shrnkingtrue\@esphack}
\def\flushcolumns{%
  \@bsphack\shrnkingfalse\@esphack}
```

Now for the last part of the show: the column balancing output routine. Since this code is called with an explicit penalty (`\eject`) there is no need to check for something special. Therefore we start by assigning the values used by `\vsplit`.

```
\def\balance@columns{%
  \splittopskip\topskip
  \splitmaxdepth\maxdepth
```

Next we measure the length of the current page and at the same time save it in box register 0.

```
\setbox\z@ \vbox{\unvbox\cc1v}\dimen@ \ht\z@
```

Then we try to find a suitable starting point for the calculation of the column height. It should be less than the height finally chosen, but large enough to reach this final value in only a few iterations.

```
\advance\dimen@\col@number\topskip
\advance\dimen@-\col@number\baselineskip
\divide\dimen@\col@number
```

At the user's request we start with a higher value (or lower, but this usually only increases the number of tries).

```
\advance\dimen@\c@unbalance\baselineskip
```

We type out statistics if we were asked to do so.

```
\ifnum\c@tracingmulticols>@ne
  \typeout{Balance columns:
    \ifnum\c@unbalance=\z@\else
      (off balance=\number\c@unbalance)\fi}%
\fi
```

Now we try to find the final column height. Everything is done in a group so as to hide the changes to register contents. We start by setting `\vbadness` to infinity (i.e. 10000) to suppress underfull box reports while we are trying to find an acceptable solution.

```
{\vbadness\@M \loop
```

In order not to clutter up TeX's valuable main memory with things that are no longer needed, we empty all globally used box registers. This is necessary if we return to this point after an unsuccessful trial. We use `\process@cols` for this purpose, starting with 1. Note the extra braces around this macro call. They are needed since PLAIN TeX's `\loop...` `\repeat` mechanism cannot be nested on the same level of grouping.

```
{\process@cols@ne{\global\setbox\count@ \box\voidb@x}}%
```

The contents of box 0 are now copied globally to box 1. (This will be the right-most column, as we shall see later.)

```
\global\setbox\@ne\copy\z@
```

Using `\vsplit` we extract the other columns from box register 1. This leaves box register 0 untouched so that we can start over again if this trial was unsuccessful.

```
{\process@cols\thr@@{\global\setbox\count@ \vsplit\@ne to\dimen@}}%
```

¹³ The value of `\dimen@` may differ from the height of box register 3 when we use the `\raggedcolumns` declaration.

After `\process@cols` has done its job we have the following situation:

```
box 0 ← all material
box 3 ← first column
box 5 ← second column
:
:
box 1 ← last column
```

We report the height of the first column.

```
\ifnum\c@tracingmulticols>@ne
  \message{@spaces First column
  = \the\ht\thr@@}fi
```

If `\raggedcolumns` is in force we also shrink the first column to its natural height and optionally inform the user.

```
\ifshrink@king \global\setbox\thr@@
  \vbox{\unvbox\thr@@}%
\ifnum\c@tracingmulticols>@ne
  \message{after shrinking
  \the\ht\thr@@}fi\fi
```

Then we give information about the last column.

```
\ifnum\c@tracingmulticols>@ne
  \message{<> last column = \the\ht\@ne}%
\typeout{}fi
```

We check whether our trial was successful. The test used is very simple: we merely compare the first and the last column. Thus the intermediate columns may be longer than the first if `\raggedcolumns` is used. If the right-most column is longer than the first then we start over with a larger value for `\dimen@`.

```
\ifdim\ht\@ne >\ht\thr@@
\global\advance\dimen@\p@%
\repeat}
```

Now we save the actual height of box register 3 (i.e. the left column) in the `(dimen)` register `\dimen@` since otherwise this information will be lost when processing the code below.¹³

```
\dimen@\ht\thr@@
```

Then we move the contents of the odd-numbered box registers to the even-numbered ones, shrinking them if requested.

```
\process@cols\z@{\tempcnta\count@
  \advance\tempcnta@one
  \setbox\count@\vtop to\dimen@ {\unvbox\tempcnta
  \ifshrink@king\vfill}fi}}
```

This will bring us into the position to apply `\page@sofar`. But first we have to set `\vsize` to a value suitable for one column output.

```
\global\vsize\@colroom
```

```
\global\advance\vsize\ht\partial@page
\page@sofar}
```

As we already know, reinserting of footnotes will be done in the macro `\endmulticols`.

Index

Italic numbers denote the pages where the corresponding entry is described, underlined numbers point to the definition, all others indicate the places where it is used.

Symbols	E	F	N	P	R
<code>\@colroom</code> ... 410, 412, 413, 415	<code>\endmulticols</code> ... <u>411</u> <code>\enough@room</code> <u>409</u> , 409, 411	<code>\flushcolumns</code> ... <u>413</u> <code>\footins</code> 410, 412, 413 <code>\ifshrnking</code> ... <u>413</u>	<code>\newif</code> ... 413	<code>\page@free</code> 410, <u>411</u> , 411 <code>\page@sofar</code> <u>412</u> , 413, 415	<code>\raggedcolumns</code> ... <u>413</u> <code>\reinsert@footnotes</code> ... 411, <u>412</u> , 413
B	H	O	P	S	
<code>\balance@columns</code> ... 411, <u>413</u>	<code>\hbadness</code> ... 410 <code>\hspace</code> ... 410, 412	<code>\output</code> ... 410, 411	<code>\pagegoal</code> ... 410 <code>\pagetotal</code> ... 410 <code>\par</code> ... 409–411 <code>\partial@page</code> 410, 411, <u>411</u> , 412, 415 <code>\postmulticols</code> ... 407, 411, <u>411</u> , 411	<code>\shrnkingfalse</code> ... 413 <code>\shrnkingtrue</code> ... 413 <code>\speci@ls</code> ... 412, <u>413</u>	<code>\textwidth</code> ... 412 <code>\tolerance</code> ... 410
C	I	M	V		
<code>\c@collectmore</code> 410, 411, <u>411</u> , 411, 413	<code>\ifshrnking</code> ... 413	<code>\mult@cols</code> ... 409, <u>409</u> <code>\mult@cols</code> ... 409, <u>409</u> <code>\multi@columnout</code> ... 410, 412	<code>\vbadness</code> ... 410, 414 <code>\vsize</code> ... 410, 413, 415		
D					
<code>\doublecol@number</code> ... 410, 411, <u>411</u>					

◊ Frank Mittelbach
 Electronic Data Systems
 (Deutschland) GmbH
 Eisenstraße 56
 D-6090 Rüsselsheim
 Federal Republic of Germany
 Bitnet: pzf5hz@drueds2

An Extension of the L^AT_EX theorem environment*

Frank Mittelbach

Abstract

The macros described in this paper yield an extension of the L^AT_EX theorem mechanism. It is designed to satisfy the different requirements of various journals. Thus, the layout of the “theorems” can be manipulated by determining a “style”. This article describes not only the use, but also the definition, of the necessary macros.

1 Introduction

For our purposes here, “theorems” are labelled enunciations, often set off from the main text by extra space and a font change. Theorems, corollaries, conjectures, definitions, and remarks are all instances of “theorems”. The “header” of these structures is composed of a label (such as THEOREM or REMARK) and a number which serializes an item in the sequence of items with the same label.

Shortly after the introduction of L^AT_EX at the Fachbereich Mathematik in Mainz, the desire to manipulate the layout of “theorems” arose. In Mainz, the following two conventions came into general use:

1. The number of the theorem is shown in the margin.
2. There is a line break at the end of the theorem header.

Additionally, some journals require different formats which depend on the “sort of theorem”: e.g. often remarks and definitions are set in \rm, while \it is employed for main theorems.

Confronted with these requirements, a theorem environment was developed in Mainz which allows separate determination of the layout of the “theorems sets”, comparable to \pagestyle.

2 The user interface

2.1 Defining new theorem sets

\newtheorem As in the original L^AT_EX version, the command \newtheorem defines a new “theorem set” or “theorem-like structure”. Two required arguments name the new environment and give the text to be typeset with each instance of the new “set”, while an optional argument determines how the “set” is enumerated:

\newtheorem{foo}{bar} The theorem set **foo** (whose name is **bar**) uses its own counter.

\newtheorem{foo2}[foo]{bar2} The theorem set **foo2** (printed name **bar2**) uses the same counter as the theorem set **foo**.

\newtheorem{foo3}[bar3][section] The theorem set **foo3** (printed name **bar3**) is enumerated within the counter **section**, i.e. with every new \section the enumeration begins again with 1, and the enumeration is composed from the section-number and the theorem counter itself.

\theoremstyle Additionally, the command \theoremstyle can define the layout of various, or all, theorem sets. It should be noted that any theorem set defined by \newtheorem is typeset in the \theoremstyle that is current at the time of the definition. Thus, the following

\theoremstyle{break}
\theoremstyle{plain}

\newtheorem{Cor}{Corollary}
\newtheorem{Exa}{Example}[section]

* This file has version number v2.0g, last revised 89/09/19, documentation dated 89/09/19.

leads to the result that the set `Cor` is formatted in the style `break`, while the set `Exa` and all the following ones are formatted in the style `plain`, unless another `\theoremstyle` follows. Since the definitions installed by `\newtheorem` are global, one also can limit `\theoremstyle` locally by grouping braces.

`\theorembodyfont` The choice of the font for the theorem body is completely independent of the chosen `\theoremstyle`; this has proven to be very advantageous. For example,

```
{\theorembodyfont{\rm} \newtheorem{Rem}{Remark}}
```

defines a theorem set `Rem`, which will be set in `\rm` in the current layout (which in our example is `plain`). As with `\theoremstyle`, the `\theorembodyfont` chosen is that current at the time of `\newtheorem`. If `\theorembodyfont` is not specified or one defines `\theorembodyfont{}`, then the font used will be that defined by the `\theoremstyle`.

`\theoremheaderfont` It is also possible to customize the font used for the theorem headers. This is, however, a global declaration, and therefore there should be at most one `\theoremheaderfont` declaration in the preamble.¹

`\theorempreskipamount` Two additional parameters affect the vertical space around the theorem environments: `\theorempreskipamount` and `\theorempostskipamount` define, respectively, the spacing before and after such an environment. These parameters apply for all theorem sets and can be manipulated with the ordinary length macros. They are rubber lengths, ('skips'), and therefore can contain plus and minus parts.

Since the definition of theorem sets should—most sensibly—be placed in the preamble, we only allow installation there. It is therefore possible to release the memory used here after `\begin{document}`, in order to make room for other applications.

2.2 Existing theorem styles

The following theorem styles exist to date:

`plain` This theorem style emulates the original L^AT_EX definition, except that additionally the parameters `\theorem...skipamount` are used.

`break` In this style, the theorem header is followed by a line break.

`marginbreak` The theorem number is set in the margin, and there is a line break as in `break`.

`changebreak` Like `break`, but with header number and text interchanged.

`change` Header number and text are interchanged, without a line break.

`margin` The number is set in the left margin, without a line break.

All styles (except `plain`) select `\sl` as the default `\theorembodyfont`.

2.3 Examples

Given the above theorem sets `Cor`, `Exa` and `Rem`, suppose that the preamble also contains the declarations:

```
\theoremstyle{marginbreak} \newtheorem{Lem}[Cor]{Lemma}
\theoremstyle{change} \newtheorem{Def}[Cor]{Definition}
\theorembodyfont{\it}
\theoremheaderfont{\sc}
```

Then the following are some typical examples of the typeset output resulting from their use.

COROLLARY 1

This is a sentence typeset in the theorem environment Cor.

¹ If it is actually necessary to have different header fonts, one has to define new theorem styles (substituting the desired font) or specify the information directly in the `\newtheorem` declaration (the unclean variant).

EXAMPLE 2.1 *This is a sentence typeset in the theorem environment Exa.*

REMARK 1 This is a sentence typeset in the theorem environment Rem.

2 LEMMA (BEN USER)

This is a sentence typeset in the theorem environment Lem.

3 DEFINITION (VERY IMPRESSIVE DEFINITION) *This is a sentence typeset in the theorem environment Def.*

The last two examples show the effect of the optional argument to a theorem environment (it is the text typeset in parentheses).

3 Acknowledgements

The publication of this set of macros was only possible with the help of Christina Busse (translating the manuscript into English), Joachim Pense (playing the rôle of typist), Chris Rowley (looking everything over) and many others providing useful suggestions.

4 Definition of the Macros

If the file has been loaded before, we abort immediately. If not, the current version of the style is shown on the screen and in the transcript file.

```
\@ifundefined{theorem@style}{}{\endinput}
\typeout{Style option: 'theorem' \fileversion \space\space
        <\filedate> (FMi)}
\typeout{English documentation as of \space\space\space
        <\docdate> (FMi)}
```

4.1 Definition of theorem styles and fonts

All the definitions in this file are done globally to allow inputting this file inside a group.

\theoremstyle Before a theorem style can be installed, the chosen style must be known. For that reason, we must test to see that `\th@{style}` is known or, more precisely, that it is different from `\relax`. If the style is not known then `\th@plain` is used.

```
\gdef\theoremstyle#1{%
  \@ifundefined{th@#1}{\@warning
    {Unknown theoremstyle '#1'. Using 'plain'}%
  \theorem@style{plain}}%
```

We save the theorem style to be used in the token register `\theorem@style`.

```
{\theorem@style{#1}}%
```

Now we “evaluate” the theorem style: this means, we call the macro `\th@{style}` which will activate the relevant definitions which are contained in a separate file.

```
\csname th@\the\theorem@style \endcsname
```

To save memory, we set two other macros to `\relax`, because they possibly are defined in the last line. (This case occurs if and only if `\th@{style}` is no longer defined as `\input....`)

```
\let\@begintheorem\relax
\let\@opargbegintheorem\relax
```

\theorem@style Obviously the token register used above has to be allocated. To assure the utmost compatibility with the original L^AT_EX definition, we set the default theorem style to `plain`, which implements the usual L^AT_EX convention.

```
\newtoks\theorem@style
\global\theorem@style{plain}
```

`\theorembodyfont` For the theorem font, we simply use a token register, whose contents can be inserted into the definition of the theorem set.

```
\newtoks\theorembodyfont
\global\theorembodyfont{}
```

`\theoremheaderfont` The font for the theorem headers is handled differently because this definition applies to all theorem styles.

```
\gdef\theoremheaderfont#1{\gdef\theorem@headerfont{#1}%
```

After using the macro once it is redefined to produce an error message.

```
\gdef\theoremheaderfont##1{%
  \typeout{\string\theoremheaderfont\space should be used
          only once.}}
\global\let\theorem@headerfont=\bf
```

`\th@plain` The different styles are defined in macros such as `\th@plain`. Since memory space is precious in “non-Big-versions”, we have to avoid offering too many unused definitions. Therefore we define these styles in separate files that can be loaded on demand. Thus the commands themselves only load these files.

```
\begin{array}{ll}
\@begintheorem & \th@plain\\
\@opargbegintheorem & \gdef\th@plain{\input thp.sty}\\
\@th@break & \gdef\th@break{\input thb.sty}\\
\th@marginbreak & \gdef\th@marginbreak{\input thmb.sty}\\
\th@changebreak & \gdef\th@changebreak{\input thcb.sty}\\
\th@margin & \gdef\th@change{\input thc.sty}\\
& \gdef\th@margin{\input thm.sty}
\end{array}
```

This list will be expanded when new styles become available. For testing, just append new theorem substyles as document options.

4.2 Definition of a new theorem set

As already pointed out, a new theorem environment can be defined in three different ways:

```
\newtheorem{Lem}{Lemma}
\newtheorem{Lem}{Lemma}[section]
\newtheorem{Lem}[Theorem]{Lemma}
```

The function of the macro `\newtheorem` is to recognize these cases and then to branch into one of the three macros `\@ynthm`, `\@xnthm` or `\@othm`. This mechanism is adopted unchanged from [1]; the essential point here is that, for example, in the second case, the arguments `Lem`, `Lemma` and `section` are passed over to the macro `\@xnthm`.

We inspect this case first because the others present fewer problems, and thus are easily derived from this one.

`\@xnthm` For our example arguments, the macro `\@xnthm` must fulfill the following:

- Define a new L^AT_EX-counter ‘Lem’
- reset this counter within a `\section`
- define the macro `\theLem`
- define the environment macros `\Lem` and `\endLem` using the current `\theoremstyle` and `\theorembodyfont`.

Obviously, all this should happen only if the first argument of `\@xnthm` (i.e. `Lem` in our example) is chosen so as not to conflict with any previously defined commands or environments. This test is performed by the L^AT_EX macro `\@ifdefinable`.

