TUGboat

Volume 16, Number 4 / December 1995

TEX Users Group

Memberships and Subscriptions

TUGboat (ISSN 0896-3207) is published quarterly by the TEX Users Group, Flood Building, 870 Market Street, #801; San Francisco, CA 94102, U.S.A.

1996 dues for individual members are as follows:

- Ordinary members: $55
- Students: $35

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

TUGboat subscriptions are available to organizations and others wishing to receive *TUGboat* in a name other than that of an individual. Subscription rates: $70 a year, including air mail delivery.

Second-class postage paid at San Francisco, CA, and additional mailing offices. Postmaster: Send address changes to *TUGboat*, TEX Users Group, 1850 Union Street, #1637, San Francisco, CA 94123, U.S.A.

Institutional Membership

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

Printed in U.S.A.

Addresses

All correspondence, payments, parcels, etc.
TEX Users Group
1850 Union Street, #1637
San Francisco,
CA 94123 USA

If you are visiting:
TEX Users Group
Flood Building
870 Market Street, #801
San Francisco,
CA 94102, USA

Telephone

+1 415 982-8449

Fax

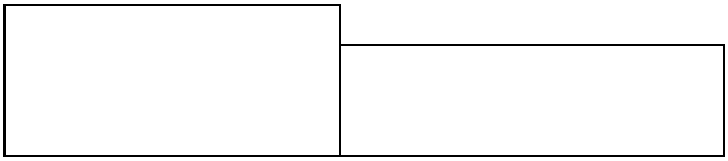
+1 415 982-8559

Electronic Mail

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

The mass market in the U.S. has always held a "good enough" mentality; consumers have always preferred a marginally adequate product that is priced lowest over a superior product at a higher price.

Caren Eliezer
The Seybold Report on Desktop Publishing (October 2, 1995)



COMMUNICATIONS OF THE TEX USERS GROUP

EDITOR BARBARA BEETON

VOLUME 16, NUMBER 4 • DECEMBER 1995
SAN FRANCISCO • CALIFORNIA • U.S.A.

TUGboat

During 1996, the communications of the TEX Users Group will be published in four issues. One issue will contain the Proceedings of the 1996 TUG Annual Meeting. One issue (Vol. 17, No. 4) will be a theme issue, edited by a guest editor, with participation by invitation.

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

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

Submitting Items for Publication

The next regular issue will be Vol. 17, No. 1; deadlines for that issue will have passed by the time this issue is mailed. Mailing is scheduled for March. Deadlines for other future issues are listed in the Calendar, page 428.

Manuscripts should be submitted to a member of the *TUGboat* Editorial Board. Articles of general interest, those not covered by any of the editorial departments listed, and all items submitted on magnetic media or as camera-ready copy should be addressed to the Editor, Barbara Beeton (see address on p. 351).

Contributions in electronic form are encouraged, via electronic mail, on diskette, or made available for the Editor to retrieve by anonymous FTP; contributions in the form of camera copy are also accepted. The *TUGboat* "style files", for use with either `plain` TEX or LATEX, are available "on all good archives". For authors who have no network FTP access, they will be sent on request; please specify which is preferred. Write or call the TUG office, or send e-mail to `TUGboat@ams.org`.

This is also the preferred address for submitting contributions via electronic mail.

Reviewers

Additional reviewers are needed, to assist in checking new articles for completeness, accuracy, and presentation. Volunteers are invited to submit their names and interests for consideration; write to `TUGboat@ams.org` or to the Editor, Barbara Beeton (see address on p. 351).

TUGboat Advertising and Mailing Lists

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

See page 351 for addresses.

Other TUG Publications

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

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

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.

Adobe Illustrator is a trademark of Adobe Systems Incorporated.
CorelDraw is a registered trademark of Corel Corporation.
Fontographer is a registered trademark of Altsys Corporation.
MS/DOS is a trademark of MicroSoft Corporation
METAFONT is a trademark of Addison-Wesley Inc.
PC TEX is a registered trademark of Personal TEX, Inc.
PostScript is a trademark of Adobe Systems, Inc.
TEX and $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX are trademarks of the American Mathematical Society.
Textures is a trademark of Blue Sky Research.
UNIX is a registered trademark of X/Open Co. Ltd.

Jon Radel
P. O. Box 2276
Reston, VA 22090-0276 U.S.A.
jon@radel.com

Sebastian Rahtz
Production Methods Group
Elsevier Science Ltd
The Boulevard
Langford Lane, Kidlington
Oxford OX5 1GB, U.K.
Sebastian.Rahtz@cl.cam.ac.uk

Tomas Rokicki
725B Loma Verde
Palo Alto, CA 94303 U.S.A.
rokicki@cs.stanford.edu

Andrey Slepukhin
Lavra
Sergiev Posad
Russia
pooh@shade.msu.ru

Christina Thiele
15 Wiltshire Circle
Nepean K2J 4K9, Ontario Canada
cthiele@ccs.carleton.ca

Norm Walsh
Online Publishing
O'Reilly & Associates, Inc.
90 Sherman Street
Cambridge, MA 02140-3244 U.S.A.
norm@ora.com

Hermann Zapf
Seitersweg 35
D-64287 Darmstadt, Germany

Jiří Zlatuška
Faculty of Informatics
Masaryk University
Burešova 20
602 00 Brno, Czech Republic
zlatuska@muni.cz

Figure 1: Flying into Moscow

Figure 2: Meeting site in Dubna

Figure 3: Anyone for sailing on the Volga?

For more information on TUG'96 — see page 428.

Opening Words

Writing the Future is Reading the Past

Michel Goossens
TUG President

The year 1995 now lies behind us, and in less than four years we will have reached the end of the 20th century, and the end of the second millennium, at least for those using a "Western" cultural reference frame. Let me note that this is by far no longer the majority of the inhabitants of this planet, so that the notion of "end of an era", is completely artificial.

Indeed, in building for the future we should always take the long view to make sure that decisions we consider urgent today have no unforeseen negative side-effects a few years from now. Knuth, when he decided to develop a system that would allow him to typeset — in a completely digital way — the second edition of "*The Art of Computer Programming*" — thought it was going to take him a "short" sabbatical year. In the end he needed seven (the magic number?) years to create TEX and METAFONT plus the Computer Modern fonts. Today many enthusiasts think they can improve in no time upon this or that aspect of digital typography, be it better-looking fonts, optimized multi-column algorithms, perfect float placement, etc. By now detailed studies by the NTS (New Typesetting System) and LATEX3 initiatives, respectively evolutionary successors to TEX and LATEX, have shown that it is far from trivial to come up with efficient general algorithms for vastly improving today's TEX/LATEX paradigms, and that several man-years of work will still be needed in these areas before Mr. Joe User will be able to profit from them directly.

Therefore, it came as a relief to me that just over two years ago the LATEX3 team decided to develop LATEX 2_ε, whose prime aim was to bring together all variant flavours of LATEX into one format, and to provide often-requested simple extensions to LATEX's original functionality (as defined by Lamport in the middle eighties). And, it can be said, we owe the LATEX 2_ε team a lot of gratitude for their foresight and thorough work.

Also in the field of the typesetting engine itself, the official release of Ω took place in March of this year, and a consolidated version 1.2 appeared in December 1995. The Ω system extends TEX's functionality mainly in the area of multi-language typesetting by extending all data-types to 16 bits, and using Unicode as internal encoding. Some applications based upon Ω are already available (see the articles of Plaice and Haralambous in recent *TUGboat*s).

In the summer, version 1 of ε-TEX was announced. This successor to TEX is, like Ω, based upon the original TEX code, and provides extensions in the areas of additional control over expansions, rescanning tokens, environmental enquiries, additional marks and debugging facilities, bi-directional typesetting, and a few supplementary primitives.

I want to stress once more that both the Ω and ε-TEX developments are clearly evolutionary, paying a lot of attention to backward compatibility, and tackle one or two precise problem areas at a time. This ensures that users are presented with a well-known and trusted upgrade path, so that the transition between present and augmented versions of TEX will be a natural process. I am therefore sure that ε-TEX and Ω will both have a brilliant future.

I am equally confident that LATEX 2_ε will, with each half-yearly release, become more robust and provide a clear reference frame to act as the definitive markup system, which can be used by all those who want to benefit from the advantages of generic markup and interchange of documents.

After the efforts of the CTAN pioneers a few years ago to build a reliable TEX Internet archive structure, you will find in this issue of *TUGboat* the conclusions of the "TDS" (TEX Directory Structure) Technical Working Group. This initiative's aim is to define a "plug-and-play" run-time directory structure for TEX files that can be used by all present-day operating systems and is ISO 9660-compliant.

In the field of fonts, an optimized version of the DC fonts has just appeared, and it is foreseen that the long-awaited European (or Extended) Computer Modern (EC) family will be available by the end of 1996. At the same time work is going on to define a "Text Companion" (TC) font to allow a clearer separation between real math and more general symbols (for currencies, trademarks, some arrows, musical notes, etc.). At present they are mixed in the CM math fonts, and it would be more logical and practical to make all general characters uniformly available to TEX users by grouping them in their own font. At the same time the LATEX3 team has published some ideas on possible layouts for 256-character math fonts,[1] so that it is hoped that also in this area a consolidation effort can take place in the not-too-distant future.

Readers of *TUGboat* will, without doubt, have

[1] See TTN 4 #2, pp. 17 – 18.

remarked the prominence given to hypertext developments this year, with articles on HTML/SGML, PDF and Acrobat, T.V. Raman's Aster, translators to and from LaTeX, etc. It is probably true to say that these new tools will become ever more important in the future and provide an ideal bridge to link excellent typography based on LaTeX with instantaneous availability of information worldwide. The same well-structured generically marked-up SGML or LaTeX document can form the basis of preparing the various views of the information, and this aspect of re-use is a sure winner for these systems in the rapidly changing world of electronic publishing.

By skimming through your four issues of *TUGboat* of 1995 you certainly will find other areas which you consider important or interesting. I would only like to mention the many articles on the non-English use of TeX (encodings, hyphenation patterns, fonts) and the revival of METAFONT as an intelligent high-level font-generation and drawing tool.

As you go through this final issue of 1995, you will also notice the use of Russian alongside English in two articles, and the TUG 1996 conference ammouncement. These articles, and the fact that Dubna in Russia was chosen to host TUG'96 show that TeX is now truly global, and it is therefore time that the TUG adapts to that situation by taking into account the wishes and needs of every TeX user in the world by acting as a central knowledge repository of TeX-related developments. TUG, as an organization, realizes that it is impossible to serve a particular user base better than local TeX User Groups, where they exist. Yet, it is important to provide a forum for discussing all TeX-related activities in a coherent framework, where all important information is kept in a unique place. TUG and *TUGboat* must even more than in the past be the voice and up-to-date road-map of the TeX community. And that is why we want to open up *TUGboat* to all TeX users and user organizations in the world.

Already in St. Petersburg Beach in Florida at TUG'95 last August, I explained that TUG was not in a rosy financial situation. With respect to last year the number of our members has decreased by about 20%, while printing and postal charges have increased. We therefore expect a shortfall of about $30,000 for 1995, and unless drastic actions are taken TUG will have great difficulties surviving.

One of our problems last in 1994/95 was the irregularity of *TUGboat*, TUG's flagship publication. For various reasons it had become impossible to keep to the three-monthly schedule of *TUGboat* and we think that quite a few of our readers did not renew their 1995 membership just for that reason. Therefore, since August, a Production Team[2] has worked very hard to publish these last five issues of *TUGboat*, and I think we have succeeded in the area of catching up with the backlog. Some compromises and trade-offs had to be taken, but we are now quite confident that this team-based production scheme will make it possible to have *TUGboat* appear regularly in the future. But at the same time we must tackle the financial problem, and, therefore, the TUG's Board of Directors has unanimously decided to discontinue the publication of *TeX and TUG News* as a separate item, and to merge it with *TUGboat*, with immediate effect. That is, there will be no further issues of TTN after 4,3. This has the advantage of publishing all TeX-related information in one place, and at the same time saves up to $10,000 in printing and postal costs. We hope to be able in the future to redirect part of those savings to improve the free services that we offer the TeX community.

TUG regularly renews part of its Board of Directors. Therefore, on page 441 you can find a nomination form for the 1996 elections, which are to take place this spring. All 1996 TUG members can stand for election and I sincerely hope that many of you will take this opportunity to show your support for TUG by running for the Board. This is without doubt one of the best ways to actively participate in the life of the TeX community and contribute in shaping its future.

Knuth promised at TUG's 10th Annual meeting in Stanford in 1989 that he would attend henceforth all TUG Conferences that would be a power of 2. And as this year's Conference in St. Petersburg Beach was TUG's $16 = 2^{2^2}$th Annual Meeting, Don Knuth honored us with his presence. I am sure that all those who attended TUG'95 and profited from his stimulating questions, interesting suggestions, and gentle remarks will already look forward to meet Don again at TUG's 32nd meeting in ... 2011. The venue? Well, that's the only unknown, isn't it, because I am confident that TeX will still be alive and well!

Let me finish by wishing you and your families all the best for the new year, with a lot of fun, motivating work, and, above all, health and happiness. And, I sincerely hope to see you all in 1996, in Dubna at TUG'96, or anywhere else in the world where our (TeX or non-TeX) roads might cross.

[2] This team, under the leadership of *TUGboat*-editor Barbara Beeton, consists of Mimi Burbank, Wietse Dol, Robin Fairbairns, Sebastian Rahtz, Christina Thiele, and myself. Malcolm Clark helped us as editor of *TUGboat* 16(2).

Editorial Comments

Barbara Beeton

A day at a small press book fair

Early this fall, over the weekend of the Columbus Day holiday (in the U.S., this is celebrated on the Monday closest to October 12, especially in locations with a sizeable population of Italian descent) I had the opportunity to attend a book fair organized by my favorite bookshop, Oak Knoll Books of New Castle, Delaware. Oak Knoll deals in "books about books" — typography, printing, design, ... bibliography — and has a strong commitment to the continued health of the book arts. In support of this, the proprieter, Bob Fleck, and his staff have undertaken to arrange "events" where small presses and craftsmen in the traditions associated with printing can get some exposure and meet prospective new customers. The book fair was the second in what promises to be an annual tradition.

The highlight of the fair was a talk by Ian Mortimer, the head of the printing house I. M. Imprimum, London, on the printing of a definitive edition of specimens of decorative wood types created in the early 1800s by Louis John Pouchee. The wood blocks, carved in solid boxwood, and intended as masters for stereotypes, were found among the effects of the Caslon type foundry at the time of its closing, without identification. The blocks are now held by the St Bride printing library in London, and the printing was undertaken under library auspices.

At the beginning of the project, nothing was known about the provenance of the types, not even the name of the creator. Nonetheless, the quality of design and workmanship of the types was exceptional, and no printed specimens were known. Aside from the effects of age — 150 years of disuse and lack of attention was evidenced in some minor warping and splitting and in the absence of some letters in a couple of the alphabets — the condition of the blocks was such that it is believed they had never before been used directly for printing, so the work described is in a very real sense a first edition.

Mr. Mortimer illustrated his description of the printing process with slides that illuminated the steps to even the least knowledgeable of the audience. I learned a great deal about printing from "real" type in order to ensure uniform distribution of ink on the printed surface — how to ensure that the type heights are even, how to use "make-ready" to permit greater pressure on areas of solid black, how to apply differential inking so that areas of great detail will not be blurred, how to use a frisket to cover those portions of the paper that should remain ink-free, ... even how to create a "rainbow" border on a traditional press (although this last was not part of the main project). Many of these techniques are simply impossible on "modern" printing equipment; however, the fine quality of the results is a fitting goal for craftsmen in the newer technologies.

Continuing his discussion of the decorative types, Mr. Mortimer showed several slides of contemporary (i.e., 1820s) posters on which they had been used; these were mainly for the theater, agricultural shows, and masonic events. The present printing project took about two years. Near the end of that time, one of the assistants in the printing house remembered seeing a specimen book with some of the designs, and undertook a search — which met with success. The cover of the specimen book identified it as being from the firm of Louis John Pouchee, and several of the types were prominently displayed there. Apparently, Pouchee was an upstart in the type business and antagonized some of the other, larger firms (typefounding was a very competitive business in the 19th century). It isn't known what actually happened, but it seems likely that a competitor bought up Pouchee's holdings and promptly made them disappear from public view, but was not so thorough that the types themselves were destroyed. The timing of this solution to the mystery was especially satisfying, as it permitted the creator of the types to be identified in the printed catalog.

One small additional fillip to the project was the use of another font of type that had apparently also not been used extensively. This was a very elegant sans serif that had been created about 1820, but failed to gain popularity at that time and was abandoned. (Sans serif types were "invented" anew at mid-century and became staples of the industry from that time.) The ability to print with such a contemporaneous type was clearly a source of great pleasure to Mr. Mortimer, and I can vouch for the highly satisfactory appearance of the results.

Following Mr. Mortimer's talk, a panel of the small press owners gave their views on questions such as these.

- Where will the book be in five years? The best answer: "where it damn well wants to." No one thought the book as produced by the small press is about to disappear, although several felt that reference and trade books would be heavily influenced by the techniques of electronic distribution.
- Does anyone running a small press (as represented on the panel) have an e-mail address? No, not one; one panelist pointed out that it

takes too long to clean the ink off one's hands for a computer to be particularly useful on an ongoing basis. But perhaps there is an intermediate approach wherein an engaged and knowledgeable bookseller can represent small presses and provide the means of introducing them to a wider audience, to the benefit of all.

- Is knowledge of the traditional printing crafts of any use to modern students of typography and design? An adamant and vocal response by Mr. Mortimer that this is the only way students are going to learn to handle space was seconded by a young woman in the audience who said that she had been in one of Mr. Mortimer's classes several years earlier and thanked him for what he had taught her.

The afternoon was occupied by a show of wares by nearly 20 small presses, and by demonstrations of printing (on a *very* small press, one made originally as a toy in a size suitable for printing business cards), bookbinding, and typefounding by current practitioners of those crafts. All the exhibitors were unfailingly helpful and the enthusiasm of the audience was obvious. I've come home with some fine new treasures and am already looking forward to next year.

For anyone interested in "books about books", Oak Knoll now has a presence on the Web, at `http://www.oakknoll.com`, or you can write to `oakknoll@ssnet.com`.

Sources of information on printing and book arts on the Web

In addition to the Oak Knoll Web page, which is also accessible through the Antiquarian Booksellers' Association of America (ABAA) World Wide Web Server, at `http://www.clark.net/pub/rmharris/booknet1.html`, several other Web sites devoted to printing and the book arts have come to my attention.

William S. Peterson, at the University of Maryland, College Park, has created a Web page "devoted to British and American fine printing of the nineteenth and twentieth centuries, with particular emphasis on William Morris and the Kelmscott Press." The address is `http://www.wam.umd.edu/~wsp/home.htm`. Even a quick glance at the page shows the hand of one who respects traditional book design. Pointers to numerous sources are given, as well as essays on the stated area of coverage. This site is under construction, and will surely expand into an even more interesting collection than it is already.

An announcement by the Catholic University of America Libraries and the School of Library and Information Science at CUA introduces "A Guide to the Book Arts and Book History on the World Wide Web", accessible at the address `http://www.cua.edu/www/mullen/bookarts.html`. This guide is "an organized list of links to Web pages that deal with book arts and book history."

From L'École des Arts visuels, Université Laval, Québec, comes the page IKONQuebec, at `http://www.ulaval.ca/ikon/HOME.HTML`. This site deals with graphic design including and beyond the book, and contains a bibliography of "les 700 meilleurs livres sur le graphisme", most of which are in either French or English.

Check them out!

A TEX Users Group for Spanish speakers

At the very end of December, a message from José R. Portillo Fernandez announced the formation of a new TEX Users Group — Grupo de Usuarios de TEX Hispanoparlantes, GUTH.

Some of the activities of GUTH will be the maintenance of a FAQ, publication of a *Boletin*, creation and maintenance of hyphenation dictionaries, styles, and a Babel package, and an FTP site. A Web page has already been installed: `http://gordo.us.es/Actividades/GUTH`.

If you are interested in the activities of this new group, but don't have Web access, you can get in touch with

José Ra Portillo Fernández
E.T.S. Arquitectura
Matemática Aplicada 1
Reina Mercedes, 2
E-41012 Sevilla (España)
E-mail: `josera@obelix.cica.es`

There is also a newsgroup about Spanish-language TEX, `es.eunet.spanish-tex`, and an associated discussion list, `spanish-tex@goya.eunet.es`.

A brief comment on the merger of *TUGboat* and *TEX and TUG News*

As you have already read in Michel Goossens' introduction to this issue, *TTN* has ceased publication after issue **4**(3). *TTN* was created as an attempt to attract members to TUG who didn't feel ready for the heavy technical matter of *TUGboat*. Although there have been differences of opinion over whether such a separate forum was necessary or successful, it is a fact that *TTN* has gathered much interesting and useful material, in a form that could easily be slipped into a purse or briefcase and carried onto an

airplane for easy reading, and I am sorry to see it end.

What I don't want to happen is for the ongoing in-print forum for this material to disappear too.

Technical material submitted to *TUGboat* is subjected to considerable scrutiny, refereed, and often edited heavily in an effort to present the facts in a style suitable for a technical journal. This will not change. However, "lighter" material, what appears in *Typographer's Inn* or '*Hey — it works!*', for example, will be accommodated without any attempt to make it hew to an inappropriate "higher standard". Peter Flynn and I will be working together to incorporate this material as a distinct and recognizable sub-publication that will retain its own personality. We have not yet had an opportunity to discuss the real details, but the current columnists will be invited — and I hope that they will agree — to continue in new surroundings.

The new, combined *TUGboat*/*TTN* will become reality with the next issue, **17**(1).

Let me take this opportunity to thank Peter Flynn, the present editor, and Christina Thiele, the founding editor (there have been no others), for their hard work. Well done!

◇ Barbara Beeton
American Mathematical Society
P. O. Box 6248
Providence, RI 02940 USA
Email: bnb@math.ams.org

Some thoughts on *TEX and TUG News*

Peter Flynn

In one way I was very fortunate to become the next editor of *TTN* after Christina Thiele. Her lead broke the new ground well, so my own spadework was much reduced. In another way she is always a hard act to follow, but I appreciate having such a high standard to work to.

As Barbara says, *TTN* was aimed at those members or potential members who did not feel the need for the technical depth of *TUGboat*, or who wanted something comprehensible which they could show to non-TEXies. I think we have been very fortunate in having writers of the technical caliber displayed in *TTN* to date who are nevertheless capable of expressing and explaining their topics in a form that doesn't require pre-reading of chapters 20–26 of *The TEXbook*.

I am looking forward to this continuing in the new format of publication, because I feel strongly that one of the best ways to persuade the less technical TEX users to start reading *TUGboat* is to get them reading *TTN*. I certainly remember my own early days with TEX, struggling to learn the operating system, editor, print spooler, job control language, and TEX all at the same time ... I would certainly have appreciated something to cut my teeth on before I developed an appetite for raw Knuth or Lamport. User support people in companies and universities have mailed me to say 'thank you' for providing this entrée to TEX, and I know some of them are now reading this.

The technical minutiæ of production are still being worked out. We have considered (and discarded, I am happy to say) suggestions like printing the *TTN* components on coloured stock, or having them bound on a perforated stub like a booklet on bowel control inside the *Reader's Digest*. I hope that our contributors, both occasional and regular, will continue to write, and I hope that the new proximity of *TTN* to *TUGboat* will mean that we can be even more encouraging to users in the future.

◇ Peter Flynn
Computer Center
University College
Cork, Ireland
Email: pflynn@curia.ucc.ie

Editor's note: Peter expresses very well what I find to be the major failing of *TUGboat* — it doesn't often contain much that can be comprehended immediately by someone who is not already relatively skilled in using TEX. Despite frequent invitations welcoming tutorial material and articles on modest but useful techniques, and an annual harangue on the subject at the summer TUG meeting, almost nothing of this sort appears in my in basket.

Even more than articles sent to me, I would very much like to find one or more volunteers who feel the same way, who are willing to listen in on the various electronic TEX discussions, talk to other TEXies, follow up on items that they think will be of general interest, and solicit even very small contributions for publication. Now that *TTN*, with its "smaller is better" flavor, is joining *TUGboat*, there will be an even greater scope for this material.

If you are such a person, and would like to help find good material to fill our pages, please send a message to TUGboat@ams.org or to Peter or me, and we will put you to work.

– bb

Editorial Comments

Barbara Beeton

A day at a small press book fair

Early this fall, over the weekend of the Columbus Day holiday (in the U.S., this is celebrated on the Monday closest to October 12, especially in locations with a sizeable population of Italian descent) I had the opportunity to attend a book fair organized by my favorite bookshop, Oak Knoll Books of New Castle, Delaware. Oak Knoll deals in "books about books" — typography, printing, design, ... bibliography — and has a strong commitment to the continued health of the book arts. In support of this, the proprieter, Bob Fleck, and his staff have undertaken to arrange "events" where small presses and craftsmen in the traditions associated with printing can get some exposure and meet prospective new customers. The book fair was the second in what promises to be an annual tradition.

The highlight of the fair was a talk by Ian Mortimer, the head of the printing house I. M. Imprimum, London, on the printing of a definitive edition of specimens of decorative wood types created in the early 1800s by Louis John Pouchee. The wood blocks, carved in solid boxwood, and intended as masters for stereotypes, were found among the effects of the Caslon type foundry at the time of its closing, without identification. The blocks are now held by the St Bride printing library in London, and the printing was undertaken under library auspices.

At the beginning of the project, nothing was known about the provenance of the types, not even the name of the creator. Nonetheless, the quality of design and workmanship of the types was exceptional, and no printed specimens were known. Aside from the effects of age — 150 years of disuse and lack of attention was evidenced in some minor warping and splitting and in the absence of some letters in a couple of the alphabets — the condition of the blocks was such that it is believed they had never before been used directly for printing, so the work described is in a very real sense a first edition.

Mr. Mortimer illustrated his description of the printing process with slides that illuminated the steps to even the least knowledgeable of the audience. I learned a great deal about printing from "real" type in order to ensure uniform distribution of ink on the printed surface — how to ensure that the type heights are even, how to use "make-ready" to permit greater pressure on areas of solid black, how to apply differential inking so that areas of great detail will not be blurred, how to use a frisket to cover those portions of the paper that should remain ink-free, ... even how to create a "rainbow" border on a traditional press (although this last was not part of the main project). Many of these techniques are simply impossible on "modern" printing equipment; however, the fine quality of the results is a fitting goal for craftsmen in the newer technologies.

Continuing his discussion of the decorative types, Mr. Mortimer showed several slides of contemporary (i.e., 1820s) posters on which they had been used; these were mainly for the theater, agricultural shows, and masonic events. The present printing project took about two years. Near the end of that time, one of the assistants in the printing house remembered seeing a specimen book with some of the designs, and undertook a search — which met with success. The cover of the specimen book identified it as being from the firm of Louis John Pouchee, and several of the types were prominently displayed there. Apparently, Pouchee was an upstart in the type business and antagonized some of the other, larger firms (typefounding was a very competitive business in the 19th century). It isn't known what actually happened, but it seems likely that a competitor bought up Pouchee's holdings and promptly made them disappear from public view, but was not so thorough that the types themselves were destroyed. The timing of this solution to the mystery was especially satisfying, as it permitted the creator of the types to be identified in the printed catalog.

One small additional fillip to the project was the use of another font of type that had apparently also not been used extensively. This was a very elegant sans serif that had been created about 1820, but failed to gain popularity at that time and was abandoned. (Sans serif types were "invented" anew at mid-century and became staples of the industry from that time.) The ability to print with such a contemporaneous type was clearly a source of great pleasure to Mr. Mortimer, and I can vouch for the highly satisfactory appearance of the results.

Following Mr. Mortimer's talk, a panel of the small press owners gave their views on questions such as these.

- Where will the book be in five years? The best answer: "where it damn well wants to." No one thought the book as produced by the small press is about to disappear, although several felt that reference and trade books would be heavily influenced by the techniques of electronic distribution.
- Does anyone running a small press (as represented on the panel) have an e-mail address? No, not one; one panelist pointed out that it

takes too long to clean the ink off one's hands for a computer to be particularly useful on an ongoing basis. But perhaps there is an intermediate approach wherein an engaged and knowledgeable bookseller can represent small presses and provide the means of introducing them to a wider audience, to the benefit of all.

- Is knowledge of the traditional printing crafts of any use to modern students of typography and design? An adamant and vocal response by Mr. Mortimer that this is the only way students are going to learn to handle space was seconded by a young woman in the audience who said that she had been in one of Mr. Mortimer's classes several years earlier and thanked him for what he had taught her.

The afternoon was occupied by a show of wares by nearly 20 small presses, and by demonstrations of printing (on a *very* small press, one made originally as a toy in a size suitable for printing business cards), bookbinding, and typefounding by current practitioners of those crafts. All the exhibitors were unfailingly helpful and the enthusiasm of the audience was obvious. I've come home with some fine new treasures and am already looking forward to next year.

For anyone interested in "books about books", Oak Knoll now has a presence on the Web, at `http://www.oakknoll.com`, or you can write to `oakknoll@ssnet.com`.

Sources of information on printing and book arts on the Web

In addition to the Oak Knoll Web page, which is also accessible through the Antiquarian Booksellers' Association of America (ABAA) World Wide Web Server, at `http://www.clark.net/pub/rmharris/booknet1.html`, several other Web sites devoted to printing and the book arts have come to my attention.

William S. Peterson, at the University of Maryland, College Park, has created a Web page "devoted to British and American fine printing of the nineteenth and twentieth centuries, with particular emphasis on William Morris and the Kelmscott Press." The address is `http://www.wam.umd.edu/~wsp/home.htm`. Even a quick glance at the page shows the hand of one who respects traditional book design. Pointers to numerous sources are given, as well as essays on the stated area of coverage. This site is under construction, and will surely expand into an even more interesting collection than it is already.

An announcement by the Catholic University of America Libraries and the School of Library and Information Science at CUA introduces "A Guide to the Book Arts and Book History on the World Wide Web", accessible at the address `http://www.cua.edu/www/mullen/bookarts.html`. This guide is "an organized list of links to Web pages that deal with book arts and book history."

From L'École des Arts visuels, Université Laval, Québec, comes the page IKONQuebec, at `http://www.ulaval.ca/ikon/HOME.HTML`. This site deals with graphic design including and beyond the book, and contains a bibliography of "les 700 meilleurs livres sur le graphisme", most of which are in either French or English.

Check them out!

A TEX Users Group for Spanish speakers

At the very end of December, a message from José R. Portillo Fernandez announced the formation of a new TEX Users Group — Grupo de Usuarios de TEX Hispanoparlantes, GUTH.

Some of the activities of GUTH will be the maintenance of a FAQ, publication of a *Boletin*, creation and maintenance of hyphenation dictionaries, styles, and a Babel package, and an FTP site. A Web page has already been installed: `http://gordo.us.es/Actividades/GUTH`.

If you are interested in the activities of this new group, but don't have Web access, you can get in touch with

José Ra Portillo Fernández
E.T.S. Arquitectura
Matemática Aplicada 1
Reina Mercedes, 2
E-41012 Sevilla (España)
E-mail: `josera@obelix.cica.es`

There is also a newsgroup about Spanish-language TEX, `es.eunet.spanish-tex`, and an associated discussion list, `spanish-tex@goya.eunet.es`.

A brief comment on the merger of *TUGboat* and *TEX and TUG News*

As you have already read in Michel Goossens' introduction to this issue, *TTN* has ceased publication after issue **4**(3). *TTN* was created as an attempt to attract members to TUG who didn't feel ready for the heavy technical matter of *TUGboat*. Although there have been differences of opinion over whether such a separate forum was necessary or successful, it is a fact that *TTN* has gathered much interesting and useful material, in a form that could easily be slipped into a purse or briefcase and carried onto an

airplane for easy reading, and I am sorry to see it end.

What I don't want to happen is for the ongoing in-print forum for this material to disappear too.

Technical material submitted to *TUGboat* is subjected to considerable scrutiny, refereed, and often edited heavily in an effort to present the facts in a style suitable for a technical journal. This will not change. However, "lighter" material, what appears in *Typographer's Inn* or '*Hey — it works!*', for example, will be accommodated without any attempt to make it hew to an inappropriate "higher standard". Peter Flynn and I will be working together to incorporate this material as a distinct and recognizable sub-publication that will retain its own personality. We have not yet had an opportunity to discuss the real details, but the current columnists will be invited — and I hope that they will agree — to continue in new surroundings.

The new, combined *TUGboat*/*TTN* will become reality with the next issue, **17**(1).

Let me take this opportunity to thank Peter Flynn, the present editor, and Christina Thiele, the founding editor (there have been no others), for their hard work. Well done!

◇ Barbara Beeton
American Mathematical Society
P. O. Box 6248
Providence, RI 02940 USA
Email: bnb@math.ams.org

Some thoughts on *TEX and TUG News*

Peter Flynn

In one way I was very fortunate to become the next editor of *TTN* after Christina Thiele. Her lead broke the new ground well, so my own spadework was much reduced. In another way she is always a hard act to follow, but I appreciate having such a high standard to work to.

As Barbara says, *TTN* was aimed at those members or potential members who did not feel the need for the technical depth of *TUGboat*, or who wanted something comprehensible which they could show to non-TEXies. I think we have been very fortunate in having writers of the technical caliber displayed in *TTN* to date who are nevertheless capable of expressing and explaining their topics in a form that doesn't require pre-reading of chapters 20–26 of *The TEXbook*.

I am looking forward to this continuing in the new format of publication, because I feel strongly that one of the best ways to persuade the less technical TEX users to start reading *TUGboat* is to get them reading *TTN*. I certainly remember my own early days with TEX, struggling to learn the operating system, editor, print spooler, job control language, and TEX all at the same time ... I would certainly have appreciated something to cut my teeth on before I developed an appetite for raw Knuth or Lamport. User support people in companies and universities have mailed me to say 'thank you' for providing this entrée to TEX, and I know some of them are now reading this.

The technical minutiæ of production are still being worked out. We have considered (and discarded, I am happy to say) suggestions like printing the *TTN* components on coloured stock, or having them bound on a perforated stub like a booklet on bowel control inside the *Reader's Digest*. I hope that our contributors, both occasional and regular, will continue to write, and I hope that the new proximity of *TTN* to *TUGboat* will mean that we can be even more encouraging to users in the future.

◇ Peter Flynn
Computer Center
University College
Cork, Ireland
Email: pflynn@curia.ucc.ie

Editor's note: Peter expresses very well what I find to be the major failing of *TUGboat* — it doesn't often contain much that can be comprehended immediately by someone who is not already relatively skilled in using TEX. Despite frequent invitations welcoming tutorial material and articles on modest but useful techniques, and an annual harangue on the subject at the summer TUG meeting, almost nothing of this sort appears in my in basket.

Even more than articles sent to me, I would very much like to find one or more volunteers who feel the same way, who are willing to listen in on the various electronic TEX discussions, talk to other TEXies, follow up on items that they think will be of general interest, and solicit even very small contributions for publication. Now that *TTN*, with its "smaller is better" flavor, is joining *TUGboat*, there will be an even greater scope for this material.

If you are such a person, and would like to help find good material to fill our pages, please send a message to TUGboat@ams.org or to Peter or me, and we will put you to work.

– bb

Software & Tools

Introduction to FasTEX: A System of Keyboard Shortcuts for the Fast Keying of TEX

Filip Machi, Jerrold E. Marsden and Wendy G. McKay

1 General Features of FasTEX

FasTEX is a system of keyboard shortcuts for speeding up the typing of TEX from the keyboard. FasTEX is currently available for the Macintosh and UNIX. It replaces any keyboard shortcut by the equivalent TEX command or group of commands in Plain TEX, $\mathcal{AMS}$-TEX, $\mathcal{AMS}$-LATEX, or LATEX. At the start of the session, the typist specifies which flavor of TEX is to be used from the set of files, one for each flavor, containing the mapping information that expands the keyboard shortcut into the corresponding TEX command(s). These files we shall refer to as the "FasTEX shortcut files".

While keying in the text of the document, the typist enters keyboard shortcuts. Expansion of a shortcut is activated by typing one of several predefined activation keys. After pressing the activation key the shortcut name is overwritten with the expansion text. It is common to use the spacebar as one of the activation keys.

For example, FasTEX replaces the keystroke sequence "xa" by `\alpha`, and the sequence "dxa" by `$\alpha$` etc. (The motivation for "xa" is that 'x' introduces any Greek letter and "a" is the Latin form of the Greek letter alpha. Similarly, the letter "d" preceding the "xa" in this example indicates that dollar signs be put around the expansion of "xa".) FasTEX can deal with long or short abbreviations with equal ease.

Using FasTEX, the typist, whether the author or another keyboard entry person, is able to keep his/her hands on the standard portion of the keyboard; and need only very rarely hit keys far from the home row of the keyboard, such as the backslash or dollar sign key.

1.1 What Types of Shortcuts Come with FasTEX?

The distributed FasTEX shortcuts are all designed to keep the typist's fingers near the home row keys. Some FasTEX shortcuts just rename standard TEX commands, while others expand into mulitple lines of TEX incantations, referred to here as templates, which can be used to simplify entering large or complex structures. Although FasTEX comes with a comprehensive, well thought out, and thoroughly tested list of shortcuts, personal shortcuts are easily customized.

Here are the kinds of shortcuts contained in the distributed FasTEX files.

1.2 Examples of Simple Expansions

These are complete expansions of some commonly used TEX commands.

- *Greek letters*

To get this	*Type*
`\alpha`	xa
`\beta`	xb
...	

The general shortcut naming rule for producing Greek letters is "x" followed by the Latin equivalent of the Greek letter, to produce the TEX command for that Greek symbol.

- *Math environment*

To get this	*Type*
`$`	d
`$\alpha$`	dxa
`$\beta$`	dxb
...	

The letter "d" at the start of a shortcut name generally means surround the expansion text with dollar signs.

- *Capital letters*

To get this	*Type*
`$A$`	dca
`\Gamma`	xcg
...	

In compound shortcut names like 'dca' and xcg', capital letters are entered by typing a "c", to indicate "Capital", before the lowercase version of the letter.

- *Calligraphic letters*

To get this	*Type*
`\cal A`	cca
`\cal B`	ccb
...	

A "c" preceding the shortcut name for a capital letter will produce the calligraphic form of that letter.

- *Superscripts*

To get this	*Type*
`^\alpha`	hxa

`^\beta`	hxb
`^4`	h4
...	

- *Subscripts*

To get this	*Type*
`_\alpha`	lxa
`_\beta`	lxb
`_4`	l4
...	

Remember this as "h" for higher, "l" for lower.

- *Fractions*

To get this	*Type*
`\frac{1}{2}`	f12
...	

- *Other Symbols* (*German or Fraktur, Open or Blackboard Bold letters*)

To get this	*Type*
`\frak G`	gmcg
`{\Bbb R}^2`	opcr2
...	

- *Word or Word-phrase abbreviations*

To get this	*Type*
`Department of Mathematics`	wcdm
`Department of Physics`	wcdp
`Euler-Poincar\'e`	wep
...	

A "w" at the start of a shortcut name means it is a word or word-phrase abbreviation.

- *Formatting features*

To get this	*Type*
`\\`	nl
`\newline`	nlin
`\hspace{0.2in}`	hsp
...	

1.3 Examples of Generic (Universal) Simple Expansions

These expansion create partial TEX commands that can be extended or completed using other shortcuts or additional user input. In general, these shortcut names have a trailing "u", meaning Universal (or Unfinished).

To get this	*Type*
`\frac{`	fu
`\int`	intu
`^{`	hu
`_{`	lu
`{\it`	hitu
`\sqrt{`	squ
...	

1.4 Complex Expansions—Templates

This category contains expansions for such things as matrices, commutative diagrams, equation environments, figures, tables, command definitions for the preamble section of a TEX file, etc. Even larger templates are available to produce skeleton versions of complete documents, such as letters and articles. Some examples are given in the display above.

1.5 The Importance of Producing Default Files

When dealing with coauthors, or anytime one is communicating documents, it is important to be able to deliver files that are as "plain vanilla" as possible so that the recipient avoids typesetting problems or troubles understanding the TEX source file for editing purposes. For example, custom definitions (or macros) can seriously interfere with this need and even make otherwise beautifully composed documents difficult for exchange between coauthors. FasTEX can help with these problems because all the shortcuts are *local* modifications to the keyboard input and do not remain in the actual text of the file created.

1.6 FasTEX is Universal, Fast, and Accurate

The FasTEX system is editor, application, and computer platform independent, and as we already indicated, can be used with plain TEX, $\mathcal{AMS}$-TEX, $\mathcal{AMS}$-LATEX, or LATEX. The use of FasTEX will *speed up the typical users typing input by a factor of about 3.*

This speed up is achieved by not only faster inputting, but through the fact that it *avoids simple typing errors* (if FasTEX is activated, the document will typeset). How many times has a document failed to typeset for you because your finger slipped when typing a backslash alpha or similar TEX command? This won't happen with FasTEX!

In addition, FasTEX is a convenient way of *remembering, storing and learning TEX commands.* FasTEX does not conflict with normal English useage; however, versions of FasTEX suitable for foreign languages require the replacement of certain shortcut names, depending on the language (some shortcut names may be words in certain foreign languages, such as "la" which is the code for "lower (subscript) a"; however, this is also a word in French but not in English).

To get this	*Type*
`\begin{equation}`	beq
`\end{equation}`	eeq
`\begin{eqnarray}`	lequ
`\lefteqn{ } \nonumber \\`	
`& &`	
`\end{eqnarray}`	
`\title{Title of paper}`	teaut
`\author{`	
`Author1`	
`\thanks{Research partially supported by ...}`	
`\\Department of Mathematics`	
`\\University of ...`	
`\\ \and`	
`Author2`	
`\thanks{Research partially supported by ...}`	
`\\Department of Physics`	
`\\State University of ... }`	
`\date{put in custom date; omit for today's date}`	
`\maketitle`	

1.7 You can get FasTEX off the WEB

As we shall explain in more detail below, FasTEX is currently available free on the World Wide Web for the Macintosh and UNIX environments. The UNIX version differs slightly from the Mac version but the basic operation and the names of the shortcuts are the same for both systems. System independence is another attractive feature when dealing with coauthors who may be using different systems but want to be able to efficiently share information about FasTEX.

1.8 Documentation

FasTEX comes with complete documentation that can be purchased from the authors.

2 A Sample Input

Here is an example of the input sequence one would use for the following equation using LATEX.

$$\phi_\alpha(x) = \int_0^x \frac{f(t)}{\alpha^2 + x^2}\, dt \qquad (1)$$

The sequence "beq xph lxa ox eq intu l0 hx fu f ot fof xa sq eb spdt eeq" produces the text in LATEX. A brief description of these shortcuts will help the reader to understand how the shortcuts names are a key to the functionality of the FasTEX system. As a typist becomes familiar with the pseudo-generic system of naming shortcuts, typing TEX documents becomes easier and faster to do.

To get this	*Type*	*Description*
`\begin{equation}`	beq	begin equation
`\phi`	xph	greek letter phi
`_\alpha`	lxa	subscript (or lower) α
`(x)`	ox	'of x' (x in parentheses)
`=`	eq	equal
`\int`	intu	set up the integral
`_0`	l0	subscript 0
`^x`	hx	superscript x
`\frac{`	fu	prepare for a fraction
`f`	f	f in numerator
`(t)`	ot	'of t' (t in parentheses)
`}{`	fof	forward fraction (go to the denominator)
`\alpha`	xa	greek letter alpha
`^2`	sq	square it (or use "h2")
`}`	eb	end brace (of fraction)
`\,dt`	spdt	thin space dt
`\end{equation}`	eeq	end equation

3 The Macintosh Version of FasTEX

The Macintosh version of FasTEX uses the shareware control panel TypeIt4Me, and the FasTEX shortcut files. You will need some version of system 7 to use FasTEX on the Mac.

The programs and files may be found on the following WEB sites.

Program	*Available from*
TypeIt4Me	http://delta.com/star.org/starhome.html
FasTEX files	http://avalon.caltech.edu/cds

3.1 Installation

After collecting the software off the WEB or using the FasTEX disk (Macintosh users can purchase a disk containing everything needed from TUG or the authors):

1. Copy the TypeIt4Me control panel to the control panels folder.
2. Copy the shortcut files to a convenient folder or on the desktop available for TypeIt4Me to open.
3. Restart the machine.
4. Open your favorite editor (Textures, Alpha, etc.) and type a shortcut such as "xa-spacebar" to test the system.

3.2 A little about TypeIt4Me

TypeIt4Me is a shareware product of Ricardo Ettore and if you really use it, you are expected to support the shareware idea and send him $30US. It is worth every penny! It is great for many purposes besides TEX: email addresses, commonly typed phrases, etc.

After restarting your machine, you will see the TypeIt4Me icon in the top left corner of your screen. Here are a few of the menu items you will find under the icon:

1. A help manual under *About TypeIt4Me* explains the workings of TypeIt4Me in detail.
2. To add an entry of your own, type it somewhere (such as your TEX editor), copy it to the clipboard, and call up the *Add an entry* item (for us, this is equivalent to the hot key "Shift-Option-C") and insert your shortcut name in the window. Now that shortcut is available!
3. To edit entries, call up this menu item (for us, it is the hot key "Shift-Option-E"). The dialog box that results is self-explanitory; you can, as the name suggests, edit any existing entry, change shortcut names, etc.
4. The preferences choice allows you to set hot keys, select what keys trigger the shortcut besides the spacebar, let you decide if you want the trigger included in the expansion, etc.
5. Below this you find date and time items.
6. The final menu items tell you what file of shortcuts you want open. On the Macintosh there is a limitation of having one file open at a time and each file is limited to 2500 shortcuts (this is due to resource editor limitations and is normally not a problem). You probably want to have your addresses, email addresses, etc., in a different file from the TEX shortcuts. FasTEX provides different files for LATEX, $\mathcal{AMS}$-TEX, and $\mathcal{AMS}$-LATEX; again, you choose the one you are working with for the particular document.

If for some reason you do not want TypeIt4Me to activate, you simply turn it off with the appropriate hot key (which we have set at Shift-Option-O)—that is an "oh". When off, the TypeIt4Me icon appears dimmed. For example, if you are typing a passage in French, you probably should turn off TypeIt4Me.