```
\gdef\@xnthm#1#2[#3]{\expandafter\@ifdefinable\csname #1\endcsname
```

Therefore, the first argument of `\@ifdefinable` is the expansion (in the example, `\Lem`) of `\csname#1\endcsname`. The second argument is executed only if the test has been completed successfully.

{%

Now we define the new counter. The names of the L^AT_EX macros employed should speak for themselves:

```
\@definecounter{#1}\@addtoreset{#1}{#3}%
```

In defining “`\theLem`” we must generate the desired macro name by use of `\expandafter` and `\csname`.

```
\expandafter\xdef\csname the#1\endcsname
```

An `\xdef` is used in order to make the definition global, and to ensure that it contains the replacement texts of `\@thmcOUNTERsep` and `\@thmcOUNTER`.² However, not everything should be expanded. For example, it saves space to use `\thesection` instead of its—at times—lengthy expansion.

```
{\expandafter\noexpand\csname the#3\endcsname
\@thmcOUNTERsep\@thmcOUNTER{#1}}%
```

Thus with the defaults of L^AT_EX, `\theLem` would be replaced by the command sequence `\thesection.\arabic{Lem}`.

We will now look at the definition of the macro which is executed at the beginning of the actual environment (in our example this macro is `\Lem`). It should be noted that we use an “`\expandafter` trick” to expand only certain parts of the replacement text at the time of the definition.

```
\def\@tempa{\global\@namedef{#1}%
\expandafter\@tempa \expandafter{%
```

First, the macro that contains the current definitions of `\@begintheorem` and `\@opargtheorem` should be called up. The name of this macro—as is already known—has the form `\th@{theorem style}`; therefore, it must be called by

```
\csname th@\the\@theorem@style
```

In addition the default theorem font should be changeable, i.e. we have to insert the contents of `\theorembodyfont`. For that reason, we expand even further, beyond `\endcsname`, and thus insert the contents of the token register `\theorembodyfont` in the replacement text.

```
\expandafter\endcsname\the\@theorembodyfont
```

Now it is time to call the macro `\@thm` which takes over the further processing. It has two arguments: the current counter name (in our example, `Lem`), and the text of the label (in our example, `Lemma`).

```
\@thm{#1}{#2}{}%
```

With this, the ‘sub-definition’ is complete. The macro `\@endtheorem` ends a theorem environment and is, so far, nothing but an `\endtrivlist`. (Hence it is defined globally, and not within the theorem styles.³) Therefore, we can set it equivalent to the macro that ends the theorem set (in our example, `\endLem`). However, if some day theorem styles exist that do change `\@endtheorem`, we would have to use the commented-out line instead.

```
\global\expandafter\let\csname end#1\endcsname\@endtheorem
% \global\@namedef{end#1}{\@endtheorem}%
```

² These two macros can be defined by the document style. Their default values produce a ‘.’ as separation and an arabic representation of the number.

³ This has to be changed as soon as theorem styles that change `\@endtheorem` exist. In such a case, all existing styles must be changed as well since they will have to reset the macro.

With these commands all the required definitions are employed, unless the test `\@ifdefinable` has failed. Therefore, we end the second argument of this macro and with it the definition of `\@xnthm`.

`}}`

- `\@ynthm` The definition of `\@ynthm` is completely analogous. In this case the new counter that is defined is not reset within another counter; thus the definition of `\the...` is simplified:

```
\gdef\@ynthm#1#2{\expandafter\@ifdefinable\csname #1\endcsname
  {\@definecounter{#1}%
   \expandafter\xdef\csname the#1\endcsname{\@thmcnter{#1}}%
```

The rest of the definition corresponds literally to that of `\@xnthm`:

```
\def\@tempa{\global\@namedef{#1}}\expandafter\@tempa
  \expandafter{\csname th@\the\theorem@style\expandafter
  \endcsname\the\theorembodyfont\@thm{#1}{#2}}%
% \global\@namedef{end#1}{\@endtheorem}%
\global\expandafter\let\csname end#1\endcsname\@endtheorem}
```

- `\@othm` The definition of `\@othm` does not contain anything new. We do not define a new counter but instead use one that has already been defined. Thus the only definition we need is that of this pseudo-counter (i.e. `\the(env. name)`).

```
\gdef\@othm#1[#2]#3{\expandafter\@ifdefinable\csname #1\endcsname
  {\expandafter\xdef\csname the#1\endcsname
   {\expandafter\noexpand\csname the#2\endcsname}}%
```

All other parts of the definition can be adopted from `\@xnthm`. We have to remember, though, that in this case the name of the current counter and the theorem label have moved to the second and third arguments.

```
\def\@tempa{\global\@namedef{#1}}\expandafter\@tempa
  \expandafter{\csname th@\the\theorem@style\expandafter
  \endcsname\the\theorembodyfont\@thm{#2}{#3}}%
% \global\@namedef{end#1}{\@endtheorem}%
\global\expandafter\let\csname end#1\endcsname\@endtheorem}
```

4.3 Macros that are employed in a theorem environment

- `\@thm` The macro `\@thm` has to increase the current counter. Then, depending on whether the environment has (or does not have) an optional argument, it has to branch into either `\@begintheorem` or `\@opargtheorem`.

```
\gdef\@thm#1#2{\refstepcounter{#1}}%
```

Now we start a `trivlist` environment, and give `\@topsep` and `\@topsepadd` the values of the skip registers `\theorempreskipamount` and `\theorempostskipamount`. The value in `\@topsep` is the vertical space that is inserted by the first (and only) `\item` in our `trivlist` whilst `\@topsepadd` is inserted by `\@endparenv` at the end of that `trivlist` environment. By using these registers, we obtain the desired space around a `theorem` environment.

```
\trivlist
\@topsep \theorempreskipamount % used by first \item
\@topsepadd \theorempostskipamount % used by \@endparenv
```

Now we have to test whether an optional argument has been given.

```
\@ifnextchar [%
```

If there is an optional argument, we will call `\@ythm`, and move the arguments read back into the input stream.

```
{\@ythm{#1}{#2}}%
```

If not, we call `\@begintheorem`. Its first argument is the name of the theorem set (hence the second argument of `\@thm`). Its second argument is the macro that produces the current number.

```
{\@begintheorem{#2}{\csname the#1\endcsname}\ignorespaces}}
```

`\@xthm` Both these macros were originally called by `\@thm`. We do not need `\@xthm` anymore, hence we reset it to `\relax`. The definition of `\@ythm` has not changed at all from its definition in L^AT_EX. In order to make the macros easier to understand, we will nevertheless present it (commented out).

```
\global\let\@xthm\relax
% \def\@ythm#1#2[#3]{\opargbegintheorem{#2}{\csname
%           the#1\endcsname}{#3}\ignorespaces}
```

The primitive `\ignorespaces` in `\@ythm` and `\@thm` is needed to remove the spaces between the `\begin{...}` and the actual text.

4.4 Definition of the theorem substyles

As already pointed out, the theorem substyles, defined below, are only loaded when necessary. Note that all these substyles, except `plain`, have `\sl` as the default body font.

4.4.1 The plain style

This style option is stored in the file `thp.sty`. As the following macros use `\@`, we have to locally set the `\catcode` of this symbol to "letter". This happens within a group, so that we do not have to worry about which `\catcode` that symbol had before.

```
\begingroup \makeatletter
```

Since we are now within a group, we must make all definitions globally. First we make sure that `theorem.sty` is loaded. This will allow us to use this file as a document style option without having to call `theorem` itself as an option. At the same time, we assure that at least version 2 is loaded, since `\theorem@style` was not defined in earlier versions.

```
\@ifundefined{theorem@style}{\input{theorem.sty}}{}
```

Now we show the version⁴ of this file:

```
\typeout{Style option: 'theorem-plain' \fileversion \space\space
       <\filedate> (FMi)}
\typeout{English documentation \spaces\spaces\spaces\space\space
       \space <\docdate> (FMi)}
```

`\th@plain` `\theoremstyle{plain}` corresponds to the original definition, except that the distances to the surrounding text are determined by the parameters `\theorempreskipamount` and `\theorempostskipamount`. First we set the default body font.

```
\gdef\th@plain{\it
```

Then we define `\@begintheorem` and `\opargbegintheorem`. These two macros define how the header of a theorem is typeset. `\opargbegintheorem` will be called if a theorem environment with an optional argument is encountered; otherwise, the header is constructed by calling `\@begintheorem`. If one of these macros is executed, we are within a `trivlist` environment started by `\@thm`. So the theorem header is produced with an `\item` command.

Instead of specifying the header font directly, all standard theorem styles use the `\theorem@headerfont` macro to allow customization. The extra space (`\labelsep`) is necessary because of problems in the `trivlist` environment.

```
\def\@begintheorem##1##2{%
  \item[\hskip\labelsep \theorem@headerfont ##1\ ##2]}%
```

⁴ This file has version number v2.1a, last revised 89/09/18, documentation dated 89/09/19.

The definition of `\@opargbegintheorem` is completely analogous. The only difference is the fact that there exists a third argument (which is the optional parameter of the environment and contains additional information about the theorem). Customarily we enclose it in parentheses.

```
\def\@opargbegintheorem##1##2##3{%
  \item[\hskip\labelsep \theorem@headerfont ##1\ ##2\ (###3)]}
```

We conclude with an `\endgroup` to restore the `\catcode` of `\@`.

```
\endgroup
```

4.4.2 The break style

This style option is stored in the file `thb.sty`. For the next two lines see the documentation for `\th@plain` on page 422.

```
\begingroup \makeatletter
\@ifundefined{theorem@style}{\input{theorem.sty}}{}
```

Now we show the version⁵ of this file:

```
\typeout{Style-Option: 'theorem-break' \fileversion \space\space
       <\filedate> (FMi)}
\typeout{English Dokumentation \@spaces\@spaces\@spaces\space\space
       \space <\docdate> (FMi)}
```

`\th@break` `\theoremstyle{break}` produces a line break after the name of the theorem. The font is `\sl`. Hence, we define `\th@break` as follows:

```
\gdef\th@break{\sl
\def\@begintheorem##1##2{\item[%
```

We run into the following problem: it is not possible to create the header with `\item[(title)]` and then start a new line by, for example, `\mbox{}\\`. Such a definition will fail whenever a list environment follows immediately. With the above construction, the `\mbox{}` causes the switch `\@inlabel` (cf. definition of `\list` and `\trivlist` in [1]) to be set to `false` and so the following list will insert additional vertical space (`\topskip`). This is quite annoying. Therefore, we create the line break within the `\item`. In order to ensure that the text will begin at the proper position in the following line, we simply pretend that the label does not take any room.⁶

```
\rlap{\vbox{\hbox{\hskip\labelsep\theorem@headerfont ##1\ ##2\%
\hbox{\strut}}}}}%
```

Again, the definition of `\@opargbegintheorem` is completely analogous.

```
\def\@opargbegintheorem##1##2##3{%
  \item[\rlap{\vbox{\hbox{\hskip\labelsep\theorem@headerfont
    ##1\ ##2\ (###3)\%
    \hbox{\strut}}}}}]}
\endgroup
```

4.4.3 The changebreak style

This style option is stored in the file `thcb.sty`.

```
\begingroup \makeatletter
\@ifundefined{theorem@style}{\input{theorem.sty}}{}
```

⁵ This file has version number v2.1a, last revised 89/09/18, documentation dated 89/09/19.

⁶ This will lead to problems whenever very high symbols occurring in the line tower into the heading. So, something else has to be done here sometime.

We show the version⁷ of this file:

```
\typeout{Style option: 'theorem-change-break' \fileversion \space\space
       <\filedate> (FMi)}
\typeout{English documentation \@spaces\@spaces\@spaces\@spaces
       \@spaces\space\space <\docdate> (FMi)}
```

- \th@changebreak** The change-break theorem style is like break but with interchange of theorem name and theorem number. Thus we define \th@changebreak as follows:

```
\gdef\th@changebreak{\sl
  \def\@begintheorem##1##2{\item
    [\rlap{\vbox{\hbox{\hspace{\labelsep \theorem@headerfont ##2\ ##1}}%
      \hbox{\strut}}}}]}
  \def\@opargbegintheorem##1##2##3{%
    \item[\rlap{\vbox{\hbox{\hspace{\labelsep \theorem@headerfont
      ##2\ ##1\ (#3)}}%
      \hbox{\strut}}}}]
  \endgroup}
```

4.4.4 The change style

This style option is stored in the file thc.sty. After startup, we show the version⁸ of this file:

```
\begingroup \makeatletter
\@ifundefined{theorem@style}{\input{theorem.sty}}{}

\typeout{Style option: 'theorem-change' \fileversion \space\space
       <\filedate> (FMi)}
\typeout{English documentation \@spaces\@spaces\@spaces\space\space
       \space\space <\docdate> (FMi)}
```

- \th@change** The change theorem style corresponds to the change break style without a linebreak after the header. To say it in another way, it's the same as the plain style but with number and name interchanged and \sl as the default font.

```
\gdef\th@change{\sl
  \def\@begintheorem##1##2{\item
    [\hspace{\labelsep \theorem@headerfont ##2\ ##1}]%
  \def\@opargbegintheorem##1##2##3{%
    \item[\hspace{\labelsep \theorem@headerfont ##2\ ##1\ (#3)}]}
  \endgroup}
```

4.4.5 The marginbreak style

This style option is the one used most often at Mainz. It is saved in the file thmb.sty. After startup we show the version⁹ of this file:

```
\begingroup \makeatletter
\@ifundefined{theorem@style}{\input{theorem.sty}}{}

\typeout{Style option: 'theorem-margin-break' \fileversion \space\space
       <\filedate> (FMi)}
\typeout{English documentation \@spaces\@spaces\@spaces\@spaces
       \@spaces\space\space <\docdate> (FMi)}
```

⁷ This file has version number v2.1a, last revised 89/09/18, documentation dated 89/09/18.

⁸ This file has version number v1.1a, last revised 89/09/18, documentation dated 89/09/18.

⁹ This file has version number v2.1a, last revised 89/09/18, documentation dated 89/09/18.

\th@marginbreak The margin break style is nearly the same as the change break style. The only difference is the placement of the theorem number. We use \rlap to place it in the left margin.

In this style \labelsep denotes the separation between the number and the text.

```
\gdef\th@marginbreak{\sl
  \def\@begintheorem##1##2{\item
    [\rlap{\vbox{\theorem@headerfont
      \hbox{\llap{##2}\hskip\labelsep ##1}%
      \hbox{\strut}}}}]%
  \def\@opargbegintheorem##1##2##3{%
    \item[\rlap{\vbox{\theorem@headerfont
      \hbox{\llap{##2}\hskip\labelsep ##1\ (###3)}%
      \hbox{\strut}}}}]%
  \endgroup}
```

4.4.6 The margin style

This style option is stored in the file thm.sty. After startup we show the version¹⁰ of this file:

```
\begingroup \makeatletter
\ifundefined{theorem@style}{\input{theorem.sty}}{}

\typeout{Style option: 'theorem-margin' \fileversion \space\space
<\filedate> (FMi)}
\typeout{English documentation \spaces\spaces\space\space\space
\space\space <\docdate> (FMi)}
```

\th@margin Again this is only a variant of the theorem styles described above without any new ideas.

```
\gdef\th@margin{\sl
  \def\@begintheorem##1##2{\item
    [\theorem@headerfont \llap{##2}\hskip\labelsep ##1]%
  \def\@opargbegintheorem##1##2##3{%
    \item[\theorem@headerfont \llap{##2}\hskip\labelsep ##1\ (###3)]}
  \endgroup}
```

4.5 Final Definitions

\theorempreskipamount **\theorempostskipamount** The skip parameters that regulate the vertical empty space before and after the theorem environment have to be allocated as well.

```
\newskip\theorempreskipamount
\newskip\theorempostskipamount
```

Since we have used the same values for all theorem sets, we now can assign them.

```
\global\setlength\theorempreskipamount{12pt plus 5pt minus 3pt}
\global\setlength\theorempostskipamount{8pt plus 3pt minus 1.5pt}
```

\@endtheorem The same holds for the macro \@endtheorem, which ends a theorem environment. Since it is the same for all theorem sets, it is removed from the macros \th@{style}. It simply ends the trivlist environment, which was begun in \@thm.

```
\global\let\@endtheorem=\endtrivlist
```

\@preamblecmds All macros defined above are to be used only in the preamble. Therefore, we insert them in \@preamblecmds (separated by the macro \do). All the commands not allowed after the \begin{document} are stored here. This is achieved by having \document

¹⁰ This file has version number v2.1a, last revised 89/09/18, documentation dated 89/09/18.

(with the help of \do) redefine all macros in \Opreamblecmds to a L^AT_EX error routine call.

```
\def\do{\noexpand\do\noexpand}
\xdef\Opreamblecmds{\Opreamblecmds \do\Oxnthm \do\Oynthm \do\Oothm
    \do\newtheorem \do\theoremstyle \do\theorembbodyfont
    \do\theoremheaderfont}
}
```

References

- [1] LAMPORT, LESLIE. *latex.tex*, version 2.09, date 13 June 1989.

Index

Italic numbers denote the pages where the corresponding entry is described, underlined numbers point to the definition, all others indicate the places where it is used.

Symbols		
\@addtoreset	420	\@xnthm 419, 426
\@begintheorem 418, <u>419</u> , 422-425	\@xthm 422
\@definecounter 420, 421		\@ynthm 421, 426
\@endtheorem 420, 421, <u>425</u>	\@ythm 421, <u>422</u>
\@opargbegintheorem 418, <u>419</u> , 422-425	N
\@othm 421, 426		\newskip 425
\@preamblecmds ... 425		\newtheorem ... 416, 426
\@thm 420, <u>421</u> , 421		\newtoks 418, 419
\@thmcounter .. 420, 421		T
\@thmcOUNTERsep ... 420		\th@break 419, 423
\@topsep 421		\th@change ... 419, <u>424</u>
\@topsepadd 421		\th@changebreak 419, <u>424</u>
		\th@margin ... 419, <u>425</u>
		\th@marginbreak 419, <u>425</u>
		◇ Frank Mittelbach
		Electronic Data Systems
		(Deutschland) GmbH
		Eisenstraße 56
		D-6090 Rüsselsheim
		Federal Republic of Germany
		Bitnet: pzf5hz@drueds2

◇ Frank Mittelbach
 Electronic Data Systems
 (Deutschland) GmbH
 Eisenstraße 56
 D-6090 Rüsselsheim
 Federal Republic of Germany
 Bitnet: pzf5hz@drueds2

Abstracts

Deutsche Kurzfassungen der TUGboat-Artikel [German Abstracts of TUGboat Articles]

Luzia Dietsche

Leitfaden für TUGboat Autoren (Ron Whitney und Barbara Beeton, S. 378)

Die Editoren der Zeitung stellen in diesem Artikel die Leitlinien vor, nach welchen Manuskripte für *TUGboat* vorbereitet sein sollten. Ursprünglich wurde *TUGboat* mit einem Makropaket, das auf plain basiert, gesetzt. Später kam eine Stylefile-option für L^AT_EX-Benutzer dazu. Für die Benutzung der auf plain-T_EX aufbauenden Makros benötigt man die zwei Files *tugbot.sty* und *tugbot.com*. Für die L^AT_EX-Makros benötigt man *ltugbot.sty* und wiederum *tugbot.com*. Dieser File enthält die meisten der gebräuchlichen Abkürzungen und kann bei Bedarf aufgestockt werden. Die Eingabe von Manuskripten sollte auch für Außenstehende übersichtlich und lesbar gestaltet werden.

Die ersten holländischen T_EX-Tage (Victor Eijkhout und Nico Poppelier, S. 316)

Am 29. und 30. Juni diesen Jahres fanden im Rechenzentrum der Universität Utrecht die ersten holländischen T_EX-Tage statt. Der erste Tag war L^AT_EX-Kursen für Anfänger und Fortgeschrittene vorbehalten, am zweiten Tag folgten Vorträge und eine Ausstellung von Firmen. In der Eröffnung sprach Kees van der Laan (1. Vorsitzender) von der Entstehung, Entwicklung und den Plänen der holländischen T_EX Gruppe NTG. Dem folgte ein Vortrag von Malcolm Clark, europäischer Koordinator der TUG, über T_EX in Europa und T_EX in der Zukunft. Außerdem wurden Vorträge über SGML und T_EX, DTP und T_EX, METAFONT, T_EX's Trennalgorithmus, "einfaches" T_EX, Folien mit L^AT_EX und T_EX für die holländische Sprache gehalten.

Benutzerleitfaden für LaTeX-help (Max Hailperin, S. 360)

Es kommt immer wieder vor, daß L^AT_EX-Benutzer für ihre Fragen, die sie trotz Ausschöpfen aller vorhandenen Quellen nicht lösen konnten, keinen Fachmann in der Nähe haben. Für solche (einfachen) Fragen wurde in den USA eine e-mail Adresse von einer Handvoll Freiwilligen einge-

richtet, an die sich die Betroffenen wenden können. Gedacht ist dieses Angebot, um Listen wie T_EXhax oder UKT_EX zu entlasten. Es funktioniert nur solange, wie es nicht mißbraucht bzw. über Gebühr beansprucht wird.

Eine Umgebung für mehrspaltige Ausgabe (Frank Mittelbach, S. 407)

Der Artikel beschreibt die Benutzung und Einbindung der *multicols* Umgebung. Diese Umgebung erlaubt den Wechsel zwischen ein- und mehrspaltigem Format innerhalb einer Seite. Fußnoten werden korrekt behandelt. Allerdings werden sie an das Ende der Seite und nicht unter die jeweilige Spalte gesetzt. In der aktuellen Version ist der "float" Mechanismus noch etwas vernachlässigt (er wird in einer späteren Version verbessert werden).

T_EX und Sprachen mit lateinischem Alphabet

(Yannis Haralambous, S. 342)

T_EX stellt für all diejenigen, denen die in einem Font vorhandenen 128 Zeichen nicht ausreichen, weitere 128 leere Stellen in 256-code Fonts zur freien Besetzung zur Verfügung. Der Autor hat nun in diesem Artikel alle (ihm bekannten) Buchstaben aus Sprachen mit lateinischem Alphabet zusammenge stellt und kommt zu dem Schluß, daß es 182 Buchstaben gibt, die bisher nicht berücksichtigt wurden. Im folgenden macht er Vorschläge, wie man diese fehlenden Zeichen sinnvoll in T_EX, bzw. die Code Tabellen einarbeiten könnte.

Anmerkungen zu "Russian T_EX" (Dimitri Vulis, S. 332)

Dimitri Vulis hat, während er die neuen cyrillischen Fonts der Universität von Washington mit seinen Trennmustern verband, eine Version von T_EX für russische Sprachen entwickelt. Er stützt sich dabei auf eine 8-bit-Codierung in seinem ASCII Computer (7-bit-Codierung ist ebenfalls vorhanden). Die Trennmuster hat er hauptsächlich von Hand gemacht, erhielt später allerdings auch einen Trennalgorithmus für russische Texte. In den Washingtoner Fonts wurde der Ligatur-Mechanismus von T_EX verwendet. Eine Transliteration von lateinischen zu cyrillischen Fonts ist ohne Schwierigkeit möglich. Da es im russischen Alphabet 32 Buchstaben gibt, mußte der Autor die \uccode und \lccode Tabellen erweitern. Er verwendet außerdem einen Präprozessor, um "Russian T_EX" in T_EX einzubinden. Dieser Präprozessor wird jedoch mit T_EX 3.0 überflüssig.

Kästen mit runden Ecken für plain \TeX (Garry Glendown, S. 385)

Garry Glendown hat ein Makro geschrieben, mit dem es möglich ist, in plain \TeX Kästen mit runden Ecken zu setzen. Verwendet werden dazu die circle Fonts, die auch in der picture-Umgebung von L^AT_EX angesprochen werden. Dabei gab es allerdings Probleme mit der Breite und dem Bezugspunkt der circle Fonts. Das Listing seiner Lösung ist dem Artikel angefügt.

Hin zu einem vollständigen und komfortablen \TeX System

(Stefan Lindner und Lutz Birkhahn, S. 368)

Die Autoren beschreiben, wie sie das erste Mal von \TeX hörten, was sie daraufhin planten (\TeX auf ATARI ST lauffähig zu machen) und wie sie ihre Pläne in die Tat umsetzten (den Quellcode von \TeX und METAFONT in C zu übersetzen). Beides, \TeX und METAFONT, war im Sommer 1988 fertig. Die entsprechenden Treiber wurden Anfang 1989 fertiggestellt. Nicht genug damit, entwickelten sie eine Benutzeroberfläche, deren Anwendung sie in dem Artikel kurz beschreiben. Für Graphikeinbindung verwenden sie pixel Format. Die Verteilung des Systems erfolgt auf der Grundlage des "shareware" Prinzips. Für die Zukunft ist u.a. die Entwicklung eines eigenen Editors geplant.

Drucken kommentierter Schachliteratur in natürlicher Schreibweise (Zalman Rubinstein, S. 387)

Der Autor stellt ein \TeX -Makro vor, mit dem es möglich ist, Schachzüge gemäß ihres Vorkommens darzustellen, Kommentare zu diesen Zügen an entsprechender Stelle einzubinden und die Darstellung entweder vom Beginn oder von einem beliebigen Punkt des Spielverlaufs ab anfangen zu lassen. Eine abgekürzte Schreibweise für Schachzüge kann im Moment noch nicht berücksichtigt werden. Ein ausführliches Eingabebeispiel ist in dem Artikel enthalten.

Der Weg zu "Äthiopischem \TeX " (Abass Andulem, S. 352)

In Äthiopien hat die Erzeugung von Schriftstücken eine lange Tradition. Mit der Einführung modernerer Druckmethoden ergab sich jedoch ein schwerwiegendes Problem, nämlich die Unmenge von Sonderzeichen in dieser Schrift. Es sind 231 Zeichen bekannt. Deshalb wurde 1987 ein Projekt mit dem Namen ETH^AT_EX gestartet, in dessen Verlauf mit Hilfe von METAFONT ein Zeichensatz generiert wurde, der alle benötigten Zeichen enthält. Ein ungelöstes Problem für die Teilnehmer an diesem Pro-

jekt stellt nun noch die Eingabe der Zeichen von der Tastatur dar. Die Gruppe arbeitet an einem "Äthiopischen Editor". Eine Fonttabelle des entwickelten Zeichensatzes ist dem Artikel angefügt.

Neugriechisch Setzen mit 128-Zeichen-Codes

(Yannis Haralambous und Klaus Thull, S. 354)

Die griechischen Fonts von Silvio Levy tragen mit ihrem 256-Zeichen-Code dem Umstand Rechnung, daß europäische Schriften für \TeX oft schwer zu handhaben sind, was Trennung oder Ligaturen angeht. Diese 256 Zeichen werden jedoch nicht von allen Druckertreibern unterstützt. Aus diesem Grund haben die Autoren die Fonts von Herrn Levy auf 128 Zeichen reduziert, wobei sie gleichzeitig den Font auf neugriechische Zeichen beschränkten. Da beim Schreiben von mathematischen Formeln zuweilen "slanted" und "small capital" Zeichen benötigt werden, entwickelten sie auch diese (reduzierten) Fonts, wodurch eine komplette Fontfamilie vorliegt. Der neueren offiziellen Verwendung eines "Ein-Akzent"-Griechisch, den griechischen Nummersymbolen und den Symbolen für zyprisches Griechisch wurde mit jeweils einem eigenen Font ebenfalls entsprochen. Als Trennmuster werden behelfsmäßig entweder die deutschen oder die portugisischen Muster verwendet.

Bibliographische Verweise; oder Variationen zu einem alten Taschenspielertrick (Lincoln K. Durst, S. 390)

Mit diesem Artikel wird ein Tutorium begonnen, das Benutzer in die Spitzfindigkeiten von \TeX einführen soll. Es ersetzt nicht die Lektüre vom \TeX -buch, soll auch nicht "das Rad neu erfinden", indem es Makropakete wie L^AT_EX oder $\mathcal{M}\mathcal{S}$ - \TeX ersetzt. Die Leser sollen einfach nur Denkanstöße erhalten, wie man mit plain \TeX arbeiten kann und was alles damit möglich ist. Die erste Folge von diesem Tutorium ist bibliographischen Verweisen gewidmet, wobei die gezeigte Lösung mehr und mehr verfeinert wird.

◊ Luzia Dietsche
Rechenzentrum der Universität
Heidelberg
Im Neuenheimer Feld 293
D-6900 Heidelberg 1
Bitnet: RZ68@HDURZ1

Editor's note: The sub-selection of articles here is due to time pressure and not other considerations such as "worthiness". In the future we intend to abstract all expository articles.

Calendar

1989

- Oct 11 UKTUG: "Fonts, Design and Use"
Aston University, Birmingham, U.K.
- Nov 23 4^e NTG-Bijeenkomst.
Katholieke Universiteit
te Tilburg, The Netherlands.
For information, contact H. Mulders
or C. G. van der Laan (Bitnet:
cgl@HGRRUG5).

1990

- Jan 8-12 Intensive Beginning/Intermed. **TEX**,
University of Houston, Clear Lake,
Houston, Texas

- Jan 15 **TUGboat Volume 11, No. 1:**
Deadline for receipt of manuscripts.

**California State University,
Northridge, California**

- Jan 15-19 Intensive Beginning/Intermed. **TEX**
Jan 22-26 Advanced **TEX/Macro Writing**

**Florida State Supercomputer Research
Institute, Tallahassee, Florida**

- Jan 15-19 Intensive Introduction to **IATEX**
Jan 22-26 Intensive Beginning/Intermed. **TEX**

**University of Maryland,
College Park, Maryland**

- Jan 8-12 Intensive Beginning/Intermed. **TEX**
Jan 15-19 Advanced **TEX/Macro Writing**
Feb 5-9 Intensive Introduction to **IATEX**

- Apr 10 **TUGboat Volume 11, No. 2:**
Deadline for receipt of manuscripts
(tentative).
May 14-15 GUTenberg meeting, Toulouse,
France

**TUG90 Conference
Texas A & M University
College Station, Texas**

- Jun 18-21 **TUG's 11th Annual Meeting**

**TEX90
University College
Cork, Ireland**

- Sep 10-12 **TUG's 1st Conference in Europe**

- Sep 18-20 **EP'90**
National Institute of Standards
and Technology, Gaithersburg,
Maryland. For information,
contact Richard Furuta
(furuta@brillig.umd.edu).

For additional information on the events listed above, contact the TUG office (401-751-7760) unless otherwise noted.

TUG90

First Announcement/
Call for Papers

TeX Users Group
11th Annual Meeting

Texas A&M University
June 18-21, 1990

Forward into the 90s

THE CONFERENCE:

In 1990, there will be *two* TUG-sponsored TeX meetings: one in Texas in June, the other in Cork, Ireland in September. The Texas meeting will be held at Texas A&M University June 18-21, 1990 (4 days).

The theme for the Texas meeting will be **FORWARD INTO THE 90s**. We celebrated our 10th Anniversary at Stanford University this past August, amidst announcements of revisions and revitalization in both TeX and LATEX. As we move into TUG's second decade, TeX will continue to grow and expand in everyday working situations. The challenge of bringing new users into contact with TeX, of stretching TeX and METAFONT to merge and work with old or new procedures, is exciting.

Plan to attend. Better yet: submit a short abstract outlining your own contribution to the new decade. For details, see below.

CALL FOR PAPERS:

Presentations should be approximately 20-25 minutes in length — roughly 6 pages of text, using a 10pt font on a 12pt baseline. Abstracts (no longer than one page) should be submitted as quickly as possible, for the **December 1st** deadline, to the Program Coordinator. Preliminary abstracts may be sent via regular mail, electronic mail, or FAX machine. See address information below.

TOPICS FOR DISCUSSION:

The following list of potential topics is neither inclusive nor exclusive; however, papers addressing TeX and METAFONT issues for the 1990s would be particularly welcomed. Theoretical analyses and actual experiences dealing with various aspects of TeX and LATEX, and METAFONT will be considered. Papers will be selected on the basis of clarity, usefulness to either the experienced or the neophyte user, and general interest to the TeX community.

- How does TeX do it? — overview of the TeX process; how TeX uses all those font files
- TeX for non-TeX users: — teaching/training personnel (support staff, students, colleagues); front-end/interface programs; from beginner to expert; documentation — what's available, how to write it; source of information: user groups, courses, bulletin boards, books, etc.
- Integrating TeX with non-TeX programs: — graphics; other software
- Fonts: — font design: how to put/pull it together; font compatibility: how to use non-TeX fonts; font experiences: samples, design histories
- Document design: — theoretical aspects of document design; styles: from technical reports to poetry
- Using TeX as part of a larger system: — extensions to TeX; subsets of TeX
- Technical nuts and bolts: — what are drivers: how do they work, how do they differ, standards; what's portability: how to make things portable; fonts and printers and drivers: making them work together; electronic sources of information: how to find them, use them; suggestions and comments