3.3 A Note about $

The dollar sign is of course the most common key needed for typesetting mathematical expressions. Because of its freqent use, we have made the single shortcut "d" for it. Surprisingly, this causes very few problems. Of course, as with all shortcuts, this does not cause a problem with a "d" occuring in a word. If you want to type a literal "d" in the middle of a formula and do not want it to come out with a dollar sign, you can use the shortcut "sd" which produces a literal "d", or you can type a "d" and (at least on the Macintosh) hold the shift key down when typing the spacebar, which keeps TypeIt4Me temporarily from activating.

3.4 A Sample Creation of a Shortcut

Let us suppose that you want to create a shortcut for a combination that occurs frequently in your work. For example, assume it is a $dx\,dy$ occurring in a double integral. First of all, decide on what spacing you want in the mathematics. For instance, let's say that the literal keystrokes you want are `\;dx\,dy` and, consistent with the FasTEX scheme of naming, you chose the shortcut name "spdxdy". Then you would create this shortcut as follows:

1. Type `\;dx\,dy` in any text editor.
2. Select and copy it to the clipboard.
3. Select "add an entry" from the TypeIt4Me menu (or hit "Shift-Option-C").
4. Type in the shortcut name spdxdy.

That's it! Now that shortcut is available.

4 The UNIX version of FasTEX

There is a version of the Macintosh FasTEX engine (TypeIt4Me) that runs on most flavors of UNIX. Called *scedit*, the UNIX version is based on the Expect program of Don Libes. Though not identical to the Mac version of TypeIt4Me, *scedit* attempts to include all of the essential functionality of the Mac version, with a few consessions to differences in the operating systems.

The shortcut files that make up FasTEX are necessarily different between the Mac and UNIX, but

both are derived from a common base file, so portability between Mac and UNIX introduces no incompatibilities.

The *scedit* program works by isolating an editor program of the user's choice from the keyboard and screen (window). All interaction between the user and the editor is intercepted by the *scedit* program, which performs the shortcut expansions.

An advantage of this approach is that shortcut expansion is independent of the windowing environment under which *scedit* runs, so for example, *scedit* will work over a telnet or modem connection. A disadvantage is that the editor program run by *scedit* can not respond to mouse events.

One benefit to using *scedit* over expansion facilities that may be native to a particular editor program is that the same shortcut files will work across different editor programs, regardless of what kind of expansion facility the editor may have, if any. For example, both vi and emacs have native expansion (abbreviation) capabilities that are incompatible with each other; but *scedit* and FasTEX works with either editor by changing a few configuration parameters.

The *scedit* program has some additional capabilities over the Mac TypeIt4Me program. One significant feature is that *scedit* can have many shortcut files active at once, unlike TypeIt4Me, which allows only one active shortcut file.

4.1 System Requirements

The *scedit* program will run on most versions of UNIX. The main requirement is that the UNIX platform have a recent version (5.7.0 or later) of the Expect program installed.

To run FasTEX on UNIX you need:

Program	*Available from*
Expect/Tcl	ftp://ftp.cme.nist.gov/pub/expect
scedit	http://avalon.caltech.edu/cds
FasTEX files	http://avalon.caltech.edu/cds

4.2 Installation

The UNIX version of FasTEX can be installed in your personal file space or in a system area. Either approach will work, and the choice depends mainly on whether you want to maintain the FasTEX system yourself or leave that up your computer's system administrator.

There are four components to the UNIX FasTEX system which need to be installed in four distinct areas. These are the *scedit* program, the FasTEX shortcut files, the documentation and the public, default configuration file. Each user may also have a personal configuration file which overrides the public one—see Customization below.

Installation consists of editing the Makefile that came with the FasTEX distribution to reflect the installation directories you chose, then running the command '`make install`'.

Within the Makefile, the variable `INSTALLDIR` should be edited to contain the name of a directory containing the sub-directories `bin`, `lib/scedit` and `man`. If any of these directories or sub-directories does not exit, you should create them first and set their permissions to allow the kind of access you think appropriate. Also, edit the variable `EXPECT` to contain the name of the directory containing the Expect executable. When you complete these two edits you may run '`make install`'.

The following example shows a typical installation of UNIX FasTEX into a personal area. To install into a system area just change the names of the destination directories (and be sure you have write permission in the affected system directories).

```
mkdir /accts/fil/fastex
chmod 755 /accts/fil/fastex
mkdir /accts/fil/fastex/bin
chmod 755 /accts/fil/fastex/bin
mkdir /accts/fil/fastex/lib
chmod 755 /accts/fil/fastex/lib
mkdir /accts/fil/fastex/lib/scedit
chmod 755 /accts/fil/fastex/lib/scedit
mkdir /accts/fil/fastex/man
chmod 755 /accts/fil/fastex/man
mkdir /accts/fil/fastex/man/man1
chmod 755 /accts/fil/fastex/man/man1

chmod 644 Makefile
vi Makefile
   ...
INSTALLDIR = /accts/fil/fastex
EXPECT = /usr/local/bin/expect
   ...
make install
```

At this point FasTEX has been installed and is ready to use. You may wish to create an alias for the *scedit* executable so that you do not have to type its full path name when you invoke it. Alternatively, you can place a symbolic link to the *scedit* executable into one of the `bin` directories listed in your shell's `PATH` variable. With this latter approach, you could make *scedit* available to the general user community by placing the symbolic link into a system `bin` directory, such as `/usr/local/bin` if that is where locally added programs are kept on your system:

```
ln -s /accts/fil/fastex/bin/scedit
                              usr/local/bin
```

Another approach is to place the symbolic link into your own personal `bin` directory if you wanted access to *scedit* to be semi-private.

4.3 Customization

The file *sceditrc*, which was installed into the lib directory during the installation procedure, contains customization parameters for *scedit*. These parameters control several aspects of the way *scedit* operates, and take effect for every user of *scedit*.

An optional file, *.sceditrc* in each user's home directory, can also contain *scedit* parameter values. These will override the values in *sceditrc* located in the lib directory. In this way each user can specify personal customization of *scedit*.

The parameters include: `FASTEX_FILES` which contains a list of shortcut file names to use by default; `FASTEX_PATH` which lists the directories to be searched for shortcut files listed on the command line; `NAME_SUFFIX` which contains the list of characters that will serve to activate shortcut expansion; `REDRAW` which holds the sequence of characters that will make your editor redraw the screen; etc. The *scedit* documentation discusses these and the other parameters in more detail.

Standard (Bourne) shell syntax can be used in the customization files. This allows a single customization file to include parameters for a variety of different editors. The default *sceditrc* file that comes with the *scedit* distribution shows how to represent separate customizations for vi and emacs, selected by examining the UNIX standard environment variable `EDITOR`.

4.4 Some specific differences between the *scedit* and TypeIt4Me versions of FasTEX

Because the UNIX version of FasTEX does not assume a windowing environment, some of the user interaction with *scedit* is necessarily different than the user interaction with TypeIt4Me. For example, *scedit* uses special escape keystrokes to perform file management operations that would be done with pull-down menus on the Mac. Most of these meta-level operations are introduced with the tilde character (`~`).

For example, the escape command `~e1`. would start a new editing session on the main shortcuts file. The display and keyboard will now be attached to this new editor session. The old editor session is waiting undisturbed in the background. When you exit this new editor session, the old one is resumed. Viewing the shortcuts in this way is helpful when you forget the name of a shortcut and want to look it up; you can then use the full search capabilities of your editor to help you locate shortcut definitions within a potentially large shortcut file.

Any changes you make to a shortcut file when viewed in this way will be incorporated back into the active *scedit* session when you exit the viewing editor. So, for example, to add a new shortcut while editing a document, use (`~ef`) to view/edit your "first" shortcut file (the "first" file is the one that *scedit* reads before the main shortcut file), add the new shortcut to the file using the editor, then write and quit the editor. The new shortcut will now be available to you as you resume editing your document.

A list of all the escape commands is available through the escape command (`~h`). To enter a actual tilde character into the text of your document, double the tilde (`~~`).

The usual UNIX job control mechanism can also be used to suspend and resume the entire *scedit* session.

◇ Filip Machi
Center for Extreme Ultraviolet Astrophysics
University of California at Berkeley 5030
Berkeley
CA 94720
USA
Email: `fil@cea.berkeley.edu`

◇ Jerrold E. Marsden
Control and Dynamical Systems
California Institute of Technology 104-44
Pasadena
CA 91125
USA
Email: `marsden@cds.caltech.edu`

◇ Wendy G. McKay
Control and Dynamical Systems
California Institute of Technology 104-44
Pasadena
CA 91125
USA
Email: `wgm@cds.caltech.edu`

Philology

The Style russianb for Babel: Problems and solutions

Olga Lapko
Irina Makhovaya

Abstract

As is the case with other languages based on non-Latin alphabets, when preparing a style one must take into account the typographic rules traditionally used in the given language. This paper describes the package `russianb`, which includes such macros as `\captionsrussian` to address the four types of standard Russian LaTeX documents, `\daterussian`, `\Asbuk` and `\asbuk` for Russian alphabet counters, and `\mathrussian` for Russian math operators. Some problems concerning the usage of this style (e.g. usage of different encodings) are described.

Introduction

As is generally known, TeX is based on the Latin alphabet and, while in theory it is possible to use TeX for other alphabets — Greek, Arabic, Cyrillic and so on — in practice, there are a lot of problems when we try to use TeX with other alphabets. The Babel package [1] is the first successful attempt to solve the problems of multilingual TeX.

In this paper we discuss the concrete difficulties we encountered while creating the `russianb`[1] file for Babel. The whole set of peculiarities of typesetting documents in Russian can be subdivided into three classes:

1. features borrowed from western European typography, especially German and French;
2. features peculiar to Russian typography only and which can be easily described in the file `russianb`;
3. features peculiar to Russian typography and which pose some difficulties in describing and using them.

We also see two very important general problems: the variety of encoding schemes which currently exist and the need for portability of the present package file to different platforms, a problem which we could solve only partially.

[1] `russianb` is an analog of the filename `germanb`; the name is chosen to avoid confusion with Russian-language Babel styles currently used with LaTeX 2.09. Currently `russianb`, in beta-version, comes as an extension to the *CyrTUG*-emTeX distribution.

Now we shall describe the macros of the file `russianb` according to the classification just outlined.

Macros borrowed from other styles

The file `russianb` was based upon the already existing versions of some already-existing styles: `german` (for LaTeX 2.09), `germanb` (for LaTeX 2_ε), and `francais` (for LaTeX 2_ε). These files have the language-specific macros which Russian typographic rules need:

1. *from `germanb`*
   - macros for French and German double quotes. *Note*: French double quotes are created by METAFONT in a Cyrillic font and have their own ligature (i.e. `<<`);
   - "shortcuts" for hyphenation in compound words and words with non-letter characters (Russian words are not as long as German ones but a few long ones do exist). And of course, as in `germanb`, the sign `"` was made active.
   - `\lefthyphenmin`–`\righthyphenmin`: for Russian (as for German) the values 2–2 are used in the hyphenation algorithms;
2. *from `francais`*
   - macros for the punctionation marks `:`, `;`, `?`, `!`, where the amount of white space is slightly increased in front of these signs: TeX looks for a space between a word and this sign, then, if there is a space, TeX "unskips" it and puts in an extra little space of `0.1em`;
   - `\frenchspacing` is switched on;
   - some additional characters (as in `francais`) are described in our style, e.g. the number sign.

Macros created in the Russian style and which pose no problems in usage

The macros described below were borrowed from various releases of the `russian.sty` file (for use with LaTeX 2.09).

1. macros for math operators whose names differ from English ones (e.g. in Russian manuscripts we write `tg` and `ctg` instead of `tan` and `cotan`);
2. additional macros for printing counter values, using uppercase and lowercase Cyrillic letters (`\Asbuk` and `\asbuk` as analogs to `\Alph` and `\alph`).

In addition, we created a shorthand macro for the Russian emdash (`"---`): in printed documents, this

sign is somewhat shorter and is surrounded by spaces (about 0.2 of the current font size; i.e. a 2pt space in a 10pt font); this emdash can never be separated from the word preceding it. *Note*: the macro for an explicit hyphen sign (`"-`) was of course rewritten because of this new macro for the Russian emdash. See examples in figs. 1 and 2.

Macros that need explanations or are difficult to use

In this section we discuss macros for breaking in-line formulas. Russian typographic tradition requires us to repeat the last sign of a broken formula on the next line.

A package, specially written by M.I. Grinchuk to solve this problem, contains macros to repeat signs when a formula is broken. The package offers two options:

1. *hand breaking*: in this case, we have the values `\binoppenalty` = `\relpenalty` = 10000; to break the formula, one must use the commands `\brokenbin{ }` and `\brokenrel{ }`;
2. *automatic breaking*: values for the same macros as above are made non-equal: `\relpenalty` > > `\binoppenalty` > 10000. Some signs are equal to `\mathcode=8000` and divided into two groups: (a) binary and relational signs `+ - < > =` allow breaks after them, and (b) the signs `* ( [ / . ,`, which prohibit breaks. On top of that, almost all mathematical signs have been rewritten using the new commands `\brkbin` and `\brkrel` to allow or prevent breaking, e.g.:

```
\def\wedge  {\brkbin{\mathchar"225E}}
\def\gg     {\brkrel{\mathchar"321D}}
\def\exists {\mathchar"0239\unbrk}
\def\bigl#1 {\mathopen{\big#1}\unbrk}
\def\bigm#1 {\mathrel {\big#1}\unbrk}
\def\langle {\delimiter"426830A \unbrk}
```

In particular, the command `\not` must be redefined as follows:

```
\def\not#1 {\brkrel{\mathchar"3236 #1}}
```

and so on. In this case TEX breaks formulas by itself but sometimes we have to handle breaking using the special commands `\unbrk` or `\allowbrk`.

This package has some drawbacks, although of a rather exotic nature:

- one must write `$x \brkbin{{+}^1} y$` instead of `$x +^1 y$`;
- math operators like `\sin` must be written with arguments in curly braces;
- in formulas such as $x + \ldots + y$ one must write `$x \unbrk + \ldots + y$` to prevent the first break (or the breaking sign must be redefined to allow look-ahead);
- in case the signs `^` and `_` are redefined, we cannot use $\mathcal{AMS}$-TEX macros such as `\Sb..\endSb` (i.e. `^\bgroup..\egroup`); in other words, it is impossible to use something like `^\leq` and `^*`.

It should be clear from the above that we must rewrite all the definitions of binary and relational operators, as well as of certain signs which cannot be followed by a line break.

At present, this style exists as an add-on[2] and can be turned on (or off) by the user. An example of how to use this package is shown in fig. 3.

Encoding and font problems

The `russianb` file was created for the "non-Latin user" who uses the Cyrillic alphabet. One of the basic problems is: there are a lot of different encodings in each of which Cyrillic letters have different codes. Because of this, our file has some particular features when compared with other files developed for the Babel system

To make this style independent of encoding, we have to use macro names instead of Cyrillic letters themselves. Russian letters and signs in this style are used in macros for the date (`\daterussian`) and the text strings for the four standard styles of LATEX (`\captionsrussian`), and also in commands for printing counter values by using Cyrillic upper- and lowercase letters (`\Asbuk` and `\asbuk`).

The macro names for Cyrillic letters and some signs are introduced with the help of a subsidiary file (e.g. `lhrcod.sty`) which corresponds to a given encoding scheme. This file also switches the Cyrillic font family.[3]

The coding schemes used in Russia usually consist of a table, where in the upper part one finds the Cyrillic letters using one of a set of layouts, while the lower part contains the Latin letters, in the usual (ASCII) layout. For successful usage of Russian hyphenation patterns Cyrillic letters are set to `\catcode\letter` (in `russianb` Cyrillic letters are restored to their normal category again in case their category had been changed). There is already a long tradition of using Russian letters in macro names

[2] It is a beta release.

[3] Currently this switching is only implemented for LATEX 2_ε.

(Russian letters are letters too, aren't they?). Moreover, there are packages which use Russian words as commands, in some special cases.[4]

At this point we should also mention that, to respect Russian typographic tradition, the `\mathcode` for Cyrillic letters is set to `70??`, i.e. class 7 (for "variable") and family 0 (`\fam0`, `\rm`).

What needs to be done

The present style is meant to be used when working with "8-bit" Russian documents, where one sometimes has fragments in the Latin alphabet, or to enter documents using transliteration.

For entering 8-bit documents the subsidiary file `russianb` (in the current version we input `cyrcod`) simply declares a new font family and encoding at the beginning of the document,[5] which then loads the Russian/English hyphenation only plus the macros for each language—this approach requires less memory (see fig. 1).

For Russian-Latin papers input with transliteration (where for each Russian letter one uses a Latin equivalent) we must declare that sometimes Latin letters represent Cyrillic ones, so we have to toggle not only hyphenation patterns and macros but (above all) fonts and encodings too (see fig. 2).

Conclusion

The main difficulty with the `russianb` style file is that we have to use macro names for Cyrillic letters. For now, these letters are described as `\def\CYRa{a}`. In this case the commands `\uppercase` and `\lowercase` don't work correctly so one has to use additional definitions for these commands.

Above we have described a few solutions to the problem of using the `russianb` file with different platforms and environments. However, because we have had almost no occasion to actually work on different environments, we cannot really say very much about possible remaining difficulties.

The present work is only a first attempt to create a Russian-language style file extension for the Babel package. We hope to provoke discussions and further work in this direction by our colleagues. We also look forward to the Ω [4] package, which should be able to solve many of the problems described in this article, especially those connected with encoding schemes and portability across different computer environments.

Acknowledgements

In conclusion we would like to mention that a lot of our colleagues have made valuable contributions to the development of the file `russianb`: Mikhail Grinchuk, Eugenii Ivanov, Sergey Lvovskii, Andrey Slepukhin, Yurii Tyumentsev, and others. We are very much obliged to all of them.

References

[1] Braams, J. "Babel, a multilingual style-option system for use with LaTeX's standard document styles." *TUGboat* 12 (2), pages 291–301, 1991.

[2] Gilenson, P. *Spravochnik tekhnicheskogo redaktora*. Moscow: Kniga, 1979.

[3] Goossens, M., F. Mittelbach, and A. Samarin. *The LaTeX Companion*. Reading, Mass.: Addison-Wesley, 1994.

[4] Haralambous, Y., and J. Plaice. "First application of Ω: Greek, Arabic, Khmer, Poetica, ISO 10646/UNICODE, etc." *TUGboat* 15 (3), pages 344–352, 1994.

[5] Khodulev A., and I. Makhovaya. "On TeX experience in Mir Publishers." *Proceedings of the Seventh European TeX Conference* (Sept. 14–18, 1992, Prague, Czechoslovakia), pages 37–42.

[6] Lapko, O. "MAKEFONT as part of *CyrTUG*-emTeX package." *Proceedings of the Eighth European TeX Conference* (Sept. 26–30, 1994, Gdańsk, Poland), pages 110–114.

◇ Olga Lapko
Irina Makhovaya
Mir Publishers
2, Pervyi Rizhskii Pereulok
Moscow, 129820, Russia
Email: `olga@mir.msk.su`
`irina@mir.msk.su`

Стиль russianb для пакета Babel: проблемы и решения

Лапко О. Г.,
Маховая И. А.

Аннотация

Как и в случае других языков, базирующихся на иных, не латинских алфавитах, при разработке соответствующего стиля следует отразить правила полиграфического оформления, традиционные для данного языка. В статье описывается стиль `russianb` с использованием макросов

[4] In a Russian document where transliteration is used it often happens that many letters are made "active", therefore the assignment `\catcode\letter` is only made inside a group when loading the hyphenation tables.

[5] Currently such a file only exists for LaTeX 2_ε.

`\captionrussian` для 4-х стандартных документов на русском языке, `\daterussian`, `\Asbuk` и `\asbuk` для перечислений при помощи букв русского алфавита и `\mathrussian` для названий математических функций и операций. Освещаются некоторые проблемы, связанные с применением данного стиля (например, использование различных схем кодировок).

Введение

Общеизвестно, что TEX основан на латинском алфавите, но теоретически возможно его использование и в иных алфавитах: греческом, арабском, кириллическом и т. д. Когда же мы хотим это реализовать на практике, то сталкиваемся с целым рядом трудностей. Первой успешной попыткой решения проблем многоязычного TEX'а был пакет Babel [1].

В настоящей работе обсуждаются конкретные трудности, с которыми мы столкнулись при создании стилевого файла `russianb`[1] для пакета Babel. Всю массу особенностей полиграфической подготовки документов на русском языке [6] можно разбить на три класса:

1. особенности, позаимствованные из полиграфии Западной Европы, большей частью из немецкой и французской;
2. особенности, присущие исключительно русской полиграфии и не вызывающие затруднений при описании их в файле `russianb`;
3. особенности, присущие русской полиграфии и вызывающие определенные сложности при их описании и использовании.

Мы также видим две очень важные общие проблемы: разнообразие схем кодировки и необходимость переносимости этого стилевого файла с платформы на платформу, которые решены нами лишь частично.

Описание макросов файла `russianb` будет вестись далее согласно этой классификации.

Макросы, позаимствованные из других стилей

Файл `russianb` был написан на основе уже существующих стилей `german` (для LaTeX2.09), `germanb` (для LaTeX 2_ε) и `francais` (для LaTeX 2_ε). В этих файлах имелись специфические для этих языков макросы, которые описывали правила, присущие и русской полиграфии.

1. *из* `germanb`
   - макросы для фванцузских и немецких двойных кавычек. *Замечание*: французские двойные кавычки в кириллическом шрифте получены при помощи METAFONT'а и имеют свою лигатуру, а именно `<<`;
   - «shorthands» для переносов составных слов и слов с небуквенными включениями (русские слова не такие длинные как немецкие, но бывают достаточно длинными тоже). Разумеется, как и в `germanb` знак `"` был сделан активным.
   - `\lefthyphenmin`–`\righthyphenmin`: в русском языке (как и в немецком) в алгоритме переносов приняты значения 2–2;
2. *из* `francais`
   - макросы для знаков пунктуации `:`, `;`, `?`, `!`, в которых размер пробела перед этими знаками немного увеличивается: TEX определяет, есть ли пробел между словом и этим знаком, и если есть, то TEX «unskips» (не пропускает) его и помещает дополнительный маленький пробел в `0.1em`;
   - включается `\frenchspacing`;
   - в нашем стиле описываются некоторые дополнительные знаки (равно как и в `francais`), например, знак номера;

Макросы, созданные специально для русского стиля и не вызывающие проблем при использовании

Приводимые ниже макросы позаимствованы из различных вариантов `russian.sty` (для LaTeX2.09):

1. Макросы для названий математических функций и операций, названия которых отличаются от соответствующих английских (например, в русских изданиях вместо `tan` и `cotan` мы пишем `tg` и `ctg`);
2. дополнительные макросы для получения перечислений русскими прописными и строчными буквами: (`\Asbuk` и `\asbuk` по аналогии с `\Alph` и `\alph`).

Создан дополнительный «shorthand» для русского тире (`"---`): в наших документах этот знак немного короче и имеет вокруг пробелы

[1] Названием стиля для пакета Babel для русского языка мы выбрали `russianb` по аналогии с `germanb`, чтобы избежать возможных коллизий с русскими стилями к LaTeX2.09. В настоящее время `russianb` представляет собой beta-версию — часть дополнения к кириллическому дистрибутиву *CyrTUG*-emTEX.