IMPORTANT DATES:

Abstract	December 1st, 1989
Notification	January 15th, 1990
Preliminary Paper	March 15th, 1990
Presentation at Texas Meeting	June 18-21, 1990
Final Paper	July 15th, 1990

PROGRAM COORDINATOR:

Christina Thiele
JPC DT1711
Carleton University
Ottawa, Canada K1S 5B6
Phone: 613-788-2340
Bitnet: wsscat@carleton.ca
FAX: 613-788-3544

PROGRAM COMMITTEE:

Shawn Farrell	Regina Girouard	Tom Reid
McGill University	American Mathematical Society	Texas A&M University

TeX USERS GROUP:

Questions concerning any other aspect of this Meeting or TeX90 (Cork, Ireland) should be directed to the TUG Office: 401-751-7760 (Eastern Time); electronic mail: tug@math.ams.com; or FAX: 401-751-1071.

5th European TeX Conference 1990
1st TeX Users Group Meeting in Europe

Announcement and
Call for Papers

TeX90

Cork, Ireland — 10–12 September 1990

THE 1990 TeX meeting in Europe is the fifth such conference organised since 1985. It also breaks new ground as the first to be a TeX Users Group (TUG) meeting outside North America. The TeX computer typesetting program is now widely established as the *de facto* standard in scientific, educational and commercial use for the setting of documents requiring very high standards of typographic control, particularly technical documents, and for applications where high quality, portability and device-independence are of importance.

Contributions are now being sought by the programme committee for presentation in September 1990. Papers may present material on a wide range of topics related to (but not limited to) the areas suggested below. Selection will be made on the basis of originality, applicability, utility and interest to the TeX user community.

Suggested topic list:

- ◊ Document origination, editing and markup
- ◊ Typographic layout and design: aesthetics, practicalities
- ◊ Specialist macros and their applications
- ◊ Database and hypertext and their relationship with TeX
- ◊ TeX training and induction: the new user
- ◊ Merging of text and graphics, new symbol sets
- ◊ Font designs and new fonts: their uses and limitations
- ◊ Output devices and drivers, en-TEXing and de-TEXing
- ◊ Merging of TeX with other systems and applications
- ◊ Structure and design of tagged and markup documents
- ◊ Multi-lingual TeX and the representation of languages
- ◊ Portability: development of new program versions
- ◊ User groups and the dissemination of information
- ◊ Conformity and standards, international and otherwise
- ◊ Novel applications of TeX, METAFONT etc

Requirements

- ◊ Authors must submit an abstract of one page (or two 80×25 screens) by 1st December 1989, by postal mail, electronic mail or facsimile (fax).
- ◊ Submissions should be sent in the first instance to the author's national user group if there is one: failing this, to one of the programme coordinators.
- ◊ Speakers selected will be notified of acceptance by 1st February 1990.
- ◊ Complete texts must be submitted by 1st April 1990: **five copies** are required on paper.
- ◊ Final copies of papers for publication in the proceedings must be submitted by 1st June 1990: **two copies** are required in camera-ready form (details of layout size and formatting will be sent to speakers at the time of notification).

Addresses

Conference Office:

TeX90 Office
Computer Centre
University College
Cork, Ireland
Telephone: +353 21 276871 ×2609
Fax: +353 21 277194
Email: tex90@vax1.ucc.ie

Programme Coordinators:

Peter Flynn (*Local Coordinator, Programme Co-Chairman*)
University College Cork (cbts8001@vax1.ucc.ie)
Dean Guenther (*Programme Co-Chairman*)
Washington State University (guenther@wsuvml.bitnet)

Organising Committee:

Ray Goucher, TeX Users Group (reg@seed.ams.com)
Dean Guenther, Washington State University
Peter Flynn, University College Cork
Kees v. d. Laan, Rijksuniversiteit Groningen (cgl@hgrrug5.bitnet)

National User Groups:

DAnTe, Germany	Joachim Lammarsch	Heidelberg University (rz92@dhdurz1.bitnet)
Nordic TeX UG	Roswitha Graham	KTH Stockholm (roswitha@duvan.nada.kth.se)
NTG, Holland	Johannes Braams	DNL Leidschendam (braams@hlsdn15.bitnet)
GUTenberg, France	Bernard Gaulle	CIRCE Orsay (ucir001@frors31.bitnet)
UKTeX	Malcolm Clark	Imperial College London (fps@vaxa.cc.imperial.ac.uk)



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

Reflections

Bart Childs

Ray Goucher requested that I write an article about my four years as President. My first thought was sure, then after thinking about what has happened, I wonder why? I am proud of my service and sincerely appreciative of the opportunities of representing you in a number of activities.

Norman Naugle introduced me to TeX after I tried to impress him with a word processor that I was supporting. We started an attempt to port TEX79 to PL/I but dropped it (like the WP) when Don announced that TEX82 and WEB would be coming. We started the port on version 0.6 and had it working with 0.9. The speed with which TeX was ported to systems is evidence that WEB is the most overlooked success in computing, even with the *Literate Programming* column in CACM.

My first TUG meeting was 1984. During the 1985 meeting the nominations committee asked Norman to consider a nomination for the Presidency. If you know Norman, you know that he would not, instead, he threw my hat in the ring. It appears that my election came about for two reasons, the (unwritten?) direction that we should have a significant leadership from academia and Don's hopes to have an officer from TEXas.

I had been to several meetings of the Steering Committee as a Site Coordinator. I recall two things from the last meeting at the '85 meeting. Don welcomed me aboard to the inner circle or some term like that. The other memory is it seemed that our primary topic of discussion was our precarious financial status and how would we declare bankruptcy if it became necessary? While flying back to TEXas, I began to wonder what I had really gotten into. I did start referring to Norman as my ex-friend.

We quickly became a stable organization, in a fiscal sense. There were several reasons and actions on the part of some TEXers that caused our improved status. I think that some of the most significant were:

1. The quality of the TeX systems and the fact that Don put it in the public domain.
2. Wider availability and distribution of TeX, especially those distributed to large numbers of users by the following who worked closely with TUG and encouraged membership:
 - The *unix* distribution by Pierre MacKay and Rick Furuta.
 - The VMS distribution by Kellerman and Smith.
 - The Personal TeX distribution of PCTeX.
 - The Addison-Wesley distribution of MicroTeX.
 - The generic distribution from DP Services and David Fuchs' public domain change files.
3. Ray Goucher's capable management, spreading the word, seeking every opportunity to find another TeX publication that we can sell, and aggressive pursuit of teaching TeX courses and expansion of the offerings, especially at Los Alamos and other labs.
4. Support, sheltering arms and facilities of the American Mathematical Society in helping us start.
5. Many volunteers who contributed so much. It would be a travesty to not name Barbara Beeton and Sam Whidden, who served us so well as Editor and Treasurer, respectively. The organization that was fostered by Richard Palais, Mike Spivak, Pierre MacKay, and many of you, was the basis of this success.

Just a few of our heroes have been mentioned. Notice that most are still active in our community. Some have changed where they work, but they are still good volunteers.

In closing, I would like to point out several indications of our maturity.

- We have become a separate legal entity. We still have a good working relationship with AMS and are a customer of theirs.
- We have moved into space of our own by leasing a renovated fire station nearby AMS.
- We supported the finishing of the TeX project at Stanford.
- We are supporting the moderating of the TeXhax bulletin board at the University of Washington.
- We subsidized the creation of teaching materials for TeX and related subjects. This documentation will grow significantly over the next year or so.
- We are supporting the finishing of BIBTeX by Oren Patashnik. It should be finished in August.
- We are actively pursuing closer involvement with TeX groups outside North America.
- The 10th TUG meeting will have significant METAFONT and WEB activities.

We generally put TUG finds and/or moral support to use as a supplement to volunteer activities that started an activity. We need to continue pushing for new horizons. Your ideas will be appreciated and volunteers should expect to get used. We have not always succeeded in following through, but the staff that is now in place will make that a rare event in the future.

Thanks for your support and I look forward to being an active worker as Past-President.

TeX USERS GROUP
Ten Years of TeX and METAFONT
Stanford University
August 20-23, 1989

Sunday - August 201:00 - 2:00 pm *Registration***Font Forum**

1:30 - 2:00	Introduction to METAFONT - Doug Henderson
2:00 - 2:30	Opening Pandora's Box - Neenie Billawala
2:30 - 3:00	Order into Chaos: Typesetting Fractal Images - Alan Hoenig
3:00 - 3:15	<i>Break</i>
3:15 - 3:45	Design of Oriental Characters with METAFONT - Don Hosek
3:45 - 4:15	Thai Languages and METAFONT - Bob Batzinger (Thailand)
4:15 - 4:45	A METAFONT-like System with PostScript Output - John Hobby
4:45 - 5:15	Migration from Computer Modern Fonts to Times Fonts - Ralph Youngen, Daniel Lattner, and William Woolf
5:15 - 5:45	Fine Typesetting with TeX Using Native Autologic Fonts - Arvin Conrad
7:00 - 10:00 pm	<i>Dinner at the Sino Restaurant</i> ¹

Monday - August 21

7:45 - 9:30 am	<i>Registration</i>
8:00 - 8:30	Introduction to TeX and TeX Systems (for New Members) - Bart Childs and Alan Hoenig
8:30 - 9:00	Introduction to TUG and TUG Officers and Staff

Keynote Address

9:00 - 11:30	The Errors of TeX - Don Knuth ²
11:30 - 1:00 pm	<i>Lunch</i> ³
1:00 - 2:30	Elections/Business Meeting
2:30 - 2:45	<i>Break</i>
2:45 - 3:00	Report from the Driver Standards Committee - Robert McGaffey
3:00 - 5:00	Output Device Manufacturer/Exhibitor Presentations: ⁴ American Mathematical Society; ArborText; Blue Sky Research; Computer Composition Corp.; Gestetner Lasers; Kinch Computer Co.; Micro Publishing Systems; Northlake Software; Personal TeX; TeXnology, Inc.; TeXplorators Corp.

Orientation

5:00 - 5:15	Things to do in and around Palo Alto - Ginger Brower
5:15 - 5:30	<i>Organization of the Birds-of-a-Feather Sessions for Sites</i> (DG, CMS, MVS, CDC, UNIX, VMS, etc.) to be held during and after the Wine & Cheese
5:30 - ???	<i>Wine & Cheese</i> - hosted by Personal TeX, Inc.

¹ Reservation required. Formerly known as the China First Restaurant.² Professor Knuth will be available through out the meeting for questions and to autograph copies of his book series "Computers and Typesetting". (A limited number of copies will be available for purchase at the Registration Desk.)³ Working lunch for the Board of Directors.⁴ Representatives are scheduled to be available throughout the meeting. Exhibit rooms will be open from 8:30 am Tuesday until 5:00 pm Wednesday.

An IBM PC and Apple Macintosh is available so that members may exchange software.

(Continued)

Tuesday – August 22**TeX Training**

- 8:30 – 9:00 am Of the Computer Scientist, by the Computer Scientist, for the Computer Scientist – Michael Doob (Canada)
- 9:00 – 9:30 Mastering TeX with Templates – Hope Hamilton
- 9:30 – 10:00 TeX and its Versatility in Office Production – Jo Ann Rattey-Hicks
- 10:00 – 10:30 TeX for the Word Processing Operator – Robin Kubek
- 10:30 – 10:45 *Break*
- 10:45 – 11:30 Site coordinators' status reports:
VAX (VMS) – David Kellerman; UNIX – Pierre MacKay;
“small” systems – Lance Carnes; IBM VM/CMS – Dean Guenther;
IBM MVS – Craig Platt; Data General & Cray – Bart Childs;
CDC Cyber – Jim Fox; Prime 50 Series – John Crawford.
- 11:30 – 11:45 *10th Anniversary Meeting Group Photograph*
- 11:45 – 1:00 pm *Lunch¹*

General Applications

- 1:00 – 1:30 TeX in México – Max Díaz (México)
- 1:30 – 2:00 TeX for 30,000 – James Haskell, Wally Deschene and Alan Stolleis
- 2:00 – 2:30 TeX Enslaved – Alan Wittbecker

Graphics Applications

- 2:30 – 3:00 Methodologies for Preparing and Integrating PostScript Graphics – Tom Renfrow
- 3:00 – 3:40 TeXpic – Design and Implementation of a Picture Graphics Language in TeX
à la pic – Rolf Olejniczak (West Germany)
- 3:40 – 4:00 *Break and organization of Topical Birds-of-a-Feather*
(Topics to be decided at this time.)

Database Applications

- 4:00 – 4:30 TeX at *Mathematical Reviews* – William Woolf and Daniel Latterner
- 4:30 – 5:00 Lexicography with TeX – Jörgen Pind (Iceland)

General Information

- 5:00 – 5:15 Status of TeX in Japan – Edgar Cooke (Japan)
- 5:15 – ??? Topical Birds-of-a-Feather sessions (IATEX, graphics, fonts, etc.)
- 6:30 – 7:30 *10th Anniversary “Mexican Fiesta” Dinner* – sponsored by the TeX Users Group
- 7:30 – ??? *Social Hour/Party Time* – hosted by ArborText

Wednesday – August 23**Help Sessions**

- 8:30 – 9:30 am TeX Help Session – Barbara Beeton
- 9:30 – 10:15 WEB Help Session – Wayne Sewell
- 10:15 – 10:30 *Break*

General Information

- 10:30 – 11:00 The State of TeX Users Groups in Europe – Malcolm Clark (England)
- 11:00 – 11:30 The UKTeX Archive at Aston University – Peter Abbott (England)
- 11:30 – 1:00 pm *Lunch¹*

¹ Working lunch for the Board of Directors.

(Continued)

TeX Tools

1:00 – 1:30	With LATEX into the Nineties – Frank Mittelbach and Rainer Schöpf (Germany)
1:30 – 2:00	TEXreation – Playing Games with TeX's Mind – Andrew Greene
2:00 – 2:30	Indexing TeX's Commands – Bill Cheswick
2:30 – 3:00	A Study in Producing Local Style Files for Letters and Memos – Steve Sydoriak
3:00 – 3:15	<i>Break</i>
3:15 – 3:45	Using WordPerfect 5.0 to Create TeX and LATEX Documents – Anita Hoover
" – "	Inserts in Multiple Columns – Gary Benson and Debi Erpenbeck
3:45 – 4:15	TeX Macros for COBOL Syntax Diagrams – Mary McClure
4:15 – 4:45	Coordinating a Procedural Language and Text Editor to Create an Efficient and Workable PC Interface for TeX – Brad Halverson and Don Riley
4:45 – 5:00	General wrap-up and closing – Bart Childs et al.

Program Coordinator: Dean Guenther – Washington State University (Pullman, Washington)
 Program Committee: Hope Hamilton – National Center for Atmospheric Research (Boulder, Colorado)
 Doug Henderson – University of California (Berkeley, California)
 Christina Thiele – Carleton University (Ottawa, Ontario)

Program: EuroTeX89**The 4th European TeX Conference****Monday, September 12, 1989**

- R. Rupprecht and A. Brüggemann-Klein:
Welcome
- B. Childs: The state of TUG and TeX
- M. Clark: The state of TUG in Europe
- B. Childs: A report on teaching TeX

TUG and national user groups [B. Childs and R. Goucher]

- J. Lammarsch: La divina comedia
- C.G. van der Laan: Getting organized in the Netherlands

The future of TeX and LATEX

- C.A. Rowley: What Don Knuth said at Stanford
- K. Thull: How to bring up TeX on a PC
- F. Mittelbach and R. Schoepf: LATEX limitations and how to get around them
- F. Mittelbach and R. Schoepf: With LATEX into the nineties

Tuesday, September 12, 1989

- J. Schrod: Changebars without \specials
- K. Neuwirth: TeX on the Amiga — hands-on experiences
- N. Eigsperger: A public domain driver for the Digiset typesetter

Foreign TeX

- M. Ryćko: Polish TeX is ready for use
- B. Jackolowski:
- M. Bryan: JIATEX

Document Structure

- M. Clark: TeX and/or SGML
- A. Brüggemann-Klein: TeX and document design
- D. Rodgers: Electronic publishing in the nineties

Social event**Wednesday, September 13, 1989****Getting information**

- P. Abbott: The UKTeX Archive at the University of Aston
- J. Lammarsch: Using LISTSERV at DHDURZ1

Fonts and graphics

- R. Olejniczak: TeXpic — design and implementation of a picture graphics language in TeX
- P.D. Bacsich: MoreMath — a new PostScript font of mathematical symbols
- D. Hosek: Design of oriental characters with METAFONT

TeX environments

- R. Wonneberger: TeX in an IDP environment
- B. Childs et al.: TeX environments
- K. Heidrich: Software-design of a complete and comfortable TeX system
- K. Baer: The use of TeX at the IKE

Production Notes

Barbara Beeton

Input and input processing

Electronic input for articles in this issue was received by mail and on floppy disk. Camera copy was accepted for one article and for several figures (see the "output" section).

Authors who had written articles previously for *TUGboat* typically submitted files that were fully tagged and ready for processing with the *TUGboat* macros—*tugbot.sty* for plain-based files and *ltugbot.sty* for those using *LATEX*. However, since the *TUGboat* macros have been completely rewritten by Ron Whitney (see the Authors' Guide, page 378, for instructions on using the new versions), all submissions were re-tagged as necessary to conform to the new input conventions.

About a fourth of the articles, and almost one-third the pages in this issue are *LATEX*. For convenience in processing, plain or *LATEX* articles were grouped whenever possible. Articles in which no, or limited, *TEX* coding was present were tagged according to the conventions of *tugbot.sty* or *ltugbot.sty* as convenient. Articles tagged according to the author's own schemes were modified sufficiently to permit them to be merged with the rest of the stream. Especial care was taken to try to identify macro definitions that conflicted with ones already defined for *TUGboat*, and *\begingroup ... \endgroup* was wrapped around any suspect article as a routine precaution.

The example page (p. 331) of Stephan v. Bechtolsheim's article was inserted using FTP and his *dvi2dvi* program. Stephan produced the *.dvi* file for that page using *TEX* and *dvi2dvi* as described in the article. When the *TUGboat* *.dvi* file for plain pages was ready (containing a blank page with proper running heads for his example page), it was FTP'd to Purdue. *dvi2dvi* then inserted the example page and the result was FTP'd back to the AMS.

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 five runs of *TEX* and two of *LATEX*, using the page number file for reference.

The following articles were prepared using *LATEX*; the starred items required the *doc*-option.

- Nelson Beebe, *Message from the New President*, page 312.
- Victor Eijkhout and Nico Poppelier, *The first Dutch T_EX days*, page 316.
- Technites, *Crossword puzzle*, page 324.
- Frank Mittelbach and Rainer Schöpf, *Towards L_AT_EX 2.10*, page 400.
- Johannes Braams, Victor Eijkhout and Nico Poppelier, *The development of national L_AT_EX styles*, page 401.
- * Frank Mittelbach, *An environment for multicol column output*, page 407.
- * Frank Mittelbach, *An extension of the L_AT_EX theorem environment*, page 416.
- Luzia Dietsche, *Deutsche Kurzfassungen der TUGboat-Artikel*, page 427.

Output

The bulk of this issue was prepared at the American Mathematical Society on a VAX 6320 (VMS) and output on an APS-μ5 using resident CM fonts and additional downloadable fonts for special purposes. The items listed below were received as camera copy; they were prepared on the devices indicated. The output devices used to prepare the advertisements were not usually identified; anyone interested in determining the device used for a particular ad should inquire of the advertiser. The appearance of the printed pages can be taken as representative of output from the devices which produced them.

- Unidentified:
 - all advertisements.
- Abass Andulem, *The road to Ethiopic T_EX*, page 352.
- Apple LaserWriter (300 dpi):
 - Jan Eric Larsson, *A chess font for T_EX*, page 351, figures only.
- HP LaserJet (300dpi):
 - Hoenig, Alan, *T_EX-PostScript output on non-PostScript devices*, p. 374, figure 1.
- Varityper VT600 (600 dpi):
 - Hoenig, Alan, *T_EX-PostScript output on non-PostScript devices*, p. 375, figure 2 only (but note remarks concerning figure).

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*

Aston University,
Birmingham, England

Bond University, School
of Information &
Computing Sciences,
Queensland, Australia

Brookhaven National Laboratory,
Upton, New York

Brown University, *Providence, Rhode Island*

California Institute of Technology,
Pasadena, California

Calvin College, *Grand Rapids, Michigan*

Carnegie Mellon University,
Pittsburgh, Pennsylvania

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

City University of New York,
New York, New York

College of St. Thomas, Computing
Center, *St. Paul, Minnesota*

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

COS Information, *Montreal, P. Q., Canada*

Data General Corporation,
Westboro, Massachusetts

DECUS, L&T Special Interest
Group, *Marlboro, Massachusetts*

Department of National Defence,
Ottawa, Ontario, Canada

Digital Equipment Corporation,
Nashua, New Hampshire

Edinboro University of
Pennsylvania, *Edinboro, Pennsylvania*

Electricité de France,
Clamart, France

Emerson Electric Company,
St. Louis, Missouri

Environmental Research Institute
of Michigan, *Ann Arbor, Michigan*

European Southern Observatory,
Garching bei München, Federal Republic of Germany

Fermi National Accelerator
Laboratory, *Batavia, Illinois*

Fordham University, *Bronx, New York*

Försvarets Materielverk,
Stockholm, Sweden

General Motors Research
Laboratories, *Warren, Michigan*

Geophysical Company of Norway
A/S, *Stavanger, Norway*

GKSS, Forschungszentrum
Geesthacht GmbH, *Geesthacht, Federal Republic of Germany*

Grinnell College, Computer
Services, *Grinnell, Iowa*

GTE Laboratories,
Waltham, Massachusetts

Harvard University, Computer
Services, *Cambridge, Massachusetts*

Hatfield Polytechnic, Computer
Centre, *Herts, England*

Hewlett-Packard Co., *Boise, Idaho*

Hutchinson Community College,
Hutchinson, Kansas

IBM Corporation, Scientific
Center, *Palo Alto, California*

Imagen, *Santa Clara, California*

Informatika, *Hamburg, Federal Republic of Germany*

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*

Kuwait Institute for Scientific
Research, *Safat, Kuwait*

The Library of Congress,
Washington D.C.

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

Louisiana State University,
Baton Rouge, Louisiana

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

Massachusetts Institute
of Technology, Artificial
Intelligence Laboratory,
Cambridge, Massachusetts

Massachusetts Institute of
Technology, Information Services,
Cambridge, Massachusetts

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

Max Planck Institut
für Mathematik, *Bonn, Federal Republic of Germany*

Max Planck Institute Stuttgart,
Stuttgart, Federal Republic of Germany

McGill University, *Montréal, Québec, Canada*

Michigan State University,
Mathematics Department,
East Lansing, Michigan

National Cancer Institute,
Frederick, Maryland

National Institutes of Health,
Bethesda, Maryland

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

National Semiconductor
Corporation, *Santa Clara, California*

New Jersey Institute of
Technology, *Newark, New Jersey*

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

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

- Northeastern University, Academic Computing Services, *Boston, Massachusetts*
- Norwegian Pulp & Paper Research Institute, *Oslo, Norway*
- Online Computer Library Center, Inc. (OCLC), *Dublin, Ohio*
- Pennsylvania State University, Computation Center, *University Park, Pennsylvania*
- Personal TeX, Incorporated, *Mill Valley, California*
- Princeton University, *Princeton, New Jersey*
- Promis Systems Corporation, *Toronto, Ontario, Canada*
- Peter Isaacson Publications, *Victoria, Australia*
- Purdue University, *West Lafayette, Indiana*
- QMS, Inc, *Mobile, Alabama*
- Queens College, *Flushing, New York*
- RE/SPEC, Inc., *Rapid City, South Dakota*
- Rice University, Department of Computer Science, *Houston, Texas*
- Rogaland University, *Stavanger, Norway*
- Royal Marsden Hospital, *Surrey, England*
- Ruhr Universität Bochum, Rechenzentrum, *Bochum, Federal Republic of Germany*
- Rutgers University, Hill Center, *Piscataway, New Jersey*
- St. Albans School, Mount St. Alban, *Washington, D.C.*
- Sandia National Laboratories, *Albuquerque, New Mexico*
- SAS Institute, *Cary, North Carolina*
- I. P. Sharp Associates, *Palo Alto, California*
- Smithsonian Astrophysical Observatory, Computation Facility, *Cambridge, Massachusetts*
- Software Research Associates, *Tokyo, Japan*
- Sony Corporation, *Atsugi, Japan*
- Space Telescope Science Institute, *Baltimore, Maryland*
- Springer-Verlag, *Heidelberg, Federal Republic of Germany*
- Stanford Linear Accelerator Center (SLAC), *Stanford, California*
- Stanford University, Computer Science Department, *Stanford, California*
- Stanford University, ITS Graphics & Computer Systems, *Stanford, California*
- State University of New York, Department of Computer Science, *Stony Brook, New York*
- Stefan Ram, Programming and Trade, *Berlin, Federal Republic of Germany*
- Stratus Computer, Inc., *Marlboro, Massachusetts*
- Syracuse University, *Syracuse, New York*
- Talaris Systems, Inc., *San Diego, California*
- TECOGRAF Software, *Milan, Italy*
- Texas A & M University, Department of Computer Science, *College Station, Texas*
- Texcel, *Oslo, Norway*
- Tribune TV Log, *Glens Falls, New York*
- TRW, Inc., *Redondo Beach, California*
- Tufts University, *Medford, Massachusetts*
- TV Guide, *Radnor, Pennsylvania*
- TYX Corporation, *Reston, Virginia*
- UNI-C, *Aarhus, Denmark*
- Universidade de Coimbra, *Coimbra, Portugal*
- Università degli Studi Milano, Istituto di Cibernetica, *Milan, Italy*
- University College, *Cork, Ireland*
- University of Alabama, *Tuscaloosa, Alabama*
- University of British Columbia, Computing Centre, *Vancouver, British Columbia, Canada*
- University of British Columbia, Mathematics Department, *Vancouver, British Columbia, Canada*
- University of Calgary, *Calgary, Alberta, Canada*
- University of California, Division of Library Automation, *Oakland, California*
- University of California, Berkeley, Computer Science Division, *Berkeley, California*
- University of California, Berkeley, Space Astrophysics Group, *Berkeley, California*
- University of California, Irvine, Department of Mathematics, *Irvine, California*
- University of California, Irvine, Information & Computer Science, *Irvine, California*
- University of California, San Diego, *La Jolla, California*
- University of California, San Francisco, *San Francisco, California*
- University of Canterbury, *Christchurch, New Zealand*
- University of Chicago, Computing Organizations, *Chicago, Illinois*
- University of Chicago, *Chicago, Illinois*
- University of Crete, Institute of Computer Science, *Heraklio, Crete, Greece*
- University of Delaware, *Newark, Delaware*
- University of Exeter, Computer Unit, *Exeter, Devon, England*
- University of Glasgow, Department of Computing Science, *Glasgow, Scotland*
- University of Groningen, *Groningen, The Netherlands*
- University of Illinois at Chicago, Computer Center, *Chicago, Illinois*

University of Illinois at Urbana-Champaign, Computer Science Department, *Urbana, Illinois*

University of Kansas, Academic Computing Services, *Lawrence, Kansas*

University of Maryland, *College Park, Maryland*

University of Massachusetts, *Amherst, Massachusetts*

Université de Montréal, *Montréal, Québec, Canada*

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

University of Ottawa, *Ottawa, Ontario, Canada*

University of Salford, *Salford, England*

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

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

University of Texas at Austin, *Austin, Texas*

University of Vermont, *Burlington, Vermont*

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

University of Western Australia, Regional Computing Centre, *Nedlands, Australia*

University of Wisconsin, Academic Computing Center, *Madison, Wisconsin*

Uppsala University, *Uppsala, Sweden*

USDA Forest Service, *Washington, D.C.*

Vereinigte Aluminium-Werke AG, *Bonn, Federal Republic of Germany*

Villanova University, *Villanova, Pennsylvania*

Vrije Universiteit, *Amsterdam, The Netherlands*

Washington State University, *Pullman, Washington*

Widener University, Computing Services, *Chester, Pennsylvania*

John Wiley & Sons, Incorporated, *New York, New York*

Worcester Polytechnic Institute, *Worcester, Massachusetts*

Yale University, Computer Center, *New Haven, Connecticut*

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

TEX Users Group Treasurer's Report as of 1989 Annual Meeting

David Ness

The financial health of *The TEX Users Group* as of June 30, 1989 is good, but somewhat short of excellent. Financial reports are presented here which describe (1) the *Balance Sheet* of the organization as prepared by Michael D. Aaronson and Associates, C.P.A. as of the close of the calendar years 1987 and 1988; and (2) Revenue and Expense statements covering (a) 1988 actual data; (b) 1989 Budget; (c) actual first half of 1989; (d) projected actual 1989; and (e) 1990 budget (first draft). Perhaps a few comments are in order.

While we hold cash (and equivalents) of about \$250,000 with relatively limited real liabilities, this is a 'cushion' of only 4-5 months of income. We would like to move this up somewhat over the forthcoming year, eventually to a point that is at least 6-8 months worth of revenues.

We project that 1989 Revenues and 1989 Expenses will each be up from their 1988 values by just about 20%. We had a small deficit in 1988 (about \$6,000 or 1.1% of expenses) and project a slightly smaller deficit this year.

Two aspects of this are probably worth commenting on. First, our current projected deficit

results from two discretionary expenditures made by the Executive/Finance Committees. We would have closed the year with a small surplus had we not decided to support the BIBTEX project (at the request of Prof. Knuth) and had we not provided some funds to support TEXhax. Your Executive/Finance Committees felt that these expenditures were very important to the membership, and were willing to incur the small projected deficit that results.

Second, by far the greatest increases on the expense side are a result of the increases in size and frequency in publication of TUGboat. Here we see the 'triple' effect of increased membership and circulation, increased number of issues and pages, and increased costs of printing and mailing.