примерно по 0.2 размера шрифта (т. е. для размера шрифта 10pt это приблизительно 2pt); причем это тире никогда не отрывается от слова, стоящего перед ним. *Замечание*: макрос для знака переноса (`"-`) был, естественно, переписан, поскольку появился новый макрос для русского тире. Пример использования этих макросов приведен на рис. 1 и 2.

Макросы, использование которых вызывает вопросы или затруднения

В этом разделе речь пойдет о макросах, позволяющих разрывать формулы в тексте. Согласно русским полиграфическим традициям знак, на котором разрывается формула, должен быть повторен на следующей строке.

В пакете, написанном для этой цели М. И. Гринчуком, содержатся макросы, позволяющие повторять знаки при разрыве формул.

Предлагается два возможных способа:

1. *разрыв вручную* — в этом случае значения `\binoppenalty` = `\relpenalty` = 10000; для разрыва формулы нужно использовать команды `\brokenbin{ }` и `\brokenrel{ }`;
2. *автоматический разрыв* — в этом случае `\relpenalty` > `\binoppenalty` > 10000; некоторые знаки равны `\mathcode=8000` и подразделяются на две группы: бинарные операции и отношения `+ - < > =` допускают после себя разрыв, а знаки `* ( [ / . ,` разрыв не допускают; кроме того почти все математические знаки переписаны с использованием новых команд `\brkbin` и `\brkrel`, чтобы допустить или предотвратить разрыв, например:

```
\def\wedge  {\brkbin{\mathchar"225E}}
\def\gg     {\brkrel{\mathchar"321D}}
\def\exists {\mathchar"0239\unbrk}
\def\bigl#1 {\mathopen{\big#1}\unbrk}
\def\bigm#1 {\mathrel {\big#1}\unbrk}
\def\langle {\delimiter"426830A \unbrk}
```

в частности, должна быть переопределена команда `\not`:

```
\def\not#1 {\brkrel{\mathchar"3236 #1}}
```

и т. д. В этом случае TEX разрывает формулы самостоятельно, но иногда приходится делать разрыв вручную при помощи специальных команд `\unbrk` или `\allowbrk`.

В данном пакете имеются небольшие огрехи, но довольно экзотические:

- следует писать `$x \brkbin{{+}^1} y$` вместо `$x +^1 y$`;
- операторы `\sin` нужно переписать с аргументами в фигурных скобках;
- в формулах типа $x + \ldots + y$ нужно писать `$x \unbrk + \ldots + y$`, чтобы избежать первого разрыва (или знаки разрыва нужно переопределять „заглядывающими вперед“);
- в случае, когда переопределены знаки `^` и `_`, мы не можем использовать макросы $\mathcal{AMS}$-TEX'а типа `\Sb..\endSb` (т. е. `^\bgroup..\egroup`), иначе говоря, невозможно использовать что-либо вроде `^\leq` и `^*`.

Из сказанного ясно, что мы должны переписать все определения для бинарных операций, отношений и для некоторых знаков, которые не допускают разрыва после себя. Иначе говоря, мы должны переписать TEX-форматы.

В настоящее время этот пакет существует как дополнительный[2] и может подключаться (или не подключаться) пользователем. Пример использования этого пакета приведен на рис. 3.

Проблемы схем кодировки и шрифтов

Файл `russianb` был создан для «нелатинского пользователя», применяющего кириллический алфавит. Одна из основных проблем здесь — существование множества различных кодировок, в которых буквы кириллицы имеют разные коды.

В связи с этим наш файл имеет, по сравнению с аналогичными стилевыми файлами для пакета Babel, некоторые специфические особенности.

Чтобы сделать этот стиль независимым от кодировки, мы должны были использовать вместо букв кириллического алфавита их макроимена. Русские буквы и знаки используются в этом стиле в макросах для даты (`\daterussian`) и в текстовых вхождениях четырех стандартных стилей LATEX'а (`\captionsrussian`), а также в командах для перечислений при помощи русских прописных и строчных букв (`\Asbuk` и `\asbuk`).

Макроимена для букв русского алфавита и некоторых знаков вводятся при помощи файла-спутника (т. е. `cyrcod.sty`), который создается в соответствии с той или иной схемой кодировки. Этот файл переключает также семейство кириллических шрифтов [3].

Используемые в России схемы кодировки обычно представляют собой таблицы, в верхней

[2] Это beta-версия пакета.

[3] В настоящее время такое переключение сделано только для LATEX 2_ε.

части которых находятся буквы кириллического алфавита в той или иной кодировке, а в нижней ее части — латинские буквы в общепринятой кодировке. Для того чтобы можно было успешно пользоваться переносами, кириллические буквы были заданы как `\catcode\letter` (в `russianb` они опять восстанавливают свое значение, в случае если их категория была переопределена). Использование в макрокомандах русских букв также давно стало традицией (русские буквы ведь тоже буквы, не так ли?). Более того, существуют пакеты, использующие в некоторых специальных случаях «русские» команды[4].

Теперь следует сказать, что `\mathcode` для кириллических букв равен `70??`, т. е. мы устанавливаем класс 7 — переменную семейства и используем шрифт из `\fam0` (`\rm`), чтобы соблюсти русские полиграфические традиции.

Что следует сделать

Предлагаемый стиль предназначен для работы с русскими 8-битовыми документами, в которых бывают отдельные фрагменты на латинице, и для набора документов с использованием транслитерации.

При наборе 8-битовых документов файл-спутник `russianb` (в текущей версии мы вводим `cyrcod`) просто декларирует новое семейство шрифтов и схему кодировки в начале документа[5] и затем втягивает только русскую/латинскую системы переносов плюс макросы для каждого из языков — такой способ требует меньше памяти (см. рис. 1).

Для русско-латинских работ, набранных при помощи транслитерации (когда для обозначения русских букв используются латинские), мы должны декларировать, что иногда латинские буквы представляют русские, так что здесь помимо переключения таблиц переносов и макросов, необходимо постоянно переключать (в первую очередь) и схемы кодировок (см. рис. 2).

Основная проблема использования файла `russianb` состоит в том, что приходится использовать макроимена для русских букв. В настоящее время эти буквы описаны как `\def\CYRa{a}`. В этом случае команды `\uppercase` и `\lowercase` работают некорректно и необходимы дополнительные определения для этих команд.

Выше описаны некоторые решения проблемы переносимости файла `russianb` на разные платформы. Но поскольку у нас не было практики работы с другими платформами, нам мало что известно о возможных подводных камнях на этом пути.

Данная работа представляет собой лишь первую попытку создания стилевого файла для русского языка для пакета Babel и мы надеемся вызвать дискуссию и дальнейшие работы в этом направлении наших коллег. Мы также надеемся, что пакет Ω [3] поможет решить описанные здесь проблемы, особенно связанные со схемами кодировки и с переносимостью на разные платформы.

Благодарности

В заключение мы хотим заметить, что существенный вклад в создание файла `russianb` внесли многие наши коллеги: Михаил Гринчук, Евгений Иванов, Сергей Львовский, Андрей Слепухин, Юрий Тюменцев и другие. Всем им мы чрезвычайно благодарны.

Список литературы

[1] J. Braams: Babel, a multilingual style-option system for use with LaTeX's standard document styles. TUGBoat, Vol. 12(1991), No. 2, 291–302.

[2] M. Goossens, F. Mittelbach, and A. Samarin: The LaTeX Companion, Addison-Wesley, Reading, MA, 1994.

[3] Y. Haralambous, J. Plaice: First application of Ω: Greek, Arabic, Khmer, Poetica, ISO 10646/Unicode, etc. TUGBoat, Vol. 15(1994), No. 3, 344–352.

[4] A. Khodulev and I. Makhovaya: On TeX experience in Mir Publishers, Proceedings of the 7th EuroTeX Conference, Prague, pp. 37–43, 1992.

[5] O. Lapko: MAKEFONT as part of *CyrTUG*-emTeX package, Proceedings of the eight European TeX Conference, Gdańsk, pp. 110–114, 1994.

[6] П. Гиленсон: Справочник технического редактора, Москва, Книга, 1979.

◇ Лапко О. Г.,
Маховая И. А.
Россия, 129820, Москва
1-й Рижский пер., д. 2
Издательство «Мир»
Email: `olga@mir.msk.su`
`irina@mir.msk.su`

[4] В русских документах с использованием транслитерации, многие буквы находятся на „активных“ кодах TeX'а, поэтому значение `\catcode\letter` задается в группе только при загрузке таблицы переносов.

[5] Такой файл сейчас имеется только для LaTeX 2_ε.

Данная распечатка демонстрирует некоторые примеры набора на русском языке с использованием 8-битной кодировки, в частности, набора кавычек, знака тире и математических формул. В русских изданиях используются:

1. кавычки „лапки“, аналогично немецким кавычкам;
2. кавычки «ёлочки», которые можно набрать двумя способами;
   (а) с помощью лигатур, которые заложены при создании шрифта, т. е. двух знаков „больше“ или „меньше“: `<<`, `>>` — вот как это выглядит: «ёлочки»;
   (б) с использованием активной кавычки: `"<` и `">`;
3. знак тире в русской полиграфии несколько короче и всегда отбивается небольшими пробелами около 0,2 кегля, см. п. 4;
4. определен знак номера —№, который набирается как `\No`.

А еще можно показать следующие формулы с измененными математическими операторами:

$$\operatorname{tg}\alpha = \frac{\sin\alpha}{\cos\alpha}, \quad \text{или} \quad \operatorname{ctg}\alpha = \frac{\cos\alpha}{\sin\alpha} \tag{1}$$

следующие формулы выделены полужирным шрифтом:

$$\boldsymbol{\operatorname{tg}\alpha = \frac{\sin\alpha}{\cos\alpha}, \quad \text{или} \quad \operatorname{ctg}\alpha = \frac{\cos\alpha}{\sin\alpha}} \tag{2}$$

Это маленький пример рубленого текста.

```
\def\theenumii{\asbuk{enumii}}

Данная распечатка демонстрирует некоторые примеры набора на русском языке с
использованием 8-битной кодировки, в частности, набора кавычек, знака тире и
математических формул. В русских изданиях используются :
\begin{enumerate}
\item кавычки "`лапки"', аналогично немецким кавычкам ;
\item кавычки "<ёлочки">, которые можно набрать двумя способами ;
\begin{enumerate}
\item с помощью лигатур, которые заложены при создании шрифта, т.~е. двух знаков
"`больше"' или "`меньше"' : \verb|<<|, \verb|>>| "--- вот как это выглядит : <<ёлочки>> ;
\item с использованием активной кавычки : \verb|"<| и \verb|">| ;
\end{enumerate}
\item знак тире в русской полиграфии несколько короче и всегда отбивается небольшими
пробелами около 0,2 кегля, см. п. 4 ;
\item определен знак номера"---\No, который набирается как \verb|\No|.
\end{enumerate}
А еще можно показать следующие формулы с измененными математическими операторами :
\begin{equation}
\tg\alpha=\frac{\sin\alpha}{\cos\alpha},\quad или
\quad\ctg\alpha=\frac{\cos\alpha}{\sin\alpha}
\end{equation}
следующие формулы выделены полужирным шрифтом :{}
{\boldmath\begin{equation}
\tg\alpha=\frac{\sin\alpha}{\cos\alpha},\quad или
\quad\ctg\alpha=\frac{\cos\alpha}{\sin\alpha}
\end{equation}}
\textsf{Это маленький пример рубленого текста.}
```

Figure 1: The LCY encoding demo.
Рис. 1: Демонстрация LCY.

Данная распечатка демонстрирует некоторые примеры набора на русском языке с использованием транслитерации, в частности, набора кавычек, знака тире и математических формул. В русских изданиях используются:

1. кавычки „лапки“, аналогично немецким кавычкам;
2. кавычки «елочки», которые можно набрать двумя способами;
   (а) с помощью знаков „больше“ или „меньше“: `<`, `>` — вот как это выглядит: «ёлочки»;
   (б) с использованием активной кавычки: `"<` и `">`;
3. знак тире в русской полиграфии несколько короче и всегда отбивается небольшими пробелами около 0,2 кегля, см. п. 4;
4. определен знак номера —№, который набирается как `\No`.

А еще можно показать следующие формулы с измененными математическими операторами

$$\mathrm{tg}\,\alpha = \frac{\sin\alpha}{\cos\alpha}, \quad \text{или} \quad \mathrm{ctg}\,\alpha = \frac{\cos\alpha}{\sin\alpha} \tag{3}$$

следующие формулы выделены полужирным шрифтом:

$$\mathbf{tg}\,\boldsymbol{\alpha} = \frac{\sin\boldsymbol{\alpha}}{\cos\boldsymbol{\alpha}}, \quad \textbf{или} \quad \mathbf{ctg}\,\boldsymbol{\alpha} = \frac{\cos\boldsymbol{\alpha}}{\sin\boldsymbol{\alpha}} \tag{4}$$

Это маленький пример рубленого текста.

```
\def\theenumii{\asbuk{enumii}}

Dannaya raspechatka demonstriruet nekotorye primery nabora na russkom yazyke s
ispolp1zovaniem transliteratsii, v chastnosti, nabora kavychek, znaka tire i
matematicheskikh formul. V russkikh izdaniyakh ispolp1zuyut{}sya :
\begin{enumerate}
\item kavychki "`lapki"', analogichno nemeckim kavychkam ;
\item kavychki "<elochki">, kotorye mozhno nabratp1 dvumya sposobami ;
\begin{enumerate}
\item s pomoshchp1yu  znakov "`bolp1she"' ili "`menp1she"' :{}
{\selectlanguage{english}\verb|<|, \verb|>|} "--- vot kak e1to vyglyadit: <e0lochki> ;
\item s ispolp1zovaniem aktivnoi0 kavychki :{} {\selectlanguage{english}\verb|"<|} i
{\selectlanguage{english}\verb|">|} ;
\end{enumerate}
\item znak tire v russkoi0 poligrafii neskolp1ko koroche i vsegda otbivaet\/sya
nebolp1shimi probelami okolo 0,2 keglya, sm. p. 4 ;
\item opredelen znak nomera"---\No, kotoryi0 nabiraet{}sya kak {\selectlanguage{english}%
\verb|\No|}.
\end{enumerate}
A eshche mozhno pokazatp1 sleduyushchie formuly s izmenennymi matematicheskimi
operatorami
\begin{equation} \tg\alpha=\frac{\sin\alpha}{\cos\alpha},\quad
\rusmath{ili}\quad\ctg\alpha=\frac{\cos\alpha}{\sin\alpha} \end{equation}
sleduyushchie formuly vydeleny poluzhirnym shriftom :{}
{\boldmath\begin{equation} \tg\alpha=\frac{\sin\alpha}{\cos\alpha},\quad
\rusmath{ili}\quad \ctg\alpha=\frac{\cos\alpha}{\sin\alpha} \end{equation}}
\textsf{E1to malenp1kii0 primer rublenogo teksta.}
```

Figure 2: The `LWN` encoding demo.
Рис. 2: Демонстрация `LWN`.

Пусть X и Y — два многообразия, а Ω_X и Ω_Y — открытые подмножества соответственно в T^*X и T^*Y. Мы показываем, что при соответствующих предположениях корректно определены функторы Φ_K, Ψ_K из $\mathbf{D}^b(Y;\Omega_Y)$ и из $\mathbf{D}^b(X;\Omega_X)$ в $\mathbf{D}^b(Y,\Omega_Y)$, задающие эквивалентности категорий. Далее, если $\chi : \Omega_X \tilde{\to} \Omega_Y$ — контактное преобразование, то мы показываем, что после сужения множества Ω_X и Ω_Y всегда можно, используя эти ядра, построить эквивалентность $\mathbf{D}^b(X;\Omega_X) \tilde{\to} \mathbf{D}^b(Y;\Omega_Y)$. Пусть теперь M — гиперповерхность в X, а N — гиперповерхность в Y. Предположим, что контактное преобразование χ преобразует $T^*_M X \cap \Omega_X$ в $T^*_N Y \cap \Omega_Y$. Если график преобразования χ ассоциирован с конормальным расслоением к некоторой поверхности S в $X \times Y$ и если в качестве ядра K выбран пучок A_S, то $\Phi_K(A_N) \simeq A_M(d)$ в $\mathbf{D}^b(X;p), (p \in \Omega_X)$, где d — сдвиг, который мы вычисляем, используя индекс инерции.

```
\def\tildeto{\leavevmode\vcenter{\baselineskip0pt\lineskip-.25ex
 \ialign{$##$\crcr\hidewidth\sim\hidewidth\crcr\to\crcr}}}
% emulation of AmSTeX symbols

\def\varOmega{\mathit{\Omega}}
\def\varPhi{\mathit{\Phi}}
\def\varPsi{\mathit{\Psi}}

\noindent Пусть $X$ и $Y$"---два многообразия, а $\varOmega_X$ и $\varOmega_Y$"---
открытые подмножества соответственно в $T^*X$ и $T^*Y$. Мы показываем, что при
соответствующих предположениях корректно определены функторы $\varPhi_K$,
$\varPsi_K$ из $\mathbf{D}^b(Y;\varOmega_Y)$ и из $\mathbf{D}^b(X;\varOmega_X)$
в $\mathbf{D}^b(Y,\varOmega_Y)$, задающие эквивалентности категорий. Далее,
если $\chi:\varOmega_X\brkrel{\tildeto}\varOmega_Y$"---контактное преобразование,
то мы показываем, что после сужения множества $\varOmega_X$ и $\varOmega_Y$
всегда можно, используя эти ядра, построить эквивалентность
$\mathbf{D}^b(X;\varOmega_X)\brkrel{\tildeto}\mathbf{D}^b(Y;\varOmega_Y)$.
Пусть теперь $M$"--- гиперповерхность в $X$, а $N$ "--- гиперповерхность в $Y$.
Предположим, что контактное преобразование $\chi$ преобразует $T^*_MX\cap\varOmega_X$
в $T^*_NY\cap \varOmega_Y$. Если график преобразования $\chi$ ассоциирован с
конормальным расслоением к некоторой поверхности $S$ в $X\times Y$ и если в
качестве ядра $K$ выбран пучок $A_S$, то $\varPhi_K(A_N)\simeq A_M(d)$ в
$\mathbf{D}^b(X;p),(p\in\varOmega_X)$, где $d$"--- сдвиг, который мы вычисляем,
используя индекс инерции.
```

Figure 3: The math breaks demo.
Рис. 3: Демонстрация разрывов в формулах.

A Package for Church Slavonic Typesetting

Andrey Slepukhin

Introduction

The multilingual ability of TEX is one of its most important properties. Due to TEX it has become possible to produce high-quality books in many different languages (sometimes with very exotic grammatical rules). During the past 16 years, TEX has became a real polyglot and it seems that it doesn't want to stop evolving. In this paper one more, maybe rather exotic, example of practical usage of TEX is considered, along with many ideas and solutions which result from five years of experience with TEX.

General solutions

What does a language-specific package have to look like from the point of view of a computer typesetting system? It must include at least the following components:

- quality fonts;
- tools for simplifying the text formatting;
- hyphenation table;
- punctuation or some other poligraphic rule description.

These requirements form the basis of SlavTEX development. The first two items have been realized satisfactorily. As to the realization of the third one — it depends on the volume of the dictionary, which is not sufficiently complete yet. The fourth item is absent because Church Slavonic has no precise rules for punctuation and other similar objects.

Fonts

Designing quality fonts is, in general, a very hard task; moreover, the author's knowledge on the subject at the beginning of this work was minimal. So, the designing of the base version of fonts took more than half a year, and various improvements are still under development. The work has been complicated by the fact that a large number of symbols (there are already 44 purely alphabetic characters) in the Church Slavonic alphabet must be included. Also the glyphs for symbols have very few shared elements. The typeface, which was in wide use at the beginning of the twentieth century, was taken as a model for the fonts created here. The following technology was applied to develop the fonts: the symbols were magnified and separate elements extracted, then base and control points of outline curves were placed manually and the METAFONT macros were designed; the symbols obtained were finally improved using the METAFONT graphic output. The current font version consists of 148 symbols including some old Slavonic letters which are no longer in use.

The diacritic problem

In developing SlavTEX, the main problem was that every word in a Church Slavonic text has at least one diacritic. None of the computer typesetting systems known to the author (except TEX, of course), contains any convenient tools for typesetting a text with accents. TEX uses the `\accent` macro for this purpose, but this macro seems to be designed for rather infrequent usage, because it gives the following undesirable effects:

- the kern between accented and previous symbols disappears;
- TEX doesn't allow any hyphens in the remainder of the word after an accented symbol and can make invalid hyphens in the initial part of the word.

These effects arise because TEX uses explicit kerning while expanding the `\accent` macro. So, it seems that the best solution for the diacritic problem (realized for many European languages, for example) is a method whereby a letter together with an accent is represented by a single character in the font. However, in the case of Church Slavonic, this solution cannot be applied in this form because there are too many possible 'letter–accent' pairs, and the limit of 256 symbols would soon be exhausted.

It would be wonderful if the following idea worked: the various 'letter-accent' combinations would be placed in a font at identical positions modulo 256 (i.e., 256 positions apart), so that their metrics would coincide. Unfortunately, it is not possible to force TEX to put a symbol with character code greater than 255 into the `dvi` file. Such a restriction is quite unexpected since the `dvi` file format supports the use of symbols with character codes up to $2^{32} - 1$. One other well-known method to deal with the accents is their realization as strongly shifted left characters of zero width. Such an option is unsatisfactory too because it does not solve the kerning problem while significantly complicating construction of the hyphenation table.

To solve the accent problem we need to understand where diacritics are placed in Church Slavonic. Some can be placed only over the first letter in the word, and some can be placed only over the last letter. These two cases are realized by the special macros `\fcaccent` and `\lcaccent`. The second macro can be written in a very simple way because it only needs to locate the accent with

the help of kerns. The macro \fcaccent has a \nobreak\hskip0pt construction in addition, which enables the hyphenation of the word after the diacritic.

As for the accents in the middle of a word, some of them are realized together with the corresponding letters, and other represent symbols, in general used to abbreviate certain words (in Church Slavonic they are called *titlo*). The words containing these symbols cannot, as a rule, be hyphenated, but it was possible to write a special macro, placing an accent and preserving the kerns both before and after the symbol. It is a quite sophisticated macro which uses such TEX commands as \futurelet.

To make inputting text easier, the symbols ’, ", ‘, ˜, _, | and < are made active and are expanded to the corresponding macros. The selection of Church Slavonic mode is realized by the \beginslav macro, and the return to normal mode is realized by the \endslav macro.

Separating colors

Another problem that had to be solved during package development was that of color separation. Almost all Church Slavonic texts are two-colored. Unfortunately, using colors via PostScript was impractical, because a PostScript-printer is a rarity in Russia. So, to make color separations and to obtain separate slides for each color, the SLITEX idea of using *invisible* fonts was applied. However, kerning problems make the use of SLITEX impossible. Indeed, having a word with a first letter emphasized by the use of another color (in Church Slavonic texts this occurs very often), SLITEX loses the required kern between the first letter and the remainder of the word when switching to the other font. So, for such cases we need special macros. To implement the color separation a special font selection scheme was designed, somewhat similar to NFSS. After including the font description file and appropriate macros, the user can declare the use of any color via the macro \newcolor(<color>). This macro gives rise to the macros \<color>g{<any text>} and \<color>. The first of these switches the color, preserving kerning, and the second switches it without preserving any implicit kern (TEX interprets this macro in the simplest way, so its usage makes sense). Now, typing \showcolor(<color>) or \hidecolor(<color>) in the input file, we can make any selection by a specified color visible or invisible in the output. The text before the first usage of \<color>g{<any text>} or \<color> will always be visible.