We prepare, with the aid of our accountant, a number of other financial statements such as *cash flows* and *statement of changes in fund balances*. Members of TUG are welcome to request these from TUG headquarters. Members are also encouraged to forward any questions about financial statements to me via TUG headquarters. While I can't guarantee that the response will always be prompt, I can promise that it will be thoughtful and eventually forthcoming.

Respectfully submitted,

David Ness

Treasurer

Philadelphia PA, 6 September 1989

TEX Users Group
Balance Sheets (unaudited)
December 31, 1987 and 1988
Prepared by Michael D. Aaronson and Associates, C.P.A

A S S E T S

	Dec. 31, 1987	Dec. 31, 1988
Current assets		
Cash	\$ 211,835	\$ 65,307
Certificates of deposit	100,000	207,505
Accounts receivable ¹	16,773	29,194
Inventory ²	15,186	25,002
Prepaid insurance	0	1,729
Total current assets	<u>343,794</u>	<u>328,737</u>
Property and equipment		
Office furniture and equipment	2,460	17,568
Computer software and equipment	0	21,885
	<u>2,460</u>	<u>39,453</u>
Less accumulated depreciation	(1,476)	(5,657)
Net property and equipment	<u>984</u>	<u>33,796</u>
Other assets		
Rent and utility deposits	0	1,880
Total other assets	<u>0</u>	<u>1,880</u>
TOTAL ASSETS	\$ 344,778	\$ 364,413

L I A B I L I T I E S A N D F U N D B A L A N C E S

Current liabilities		
Accounts payable	\$ 19,846	\$ 5,924
Accrued payroll and payroll taxes payable	1,454	3,697
Accrued vacation payable	5,513	4,896
Accrued pension expense	0	7,071
Other payables ³	7,486	7,486
Deferred income ⁴	<u>56,200</u>	<u>88,267</u>
Total current liabilities	<u>90,499</u>	<u>117,341</u>
Total liabilities	<u>90,499</u>	<u>117,341</u>
Commitments and contingencies ⁵		
Fund balances ⁶		
Property and equipment funds	984	33,796
Unrestricted funds	<u>253,295</u>	<u>213,276</u>
Total fund balances	<u>254,279</u>	<u>247,072</u>
TOTAL LIABILITIES AND FUND BALANCE	\$ 344,778	\$ 364,413

¹ TUG uses the specific identifying method in identifying potential bad debts. No receivables were considered uncollectible at December 31, 1988.

² Inventory consists of books, past issues of TUGboat, software, and other materials both purchased and created inhouse. Inventory is valued at average cost based on the FIFO method of evaluation.

³ Other payables consists of pre-1988 tax sheltered annuity payroll deductions for a staff member which cannot be deposited with a pension agent until TUG's tax exempt status is determined (see also Note 5 to the financial statements).

⁴ Deferred income consists of cash collected in the current period applicable to membership dues or course registrations for the subsequent year.

⁵ a) Commitments: Operating lease - space rental: TUG rents office space under a three year operating lease which expires in May, 1991. Monthly rent is \$800.

b) Contingencies: IRS exempt status: TUG has an application for tax exempt status under Section 501(C)(3) of the Internal Revenue Code pending before the IRS. In November 1988, the IRS denied TUG's application and TUG has filed an appeal. The ultimate disposition of this matter cannot be determined but TUG's management is confident it will be successful in gaining tax exempt status under either section 501(C)(3) or another section of the Code governing exempt organizations. However, should TUG be judged a taxable organization, income taxes would become due and payable. No provision in these financial statements has been made for this contingent liability.

⁶ Prior to January 1, 1988, TUG kept its books on a cash basis. Fund balances consisted of cash in bank accounts, on hand, and certificates of deposit. An accrual basis balance sheet as of December 31, 1987 consisting of estimated asset and liability accounts was established from an examination of TUG's financial records as of that date for the purposes of facilitating the change in accounting method to the accrual basis as of January 1, 1988.

TEx Users Group
1989 Revenue and Expense Statement
January 1 – June 30 with projections and comparisons

R E V E N U E

	Actual 1988	Budget 1989	Actual 1st half 1989	Projected 1989	Budget 1990
Membership Income					
Individual	\$ 81,727	\$ 106,500	\$ 58,748	\$ 113,000	\$ 127,500
Institutional—educational	31,195	38,000	25,165	48,000	54,000
Institutional—noneducational	<u>16,121</u>	<u>22,000</u>	<u>3,437</u>	<u>6,500</u>	<u>7,500</u>
	<u>129,113</u>	<u>166,500</u>	<u>87,350</u>	<u>167,500</u>	<u>189,000</u>
Annual Meeting and Course Income					
Meeting	34,080	40,000	15,116	44,000	42,000
Regional courses	63,577	60,000	61,437	94,000	103,500
In-house courses	<u>65,017</u>	<u>50,000</u>	<u>32,060</u>	<u>37,000</u>	<u>44,500</u>
	<u>162,674</u>	<u>150,000</u>	<u>108,613</u>	<u>175,000</u>	<u>190,000</u>
Sales Income					
Resale of books	116,372	140,000	56,981	126,500	139,000
Resale of software	21,342	30,000	21,430	47,500	52,000
In-house publications and software	36,837	40,000	22,326	49,500	59,500
Video tape rental	1,600	3,000	800	1,500	1,500
Back issues	18,471	20,000	93,151	21,000	25,000
Shipping and handling fees	9,616	16,000	4,547	10,000	10,500
TEx and Metafont drawings	<u>891</u>	<u>1,000</u>	<u>0</u>	<u>0</u>	<u>0</u>
	<u>205,129</u>	<u>250,000</u>	<u>115,415</u>	<u>256,000</u>	<u>287,500</u>
Other Income					
Advertising	21,985	37,000	18,560	37,000	40,500
Mailing lists	1,746	2,000	1,113	2,000	2,000
Contributions	9,233	10,000	3,584	6,500	6,500
Promotional items	1,523	2,000	0	2,000	2,000
Interest	<u>16,162</u>	<u>17,000</u>	<u>6,542</u>	<u>15,500</u>	<u>15,500</u>
	<u>50,649</u>	<u>68,000</u>	<u>6,542</u>	<u>63,000</u>	<u>66,500</u>
TOTAL REVENUE	\$ 547,565	\$ 634,500	\$ 341,197	\$ 661,500	\$ 733,000

TeX Users Group
1989 Revenue and Expense Statement
January 1 – June 30 with projections and comparisons

E X P E N S E S

	Actual 1988	Budget 1989	Actual 1st half 1989	Projected 1989	Budget 1990
TUGboat Expenses					
Editorial	\$ 22,977	\$ 16,500	\$ 18,732	\$ 26,500	\$ 24,000
Computing	4,321	5,000	7,788	12,000	13,500
Printing	28,177	37,000	31,316	55,000	51,000
Mailing	<u>13,325</u>	<u>10,000</u>	<u>10,437</u>	<u>21,000</u>	<u>24,000</u>
	<u>68,800</u>	<u>68,500</u>	<u>68,273</u>	<u>114,500</u>	<u>113,500</u>
Annual Meeting and Course Expenses					
Meeting	10,683	10,000	8,622	18,000	17,000
Regional courses	41,620	28,000	16,052	32,000	35,000
In-house courses	<u>29,006</u>	<u>16,500</u>	<u>21,300</u>	<u>24,500</u>	<u>29,500</u>
	<u>81,309</u>	<u>54,500</u>	<u>45,974</u>	<u>74,500</u>	<u>81,500</u>
Sales Expenses					
Cost of goods sold (resale)	93,203	100,000	56,000	98,500	108,500
Cost of goods sold (software)	8,710	12,000	6,882	16,500	18,000
Cost of goods sold (in-house)	13,726	15,500	11,830	27,500	33,000
Bankcard processing fees	875	700	1,465	3,000	3,500
Bad debt expense	2,456	500	0	0	500
Shipping	<u>9,616</u>	<u>16,000</u>	<u>4,539</u>	<u>10,000</u>	<u>10,500</u>
	<u>128,586</u>	<u>144,700</u>	<u>67,379</u>	<u>155,500</u>	<u>174,000</u>
Personnel Expenses					
Salaries and wages	126,926	140,000	69,892	140,000	147,000
Payroll Taxes	15,898	22,000	7,534	15,000	15,000
Health Insurance	7,538	11,000	5,481	11,000	11,500
Pensions	15,310	42,000	8,276	16,500	17,000
Other employee benefits	<u>1,087</u>	<u>2,000</u>	<u>1,110</u>	<u>2,000</u>	<u>2,500</u>
	<u>166,759</u>	<u>217,000</u>	<u>92,293</u>	<u>184,500</u>	<u>193,500</u>
Operational Expenses					
Rent	8,225	12,800	6,059	12,000	13,000
Phone	4,429	6,000	2,687	5,500	6,000
Utilities	2,360	2,500	1,613	3,000	3,500
Office supplies	8,530	8,500	3,231	6,500	7,000
Bank lockbox fees	2,703	3,500	1,767	3,500	4,000
Postage and mailing	16,764	17,000	9,465	19,000	20,500
Printing/photocopying	11,453	18,000	8,603	17,000	18,000
Business Insurance	3,900	2,500	4,320	8,500	9,000
Legal and accounting fees	12,276	5,000	2,078	4,000	4,000
Off-site computer usage	4,444	1,500	661	1,500	1,500
Equipment depreciation	4,981	6,000	4,451	9,500	10,500
Other	<u>2,663</u>	<u>4,000</u>	<u>4,652</u>	<u>8,000</u>	<u>19,000</u>
	<u>82,728</u>	<u>87,300</u>	<u>48,250</u>	<u>98,000</u>	<u>116,000</u>
Other Expenses					
TUG Committees	11,307	10,000	5,853	11,500	12,500
TeXhax	3,045	12,000	6,271	12,500	13,000
BIBTEX Project	0	0	10,000	10,000	0
Staff training	3,572	5,000	671	1,500	1,500
Exhibits/meetings	<u>7,646</u>	<u>8,000</u>	<u>2,754</u>	<u>5,500</u>	<u>5,500</u>
	<u>25,570</u>	<u>35,000</u>	<u>25,549</u>	<u>41,000</u>	<u>32,500</u>
TOTAL EXPENSES	<u>\$ 553,752</u>	<u>\$ 607,000</u>	<u>\$ 349,055</u>	<u>\$ 668,000</u>	<u>\$ 710,500</u>
NET INCOME	<u>\$ (6,187)</u>	<u>\$ 27,500</u>	<u>\$ (7,858)</u>	<u>\$ (6,500)</u>	<u>\$ 22,500</u>

Request for Information

The TeX Users Group maintains a database and publishes a membership list containing information about the equipment on which TeX is (or will be) installed and about the applications for which TeX is used. This list is updated periodically and distributed to members with TUGboat, to permit them to identify others with similar interests. Thus, it is important that the information be complete and up-to-date.

Please answer the questions below, in particular those regarding the status of TeX and the hardware on which it runs. (Operating system information is particularly important in the case of IBM mainframes and VAX.) This hardware information is used to group members in the listings by computer and output device.

If accurate information has already been provided by another TUG member at your site, indicate that member's name and the same information will be repeated automatically under your name. If your current listing is correct, you need not answer these questions again. Your cooperation is appreciated.

- Send completed form with remittance (checks, money orders, UNESCO coupons) to:

TeX Users Group
P. O. Box 594
Providence, Rhode Island 02901, U.S.A.

- For foreign bank transfers direct payment to the TeX Users Group, account #002-031375, at:

Rhode Island Hospital Trust National Bank
One Hospital Trust Plaza
Providence, Rhode Island 02903-2449, U.S.A.

- General correspondence about TUG should be addressed to:

TeX Users Group
P. O. Box 9506
Providence, Rhode Island 02940-9506, U.S.A.

Name: _____
Home [] Address: _____
Bus. [] _____

QTY	ITEM	AMOUNT
	1989 TUGboat Subscription/TUG Membership (Jan.-Dec.) - North America New (first-time): [] \$35.00 each Renewal: [] \$45.00; [] \$35.00 - reduced rate if renewed before February 1, 1989	
	1989 TUGboat Subscription/TUG Membership (Jan.-Dec.) - Outside North America New (first-time): [] \$45.00 each Renewal: [] \$50.00; [] \$45.00 - reduced rate if renewed before February 1, 1989	
	TUGboat back volumes 1980 1981 1982 1983 1984 1985 1986 1987 1988 Circle volume(s) desired: vol. 1 vol. 2 vol. 3 vol. 4 vol. 5 vol. 6 vol. 7 vol. 8 vol. 9 Indiv. issues \$18.00 ea. \$18 \$50 \$35 \$35 \$35 \$50 \$50 \$50 \$50	

Air mail postage is included in the rates for all subscriptions and memberships outside North America.
Quantity discounts available on request.

TOTAL ENCLOSED: _____
(Prepayment in U.S. dollars required)

Membership List Information

Institution (if not part of address): _____

Date: _____

Title: _____

Status of TeX: [] Under consideration

Phone: _____

[] Being installed

Network address: _____

[] Up and running since: _____

[] Arpanet [] BITnet
[] CSnet [] uucp
[] JANET [] other _____

Approximate number of users: _____

Specific applications or reason for interest in TeX:

Version of TeX:

My installation can offer the following software or technical support to TUG:

[] Pascal

Please list high-level TeX users at your site who would not mind being contacted for information; give name, address, and telephone.

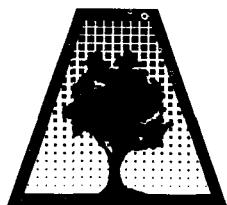
[] C

[] other (describe)

From whom obtained: _____

Hardware on which TeX is used:

Computer(s)	Operating system(s)	Output device(s)
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____



T_EX FOR THE HP 9000

ARBORTEXT INC.

ArborText introduces T_EX for the HP 9000,
Series 300 workstations.

The new release includes the
T_EX Package, Preview for
the X Window System,
DVILASER for
PostScript and
HP printers.

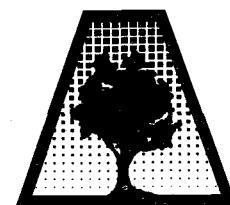
COMING SOON

- Improved DVILASER/PS
with expanded support for
Encapsulated PostScript®

- X11 versions of Preview for Sun and
Apollo workstations

- Enhanced μT_EX with support for extended or
expanded memory

COMMITTED TO THE FUTURE



ARBORTEXT INC.

535 West William Street, Suite 300, Ann Arbor, MI 48103 - (313) 996-3566 • FAX (313) 996-3573

Apollo is a trademark of Apollo Computer, Inc. HP 9000 is a trademark of the Hewlett-Packard Corp. PostScript is a registered trademark of Adobe Systems, Inc. Sun is a trademark of Sun Microsystems, Inc. T_EX is a trademark of the American Mathematical Society. The X Window System is a trademark of the Massachusetts Institute of Technology.

TEX Users

Take Note . . .

Computer Composition Corporation offers the following services to those who are creating their technical files using **TEX**:

- Convert your DVI files to fully paginated typeset pages on our APS-5 phototypesetters at 1400 dpi resolution.
- Files can be submitted on magnetic tape or PC diskettes.
- Provide 300 dpi laser-printed page proofs which simulate the typeset page. (Optional service \$1.50 per page)
- Macro writing and keyboarding from traditionally prepared manuscripts **in several typeface families** via the **TEX** processing system. Send us your manuscript for our review and quotation.
- Full keylining and camera work services, including halftones, line art, screens and full-page negatives or positives for your printer.
- Quick turnaround (**usually less than 48 hours!**) on customer supplied DVI files of 500 typeset pages or less.
- From DVI files: first 100 typeset pages at \$4.75 per page; 100 pages and over at \$3.50 per page. **Lower prices for slower turnaround service.**

*For further information and / or a specific quotation,
call or write Frank Frye or Tim Buckler*

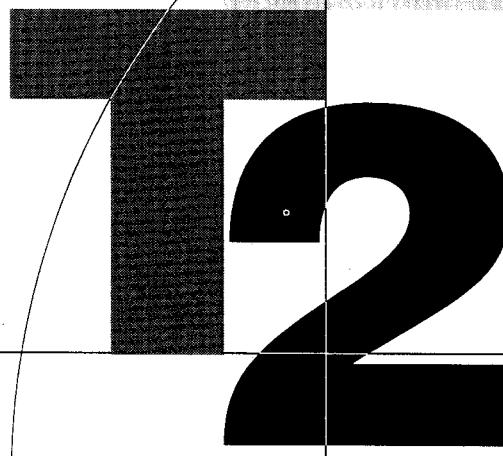


COMPUTER COMPOSITION CORPORATION

1401 West Girard Avenue • Madison Heights, MI 48071

(313) 545-4330 FAX (313) 544-1611

— Since 1970 —

T_EX Device Interfaces for VMS**PostScript****LaserJet****LN03**A large graphic element consisting of three stylized numbers: a dark gray '7' on the left, a white '1' in the middle, and a black '2' on the right. The '1' has a small dot at its top center. The '2' has a small circle at its bottom center. The entire graphic is set against a background of fine horizontal lines forming a grid.

Northlake Software
812 SW Washington, Suite 1100
Portland, Oregon 97205 USA

503-228-3383 fax 503-228-5662

The VMS T_EX specialists



TYPESETTING: JUST \$2.50 PER PAGE!

Send us your *TEX* DVI files and we will typeset your material at 2000 dpi on quality photographic paper — \$2.50 per page!

Choose from these available fonts: Computer Modern, Bitstream Fontware™, and any METAFONT fonts. (For each METAFONT font used other than Computer Modern, \$15 setup is charged. This ad was composed with PCTEX® and Bitstream Dutch (Times Roman) fonts, and printed on RC paper at 2000 dpi with the Chelgraph IBX typesetter.)

And the good news is: just \$2.50 per page, \$2.25 each for 100+ pages, \$2.00 each for 500+ pages! Laser proofs \$.50 per page. (\$25 minimum on all jobs.)

Call or write today for complete information, sample prints, and our order form. **TYPE 2000, 16 Madrona Avenue, Mill Valley, CA 94941. Phone 415/388-8873.**

— TYPE —
2000

Public Domain T_EX