Numbering

In Church Slavonic, literal numeration is accepted, which can be described by the following algorithm:

> Given an integer $n \geq 0$, let $S(n)$ be its representation in Church Slavonic. Consider the following table:

n	$S(n)$	n	$S(n)$	n	$S(n)$
1	а҃	10	і҃	100	р҃
2	в҃	20	к҃	200	с҃
3	г҃	30	л҃	300	т҃
4	д҃	40	м҃	400	у҃
5	є҃	50	н҃	500	ф҃
6	ѕ҃	60	ѯ҃	600	х҃
7	з҃	70	ѻ҃	700	ѱ҃
8	и҃	80	п҃	800	ѿ҃
9	ѳ҃	90	ч҃	900	ц҃

The representation of zero is absent in Church Slavonic, but let it be empty for convenience.

If $10 \leq n < 20$, then $S(n) = S(n \bmod 10)S(10)$. If $20 \leq n < 100$, then $S(n) = S(n-(n \bmod 10))S(n \bmod 10)$. If $100 \leq n < 1000$, then $S(n) = S(n-(n \bmod 100))S(n \bmod 100)$. If $1000 \leq n < 10000$, then $S(n) = \text{҂}S(n-(n \bmod 1000))S(n \bmod 1000)$.

There are disagreements about the representation of numbers greater than 9999. In the SlavTEX package, we use a modern option, where the rule $S(n) = \text{҂}S(n-(n \bmod 1000))S(n \bmod 1000)$ is true for all numbers ≥ 1000. The macro \slnum(<number>) automatically generates the number representation in Church Slavonic. For example, \slnum(1995) gives ҂ацч҃є. One has to be careful since this macro is valid only inside Church Slavonic mode.

TEX without encoding

During the development of SlavTEX, an idea appeared which solves the compatibility problem when transferring any package to another platform. This problem is especially acute in Russia because Russian letter encodings on different platforms do not coincide. A version of TEX Cyrillization made by CyrTUG is specific for PC-compatible computers under MS-DOS. This results in a justified unhappiness amongst the many UNIX users in big research institutes which need TEX most of all.

The procedure to easily transfer any TEX package to different platforms is given below:

- The encoding table containing a map between character codes and their symbolic names (for example, like PostScript names) must be defined for each specific platform and font family.

- A utility (which can even be written in TEX!) must be created to generate two files from the original encoding table — a TEX encoding table and a METAFONT encoding table.
- A set of METAFONT macros must be added to redefine the `beginchar` macro; it must allow the use of symbolic names instead of character codes by declaring `usenames:=1` or whatever like this.
- A hyphenation table must be written, using symbolic names; when generating the base file, TEX should first read the encoding, then it should convert the original hyphenation table into a temporary file using the current encoding and then it should read the file obtained.
- `\catcode`, `\lccode` and `\uccode` should be defined using symbolic names.

A variant of this idea is implemented in the latest version of the package presented and is now being tested. It is hoped that new Cyrillization versions from CyrTUG will be written in the form described above. It would facilitate the work of the many TEX users in Russia and of people who need to typeset Russian (or other Cyrillic) texts.

Type 1 from METAFONT?

One more idea, implemented as part of the SlavTEX project, was inspired by an article by Jackowski and Ryćko [2]. Moreover, the arrival of a PostScript-printer on the author's desk helped stimulate its realization. There exists a set of METAFONT macros with which one can obtain a text representation of Type 1 fonts from METAFONT sources and from that, a downloadable font by L. Hetherington's Type 1 utilities. This macro package was initially designed to solve the specific problem of representating Church Slavonic in Type 1 format, but the conversion of Computer Modern fonts (and others) is also possible. This work deserves separate treatment and is not described in this paper.

Problems and plans

The most important problem today is adapting SlavTEX to LATEX 2_ε, or rather, its realization as a LATEX 2_ε package. The author also plans to develop a package for typesetting music in non-linear notation (so-called *krjuki*), in use before the eighteenth century. The following problems associated with Church Slavonic typesetting also should be noted: designing a font of initial caps and a special font for headings. In this font different combinations of letters must have specific glyphs (this task seems to be rather gigantic, because such a font would have a monstrous number of symbols and ligatures).

It would be nice if others shared the author's interest in the problem of Church Slavonic and ancient texts. Maybe at some future date, a multilingual edition of the Bible (in Church Slavonic, Greek, Latin, Hebrew ... what else?) produced with TEX could come into existence.

Examples

A simple example of a Church Slavonic text:

```
г|сди <i_исе х|срт‘е, с_не б_жій,
пом’илуй м‘я гр’эшнаго
```

and the result of its compilation:

An example of color separation: a sequence of macros

```
\beginslav\family(slav)\size(12)%
\black%
\def\pray{%
\redg Г|с{ди} <i_исе х|срт‘е, с_не
б_жій, пом’илуй м‘я гр’эшнаго
}%
\black%
\hidecolor(black)%
\showcolor(red)%
\par\noindent\pray
\hidecolor(red)%
\showcolor(black)%
\par\noindent\pray
\showcolor(red)%
\par\noindent\pray
\endslav
```

gives the result

This example shows that accents can be placed over a group of characters, and not only over a single character.

References

[1] Alipiy, Ieromonakh (Gamanovich). *Grammatika tserkovno-slavjanskogo jazyka*. Moscow: Palomnik, 1991.

[2] Jackowski, Bogusław, and Marek Ryćko. "Labyrinth of METAFONT paths in outline", *Proceedings of the Eighth European TEX Conference* (Sept. 26–30, 1994, Gdańsk, Poland), 18–32.

[3] Donald E. Knuth. *The TEXbook*. Addison Wesley, Reading, MA, 1990.

[4] Donald E. Knuth. *The* METAFONT*book*. Addison Wesley, Reading, MA, 1990.

[5] *Slovar' russkogo jazyka XI–XVII vv.* Moscow: Nauka, 1975.

◇ Andrey Slepukhin
Lavra, Sergiev Posad, Russia
Email: pooh@shade.msu.ru

Пакет для набора церковно-славянских текстов

Андрей Слепухин

Введение

Одним из самых важных свойств TEX'а безусловно является его многоязычность. Благодаря TeX'у стало возможным издание книг на самых разных языках (порой с весьма прихотливыми грамматическими правилами) с высочайшим полиграфическим качеством. За 10 с лишним лет своего существования TEX стал настоящим полиглотом и, похоже, не собирается останавливаться на достигнутом. В этой статье рассматривается еще один, быть может, довольно экзотический пример применения многоязычности TEX'а на практике, а также многие идеи и решения, возникшие в результате 5-летнего опыта работы с TEX'ом.

Общий обзор

Что с точки зрения компьютерной издательской системы должен представлять собой пакет для работы с каким-либо языком? Он должен содержать по крайней мере следующие компоненты:

- качественные шрифты;
- средства облегчения набора текста;
- таблицу переносов;
- описание полиграфических правил, используемых в данном языке;

Эти требования и стали основой при разработке пакета CLiCkTEX. Первые два пункта удалось реализовать вполне удовлетворительно, для качественной реализации третьего пока не хватает необходимого объема словаря, а четвертый практически отсутствует, так как в церковнославянском языке каких-либо полиграфических правил просто нет.

Шрифты

Разработка шрифтов вообще является очень трудоемким занятием, к тому же познания автора в этом вопросе к началу работы были минимальными. Поэтому на разработку базового варианта шрифтов ушло больше полугода, а внесение различных улучшений продолжается до сих пор. Среди факторов, усложнивших работу, следует отметить то, что церковно-славянский алфавит содержит большое количество символов (только букв—44), и начертания этих символов имеют мало похожих элементов. В качестве образца был взят шрифт, получивший широкое распространение в начале XX века. Технология разработки шрифта была следующая: символы увеличивались, разбивались на отдельные элементы, затем вручную приблизительно проставлялись опорные точки и эти элементы описывались с помощью макрокоманд METAFONT'а; полученные символы далее доводились с использованием графического вывода METAFONT'а. В настоящий момент шрифт содержит 148 символа, включая некоторые буквы старо-славянского языка, вышедшие из употребления.

Проблема диакритических знаков

Основная проблема, возникшая при разработке пакета CLiCkTEX, была связана с тем, что в церковно-славянских текстах каждое слово имеет по крайней мере один диакритический знак. Ни одна из известных автору систем компьютерного набора не имеет удовлетворительных средств для набора текстов с диакритическими знаками. Лучший из известных способ предлагает TEX. Для этой цели он использует макрокоманду `\accent`, но она, видимо, была рассчитана лишь на достаточно редкое применение, потому что использование этой макрокоманды вызывает два нежелательных эффекта:

- исчезает керн между акцентируемым символом и предыдущим;
- конец слова, начиная с акцентируемого символа, не переносится вообще, а начало может переноситься неправильно;

Эти эффекты возникают из-за того, что TEX, при реализации макрокоманды `\accent` использует явный керн. Поэтому, вероятно лучшим решением (что и реализовано, например, для некоторых европейских языков) является реализация буквы вместе с акцентом в виде отдельного символа в шрифте. Однако, в случае с церковнославянским языком такое решение в чистом виде не проходит, поскольку возможных комбинаций буква—диакритический знак так много, что 256 символов в шрифте просто не хватит! Наверное, очень хорошим решением проблемы было бы следующее: различные пары буква—диакритический знак имеют в шрифте позиции совпадающие по модулю 256, и их метрики в TFM-файле совпадают. К сожалению, никаким

образом от TEX'а нельзя добиться, чтобы он в DVI-файл вывел символ с кодом большим, чем 256. Такое ограничение тем более непонятно, поскольку формат DVI-файла поддерживает использование символов с кодами до $2^{32} - 1$. Еще один известный способ борьбы с диакритическими знаками—делать их в виде символов с нулевой шириной и сильно смещенных влево. Такой вариант тоже является неудовлетворительным, поскольку проблему с кернингом он не решает и, кроме того, построение таблицы переносов в этом случае было бы делом весьма затруднительным.

Чтобы решить проблему, пришлось понять, по какому принципу расставляются диакритические знаки в церковно-славянском языке. Оказалось, что некоторые из них встречаются только над первой буквой слова, а некоторые—только над последней. Эти два случая реализуют специальные макрокоманды `\fcaccent` и `\lcaccent`. Написание последней не представляет особого труда, так как она лишь размещает с помощью кернов диакритический знак. В макрокоманде `\fcaccent` кроме этого, используется конструкция `\nobreak\hskip0pt`, которая после выравнивания диакритического знака с помощью кернов разрешает перенос слова.

Что касается тех диакритических знаков, которые встречаются в середине слова, то часть из них реализована вместе с соответствующими буквами в виде отдельных символов, а часть представляет собой знаки, применяемые, в основном, для сокращения написания некоторых слов (в церковно-славянском языке они носят название "титло"). Слова, содержащие эти знаки, как правило, не переносятся, поэтому стало возможным написать макрокоманду, которая ставит над символом акцент, сохраняя керн как до, так и после этого символа. Это довольно хитрая макрокоманда, в частности, для ее написания пришлось применить такое средство, как `\futurelet`.

Для удобства набора текста символы `'`, `"`, `` ` ``, `~`, `_`, `|` и `<` сделаны активными и раскрываются в соответствующие макрокоманды. Переход в церковно-славянский режим осуществляется макрокомандой `\beginslav`, а выход—макрокомандой `\endslav`.

Разделение цветов

Другая проблема, с которой пришлось столкнуться при написании пакета—это разделение цветов. Практически все церковно-славянские тексты являются двуцветными. К сожалению, от реализации разделения цветов через PostScript пришлось отказаться, так как в России PostScript-принтер все еще является большой редкостью. поэтому чтобы реализовать разделение цветов и получить отдельные слайды для каждого текста, была использована идея SLITEX'а об использовании "невидимых" шрифтов. Однако, из-за проблем с кернингом, использовать SLITEX в чистом виде не удается. В самом деле, если у нас есть слово, первая буква которого выделена другим цветом (а такое в церковно-славянских текстах случается очень часто), то керн между выделенной буквой и остальным словом пропадает в связи с переключением на другой шрифт. Поэтому, для такого рода выделений приходится использовать специальные макрокоманды. Чтобы реализовать разделение цветов, была реализована специальная система подключения шрифтов, немного похожая на NFSS. После этого пользователь может описать используемые цвета макрокомандами `\newcolor(<цвет>)`. Эта макрокоманда порождает макрокоманды `\<цвет>g{<текст>}` и `\<цвет>`. Первая из них переключает цвет с сохранением кернинга, вторая—без сохранения кернинга (она более просто интерпретируется TeX'ом, поэтому ее использование имеет смысл). Теперь, если во входном файле указать `\showcolor(<цвет>)`, то выделения данным цветом будут видны при печати. Текст, расположенный до первой макрокоманды `\<цвет>` или `\<цвет>g` будет видимым независимо от макрокоманд `\showcolor`.

Нумерация

В церковно-славянском языке принята буквенная нумерация, описываемая следующим алгоритмом:

Пусть n—целое число, $n \geq 0$, а $S(n)$—его представление в церковно-славянском языке. Для начала введем таблицу:

n	$S(n)$	n	$S(n)$	n	$S(n)$
1	а	10	і	100	р
2	в	20	к	200	с
3	г	30	л	300	т
4	д	40	м	400	у
5	е	50	н	500	ф
6	ѕ	60	ѯ	600	х
7	з	70	о	700	ѱ
8	и	80	п	800	ѿ
9	ѳ	90	ч	900	ц

Представление числа 0 в церковно-славянском языке отсутствует, но для удобства будем считать его пустым.

Если $10 \le n < 20$, то $S(n) = S(n \bmod 10)S(10)$. Если $20 \le n < 100$, то $S(n) = S(n - (n \bmod 10))S(n \bmod 10)$. Если $100 \le n < 1000$, то $S(n) = S(n - (n \bmod 100))S(n \bmod 100)$. Если $1000 \le n < 10000$, то $S(n) =_{\text{ѧ}} S(n - (n \bmod 1000))S(n \bmod 1000)$.

Относительно представления чисел, больших, чем 9999, существуют разногласия, в пакете ЄлѣкTEX реализован современный вариант, где правило $S(n) =_{\text{ѧ}} S(n - (n \bmod 1000))S(n \bmod 1000)$ распространяется на все числа ≥ 1000. Автоматическую генерацию записи чисел в церковнославянском языке реализует макрокоманда `\slnum(<число>)`. Например, `\slnum(1995)` даст ѧцѱѳ. Будьте внимательны: макрокоманда `\slnum` действует только в церковно-славянском режиме.

TEX без кодировок

В процессе работы над ЄлѣкTEX'ом появилась идея, которая позволяет решить проблемы совместимости при переносе какого-либо пакета на другую платформу. Эта проблема особенно актуальна в условиях России, поскольку на разных платформах кодировка русских букв различна. Версия кириллизации TEX'а, распространяемая CyrTUG, рассчитана в основном на пользователей IBM PC-совместимых компьютеров, работающих под MS DOS. Это, в частности, вызывает справедливое недовольство многочисленных пользователей UNIX'а, среди которых—крупные научные институты, которые в TEX'е нуждаются больше всего.

Идея легкого переноса на другую платформу заключается в следующем:

- Вводится таблица кодировки для конкретной платформы и конкретного семейства шрифтов, содержащая соотношение между символьными кодами и их именами (например, используемыми в PostScript'е);
- Определяется утилита (она может быть написана даже на TEX'е!), которая из таблицы кодировки генерирует два файла: первый—таблица кодировки для METAFONT'а, второй—для TEX'а
- Добавляется набор макрокоманд для METAFONT'а, который переопределяет макрокоманду `beginchar` так, что при установке `usenames:=1` в ней могут быть использованы символьные имена;
- Таблица переносов описывается с использованием символьных имен; при генерации формата TEX читает таблицу кодировки, затем конвертирует таблицу переносов в промежуточный файл с использованием текущей кодировки и читает полученный файл;
- `\catcode`, `\lccode` и `\uccode` описываются также с использованием символьных имен;

Вариант этой идеи реализован в последней версии представленного пакета и находится в процессе тестирования. Хочется надеяться, что аналогичным образом будут оформлены и последующие версии кириллизации TEX'а, распространяемые CyrTUG, во всяком случае это облегчило бы работу для многих пользователей TEX'а в России, а также для всех тех, кому необходимо работать с русскими текстами.

Type 1 из METAFONT'а?

Еще одна идея, реализованная при разработке ЄлѣкTEX'а была навеяна статьей [3], а также появлением на столе автора PostScript-принтера. В настоящее время имеется набор макроопределений для METAFONT'а, с помощью которого можно получить шрифт в формате Type 1 в текстовом виде, а затем, используя Type 1 утилиты, написанные L. Hetherington'ом,—в виде загружаемых шрифтов. С помощью этого набора макрокоманд можно получить представление в виде Type 1 и шрифтов Computer Modern. Эта работа заслуживает отдельного рассмотрения и, поэтому в данной статье не описывается.

Проблемы и планы

Самой важной из существующих на сегодняшний день проблем остается подключение пакета ЄлѣкTEX к LATEX 2_ε, точнее его оформление как LATEX 2_ε-пакета. В будущем автор также собирается заняться разработкой пакета для записи музыки в безлинейной нотации (так называемые "крюки"), использовавшейся в России вплоть до XVII века. В числе проблем, связанных с церковно-славянским языком, можно упомянуть разработку шрифтов для буквиц, а также—специального шрифта для заголовков, в котором различные буквосочетания имеют различные начертания (эта задача кажется весьма фантастичной, так как подобный шрифт должен иметь очень много символов и огромное число лигатур).

Хочется надеяться, что кто-нибудь разделит интерес автора к проблеме церковнославянских и древних текстов. Может быть, когда-нибудь появится многоязычное издание Библии (на церковно-славянском, греческом, латинском, еврейском ... на каком еще?), выполненое с помощью TEX'а.

Примеры

Простой пример набора текста на церковно-славянском языке:

```
г|сди <i_исе х|срт‘е, с_не б_жій,
пом’илуй м‘я гр’эшнаго
```

и результат его компиляции:

гди їйсе хрт҃е, сйе бжїй, помúлуй мѧ грѣшнаго

Пример цветоделения: последовательность команд

```
\beginslav\family(slav)\size(12)%
\def\pray{%
\redg Г|с{ди} <i_исе х|срт‘е, с_не
б_жій, пом’илуй м‘я гр’эшнаго
}%
\black%
\hidecolor(black)%
\showcolor(red)%
\par\noindent\pray
\hidecolor(red)%
\showcolor(black)%
\par\noindent\pray
\showcolor(red)%
\par\noindent\pray
\endslav
```

приводит к следующему результату:

Г

гди їйсе хрт҃е, сйе бжїй, помúлуй мѧ грѣшнаго
Гди їйсе хрт҃е, сйе бжїй, помúлуй мѧ грѣшнаго

В этом примере хорошо видно, что акцент можно ставить не только над одним символом, но и над сочетанием символов.

Список литературы

[1] Donald E. Knuth. *The TEXbook.* Addison Wesley, Reading, MA, 1990.

[2] Donald E. Knuth. *The* METAFONT*book.* Addison Wesley, Reading, MA, 1990.

[3] Bogusław Jackowski, Marek Ryćko. Labyrinth of METAFONT paths in outline. *EuroTEX Proceedings, 1994: 18–32.*

[4] Иеромонах Алипий (Гаманович). *Грамматика церковно-славянского языка.* Паломник, Москва, 1991.

[5] *Словарь русского языка XI–XVII вв.* Наука, Москва, 1975.

◇ Андрей Слепухин
Лавра, Сергиев Посад, Россия
Email: pooh@shade.msu.ru

Production Notes on the Russian Papers

Michel Goossens

The two previous articles both have their first part in English, then an equivalent text in Russian, and finally a part with examples in both languages. Apart from these similarities the Russian text was coded in completely different ways for both articles.

The Lapko-Makhovaya article uses the KOI8 encoding popular in the Unix community in Russia. The KOI8 input was matched by hyphenation patterns and a font layouts using the same encoding.

On the other hand the Slepukhin article was coded in the Alternativniy encoding popular on PC machines, and the normal Russian and SlavTEX fonts matched that encoding, as did the Russian hyphenation patterns needed to run that paper.

Both Russian encodings are 8-bits wide and coincide with ASCII in positions 0 to 127, but they place the Cyrillic characters in completely different locations between 128 and 255[1].

To run both articles two different formats had to be generated. The first contained the English and Russian hyphenation patterns using the KOI8 encoding, the second one English and Russian in the Alternativniy encoding. Also different style files and font encodings were necessary. Both font instances are based on the LH Cyrillic font family developed by Olga Lapko and her colleagues of CyrTUG [1], only the encoding is different.

Presently CyrTUG, the Cyrillic TEX Users Group has a working group coordinated by Olga Lapko trying to come up with a unique encoding, but, as stated in the `babel` article, we will probably have to wait until the Ω system is ready to try and solve the problems discussed in Section “Encoding and font problems”.

Finally, as a technical aside, the picture on the next page, showing an example of SlavTEX output, had its text typeset with the SlavTEX system, while the first letter of the text and the surrounding frame are bitmaps.

References

[1] Olga G. Lapko. MAKEFONT as part of the CyrTUG-EMTEX package. *Proceedings of the 8th European TEX Conference*, pages 110–118, Gdańsk, Poland, 1994.

◇ Michel Goossens
CERN, Geneva, Switzerland
Email: goossens@cern.ch

[1] More information on KOI8 can be found at the URL `http://www.nar.com/tag/koi8_explained.html`.

МОЛИ́ТВА РАЗРѢШИ́ТЕЛЬНАѦ

Ѿ ІЕРЕ́А

НАД҆ ПРЕСТА́ВЛЬШИМСѦ ЧТО́МАѦ

Гдⷭ҇ь на́шъ і҆и҃съ хрⷭ҇то́съ, бж҃е́ственною свое́ю бл҃года́тїю, да́ромъ же и҆ вла́стїю, да́нною ст҃ы̑мъ є҆гѡ̀ ᲂу҆чн҃кѡ́мъ и҆ а҆п҃лѡмъ, во є҆́же вѧза́ти и҆ рѣши́ти грѣхи̑ человѣ́кѡвъ, [ре́къ и҆̑мъ: прїими́те дх҃а ст҃а́го, и҆́хже ѿпꙋсти́те грѣхи̑, ѿпꙋстѧ́тсѧ и҆̑мъ: и҆́хже ᲂу҆держи́те, ᲂу҆держа́тсѧ: и҆ є҆ли̑ка а҆́ще свѧ́жете и҆ разрѣши́те на земли̑, бꙋ́дꙋтъ свѧ̑зана и҆ разрѣшє́на и҆ на нб҃си̑] ѿ ѻ҆нѣ́хъ же и҆ на ны̀ дрꙋ́гъ дрꙋгопрїима́тельнѡ прише́дшею, да сотвори́тъ чрез̾ менѐ смире́ннаго проще́нно и҆ сїѐ по дꙋ́хꙋ ча́до [и҆́мкъ] ѿ всѣ́хъ, є҆ли̑ка ꙗ҆́кѡ человѣ́къ согрѣшѝ бг҃ꙋ сло́вомъ, и҆лѝ дѣ́ломъ, и҆лѝ мы́слїю и҆ всѣ́ми свои́ми чꙋ́вствы, во́лею и҆лѝ нево́лею, вѣ́дѣнїемъ и҆лѝ невѣ́дѣнїемъ. А҆́ще же под̾ клѧ́твою и҆лѝ ѿлꙋче́нїемъ а҆рхїере́йскимъ и҆лѝ і҆ере́йскимъ бы́сть, и҆лѝ а҆́ще клѧ́твꙋ ѻ҆тца̀ своегѡ̀ и҆лѝ ма́тере своеѧ̀ наведѐ на сѧ̀, и҆лѝ своемꙋ̀ прокла́тїю подпадѐ, и҆лѝ клѧ́твꙋ престꙋпѝ, и҆лѝ и҆ны́ми нѣ́кїими грѣхи̑ ꙗ҆́кѡ человѣ́къ свѧза́сѧ: но ѡ҆ всѣ́хъ си́хъ се́рдцемъ сокрꙋше́ннымъ пока́ѧсѧ, и҆ ѿ тѣ́хъ всѣ́хъ вины̀ и҆ ᲂу҆́зы да разрѣши́тъ є҆гѡ̀ [ю҆̀]: є҆ли̑ка же за не́мощь є҆стества̀ забве́нїю предадѐ и҆ та̑ всѧ̑ да прости́тъ є҆мꙋ̀ [є҆́й], чл҃вѣколю́бїѧ ра́ди своегѡ̀, мл҃твами прест҃ы́ѧ и҆ преблг҃ослове́нныѧ вл҃чцы на́шеѧ бцⷣы и҆ приснодв҃ы мр҃і́и, ст҃ы́хъ сла́вныхъ и҆ всехва́льныхъ а҆п҃лъ и҆ всѣ́хъ ст҃ы́хъ, а҆ми́нь.

An example of SlavTEX—Пример Слв҃TEXа

Fonts

Release 1.2 of the dc-fonts: Improvements to the European letters and first release of text companion symbols

Jörg Knappen

Abstract

I describe the improvements made to the dc fonts in release 1.2, and the text companion font added. The ec fonts will finally replace the present 128-character cm fonts as the default fonts of LaTeX.

1 Introduction

In 1990 at the TUG meeting in Cork, Ireland, the European TeX user groups agreed on a 256-character encoding supporting many European languages with latin writing. This encoding is both an *internal encoding* for TeX and a *font encoding*. This double nature is a consequence of the fact that both kind of encodings cannot be entirely separated within TeX.

The design goals of the Cork encoding are to allow as many languages as possible to be hyphenated correctly and to guarantee correct kerning for those languages. Therefore it includes many ready-made accented letters.

It also includes some innovative features which have not become very popular yet, though they deserve to become so. The first to mention is a special, zero-width invisible character, the compound word mark (cwm). The second is the separation of the two characters ⟨hyphen⟩ and ⟨hyphenchar⟩. By appropriate design of the hyphenchar glyph, hanging hyphenation can be achieved.

The final version of the Cork encoded fonts will be called ec (European Computer Modern or Extended Computer Modern) fonts. The current version, called dc fonts, is an intermediate step towards the final version. Note that in the case of bug fixes and improvements, the metrics may change.

The need for a text companion font was first articulated in the discussion of new 256 character mathematical fonts in 1993. In order to achieve a better orthogonality between text and math, some text symbols stored in the math fonts should be moved to the text companion fonts[1]. The text companion fonts are also the ideal place to store some new characters, like currency symbols.

[1] The archives of the math-font-discuss mailing list are available for ftp on `ftp.cogs.susx.ac.uk` in directory `pub/tex/mathfont`.

2 Supported Languages

The following languages are supported by the Cork encoding: Afrikaans, Albanian, Breton, Croatian, Czech, Danish, Dutch, English, Estonian, Faroese, Finnish, French, Frisian, Gaelic, Galician, German, Greenlandic, Hungarian, Icelandic, Irish (modern orthography), Italian, Letzeburgish, Lusatian (Sorbian), Norwegian, Polish, Portuguese, Rhaetian (Rumantsch), Rumanian, Slovak, Slovene, Spanish, Swedish, Turkish. Many non-European languages using the standard latin alphabet (e.g., Bahasa Indonesia, Suaheli) are also supported.

In Europe, the following languages aren't supported: Azeri, Basque, Catalan, Esperanto, Irish (old orthography), Latvian, Lithuanian, Maltese, Sami, Welsh. Of course, Greek and all languages with cyrillic writing are outside the scope of the Cork encoding.

3 Improvements to the dc Fonts

3.1 Accents

In good typography, the accent marks should look different for capitals and lowercase letters respectively. The accent over a capital should be of a 'flat' design, while the accent on a lowercase letter should be 'steep'. The Computer Modern fonts by D. E. Knuth only have steep accents, suitable for lowercase letters.

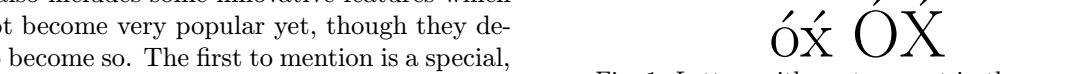


Fig. 1: Letters with acute accent in the cmr font

There are no pre-accented letters provided, which leads to problems with proper hyphenation and kerning. However, the floating accent approach guarantees the consistency of all accented letters.

With the (now out-of-date) version 1.1 of the dc fonts, the situation is different. We have predesigned accented letters for all languages included in the ISO standards 8859-1 and 8859-2. If an 'exotic' accented letter is needed, it does not fit with the provided ones.

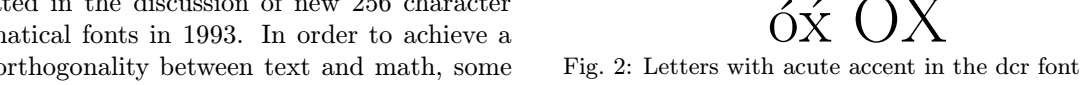


Fig. 2: Letters with acute accent in the dcr font (v1.1)

Note that the floating acute accent is the same for capitals and lowers, but different from both, being even steeper than the lowercase one.

With version 1.2 of the dc fonts, all inconsistencies have gone. The accents are different between capitals and lowers as they should be, and floating accents can be applied in a consistent manner.

Fig. 3: Letters with acute accent in the dcr font (v1.2)

Since the Cork encoding provides only one slot for each accent, the capital acute accent is taken from the *text companion* font tcr. This is possible because TEX allows cross-font accenting.

The acute accent and the readily accented letters were taken with kind permission of the authors from the polish pl fonts, which provide the highest available quality for these shapes. The hungarian double acute accent and the grave accent follow the design of the acute accent.

3.2 Quotation marks

The design of quotation marks provides another challenge for the ec fonts. In the Computer Modern fonts, they are optimised to english usage.

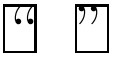


Fig. 4: Quotation marks in the cmr font

They lie asymmetrically in their boxen, which makes a wider space before and after a quotation. However, this kind of design produces a disaster, if the same english opening quotation mark is used as a german or polish closing quotation mark. Currently, macros have to compensate for this.

In the dc fonts 1.2, the quotation marks lie symmetrically in a tighter box, and the additional space is created by kerning against the `boundarychar`.

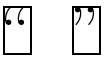


Fig. 5: Quotation marks in the dcr font (v1.2)

The boundarychar feature was introduced with TEX3 and METAFONT2; it is reasonable to assume that nowadays every TEX user has access to these or later versions[2].

3.3 Miscellaneous

The shapes for polish letters are now taken from the polish pl font, leading to improved shapes on the ogoneked letters and the crossed l.

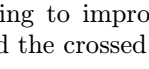


Fig. 6: Polish special letters in the dcr font (v1.2)

With the help of the czechoslovak TEX users' goup, the shapes of czech and slovak special letter have been improved, too.

[2] Maybe it was not a reasonable assumption in 1990, when the Cork encoding was born and the above mentioned versions were brand new.

Fig. 7: Some czech and slovak special letters in the dcr font (1.2)

The height of umlaut dots has been adjusted to the value contributed by the czechoslovak group (ä occurs in slovak); the value used in version 1.1 of the dc fonts was considered too low even by german users.

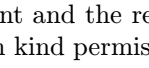


Fig. 8: The letter ä in cmr, dcr v1.1, and dcr v1.2

The `hyphenchar` is now designed to hang out of its bounding box, thus allowing for hanging hyphenation.

Fig. 9: Hyphen and hyphenchar with their bounding box

The release of version 1.2 also contains a new shape, a classical serif italic font. It was already prepared for version 1.1, but no parameter and driver files were present for it. It is an italic with upper serifs instead of ingoing hooks. This paragraph is typeset using the dcci font to show its appearance.

4 The tc Fonts

4.1 A text symbol encoding

Over the years, many reasons have accumulated for a new text symbol encoding. There are some text symbols currently stashed in the math fonts, the footnote marks, and the bullet (•) are among them.

Fig. 10: Footnote symbols from the cm math fonts

In 256-character math fonts they should not be preserved, but moved to a text symbol encoding.

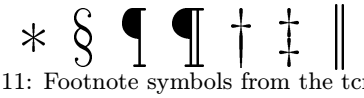


Fig. 11: Footnote symbols from the tcr font

I have added serifs to the paragraph sign (¶) in the serif typefaces, and I have added another one having only one vertical stroke. The design of the section sign (§) was improved significantly, as can be seen from the boldface glyphs.

Fig. 12: Old and new design of the section mark

ISO standards 8859-1 (Latin 1), 8859-2 (Latin 2), and 6937 contain several custom signs. It will be easier to typeset text encoded according to those standards if the necessary symbols were easily accessible through a text symbol font.

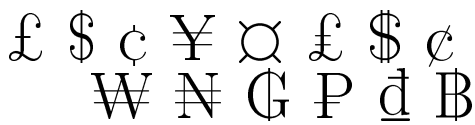


Fig. 13: Currency signs from the tcr font

Finally, I wanted to have different style accents for capitals and lowercase letters. Since the Cork encoding does not have space for another fourteen accent glyphs, I decided to have the lowercase accents, which are needed far more often, in the dc fonts, and to put the capital accents into the text companion fonts.

The users of commercial fonts also want to access all glyphs stored in those fonts. Since most of those glyphs are textual, they all should be included into a text symbol font encoding.

4.2 The font encodings TS1 and TSA

For mainly technical reasons, I think the candidates for a text symbol encoding should be distributed over two fonts, their encoding named TS1 and TSA respectively. There are important differences between the technologies supported by METAFONT and TEX compared to the path most commercial font suppliers choose.

The Computer Modern family of fonts supports the notion of a designsize, i.e., there are subtle differences between the shapes at different point sizes as illustrated in the next section. TEX is able to raise and lower letters, thus it does not need an already raised digit to produce a superscript. It can also produce nice fractions using a macro from the TEXbook, exercise 11.6, as 1/2, 1/4, and 1/6.

Most commercial vendors took the easier path; their fonts come in only one size and are scaled up and down to other sizes. Thus, a small superscript does not look right, and to compensate for this a predesigned superscript is added to the fonts. A subscript, too, because earlier text processors weren't able to raise or lower letters. For similar reasons, fraction glyphs were provided, or fraction were contructed out of a sequence ⟨superscript digit⟩ ⟨fraction⟩ ⟨subscript digit⟩, where fraction is a special slash to construct fractions.

On the other hand, it is almost impossible to follow this path with TEX and METAFONT: The size of the superscripts can be influenced by TEX macros, and therefore there is no unique 'virtual designsize' for ready-made superscripts.

The selection of superscripts offered by commercial vendors is at the moment rather sparse; many often needed ones are lacking.

Fig. 14: Some superscript letters lacking in expert fonts

Therefore the rule of thumb for the distribution of glyphs is the following: Put all glyphs wich can be conveniently made with METAFONT and are needed with TEX into the encoding TS1, and put the remaining glyphs, mainly superscripts and subscripts, into the encoding TSA. There are some duplications and deviations from this rule of thumb, e.g., superscript 1, 2, and 3 are part of ISO 8859-1, thus they occur in TS1 as well as in TSA.

5 Standard Control Sequences

The following standard control sequences are assigned with LATEX's T1 encoding for the dc fonts:
\r Ring accent (\r u gives ů)
\k Ogonek (\k e gives ę)
\dh, \DH Icelandic letter edh (ð, Ð)
\dj, \DJ Letter d with stroke (đ, Đ)
\ng, \NG Letter eng (ŋ, Ŋ)
\th, \TH Icelandic letter thorn (þ, Þ).

The package textcomp by Sebastian Rahtz assigns standard names to all text companion symbols. The documentation prints a nice table.

6 Ligatures

In the proportional fonts, the following ligatures are implemented:

--	– (en dash)
---	— (em dash)
‘‘	“ (english opening quotes, german closing quotes)
’’	” (english and polish closing quotes)
,,	„ (german and polish opening quotes)
<<	« (french opening quotes)
>>	» (french closing quotes)
!‘	!‘ (spanish opening exclamation mark)
?‘	?‘ (spanish opening question mark)
fi	fi
ff	ff
fl	fl
ffi	ffi
ffl	ffl

There is another important ligature, not shown above:
⟨hyphen⟩⟨hyphenchar⟩ gives ⟨hyphenchar⟩.
This allows the implementation of new hyphenation patterns with hyphenchar *´127* allowing hyphenation of words containing explicit hyphens. This ligature was missing by accident in the September

1995 release of dc 1.2, but has been added in patchlevel 1 released in December 1995.

7 New Names of the Font Files

Currently, the extended Computer Modern fonts have the prefix dc. This prefix will change to ec with the final release after another round of bug fixing. I hope to make the transition from dc to ec in about one year. The text companion fonts have the prefix tc, which is not subject to change. However, later releases may include more characters and therefore have different checksums. No characters shall be removed from the tc fonts.

Most of the dc fonts can be generated at any size one wants in the range from 5pt to 100pt. For each size, a unique name is needed.

With release 1.2 of the dc fonts, a new, more precise naming scheme is in effect. Since there are widely used operating sytems limiting the file name to 8 characters (plus an extension of 3 characters), the following scheme is used:

- The first two letters (either dc or tc) denote the encoding and the general design of the font.
- The following one or two letters denote the family, shape, and series attributes of the font, e.g., r for roman, bx for bold extended, it for italic, or bi for bold extended italic. A complete overview is given at the end of this section.
- The following four digits give the design size in TEX's points multiplied by 100, e.g., 1000 denotes ten point, 1440 denotes magstep 2, i.e., 14.4 point, and 0500 denotes five point.

Here are the implemented styles:

Roman family: r roman, b bold, bx bold extended, sl slanted, bl bold extended slanted, cc caps and small caps, ti (text) italic, bi bold extended italic, u unslanted italic, ci classical serif italic (new design).

Sans serif family: ss sans serif, si sans serif inclined (slanted), sx sans serif bold extended, so sans serif bold extended oblique (slanted).

Typewriter family: tt typewriter, tc typewriter caps and small caps, st slanted typewriter, it italic typewriter, vt variable width typewriter.

Various other fonts: bm variant bold roman, dh dunhill, fb Fibonacci parameters, ff funny, fi funny italic.

Here are some examples:

dcr1000	European Computer Modern roman at 10pt
tcr1000	Text companion symbols roman at 10pt
dcss1728	European Computer Modern sans serif at 17.28pt
dcbx0900	European Computer Modern roman bold extended at 9pt

Some remaining fonts come at one size only; those are:

dcssdc10	sans serif demi-bold condensed
dcsq8	sans serif quotation
dcqi8	sans serif quotation inclined
dclq8	latex sans serif quotation
dcli8	latex sans serif quotation inclined
idclq8	invisible latex sans serif quotation
idcli8	invisible latex sans serif quotation inclined.

The last four fonts are for the slides document class, which replaces old SLiTEX. They contain a special version of the capital letter 'I'.

8 Upgrading to ec

Here is the following non-official schedule for the upgrade from the dc to ec fonts: One intermediate release (1.3) shall come out in spring 1996; the final release of the ec fonts shall be made in autumn 1996. Afterwards, the fonts will be frozen and only necessary bug fixes will be applied.

A The Cork Encoding

position (octal)	description
Accents for lowercase letters	
000	grave
001	acute
002	circumflex
003	tilde
004	umlaut
005	hungarian
006	ring
007	hachek
010	breve
011	macron
012	dot above
013	cedilla
014	ogonek
Miscellaneous	
015	single base quote
016	single opening guillemet
017	single closing guillemet
020	english opening quotes

021	english closing quotes
022	base quotes
023	opening guillemets
024	closing guillemets
025	en dash
026	em dash
027	compound word mark (invisible)
030	perthousandzero
031	dotless i
032	dotless j
033	ligature ff
034	ligature fi
035	ligature fl
036	ligature ffi
037	ligature ffl
040	visible space

ASCII

041	exclamation mark
042	straight quotes
043	hash mark
044	dollar sign
045	percent sign
046	ampersand
047	apostrophe
050	opening parenthesis
051	closing parenthesis
052	asterisk
053	plus sign
054	comma
055	hyphen (note: not minus sign)
056	full stop
057	solidus
060	digit 0
...	
071	digit 9
072	colon
073	semicolon
074	less than sign
075	equals sign
076	greater than sign
077	question mark
080	commercial at
081	capital letter A
...	
132	capital letter Z
133	opening square bracket
134	backslash
135	closing square bracket
136	ASCII circumflex
137	underscore
140	opening quote (not ASCII grave!)
141	lowercase letter a
...	
172	lowercase letter z
173	opening curly brace
174	vertical bar
175	closing curly brace
176	ASCII tilde
177	hyphenchar (hanging)

Letters for eastern European languages (from Latin-2)

200	capital letter A with breve
201	capital letter A with ogonek
202	capital letter C with acute
203	capital letter C with hachek
204	capital letter D with hachek
205	capital letter E with hachek
206	capital letter E with ogonek
207	capital letter G with breve
210	capital letter L with acute
211	capital letter L with hachek
212	capital letter crossed L
213	capital letter N with acute
214	capital letter N with hachek
215	capital letter Eng
216	capital letter O with hungarian double acute
217	capital letter R with acute
220	capital letter R with hachek
221	capital letter S with acute
222	capital letter S with hachek
223	capital letter S with cedilla
224	capital letter T with hachek
225	capital letter T with cedilla
226	capital letter U with hungarian double acute
227	capital letter U with ring
230	capital letter Y with diaeresis
231	capital letter Z with acute
232	capital letter Z with hachek
233	capital letter Z with dot
234	capital letter IJ
235	capital letter I with dot
236	lowercase letter d with bar
237	section sign
240	lowercase letter a with breve
241	lowercase letter a with ogonek
242	lowercase letter c with acute
243	lowercase letter c with hachek
244	lowercase letter d with hachek
245	lowercase letter e with hachek
246	lowercase letter e with ogonek
247	lowercase letter g with breve
250	lowercase letter l with acute
251	lowercase letter l with hachek
252	lowercase letter crossed l
253	lowercase letter n with acute
254	lowercase letter n with hachek
255	lowercase letter eng
256	lowercase letter o with hungarian double acute
257	lowercase letter r with acute
260	lowercase letter r with hachek
261	lowercase letter s with acute
262	lowercase letter s with hachek
263	lowercase letter s with cedilla
264	lowercase letter t with hachek
265	lowercase letter t with cedilla

266	lowercase letter u with hungarian double acute
267	lowercase letter u with ring
270	lowercase letter y with diaeresis
271	lowercase letter z with acute
272	lowercase letter z with hachek
273	lowercase letter z with dot
274	lowercase letter ij
275	spanish inverted exclamation mark
276	spanish inverted question mark
277	pound sign

Letters for western European languages (from Latin-1)

300	capital letter A with grave
301	capital letter A with acute
302	capital letter A with circumflex
303	capital letter A with tilde
304	capital letter A with diaeresis
305	capital letter A with ring
306	capital letter AE
307	capital letter C with cedilla
310	capital letter E with grave
311	capital letter E with acute
312	capital letter E with circumflex
313	capital letter E with diaeresis
314	capital letter I with grave
315	capital letter I with acute
316	capital letter I with circumflex
317	capital letter I with diaeresis
320	capital letter Edh (D with bar)
321	capital letter N with tilde
322	capital letter O with grave
323	capital letter O with acute
324	capital letter O with circumflex
325	capital letter O with tilde
326	capital letter O with diaeresis
327	capital letter OE
330	capital letter O with slash
331	capital letter U with grave
332	capital letter U with acute
333	capital letter U with circumflex
334	capital letter U with diaeresis
335	capital letter Y with acute
336	capital letter Thorn
337	capital letter Sharp S (deviating from Latin-1)
340	lowercase letter a with grave
341	lowercase letter a with acute
342	lowercase letter a with circumflex
343	lowercase letter a with tilde
344	lowercase letter a with diaeresis
345	lowercase letter a with ring
346	lowercase letter ae
347	lowercase letter c with cedilla
350	lowercase letter e with grave
351	lowercase letter e with acute
352	lowercase letter e with circumflex
353	lowercase letter e with diaeresis
354	lowercase letter i with grave
355	lowercase letter i with acute
356	lowercase letter i with circumflex
357	lowercase letter i with diaeresis
360	lowercase letter edh
361	lowercase letter n with tilde
362	lowercase letter o with grave
363	lowercase letter o with acute
364	lowercase letter o with circumflex
365	lowercase letter o with tilde
366	lowercase letter o with diaeresis
367	lowercase letter oe
370	lowercase letter o with slash
371	lowercase letter u with grave
372	lowercase letter u with acute
373	lowercase letter u with circumflex
374	lowercase letter u with diaeresis
375	lowercase letter y with acute
376	lowercase letter thorn
377	lowercase letter sharp s (deviating from Latin-1)

B The Text Companion Encoding

position description
(octal)

Accents for capital letters

000	grave
001	acute
002	circumflex
003	tilde
004	umlaut
005	hungarian
006	ring
007	hachek
010	breve
011	macron
012	dot above
013	cedilla
014	ogonek

Miscellaneous

015	base single straight quote
022	base double straight quotes
025	twelve u dash
026	three quarters emdash
030	left pointing arrow
031	right pointing arrow
032	tie accent (lowercase)
033	tie accent (capital)
040	blank symbol
044	dollar sign
047	straight quote
052	centered star
057	fraction

Oldstyle digits

060	oldstyle digit 0
061	oldstyle digit 1
062	oldstyle digit 2
063	oldstyle digit 3
064	oldstyle digit 4
065	oldstyle digit 5
066	oldstyle digit 6
067	oldstyle digit 7
070	oldstyle digit 8
071	oldstyle digit 9

Miscellaneous

115	mho sign
117	big circle
127	ohm sign
136	arrow up
137	arrow down
140	backtick (ASCII grave)
142	born
144	died
154	leaf
155	married
156	musical note
176	low tilde
177	short equals

TS1-symbols

200	ASCII-style breve
201	ASCII-style hachek
202	double tick (ASCII double acute)
203	double backtick
204	dagger
205	ddagger
206	double vert
207	perthousand
210	bullet
211	centigrade
212	dollaroldstyle
213	centoldstyle
214	florin
215	colon
216	won
217	naira
220	guarani
221	peso
222	lira
223	recipe
224	interrobang
225	gnaborretni
226	dong sign
227	trademark

Symbols from ISO 8859-1 (Latin-1)

242	cent
243	sterling
244	currency sign
245	yen
246	broken vertical bar
247	section sign
250	high dieresis
251	copyright
252	feminine ordinal indicator
254	logical not
256	circled R
257	macron
260	degree sign
261	plus-minus sign
262	superscript 2
263	superscript 3
264	tick (ASCII-style acute)
265	micro sign
266	pilcrow sign
267	centered dot
271	superscript 1
272	masculine ordinal indicator
274	fraction one quarter
275	fraction one half
276	fraction three quarters
326	multiplication sign (times)
366	division sign

◇ Jörg Knappen
Barbarossaring 43
D-55118 Mainz
Germany
Email: knappen@vkpmzd.kph.uni-mainz.de

Graphics

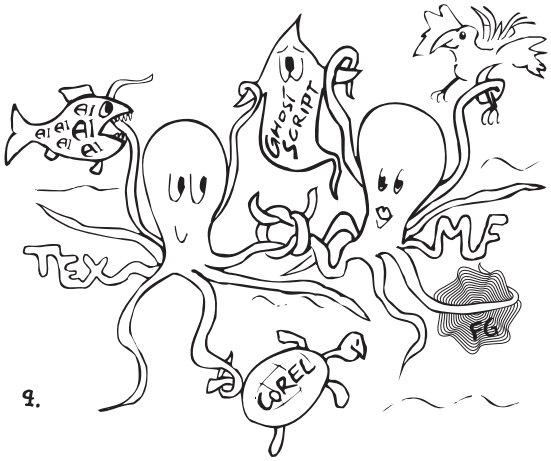
A METAFONT–EPS interface

Bogusław Jackowski

Do not explain too much.
W. Strunk Jr. and E. B. White,
The Elements of Style

Introduction

TEX is not a lion, TEX is an octopus... This sounds like heresy, but it is my deepest conviction that one of the most wonderful features of the TEX/METAFONT system is its openness, i.e., the capability of collaboration with other systems. Hence the association with an octopus:



The paper illustrates this statement by presenting a brief description of an interface for METAFONT-to-PostScript, MFTOEPS. The kernel of the package is a METAFONT program (`MFTOEPS.MF`) which provides necessary definitions for translating the description of graphic objects from METAFONT to PostScript. The PostScript code is written to a log file. It can be extracted from the log file either manually or with the help of additional utilities. There are two programs in the package for performing this task: an AWK program and a TEX program, the latter a bit slower but more universal.