The authorized and current versions T_EX software are available from *Maria Code - Data Processing Services* by special arrangement with Stanford University and other contributing universities. The standard distribution tape contains the source of T_EX and METAFONT, the macro libraries for A_MS-T_EX, L_AT_EX, SliT_EX and HP T_EX, sample device drivers for a Versetec and LN03 printers, documentation files, and many useful tools.

Since these are in the public domain, they may be used and copied without royalty concerns. They represent the official versions of T_EX. A portion of your tape cost is used to support development at Stanford University.

If you have a DEC VAX/VMS, IBM CMS, IBM MVS or DEC TOPS operating system, you will want to order a special distribution tape which contains "ready-to-run" T_EX and METAFONT. If you do not have one of these systems, you must perform a more involved installation which includes compiling the source with your Pascal compiler. Ready-to-run versions of T_EX are available for other systems from various sources at various prices. You may want to examine these before ordering a standard distribution tape.

The font tapes contain GF files for the Computer Modern fonts. While it is possible to generate these files yourself, it will save you a lot of CPU time to get them on tape.

All systems are distributed on 9 track, 1600 bpi magnetic tapes. If both a distribution tape and a font tape are ordered, they may be combined on a single 2400' reel, space permitting.

Your order will be filled with the current versions of software and manuals at the time it is received. If you want a specific version, please indicate that on your order.

Please use the form on the next page for your order. Note that postage, except domestic book rate is based on the item weights in pounds. If you want to place your order by telephone, please call (408) 735-8006 between 9:00 am and 2:00 pm West Coast time. Do not call for technical assistance since no one there can help you.

We normally have a good stock of books and tapes, so your order can be filled promptly — usually within 48 hours.

Make checks payable to *Maria Code - Data Processing Services*. Export orders must have a check drawn on a US bank or use an International Money Order. Purchase orders are accepted.

TeX Order Form

TeX Distribution tapes:

- Standard ASCII format
- Standard EBCDIC format
- Special VAX/VMS format Backup
- Special DEC 20/TOPS 20 Dumper format
- Special IBM VM/CMS format
- Special IBM MVS format

Tape prices: \$92.00 for first tape,
\$72.00 for each additional tape.

Font Library Tapes (GF files)

- 300 dpi VAX/VMS format
- 300 dpi generic format
- IBM 3820/3812 MVS format
- IBM 3800 CMS format
- IBM 4250 CMS format
- IBM 3820/3812 CMS format

Total number of tapes _____

Postage: allow 2 lbs. for each tape

Documents:

	Price \$	Weight	Quantity
TeXbook (vol. A) softcover	27.00	2	_____
TeX: The Program (vol. B) hardcover	40.00	4	_____
METAFONT book (vol. C) softcover	22.00	2	_____
METAFONT: The Program (vol. D) hardcover ..	40.00	4	_____
Computer Modern Typefaces (vol. E) hardcover	40.00	4	_____
IATEX document preparation system	27.00	2	_____
WEB language *	12.00	1	_____
TeXware *	10.00	1	_____
BibTeX *	10.00	1	_____
Torture Test for TeX *	8.00	1	_____
Torture Test for METAFONT *	8.00	1	_____
METAFONTware *	15.00	1	_____
Metamarks *	15.00	1	_____

* published by Stanford University

Payment calculation:

Number of tapes ordered	_____	Total price for tapes	_____
Number of documents ordered	_____	Total price for documents	_____
Add the 2 lines above			

Orders from within California: Add sales tax for your location. _____

Shipping charges: (for domestic book rate, skip this section)

Total weight of tapes and books _____ lbs.

Check One	<input type="checkbox"/> domestic priority mail	rate \$1.50/lb.
	<input type="checkbox"/> air mail to Canada and Mexico:	rate \$2.00/lb.
	<input type="checkbox"/> export surface mail (all countries):	rate \$1.50/lb.
	<input type="checkbox"/> air mail to Europe, South America:	rate \$5.00/lb.
	<input type="checkbox"/> air mail to Far East, Africa, Israel:	rate \$7.00/lb.

Multiply total weight by shipping rate. Enter shipping charges: _____

Total charges: (add charges for materials, tax and shipping) _____

Send to: Maria Code, DP Services, 1371 Sydney Drive, Sunnyvale, CA 94087.

Include your name, organization, address, and telephone number.

Are you or your organization a member of TUG? _____

Stocking Stuffers From Personal T_EX.

Get a 20% Discount on all orders placed before Jan. 10, 1990!

PTIJET FOR HP DESKJET. Full featured printer driver for HP DeskJet, PLUS. Laser quality output. **\$119**

PCT_EX + PTILASER + PTIVIEW. T_EX82, Version 2.9: professional formatting and typesetting results—for amateur prices. Includes INIT_EX, LaT_EX, AMS-T_EX, VANILLA Macro Pak, PCT_EX and LaT_EX manuals. Plus a PTILaser device driver, to take full advantage of your laser printer. Top performance and low cost make this our most popular package. **\$499**

PCT_EX + PTILASER. As above, but without the PTIView screen previewer. **\$399**

PCT_EX + PTIDOT + PTIVIEW. This package gives you all the T_EX and PTIView benefits, together with our dot-matrix device driver for reliable, low cost printing. **\$399**

PCT_EX + PTIJET + PTIVIEW. Same as the above package, but with PTIJet instead of PTIDot. Laser quality output **\$429**

PCT_EX + PTIJET. As above, without the PTIView screen previewer. **\$329**

PCMF—METAFONT for the PC. Lets you design fonts and create graphics. (Not for the novice.) pcMF Version 1.7 **\$195**

PTI LASER HP+, SERIES II. This device driver for the HP LaserJet Plus and Series II laser printers takes full advantage of the 512K resident memory. **\$195**

PTI LASER POSTSCRIPT. Device driver for PostScript printer; allows the resident fonts and graphic images to be used in T_EX in documents. **\$195**

PTI FONTWARE Interface Package. Software to generate Bitstream outline fonts at any size. (The Interface is necessary to use Bitstream fonts. Fonts are not included—order below) **\$95**

PTI FONTWARE WITH SWISS or DUTCH. Same as above but includes your choice of either Swiss or Dutch at a special bundled price **\$179**

PTIVIEW. Now get the viewer that was designed to take full advantage of PCT_EX. It works with all fonts, drivers and graphics cards, plus it contains many useful Zoom features! **\$139**

THE TABLE MACRO Package. Written by Michael Wichura of PiC_EX fame is a must for generating complex tables. This program is supported by a well-written manual and complete installation guide. **\$79**



To order, just dial

(415) 388-8853

12 Madrona Avenue Mill Valley, CA 94941 FAX: (415) 388-8865 VISA, MC accepted.

Requires: DOS 2.0 or later, 512K RAM, 10M hard disk. T_EX is an American Mathematical Society TM. PC T_EX is a Personal T_EX, Inc. TM. Manufacturers' names are their TMs. Outside the USA, order through your local PC T_EX distributor. Inquire about available distributorships and site licenses. This ad was produced using PCT_EX and Bitstream fonts.

**Publishing Companion translates
WordPerfect**

to

TEX

It doesn't take a TExpert to use TEX.

With **Publishing Companion**, you can publish documents using TEX with **little or no TEX knowledge**. Your WordPerfect files are translated into TEX files, so anyone using this simple word processor can immediately begin typesetting their own documents!

And now, K-Talk introduces **Publishing Companion** version 2.0, which translates **WordPerfect 5.0** files into TEX.

Other word processors are supported using Mastersoft's WordForWord file conversion utility, \$70.

Special Introductory Offer

Retail Price	\$249
Academic Discount Price	\$199
Introductory Price	\$179

This offer good until January 31, 1990. Upgrade from Publishing Companion v.1.XX is \$49.

For the power of TEX with the ease of a word processor, **Publishing Companion** is your "best friend" for desktop publishing.

For more information or to place an order, call or write:



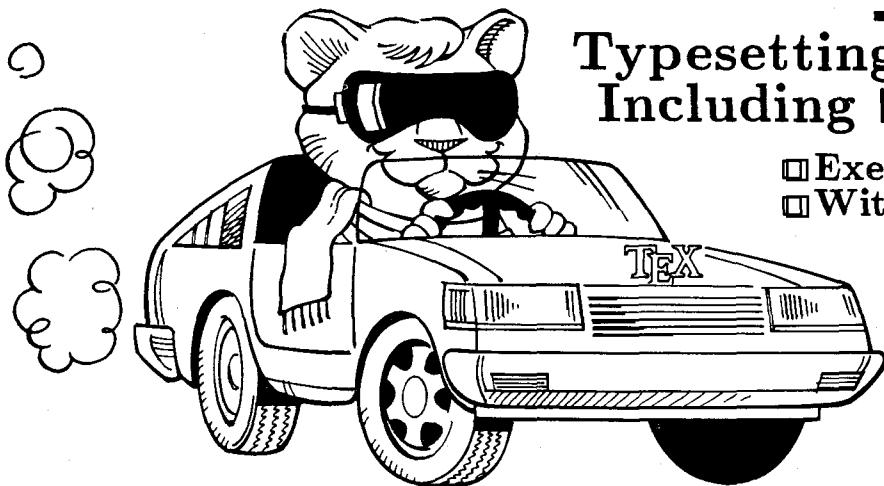
50 McMillen Ave
Columbus, Ohio 43201
(614) 294-3535

**DESKTOP PUBLISHING HAS NEVER BEEN SIMPLER
AND WILL NEVER BE THE SAME**

TurboTeX

**Typesetting Software
Including METAFONT**

Executables \$150
 With source \$300



TurboTeX Release 2.0 software offers you a complete typesetting package based on the TeX 2.95 and METAFONT 1.7 standards: preloaded plain TeX, L^AT_EX, INITEX, VIRTEX, and plain METAFONT interfaced to CGA/EGA/VGA/Hercules graphics; TRIP and TRAP certification; Computer Modern and L^AT_EX fonts, and printer drivers for HP LaserJet Plus/Series II, PostScript, and dot-matrix printers. New features in the HP LaserJet driver put PCX or TIFF graphics files directly into your TeX documents. This wealth of software fills over 10 megabytes of diskettes, and runs on your IBM PC, UNIX, OS/2, or VAX/VMS system.

■ Power Features: TurboTeX brings big-machine performance to your small computer. TurboTeX breaks the 640K memory barrier under MS-DOS on the IBM PC with our virtual memory sub-system. You'll have the same sized TeX that runs on multi-megabyte mainframes, with plenty of memory for large documents, complicated formats, and demanding macro packages that break other TeX implementations. On

larger computers, TurboTeX runs up to 3 times faster in less memory than the Stanford Pascal distribution.

■ Source code: Order the TurboTeX source in portable C, and you will receive more disks with over 85,000 lines of generously commented TeX, TurboTeX, METAFONT, and printer driver source code, including: our WEB system in C; PASCHAL, our proprietary Pascal-to-C translator; and preloading, virtual memory, and graphics code. TurboTeX meets C portability standards like ANSI and K&R, and is robustly portable to a growing family of operating systems.

■ TeX-FAX: Connects TeX to the FAX revolution. Send perfect TeX output instantly, anywhere, without scanning. With TeX-FAX, any FAX machine in the world becomes your output device! Complete with PC board and software for \$395 (4800 bps) or \$795 (9600 bps).

■ Desktop Publishing Interface Option: Converts TeX output (such as equations or tables) for direct use in programs like Ventura Publisher and Pagemaker. (\$50 for PC).

■ Availability & Requirements: TurboTeX executables for IBM PC's include the User's Guide and require 640K and hard disk. Order source code (includes Programmer's Guide) for other machines. Source compiles with Microsoft C 5.0 or later on the PC; other systems need 1 MB memory and a C compiler supporting UNIX standard I/O. Media is 360K 5-1/4" PC floppy disks; other formats at extra cost.

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

Ordering TurboTeX

Order by phone, FAX, or mail. Terms: Check with order (free media and ground shipping in US), VISA, Mastercard (free media, shipping extra); Net 30 to well-rated firms and public agencies (shipping and media extra). Discounts available for quantities or resale.

Ask for the free, 50-page Buyer's Guide.

The Kinch Computer Company

PUBLISHERS OF TURBOTeX

501 South Meadow Street
Ithaca, New York 14850
Telephone (607) 273-0222
FAX (607) 273-0484

Kinch

The Joy of \TeX



A Gourmet Guide to Typesetting with the $\text{AMS}\text{-}\text{\TeX}$ macro package

M. D. SPIVAK, Ph.D.

The Joy of \TeX is the user-friendly user's guide for $\text{AMS}\text{-}\text{\TeX}$, an extension of \TeX , Donald Knuth's revolutionary program for typesetting technical material. $\text{AMS}\text{-}\text{\TeX}$ was designed to simplify the input of mathematical material in particular, and to format the output according to any of various preset style specifications.

There are two primary features of the \TeX system: it is a computer system for typesetting technical text, especially text containing a great deal of mathematics; and it is a system for producing beautiful text, comparable to the work of the finest printers.

Most importantly, \TeX 's capabilities are not available only to \TeX pers. While mathematicians and experienced technical typists will find that \TeX allows them to specify mathematical formulas with great

accuracy and still have control over the finished product, even novice technical typists will find the manual easy to use in helping them produce beautiful technical \TeX t.

This book is designed as a user's guide to the $\text{AMS}\text{-}\text{\TeX}$ macro package and details many features of this extremely useful text processing package. Parts 1 and 2, entitled "Starters" and "Main Courses," teach the reader how to typeset most normally encountered text and mathematics. "Sauces and Pickles," the third section, treats more exotic problems and includes a 60-page dictionary of special \TeX niques.

Exercises sprinkled generously through each chapter encourage the reader to sit down at a terminal and learn through experimentation. Appendixes list summaries of frequently used and more esoteric symbols as well as answers to the exercises.

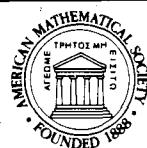


ISBN 0-8218-2999-8, LC 85-7506
290 pages (softcover), 1986
AMS Indiv. Memb. \$29, AMS Inst.
Memb. \$32, List price \$36
To order specify JOYT/T

Free shipment by surface. For air
delivery, add: 1st book \$5, each
add'l \$3, max. \$100

PREPAYMENT REQUIRED. Order from
American Mathematical Society
P.O. Box 1571
Annex Station
Providence, RI 01901-1571
or call 800-556-7774 to use VISA or MasterCard.
Prices subject to change.

Updated T_EX Products



A_MS-T_EX Version 2.0 (Available January 1990)

A_MS-T_EX, the T_EX macro package that simplifies the typesetting of complex mathematics, has been updated to version 2.0. *A_MS-T_EX* is intended to be used in conjunction with AMSFonts 2.0 (see below). However, if the purchaser does not need the extra symbols and Euler characters found in AMSFonts, *A_MS-T_EX* can be run without AMSFonts. (*A_MS-T_EX* 2.0 **cannot** be used with previous versions of AMSFonts.) *A_MS-T_EX* is available on IBM or Macintosh diskettes—either format may be uploaded to many mainframe computers.

AMSFonts Version 2.0 (Available January 1990)

AMSFonts 2.0 are designed either for use with *A_MS-T_EX* 2.0, or for use with Plain T_EX. (AMSFonts 2.0 **cannot** be used with previous versions of *A_MS-T_EX*.) There will be two distributions of fonts: one that can be used on PC's as well as mainframes, and one that can be used on a Macintosh with *Textures*. The fonts that will be included on these distributions are:

Font Name	Description	Point Sizes	Font Name	Description	Point Sizes
CMCSC	CM Caps and Small Caps	8-9*	MSAM	Symbols	5-10
CMMIB	CM Math Italic Boldface	5-9*	MSBM	Symbols (w/Blackboard Bold)	5-10
CMBSY	CM Bold Symbols	5-9*	WNCYR	Cyrillic Upright	5-10**
EURB	Euler Cursive Boldface	5-10	WNCYI	Cyrillic Italic	5-10**
EURM	Euler Cursive Medium	5-10	WNCYB	Cyrillic Boldface	5-10**
EUFB	Euler Fraktur Boldface	5-10	WNCYSC	Cyrillic Caps and Small Caps	10**
EUFM	Euler Fraktur Medium	5-10	WNCYSS	Cyrillic Sans Serif	8-10**
EUSB	Euler Script Boldface	5-10			
EUSM	Euler Script Medium	5-10			

* 10 point is included in the standard T_EX distribution.

** Developed by the University of Washington

AMSFonts for PC or mainframe

- Font Resolution: 118, 180, 240, 300, 400 dpi (one resolution per order).
- Magnification: All the standard T_EX magnifications will be included. The standard magnifications are: 100, 109.5, 120, 144, 172.8, 207.4, and 248.8%.