The PostScript files (specifically, Encapsulated PostScript files) produced by MFTOEPS are readable by some popular graphics programs, namely, by Adobe Illustrator (Macintosh and PC compatibles), CorelDraw! (PC compatibles), and Fontographer (Macintosh and PC compatibles). In other words, graphic objects programmed using METAFONT can be further processed by these programs.

It should be stressed that it not the idea of employing METAFONT to produce PostScript code that is important here. A much better tool for this purpose is J.D. Hobby's METAPOST. It is possibile to further process the objects generated by MFTOEPS that makes this package worthy of mention.

Overview of the MFTOEPS package

The `MFTOEPS.MF` program contains the definitions of the following macros which are meant to be used for generating EPS files:

```
eps_mode_setup    fix_line_width
write_preamble    fix_line_join
write_postamble   fix_line_cap
find_BB           fix_miter_limit
set_BB            fix_dash
fill_C            fix_fill_cmyk
draw_C            fix_draw_cmyk
clip_C
```

Obviously, not all possibilities of PostScript are exploited, but the main idea was to provide a *simple tool* for producing output "eatable" by programs which are not PostScript interpreters. Therefore only a small subset of the PostScript language can be taken into account. Nevertheless, these 15 commands are enough to produce an innumerable variety of graphic objects.

METAFONT programs using MFTOEPS have the following structure:

```
1. input mftoeps;
2. eps_mode_setup; % instead of mode_setup
3. ⟨ METAFONT code ⟩
4. find_BB ⟨ list of paths ⟩;
5. write_preamble jobname;
6. ⟨ METAFONT code containing fill_C, draw_C,
     clip_C, etc. ⟩
7. write_postamble;
8. end.
```

The structure seems straightforward, except for some notational details which will be explained in a moment. Perhaps only the fourth line needs a few remarks. A properly formed EPS file should contain the coordinates of the corners of the bounding box in a comment line at the beginning of the file. Macro `write_preamble` needs to know the respective coordinates, as it is responsible for generating the header of an EPS file. Macro `find_BB` simply prepares the data for `write_preamble`.

As you can see, using the plain `beginchar` and `endchar` commands is not essential, although usually it is convenient to make use of them.

Synopsis of the interface of the MFTOEPS package

Conventions: In the following I shall use the words *number*, *pair*, *string*, and *path* as abbreviations for *numeric expression*, *pair expression*, *string expression*, and *path expression*, respectively. The angle brackets, ⟨ and ⟩, used for marking parameters of macros, are "meta-characters," i.e., they do not belong to the METAFONT code.

COMMAND:
`eps_mode_setup`

USAGE:
`eps_mode_setup` ⟨ *an optional number (0 or 1)* ⟩`;`

REMARKS:
This command should be used *instead of* the usual `mode_setup` command. The forms `eps_mode_setup` and `eps_mode_setup 1` are equivalent. One of them (preferably the former) should be used for normal processing, i.e., for generating EPS files. Invoking `eps_mode_setup 0` is meant primarily for testing purposes and is supposed to be used by experienced programmers who know what they are doing. There is a string variable, `extra_eps_setup`, similar to `extra_mode_setup`; at the end of `eps_mode_setup` the command `scantokens extra_eps_setup` is invoked, enabling user-oriented adjustments, e.g., changing the default resolution.

COMMAND:
`write_preamble`

USAGE:
`write_preamble` ⟨ *string* ⟩`;`

REMARKS:
This command initializes the process of writing the PostScript code. The string expression is the name (without extension) of the resulting EPS file; the extension is always `EPS`. METAFONT is switched to `batchmode` in order to avoid slowing down the process by writing mess(ages) to the terminal. Inspection of the log file is thus highly recommended.

COMMAND:
`write_postamble`

USAGE:
`write_postamble;`

REMARKS:
This command ends the writing of the PS code, switches METAFONT back to `errorstopmode`, and performs necessary "last minute" actions (see below).

COMMANDS:
`set_BB find_BB reset_BB`

USAGE:
`set_BB` ⟨ *four numbers or two pairs separated by commas* ⟩`;`
`find_BB` ⟨ *a list of paths separated by commas* ⟩`;`
`reset_BB;`

REMARKS:
The commands `set_BB` or `find_BB` should be invoked prior to invoking `write_preamble`. `set_BB` sets the coordinates of the corners of the bounding box of a graphic object; it is useful when the bounding box of a graphic object is known in advance or if it is required to force an artificial bounding box. `find_BB` computes the respective bounding box for a list of paths; if several `find_BB` statements are used, the common bounding box is calculated for all paths that appear in the arguments. The result is stored in the variables `xl_crd`, `yl_crd`, `xh_crd`, and `yh_crd`. There are two functions, `llxy` and `urxy`, returning pairs `(xl_crd,yl_crd)` and `(xh_crd,yh_crd)`, respectively. The last command, `reset_BB`, makes `xl_crd`, `yl_crd`, `xh_crd`, and `yh_crd` undefined (the initial situation); `reset_BB` is performed by the `write_postamble` macro, which is convenient in the case of generating several EPS files in a single METAFONT run.

COMMANDS:
`fill_C draw_C`

USAGE:
`fill_C` ⟨ *a list of paths separated by commas* ⟩`;`
`draw_C` ⟨ *a list of paths separated by commas* ⟩`;`

REMARKS:
These commands are to be used instead of the usual METAFONT `fill` and `draw` ones. They cause a list of paths followed by the PostScript operation `eofill` (`fill_C`) or `stroke` (`draw_C`) to be translated to a PostScript code. The list of paths constitutes a single curve in the sense of PostScript.

COMMAND:
`clip_C`

USAGE:
`clip_C` ⟨ *a list of paths separated by commas, possibly empty* ⟩`;`

Remarks:
The macro `clip_C` with a non-empty parameter works similarly to the `fill_C` command, except that the `eoclip` operator is issued instead of `eofill`. This causes an appropriate change to the current clipping area. According to PostScript's principles, the resulting area is a set product of the current clipping area and the area specified in the argument of the `eoclip` command. The empty parameter marks the end of the scope of the most recent `clip_C` command with a non-empty parameter. In other words, nested `clip_C` commands form a "stack" structure. If needed, the appropriate number of parameterless `clip_C` commands is issued by the `write_postamble` macro, thus the user does not need to worry about it. Warning*: files produced using* `clip_C` *are interpreted properly by Adobe Illustrator (provided path directions are defined properly) but not by CorelDraw! (ver. 3.0).*

Commands:
`fix_line_width fix_line_join`
`fix_line_cap fix_miter_limit`
`fix_dash`

Usage:
`fix_line_width` ⟨ *a non-negative number (dimension)* ⟩`;`
`fix_line_join` ⟨ *a number (0, 1 or 2)* ⟩`;`
`fix_line_cap` ⟨ *a number (0, 1 or 2)* ⟩`;`
`fix_miter_limit` ⟨ *a number* ≥ 1 ⟩`;`
`fix_dash` (⟨ *a list of numbers (dimensions) separated by commas, possibly empty* ⟩) ⟨ *a number (dimension)* ⟩

Remarks:
These command are to be used in connection with the `draw_C` command. `fix_line_width` fixes the thickness of the outline. The other four commands correspond to PostScript operations `setlinejoin`, `setlinecap`, `setmiterlimit`, and `setdash` (see the *PostScript Language Reference Manual* for details). All commands should be used after `write_preamble`, as `write_preamble` sets the default thickness (0.4pt), default line join (0), default line cap (0), default miter limit (10), and a solid line as a default for stroking (`fix_dash () 0`).

Commands:
`fix_fill_cmyk fix_draw_cmyk`

Usage:
`fix_fill_cmyk` ⟨ *four numbers separated by commas* ⟩`;`
`fix_draw_cmyk` ⟨ *four numbers separated by commas* ⟩`;`

Remarks:
These commands define the colours of the interiors of graphic objects (`fix_fill_cmyk`) and colours of outlines (`fix_draw_cmyk`) using the cyan-magenta-yellow-black model (the basic model of the MFTOEPS package). They should be used after `write_preamble` (because `write_preamble` defines the black colour as a default for both macros) and prior to invoking the corresponding `fill_C` and `draw_C` commands. There are also (just in case) macros `fix_fill_rgb` and `fix_draw_rgb` using red-green-blue model; the argument to both of these macros is a triple of numbers. (The user can control the process of conversion from RGB to CMYK by the redefinition of macros `under_color_removal` and `black_generation`.) The numbers forming the arguments of the macros are supposed to belong to the interval [0...:1].

Besides the fifteen basic macros there are two functions and two control variables that may be of some interest for a virtual user of the MFTOEPS package:

Additional functions:
`pos_turn neg_turn`

Usage:
`pos_turn (`⟨ *path* ⟩`)`
`neg_turn (`⟨ *path* ⟩`)`

Remarks:
Each function returns the path passed as the argument, except that the orientation of the path is changed, if necessary: `pos_turn` returns paths oriented counter-clockwise, `neg_turn` — oriented clockwise. This may be useful for creating pictures which are to be processed further by Adobe Illustrator, because this program is sensitive to the orientation of paths.

Control variable:
`yeseps`

Remarks:
No EPS file will be generated unless the variable `yeseps` is assigned a definite value. It is advisable to set this variable in a command line (see section "Examples").

Control variable:
`testing`

Remarks:
If the variable `testing` is assigned a definite value, the whole PostScript code is flushed to the terminal, thus slowing down significantly the process of generating an EPS file (cf. the description of the `write_preamble` command).

Examples

All sample programs in this section are presented *in extenso*. The reader is not supposed to study the code thoroughly. Nevertheless, I prefer to leave the reader to decide which parts of the code are to be skipped.

Let us start with a trivial example of a "pure" METAFONT program:

```
beginchar(48, % ASCII code
 2cm#, % width
 1cm#, % height
 0cm#  % depth
 );
 fill unitsquare xscaled w yscaled h;
endchar;
end.
```

The program, obviously, generates a font containing one character: a darkened rectangle 2cm×1cm. In order to generate an EPS file containing the same figure, a few modifications are necessary:

```
input mftoeps;
eps_mode_setup;
beginchar(48, % just something
 2cm#, % width
 1cm#, % height
 0cm#  % depth
 );
 set_BB 0,-d,w,h; % coordinates
                  % of the corners
                  % of the bounding box
 write_preamble "rectan";
 fill_C unitsquare xscaled w yscaled h;
 write_postamble;
endchar;
end.
```

Four new commands have appeared: `eps_mode_setup`, `set_BB`, `write_preamble` and `write_postamble`; moreover, `fill` has been replaced by `fill_C`. This is a usual routine for converting an "ordinary" METAFONT program to a form suitable for generating EPS files. Obviously, `draw` should be replaced by `draw_C`, and `filldraw` — with the two operations `fill_C` and `draw_C`. In the latter case the order of the operations `fill_C` and `draw_C` is significant if the drawing and filling colours are different.

Having made this change you can easily generate the respective EPS file, provided you are a DOS user. Assume that the modified program is stored in the file `RECTAN.MF`. In the package MFTOEPS you will find a DOS batch, `M2E.BAT` (subdirectory `PROGS`), which — perhaps after slight adjustments — can be used for this task. It is enough to write

```
m2e rectan
```

(no extension, please) from the command line in order to obtain the required `RECTAN.EPS` file. The batch makes use of AWK for extracting the PostScript code from the log file. There is also an alternative batch, `M2E-ALT.BAT`, that employs TEX for this purpose. In both batches METAFONT is called in the following way:

```
mf386 &plain \yeseps:=1; input %1
```

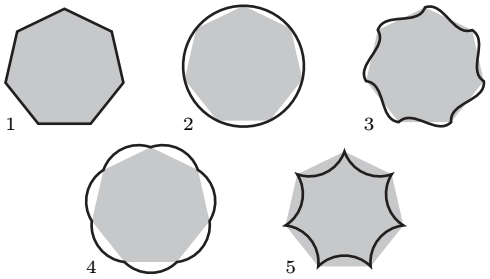
Observe the assignment `yeseps:=1`. In fact, assigning a definite (arbitrary) value to the `yeseps` variable triggers the action of generating an EPS file.

I hope that making scripts for other operating systems is not found to be extremely difficult. I would be very much obliged if others could contribute such scripts to the package.

Let us consider now a more complex example. Suppose that the file `POLYGON.MF` contains the following definitions:

```
vardef regular_polygon(expr n) =
% n is the number of vertices;
% the diameter of the circumscribed
% circle is equal to 1, its centre
% is in the origin
 (up % first vertex
  for i:=1 upto n-1:
   --  % next vertices:
   (up rotated (i*(360/n)))
  endfor
  -- cycle) scaled .5
enddef;
vardef flex_polygon(expr n,a,b) =
% n is the number of vertices,
% a, b are the angles (at vertices)
% between a tangent to a ``flex side''
% and the corresponding secant
 save zz;
 pair zz[ ]; % array of vertices
 for i:=0 upto n-1:
  zz[i]:=up rotated (i*(360/n));
 endfor
 (zz[0] {(zz[1]-zz[0]) rotated a}
  for i:=1 upto n-1:
   .. {(zz[i]-zz[i-1]) rotated b}
   zz[i]
   {(zz[(i+1) mod n]-zz[i]) rotated a}
  endfor
  .. {(zz[0]-zz[n-1]) rotated b} cycle)
 scaled .5
enddef;
```

The first function, `regular_polygon`, returns a closed path that is — as the name suggests — a regular polygon with a given number of vertices. The second function, `flex_polygon`, returns a curve that is in a sense a "generalized polygon" — the following examples show why this epithet is appropriate:



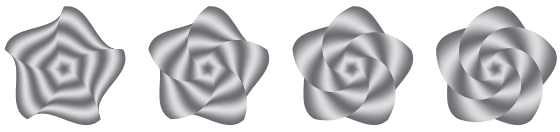
The first picture was generated by the following program:

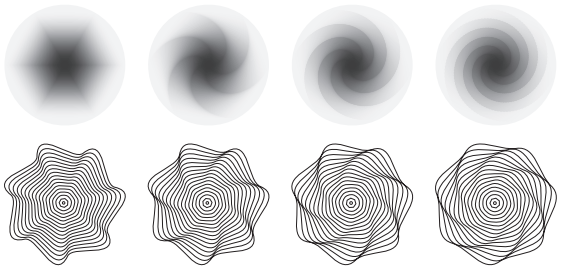
```
1. input polygons;
2. input mftoeps;
3. eps_mode_setup;
4. beginchar(0,16mm#,16mm#,0);
5.  path P[ ]; % ``room'' for two polygons
6. % preparing:
7.  P[1]:=regular_polygon(7)
8.   scaled w shifted (.5w,.5h);
9.  P[2]:=flex_polygon(7,0,0)
10.   scaled w shifted (.5w,.5h);
11. % exporting:
12.  find_BB P[1], P[2];
13.  write_preamble jobname;
14. % 25 percent of black for filling:
15.  fix_fill_cmyk 0,0,0,.25;
16.  fix_line_width 1pt;
17.  fill_C P1; draw_C P2;
18.  write_postamble;
19. endchar;
20. end.
```

The remaining four figures can be obtained by simple modifications of line 9 of the program:

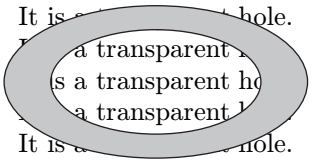
```
P[2]:=flex_polygon(7,-180/7,180/7)  % 2
P[2]:=flex_polygon(7,45,45)         % 3
P[2]:=flex_polygon(7,-45,45)        % 4
P[2]:=flex_polygon(7,45,-45)        % 5
```

These fairly trivial objects can be used to achieve some rather non-trivial effects (METAFONT sources are included in the MFTOEPS package):





So far the examples have contained `fill_C` and `draw_C` commands with arguments being single paths. PostScript, in contrast to METAFONT, accepts groups of paths as a single curve. Therefore the `fill_C` and `draw_C` commands were defined to accept the lists of METAFONT paths as arguments. In the resulting PostScript code they constitute a single object. The main reason is that such objects may contain transparent holes. This enables achieving such effects as:



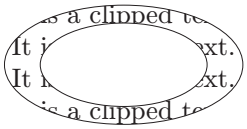
The graphic object was generated by the following simple program:

```
1. input mftoeps; eps_mode_setup;
2. w#=4cm#; h#=2cm#; define_pixels(w,h);
3. set_BB origin, (w,h);
4. write_preamble jobname;
5. % 25 percent of black for filling:
6.  fix_fill_cmyk 0,0,0,.25;
7.  fix_line_width 1pt;
8.  for oper:="draw_C", "fill_C":
9.  scantokens oper
10. % outer edge:
11.   fullcircle
12.    xscaled w yscaled h
13.    shifted (.5w,.5h),
14. % inner edge:
15.   reverse fullcircle
16.   xscaled .7w yscaled .7h
17.   shifted (.5w,.5h);
18.  endfor
19. write_postamble;
20. end.
```

One innocent trick was used in order to shorten the code: the loop in the combination with the `scantokens` command (lines 8 and 9). It is advisable to have paths that form transparent holes appropriately oriented — therefore the operator `reverse` is used in line 15. A TEX code for obtaining the above figure is obvious: it is enough

to put the picture on top of a text box, using, for example, the `\llap` command.

Removing the command `fix_fill_cmyk` (line 6) and replacing the command `fill_C` (line 8) by `clip_C` gives the opportunity to obtain yet another effect:



In this case, however, the TEX code is somewhat complicated, since macros for inclusion of an EPS file (I use Tomas Rokicki's `EPSF.TEX`) embed the code of the EPS file into a PostScript `save`–`restore` group. A clipping path is subjected to such a grouping, contrary to the state of the currently painted picture. Therefore some `\special` hackery is needed (the respective TEX source is included with samples in the MFTOEPS package).

The distinction between single and multiple paths in the context of drawing outlines (`draw_C`) is meaningless.

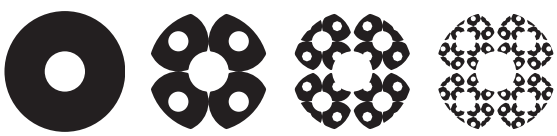
The final example shows how to use clipping to generate a geometric figure known as "Sierpiński's carpet". In order to construct the "carpet" you start with a square with a central hole; this hole is a square with each edge one-third the length of the edge of the original square. Now you divide the original figure into nine squares and replace all filled small squares with a copy of the square with the central hole, scaled down to fit the area of the small square. Then you apply the same procedure to the smaller squares, an so on, *ad infinitum*.

Here you have the program accomplishing this task (infinity "equals" three):

```
1. input mftoeps; eps_mode_setup;
2. % ---
3. def ^ = ** enddef; % syntactic sugar
4. primarydef i // n = % ditto
5.  (if n=0: 0 else: i/n fi)
6. % why not to divide by 0?
7. enddef;
8. def shifted_accordingly(expr i,j,n,D)=
9.  shifted ((i//n)[0,w-D],(j//n)[0,w-D])
10. enddef;
11. % ---
12. w#=16mm#; h#=16mm#; define_pixels(w,h);
13. for N:=1,2,3: % 4, 5, 6, ..., infinity
14.  set_BB 0,0,w,h;
15.  write_preamble jobname & decimal(N);
16.  D:=3w;
17.  for n:=
18.   0 for q:=1 upto N-1: , 3^q-1 endfor:
19. % i.e.:
20. % ``for n:=0, 3^1-1, ..., 3^(N-1)-1:''
21.  path p[], q[]; D:=1/3D; k:=-1;
22.  for i:=0 upto n: for j:=0 upto n:
23.   k:=k+1;
24.   p[k]=unitsquare scaled D
25.    shifted_accordingly(i,j,n,D);
26.   q[k]=reverse unitsquare scaled 1/3D
27.    shifted (1/3D,1/3D)
28.    shifted_accordingly(i,j,n,D);
29.  endfor; endfor;
30.  clip_C p0, q0
31.   for i:=1 upto k:
32.    , p[i], q[i]
33.   endfor;
34. endfor;
35. fill_C unitsquare scaled w;
36. write_postamble;
37. endfor;
38. % ---
37. end.
```

The program is lengthy mainly because of technical details that are not especially interesting; however, there are three points worthy of comment. First, observe that a couple of EPS files are produced in one METAFONT run (the loop in line 13 is relevant here); second, loops are used to form arguments to the loop in line 18 and to the `clip_C` command in line 30 — it is a very useful feature of METAFONT that loops behave exactly like macros; and third, observe that the operation `fill_C` is used only once. The resulting EPS files are shown in the following picture:

You may argue that such a figure can be generated easily in a simpler way, without clipping. True, yet I like this approach — can you imagine a straightforward method for generating a "circular carpet" without clipping? Moreover, one can use



clipping in more complicated situations, not only for filling. But, on the other hand, finding the precise bounding box for a clipped figure becomes a non-trivial task. You must remember, moreover,

that clipping consumes a lot of the resources of a PostScript interpreter, thus it should be used with great care.