- Format: Available on both 3.5" and 5.25" diskettes, in either high or low density (the default will be 5.25", high density).

AMSFonts for use on a Macintosh with *Textures*

- Font Resolution: 72, 144, and 300 dpi (all resolutions included in each order).
- Magnification: The standard distribution will include fonts at 100% and 120%. An extended distribution, which will contain all the standard T_EX magsteps, will also be available.
- Format: The Macintosh fonts will be available on double-sided double-density 3.5" diskettes.

PRICES: *A_MS-T_EX*: List \$30, AMS member \$27

AMSFonts: List \$45, AMS member \$41

A_MS-T_EX and AMSFonts: List \$65, AMS member \$59

Shipping and handling: \$8 per order in the US and Canada, \$15 elsewhere.

HOW TO ORDER: Prepayment is required. Free upgrades are available for those who have purchased previous versions. When ordering AMSFonts for the PC, specify desired resolution, diskette size, and diskette density.

For more information: Call the AMS at (401) 272-9500, or (800) 556-7774 in the continental U.S. or write to: T_EX Library, American Mathematical Society, P.O. Box 6248, Providence, RI 02940.

TEX Plus

TM

NOW
only

\$195.00

Price includes MAXview
screen previewer until
November 30



TEX Plus provides a quality TEX environment for PCs and compatibles at an unbeatable price. TEX Plus includes:

TEXWRITE: a specially designed editor/interface with TEX related functions, multiple file handling, access to TEX and drivers from within the editor, and more.

CTEX: a full implementation of TEX version 2.98 that includes virtual memory support to enable you to run your favorite macro package in a minimum of RAM.

TEXPRINT drivers: device drivers for the HP LaserJet Plus/Series II and PostScript printers with features that include landscape printing, support for native fonts, font conversion utilities and more, from a menu-driven interface.

For more information or to place an order contact:

In Canada:

Micro Publishing Systems, Inc.,
1273 Clyde Avenue,
West Vancouver, B.C., V7T 1E6
(604) 926-0500

In the United States:

Oregon House Software,
12894 Rices Crossing Road,
Oregon House, CA 95962
(916) 692-1377

Also available from the TEX Users Group.

MacroTEX

A T_EX Macro Toolkit.

Clients include:

MacroTEX:
*Publishing Companies,
 Research Labs,
 Academic Departments,
 T_EX Users world-wide.*

**Macro Writing for
 Publishing:**
*Addison-Wesley
 MIT Press*

*Brooks-Cole/Wadsworth
 Prentice-Hall
 Academic Press
 John Wiley and Sons
 Pronk and Associates*

**Macro Writing for
 Technical Documents:**
*Shell Research
 American Physical Society
 Technical Typesetting*

**Macro Writing for
 Software Companies:**
*Alpha Software
 Cytel Software
 Intermetrics Corp.
 Saddlebrook Corp.
 Grass Valley Group
 Technical Support Software*

**T_EX-PostScript
 Interactive Macros:**
MIT Press, Addison-Wesley

Teaching T_EX:
*Beginning, Intermediate and
 Advanced Macro Writing;
 DEC, MIT, TUG, NCAR.*

Maximum flexibility

Style files: Generic, Book, Report, Software Documentation, Letter and Note style, each notated and easily adapted to your design.

Macros to help form your own font families, including PostScript font families.

Modular format. Separate files are used for separate functions: listing macros, tables macros, or indexing macros, for example, are called in only when needed.

All or portions of MacroTEX may be added to existing macro files.

Source code included.

Minimum hassle

All the features you need for document preparation:

- Table of Contents, List of Figures, List of Tables generation
- Multilevel headline and footline, Modular page numbering
- Section heads, Cross-referencing, Listing
- Complete table macros, including tables that continue across pages
- Figure, Table and Program Captions
- Partial page figure that text will wrap around
- Theorem environments, Left-justified equations
- Verbatim that continues across pages, Screen Simulation
- Bibliography, Glossary, Index generation and formatting
- Utilities: Margin control, Footnotes and Endnotes, Diagonal lines, Drop Caps, Margin notes, Mailing labels, Font charts...
- Slides

Single User: \$200, Site Licenses Available.

Coming soon: **Post-T_EX**, our new T_EX-PostScript package that passes information from PostScript to T_EX to insert encapsulated PostScript graphics. It passes information from T_EX to PostScript to position grey screens behind code examples, highlight screen simulations, and position a grey screen in a particular table column entry, as well as and other useful T_EX-PostScript macros. Post-T_EX works with both T_EX and L_AT_EX.

T_EXnology Inc.

Amy Hendrickson

*57 Longwood Avenue, Brookline MA 02146
 T_EX, L_AT_EX, and PostScript Consulting*

617 / 738-8029

MicroPress presents

TEX for the 90's

VectorTM TEX

Retain all the advantages of TEX and

- Save megabytes of storage.
- Instantly generate any font in any size in any variation (5-70 points).
- Automatically create compressed, slanted, smallcaps, outline or shaded fonts.
- Use either CM-compatible Vector fonts or MicroPress professional typefaces, including Tempo-Roman, Avon Guard, Helvettot,...

Includes the VTEx typesetter, 10 instantly scalable typefaces, VVIEW (arbitrary magnification on EGA, CGA, VGA, Hercules, AT&T,...), VLASER (HP LaserJet), VPOST (PostScript), VDOT(Epson, Panasonic, NEC, Toshiba, Proprinter, Star, Deskjet) and manuals.

List price \$399 Introductory offer \$249

Introductory offer expires on January 1, 1990. S/H \$5. COD add \$5. WordPerfect Interface add \$100. Demo \$3. Site licenses available. Dealers inquires welcome. Professional typefaces and METAFONT sources available for older implementations of TEX. Discounts for PCTEX users.



67-30 Clyde Street, Rm. 2N
Forest Hills, New York 11375
(718)-575-1816

VTEx is a trademark of MicroPress Inc.

Other products are trademarks of respective companies.

This ad was typeset by VTEx using scalable fonts on a LaserJet.



Announcement and Call for Participation Electronic Publishing '90

International Conference on Electronic Publishing, Document Manipulation, and Typography
September 18-20, 1990
National Institute of Standards and Technology
Gaithersburg, MD (Washington, DC area)

EP90, the third in a series of international conferences, will bring together researchers in the areas of electronic publishing, document manipulation, and typography. Our definition of "electronic publishing," encompasses all aspects of computer-assisted preparation, presentation, transmittal, storage, and retrieval of linear and non-linear documents. It includes the design of the related computer systems, the design of their components, and the theory that underlies such systems. Careful presentation of important earlier results inadequately described in the open literature also is appropriate. Papers should present previously unpublished original research results and should be supported by experience. Guidelines for the exhibition of commercial and research systems are available from the conference address.

Specific topic list:

- Document preparation systems: design, concepts, experience, theoretical and algorithmic foundations
- Document component identification and manipulation.
- Hypertext systems, particularly those that provide insights on the characteristics of these systems.
- Font design and use: design and evaluation of computer-based tools, techniques and goals, visual issues.
- Representations specialized for electronic display: fonts, presentations, etc.
- Graphics and document illustration.
- Page description languages.
- Critical analyses of proposed and established international standards. Experience with standards.
- Managing the complexities introduced by scale. Scaling up to large documents.
- Hardware-environment issues: Printers, displays, networks, workstations.
- Distributed document manipulation systems (in the sense of distributed processing).
- Specialized documents (e.g., catalogs, programs, manuals, & proposals). Do they differ from generic documents?
- Text and document recognition (recognition of physical and/or logical structure from a printed document).
- Heterogeneous target reader populations (e.g., multi-lingual).
- Application of database technology to document preparation.
- Integration of documentation tools with other tools, e.g., CASE, CAD-CAM.

We request seven copies of full papers (in English), limited to the equivalent of ten pages, 10 point on 12.

Conference Address:

Lawrence A. Welsch/EP90
Building 225, Room B252
National Institute of Standards and Technology
Gaithersburg, MD U.S.A. 20899
Telephone: (301) 975-3345 (8:30-5:00 U.S. Eastern Time)
Electronic mail: ep90@asl.ncsl.nist.gov FAX: (301) 590-0932

Important dates:

January 31, 1990: full papers due
April 1, 1990: acceptance notification
May 15, 1990: final paper due
September 18-20, 1990: EP90

Peter King (Conference Chair), University of Manitoba, Canada
Richard Furuta (Program Chair), University of Maryland, USA
Debra Adams (Exhibition Chair), Xerox Palo Alto Research Center, USA
Larry Welsch (Local Arrangements and Publicity Chair), NIST, (National Institute of Standards and Technology), USA

Jacques André
Patrick Baudelaire
Richard J. Beach
Charles Bigelow
David F. Brailsford
Heather Brown

Donald D. Chamberlin
Giovanni Coray
R. W. Davy
Irene Greif
Vania Joloboff
Brian Kernighan

Dario Lucarella
Pierre MacKay
Norman Meyrowitz
Robert A. Morris
Jurg Nievergelt
Vincent Quint

Brian Reid
Richard Rubinstein
Alan Shaw
Andries van Dam
Hans van Vliet
Jan Walker

Sponsor: National Institute of Standards and Technology
Co-sponsors: ACM (pending), EPSIG/American Association of Publishers, University of Maryland, Xerox PARC



Publishing Services



From the Basic

The American Mathematical Society can offer you a basic **TeX** publishing service. You provide the DVI file and we will produce typeset pages using an Autologic APS Micro-5 phototypesetter. The low cost is basic too: only \$5 per page for the first 100 pages; \$2.50 per page for additional pages, with a \$30 minimum. Quick turnaround is important to you and us ... a manuscript up to 500 pages can be back in your hands in just one week or less.

To the Complex

As a full service **TeX** publisher, you can look to the American Mathematical Society as a single source for all your publishing needs.

Macro-Writing

TeX Problem Solving

Autologic Fonts

Keyboarding

Art and Pasteup

Camera Work

Printing

Binding

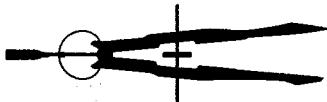
For more information or to schedule a job, please contact Regina Girouard, American Mathematical Society, P.O. Box 6248, Providence, RI 02940 or call 401-272-9500 or 800-556-7774 in the continental U.S.

DeskJet driver for **TeX**

With The Toolsmith's DVI driver and the Hewlett-Packard DeskJets you get:

- Print Quality—the precision of laser printers (300 dpi) without the price.
- Speed—with downloaded fonts, ≈1 page per minute on a DeskJet, faster on a Plus.
- Quiet—non-impact printing and no fan.

You are only limited by the page size and your **TeX**pertise. \$100 for the DeskJet DVI driver (shipping and any sales tax included). To order or for more information, call or write:



The Toolsmith

P.O. Box 5000
Davis, CA 95617
(916) 753-5040

Requires IBM PC or compatible, hard disk, 512K or more RAM, and MS-DOS 2.11 or later. This ad, including the logo, was created on a DeskJet with **TeX** and METAFONT.

THE WRITE STUFF, TEXNICALLY SPEAKING

TeX is the write stuff

***TeX** is the powerful publishing system that is guaranteed to make any document you write easier to read... and guaranteed to make that document say the right stuff about you!*

*Micro Programs, Inc. is your source for **TeX** and related ArborText products for IBM PC and Sun workstations.*

Call Bob Harris on (516) 921-1351 and get the name of the dealer nearest you.

MICRO PROGRAMS, INC.
251 JACKSON AVENUE
SYOSSET
NY 11791

Integrated Computer Software, Inc

Stephan v. Bechtolsheim

T_EX Related Services:

- Macro package design
- Specialized output routines
- Training
- Consulting

dvi2dvi Processor:

- Contact me for further details.
- See also article in this issue.

POSTSCRIPT Related Services:

- Programming and Training

X Window System Programming

L_AT_EX Related Services:

- Style file modifications
- Training
- Consulting

"T_EX in Practice" (Springer Verlag)

Comprehensive book on T_EX,
1100 pages, to be published
in March of 1990.

T_EXPS Software Package

NeWS Programming

*2119 Old Oak Drive
West Lafayette, IN 47906*

(317) 463 0162

Index of Advertisers

- | | |
|-----------|------------------------------------|
| 454-5,460 | American Mathematical Society |
| 445 | ArborText |
| cover 3 | Blue Sky Research |
| 446 | Computer Composition |
| 459 | EP'90 |
| 449-50 | DP Services |
| 461 | Integrated Computer Software, Inc. |
| 452 | K-Talk Communications |
| 453 | Kinch Computer Company |
| 460 | Micro Programs, Inc. |
| 456 | Micro Publishing Systems, Inc. |
| 447 | Northlake Software |
| 451 | Personal T _E X Inc. |
| 462 | T _E X Users Group |
| 457 | T _E Xnology, Inc. |
| 460 | The Toolsmith |
| 448 | Type 2000 |
| 458 | VectorT _E X |

TEX USERS GROUP

Have you ever wished you knew how to use the full potential of TeX or L^AT_EX properly?

Can't get away to take a course? Let us help you!

Here's your opportunity to learn more about **TEX**.

If you know of six to fifteen individuals interested in learning more about T_EX or L_AT_EX, we'll come to you!* Check around your own organization; check with other organizations in your locality. (If you have less than six, let us know; we may be able to combine two or more small groups.)

NAME OF CONTACT/TELEPHONE: _____

Names of individuals interested in courses Dept./Co. Level of Instruction

Please check one:

- We have a training facility equipped with computer terminals, PC's or Macs.
 We do not have a training facility.

Notes:

Completion of this inquiry does not obligate you in any way.

Fees will vary depending on the course offered and the number of participants.

Some of the courses we offer are:

- **TEX**: all levels, macro writing,
output routines, wizard
 - **LATEX**: all levels, style files
 - **METAFONT**
 - **PostScript**

If you would like to have detailed descriptions of the courses we offer or need additional information, contact Charlotte Laurendeau at the TUG office:

P. O. Box 9506
Providence, RI 02940, U.S.A.
401-751-7760

In addition to having conducted these courses over the past several years throughout the U. S. and Europe as part of our Regional Course program, we have offered courses on-site for a number of organizations, including:

American Mathematical Society,
Beckman Instruments, Inc.,
Brookhaven National Lab,
Digital Equipment Corp.,
Digital Equipment Corp. (England),
Institute for Advanced Study (Princeton),
Lawrence Livermore National Lab,
Lawrence Berkeley National Lab,
Los Alamos National Lab,
Rutgers University,
Science Applications (Las Vegas),
University of Delaware,
University of Washington, and
Woods Hole Oceanographic Institute.

In many cases TUG has conducted several different courses for each these organizations, especially Los Alamos National Lab, where more than two dozen courses at all levels of *TEX* and *LATEX* have been conducted since 1985.

*Based on availability of a properly equipped training facility.