Final remarks

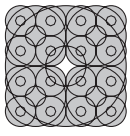
The MFTOEPS package was not devised as a competitor to such giants as Adobe Illustrator or CorelDRAW!. On the contrary, it can be regarded as their little ally. Interactive programs don't cope particularly well with tasks that bear logical structure. In such cases METAFONT — with its wealth of *programmable* path operations, absent "by definition" from the menus of interactive programs — is certainly a preferable tool.

One of the advantages of the applied approach is its portability — the only software needed is METAFONT and either AWK or TEX. Another advantage is its flexibility. It is not particularly difficult to modify the MFTOEPS package to produce another PostScript dialect, if for some reason the dialect of Adobe Illustrator is inconvenient. MFTOEPS can also be modified to produce output in other lingos, e.g., HP-GL (Hewlett-Packard Graphic Language).

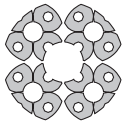
There is still a lot of work to be done. Of course, every program can be improved, but perhaps more important would be preparing a library of METAFONT routines useful for creating objects with a vector representation.

For example, it would be convenient to have a procedure which, for a given set of graphic objects finds a single curve (outline) filling of which would give the same optical result. In other words, such a procedure would perform the task of finding an outline for a set union of graphic objects. Such a procedure is known as *removing overlaps*. The example of the "circular carpet" (see above) illustrates a similar problem: to find an outline for a set intersection of a group of graphic objects.

If the carpet is generated using clipping, the PostScript file contains, in fact, the following elements:



They are partially invisible because of clipping, still they are there. In some contexts, e.g., if the figure is to be cut on a cutting plotter, it is crucial to replace such a multiplicity of objects by a single object:



Note that routines for finding the outline of a set union or a set intersection of a group of graphic objects are not MFTOEPS-oriented. A package providing tools for programming such operations, ROEX, is already available. Perhaps it is most useful in the context of exporting to EPS, however, it can be used with plain METAFONT, and — I guess — with METAPOST as well.

Universal routines of this kind are important from the point of view of the openness of the TEX/METAFONT system, and its openness — as was already mentioned — is one of the most powerful features of the system.

Note also that the openness of a system concerns both *output* and *input*. MFTOEPS accomplishes the first part of the conjunction, but one can think also about an export from PostScript to METAFONT. A package accomplishing this task, `PS_CxONV`, has been recently released as a public domain contribution. Its kernel is a converter, written in PostScript and using the GHOSTSCRIPT interpreter of PostScript, which translates a general PostScript code into a canonical EPS form (there exists a similar program in the standard GHOSTSCRIPT distribution, namely, `PS2AI`, written by Jason Olszewski, but it does not fit this particular problem); the result of such a conversion can be translated to a METAFONT program using the AWK-based utility, EPSTOMF, also recently released into the public domain. This would complete a link between METAFONT and PostScript. I do believe that providing such links is one of the most efficient routes towards a limitless development of the TEX/METAFONT system.

Glossary

AWK: a simple yet powerful batch text processor.

Bounding box: the smallest rectangle surrounding the glyph of a picture; coordinates of its lower left and upper right corners (in big points) should appear in a structural comment in a header of an EPS file.

EPS file: Encapsulated PostScript file; a single-page PostScript document; the purpose of the EPS file is to be included ("encapsulated") as a part of other PostScript programs and to exchange graphic data among applications.

Even-odd rule: a rule that specifies the interior of a (multiple) path in the following way: if

for a given point and for any ray drawn from this point to infinity, the number of intersection points of the ray and the path is odd, the point is inside; if the number is even, the point is outside; command `eofill` and `eoclip` operators follow this rule.

Path orientation: nodes of a closed single path are ordered; if traversing a path following the order of its nodes results in a counter-clockwise turn(s), the path is positively oriented, if it results in a clockwise turn(s), its orientation is negative; the number of turns (signed) is called a turning number (METAFONT) or a winding number (PostScript); the operators `fill` and `clip` make use of a winding number, the operators `eofill` and `eoclip` ignore it.

Availability

The packages MFTOEPS, EPSTOMF and `PS_CONV` can be found at

`ftp.pg.gda.pl`

in the directories

`/pub/TeX/GUST/contrib/MF-PS/MFTOEPS`
`/pub/TeX/GUST/contrib/MF-PS/EPSTOMF`
`/pub/TeX/GUST/contrib/PS/PS_CONV`

References

[1] Adobe Systems Inc. *PostScript Language Reference Manual*. Reading, Mass.: Addison-Wesley, 1991.

[2] Aho, A.V., B.W. Kernighan, P.J. Weinberger. *The* AWK *Programming Language*, Reading, Mass.: Addison-Wesley, 1988.

[3] Jackowski, B., M. Ryćko. "Labyrinth of METAFONT paths in outline." *Proceedings of the Eighth European TEX Conference* (Sept. 26–30, 1994, Gdańsk, Poland), pages 18–32.

[4] Knuth, D.E. *The* METAFONT*book*. Reading, Mass.: Addison-Wesley, 1992. D. E. Knuth: *The* METAFONT*book*, Addison-Wesley, 1992.

◇ Bogusław Jackowski
BOP s.c., Gdańsk, Poland

Technical Working Group Reports

A proposed standard for specials

Tomas G. Rokicki

Introduction

This note presents the current state of the draft standard, as presented by the author and Michael Sofka at the standards session at TUG'95.

1 Identifying syntax

Standard specials shall be syntactically identified by beginning with a colon (':'). All specials beginning with a colon shall follow the guidelines established here.

Any special beginning with a colon, followed by an agreed keyword with agreed semantics, shall be interpreted according to the rules set out in this document and according to the agreed semantics of that keyword.

Any special beginning with two consecutive colons shall be considered an experimental special. It will be interpreted following the syntax and scoping semantics specified in this document, but individual drivers are free to interpret these specials however they wish. This convention allows experimentation with specials in conjunction with the scoping mechanism described here.

2 Syntax

Standard specials shall consist of a sequence of the 95 printable ASCII characters plus the tab character. Tabs will always be interpreted as spaces. Each standard special shall begin with a colon, optionally followed immediately by another colon. Following this shall be a sequence of elements separated by whitespace. Whitespace is also allowed between the colon (or optional colon) and the first element.

Whitespace shall consist of any number of tab or space characters.

The elements shall fall in the following categories: symbol, keyword, key/value pair.

We also define the syntax of numbers, dimensions, and lists, for convenience.

A symbol is any sequence of characters. If the symbol consists of only the 95 printable ASCII characters, and does not contain a double quote, equals sign, space, tab, backslash, or comma, it can be

specified by the sequence of characters that comprise it; when a symbol is so specified, it is called a simple symbol. Otherwise, it can be specified by enclosing the exact characters in a pair of double quotes. Any double quote or backslash within the symbol must be preceded by a backslash. When a symbol is specified through the use of double quotes, it is a quoted symbol.

A keyword is a simple symbol.

A number is a simple symbol that consists of an optional negative sign followed by a sequence of at least one digit, with one optional period (‘`.`’) included in the sequence of digits.

A dimension is a symbol that obeys the rules of a number except that it is followed immediately (with no intervening whitespace) by one of the following pairs of characters: ‘`bp`’, ‘`cm`’, ‘`dd`’, ‘`in`’, ‘`mm`’, ‘`pc`’, ‘`pt`’, ‘`sp`’.

An equals sign shall appear only as part of a quoted symbol, or to separate a keyword and some other symbol forming a key/value pair.

A list is a sequence of symbols separated by commas with no intervening whitespace.

Figure 1 gives a BNF formulation for standard specials. Square brackets enclose optional components. Quotes enclose literal characters. Parentheses group. Angle brackets enclose nonterminals. An asterisk represents zero or more occurrences; a plus sign indicates at least one and possibly more occurrences. Vertical bars indicate choice.

In general, case is significant in specials.

3 Specials scoping and infrastructure

Specials exist in the DVI file as just another sequential command. Yet, often we desire a special to have a scope. One use of specials is to modify the way certain commands in the DVI file are interpreted. For instance, a special might indicate that all subsequent characters until an overriding special shall be rendered in a specific color. Another special might indicate that a certain set of pages is to be printed on different media. Yet a third special might indicate that the background color of the entire document should be mauve.

For this reason, we are introducing standard scoping semantics for standard specials.

3.1 Flat DVI files

Ideally, scoping could always be handled simply by placing an appropriate special at the beginning and end of each scoped region, and no further action would need to be taken. For various reasons, as we shall describe, this is not always possible or convenient. Nonetheless, such a ‘flat’ scoping would serve as an ideal model for the driver writer, and its semantics should form a base on which more complex scoping rules can be built.

In this section, we describe how flat DVI files are interpreted. At the minimum, each driver should be capable of handling flat DVI files.

A flat DVI file is one that can be processed (with respect to specials) with no prescanning or preprocessing. Each special is located syntactically where it belongs semantically. In addition, assuming the beginning of the first page has been scanned for document global specials, each page can be processed and reprocessed independently of any other and in random order, skipping arbitrary sequences of pages. Thus, a flat DVI file is ideal for quick browsing and previewing.

3.1.1 Object specials

The first category of specials, called *object* specials, is those specials that themselves render objects to the page, but do not affect the rendering of other objects. One such example is a special that indicates that a particular graphical figure should be rendered at a particular location on the page.

All object specials shall begin with the keyword ‘`object`’.

Object specials take several implicit parameters that affect how they are rendered. These implicit parameters include the current DVI location on the page, and the DVI magnification.

Unless otherwise specified for a particular object special, all object specials shall be interpreted such that their lower left-hand corner is rendered at the current DVI location. In addition, all object specials shall be scaled by the DVI magnification in effect.

Object specials cannot take the optional scoping keywords described in section 3.2.

The initial syntax for object specials is as follows:

```
<object-specifier> := ':' [':'] <w> 'object' <w>
```

3.1.2 Attribute specials

The second category of specials is attribute specials. These specials affect the way the page is rendered. Normally, an attribute special affects the rendering state for subsequent DVI commands until overridden by another attribute special that affects the same rendering attribute. We shall discuss how our scoping mechanism can change these rules in section 3.2.

All attribute specials begin with the keyword ‘`attribute`’.

```
<standard-special> := ':' [':'] [<whitespace>] (<element> <whitespace>)+
<element>          := <key-value>
                   |  <simple-symbol>
<whitespace>       := (tab | ' ')+
<w>                := <whitespace>
<key-value>        := <simple-symbol> ['=' <symbol-or-list>]
<symbol-or-list>   := <symbol> | <list>
<list>             := <symbol> (',' <symbol>)*
<symbol>           := <simple-symbol>
                   |  <quoted-symbol>
<simple-symbol>    := (<printable-char-except-space,tab,comma,backslash,equals,doublequote>)+
<quoted-symbol>    := '"' (<quoted-char>)* '"'
<quoted-char>      := <space-or-printable-char-except-backslash,doublequote>
                   |  '\' '\' | '\' '"'
<number>           := ['-'] (<digit>)+ [. (<digit>)*]
                   |  ['-'] . (<digit>)+
<dimension>        := <number> <unit>
<unit>             := 'bp' | 'cm' | 'dd' | 'in' | 'mm' | 'pc' | 'pt' | 'sp'
```

Figure 1: A BNF grammar for specials

One interesting case should be mentioned here. In a flat DVI file, it is possible for an attribute special to contain the scoping keyword '`pop`'. If the '`attribute`' keyword in a flat DVI file is followed immediately by the keyword '`pop`', then that corresponding attribute in the rendering state is set (back) to its initial state. (This is necessary because, when flattening scoped specials, the initial state might not be known for all attributes; this provides a convenient way to access that default initial value.)

The initial syntax for attribute specials is as follows:

```
<attribute-specifier> := ':' [':'] <w>
                         'attribute'
                         (<w> <scope>)* <w>
<scope>               := 'push' | 'pop'
                      |  'page' | 'global'
```

3.2 Scoped specials

Sometimes a special must occur at a syntactic location different from where it semantically affects the rendering state. One example of this is where an attribute affecting the background color of the paper is specified as part of the running text of a document, in a document with headers. Normally, this special will follow the headline in the DVI file, because TEX's output routine typically ships the header before the whatsits attached to the body text. So by the time the DVI driver sees the special, it has probably already rendered the header, so it may be difficult or inconvenient to change the background color of the sheet at this point.

Another example is when a colored paragraph is broken across a page boundary, and the DVI driver wishes to render only the second page, without scanning the first page. In the absence of specials, this is easily done. However, if there is only a single special specifying the red color at the beginning of the paragraph (on page one), there is no indication in page two that the color should still be red.

As a final example, consider the use of nested attribute specials. One word in a blue paragraph is to be colored green. The special at the end of the green word indicates that the 'previous' color state should be restored. In this case, the special at the beginning of the paragraph, indicating that the paragraph should be blue, is also semantically visible at the end of the green word.

The scoping rules we introduce in this section introduce a scoping mechanism, and define how specials that use this mechanism, and thus cause a DVI file to be scoped (rather than flat), can be transformed into flat specials for easier rendering.

3.2.1 Stacks

The first mechanism is a convenient 'stack' mechanism that allows the previous state of a particular attribute to be easily restored. The two keywords that indicate this mechanism should be used are '`push`' and '`pop`'.

If the keyword '`attribute`' in a special is followed by the keyword '`push`', then the current state of that attribute is pushed onto a stack internal to the DVI processor. Then, the remainder of the special is used to determine the new value of the attribute.

If the keyword '`attribute`' in a special is followed by the keyword '`pop`', then the previously saved value of the attribute is restored. Any actual attribute value specified in the special is ignored.

If the stack is empty when a '`pop`' special is encountered, then the value of the attribute is set to the default initial value for that attribute.

Attribute specials that use neither push nor pop are still fully legal; they affect the current setting of the attribute, but do not affect the stored stack.

Note that each attribute has its own independent stack. Thus, the following sequence of specials is perfectly legal:

```
:attribute push color red
:attribute push trap true
:attribute pop color
:attribute pop trap
```

DVI drivers should support a stack depth for each attribute of at least twenty levels.

3.2.2 Page

Some specials affect the page, or paper, or media, rather than the rendering of characters or rules. Specials that specify the paper size or background color are examples of these. Such specials should ordinarily occur before any characters or rules or object specials on the page itself; such is the case for flat DVI files.

As discussed before, it is sometimes inconvenient or difficult to ensure that these specials actually occur at the beginning of the page itself. Thus, we define the scoping keyword '`page`' that indicates this special should semantically appear at the beginning of the page. Thus, if a page contains a single (hypothetical) background color attribute, specified as

```
:attribute page backgroundcolor mauve
```

anywhere on the page, then the page should be rendered with a mauve background.

Note that there is no predefined correlation between attributes that are specified page and those that are local. Thus, a special such as

```
:attribute backgroundcolor mauve
```

that occurs between two characters on a page makes little sense. Indeed, where the special obviously affects the entire page, but it is not specified with a page keyword, the operation of the DVI driver shall be undefined.

If multiple page attribute specials for the same attribute appear on a page, they shall all be semantically moved to the top of the page — in the same order as they occur on the page.

3.2.3 Global

Some specials affect the document as a whole, or it is desired that they affect the document as a whole. Specials that define a paper size are one example of these. Generally, such specials should occur before any characters or rules or object specials on the first page.

As discussed before, it is sometimes inconvenient or difficult to ensure that these specials actually occur at the beginning of the page. Thus, we define the scoping keyword '`global`' that indicates this special should semantically appear at the beginning of the entire document. Thus, if a document contains a single (hypothetical) paper size attribute, specified as

```
:attribute global papersize 8.5in 11in
```

anywhere in the document, then the entire document should be rendered on $8.5'' \times 11''$ paper.

For pragmatic reasons (we may not want to, or be able to, prescan the entire document), we require that such global specials occur somewhere on the first page in order to take effect.

Note that there is no predefined correlation between the attributes that are specified global and those that are local or page-specific.

If multiple global attributes for the same attribute appear in a document, all shall be semantically moved to the front of the document — in the same order as they occur in the document.

3.3 Flattening process

The scoping rules are concentrated on the distinction between a special's syntactic and semantic location. The process of flattening a DVI file resolves these differences, by eliminating all stack operations (except for defaulting pops) and by moving all specials to their logical semantic location. This operation converts a scoped DVI file to a flat DVI file.

When flattening occurs, object specials remain in their original location.

For instance, consider a scoped DVI file that appears as follows:

```
Page 1:
   <text>
   :attribute push color red
   <text>
   :attribute global backgroundcolor mauve
Page 2:
   <text>
   :attribute push color green
   <text>
   :attribute color blue
   <text>
   :attribute page papersize 8.5in 11in
   :attribute pop color
   <text>
Page 3:
   <text>
   :attribute pop color
   <text>
```

This would be flattened into the following flat DVI file:

```
Page 1:
   :attribute global backgroundcolor mauve
   <text>
   :attribute color red
   <text>
   :attribute pop color
Page 2:
   :attribute page papersize 8.5in 11in
   :attribute color red
   <text>
   :attribute color green
   <text>
   :attribute color blue
   <text>
   :attribute color red
   <text>
   :attribute pop papersize
   :attribute pop color
Page 3:
   :attribute color red
   <text>
   :attribute pop color
   <text>
```

If a page reversal program that obeys the specials is run, the following DVI file would result. Note that the only difference is that the global specials are moved from the original first page to the new first page.

```
Page 3:
   :attribute global backgroundcolor mauve
   :attribute color red
   <text>
   :attribute pop color
   <text>
Page 2:
   :attribute page papersize 8.5in 11in
   :attribute color red
   <text>
   :attribute color green
   <text>
   :attribute color blue
   <text>
   :attribute color red
   <text>
   :attribute pop papersize
   :attribute pop color
Page 1:
   <text>
   :attribute color red
   <text>
   :attribute pop color
```

This flattening can be performed with a special DVI to DVI processor (which we hope to provide). The use of such a preprocessor will allow fancy scoped specials to be used in an environment that supports only flat specials. For DVI files that are intended to be widely distributed and portable, such a flattening should probably be done.

This flattening can also be performed dynamically, as a DVI file is being created or read from disk, so long as single-page scanning is available. (We hope to provide code for this as well, that will allow easy integration of scoped specials into previewers and printer drivers with a minimum of effort.)

4 Proposed object specials

The primary and most important proposed object special is that for encapsulated PostScript file inclusion.

4.1 EPSF inclusion

The syntax for the encapsulated PostScript inclusion special is as follows:

```
<epsf-special>  := <object-specifier>
                   ('epsf' | 'psfile') '='
                   <symbol>
                   (<w> <epsf-keyword>)* [<w>]

<epsf-keyword> := 'width=' <dimen>
                | 'height=' <dimen>
                | 'scale=' <number>
                     [',' <number>]
                | 'clip=' ('on' | 'off')
                | 'boundingbox='
                     <number> ',' <number> ','
                     <number> ',' <number>
```

In this special, the symbol following the keyword '`epsf`' or '`psfile`' is interpreted as a filename

containing a encapsulated PostScript file for inclusion. This file should follow the Adobe standards for EPSF files; otherwise the effects of this special are undefined.

Positioning occurs as follows. First, a bounding box is determined. If one is specified in the special it is used. Otherwise, if none is specified in the special and the keyword is given as '`epsf`', the EPSF document itself is scanned for a bounding box.

The bounding box values at this point, interpreted in PostScript units, map the region of the illustration that will be included.

The TEX width is set to the horizontal size of the bounding box, and the TEX height is set to the vertical size of the bounding box, both interpreted as 72 units to the inch. Consideration of any width, height, and scale parameters may further affect these values, as described below.

Next, the optional width and height specifiers are considered. If neither is given, this step is omitted. If both are given, their values replace the TEX height and width set before. If only one is given, it replaces the corresponding TEX value, and the other TEX value is set to preserve the aspect ratio.

Next, if the '`scale`' keyword is given, the TEX height and width are further modified. If only one numeric parameter is given, both the width and height are multiplied by this parameter. Otherwise, the width is multiplied by the first parameter and the height is multiplied by the second parameter.

Finally, the width and height values are multiplied by the current DVI magnification in effect.

The resulting values describe the size of a rectangle on the DVI page. The lower left hand corner of this rectangle is positioned at the current DVI location. The geometric mapping from the original bounding box to this rectangle defines the scaling that is performed on the EPSF file when it is rendered.

If the clipping keyword is specified to be '`on`', or if no clipping keyword is specified, the rendering of the EPSF file will be constrained to fall within the DVI rectangle calculated above. Otherwise, no clipping will be performed, and if the EPSF file renders outside its bounding box borders, portions of the image will also be rendered outside the DVI rectangle.

4.1.1 PS *vs.* EPSF

The effects of the keyword '`epsf`' and the keyword '`psfile`' are almost identical — with one minor difference. If '`epsf`' is specified, then the bounding box keyword and values are optional; if they are not specified, the bounding box is read from the EPSF file. If '`psfile`' is specified, the bounding box must be specified; the operation of the special is undefined if no bounding box is specified. (The reasoning behind this seeming inconsistency is partially political and partially religious.)

In order to include a normal EPSF image in its entirety, either the '`psfile`' or the '`epsf`' keyword can be used; if a bounding box value is specified in the special command, the semantics are equivalent.

If only a rectangular subportion of an EPSF image or PS page is to be rendered, then the '`psfile`' keyword should be specified, along with a bounding box describing precisely what portion of the image should be included. Clipping should be turned on to ensure that the rest of the image does not show up outside the DVI rectangle.

4.1.2 Bounding box

The bounding box is specified as a comma-separated list of numbers. These numbers are interpreted as PostScript units. There are 72 PostScript units to the inch. The four numbers are, in sequence, the x-coordinate of the lower left corner, the y-coordinate of the lower left corner, the x-coordinate of the upper right corner, and the y-coordinate of the upper right corner. All four must be specified.

Note that we do not restrict these numbers to be integers. While Adobe requires these to be integers in their Document Structuring Conventions, some applications generate floating point numbers. In addition, the higher precision afforded by floating point might be useful in some circumstances.

4.1.3 Scaling

The scaling parameter consists of one or two numbers separated by a comma. These are used to specify a scaling ratio for an EPSF image. If only one number is specified, the scaling preserves the aspect ratio of the image.

4.1.4 Clipping

Clipping can be set to either '`on`' or '`off`'. The default is '`on`'.

4.1.5 Rotation

Rotation is not currently supported, although it is currently under discussion.

5 Proposed attribute specials

5.1 Color

Under development.

5.2 Background color

Under development.

5.3 Paper size

Paper size and orientation are important characteristics of the document, and should be specified in the document itself rather than on the driver command line. The syntax for the paper size special is:

```
<papersize-special> := <attribute-specifier>
                       'papersize'
                       [<w> <dimension>
                        <w> <dimension>] [<w>]
```

The value (composed of two dimensions) may be omitted only if one of the specified scoping operators is '`pop`'.

The dimensions specify horizontal and then vertical size.

As is conventional, the DVI origin is located one inch down and one inch from the left of the top left corner of the paper.

A typical papersize special is

```
:attribute global papersize 8.5in 11in
```

To specify that landscape letter size is in effect for the current page forward, use

```
:attribute page push papersize 11in 8.5in
```

followed by

```
:attribute page pop papersize
```

on the page you wish to return to portrait.

6 Document history

This section shall record a history of the changes to this document.

22 September 1995: Originated by Tomas Rokicki on the basis of extensive discussions at TUG'94 and TUG'95, discussion on the TWG-DVI mailing list, and discussion at a meeting at MSRI in December of 1994.

10 January 1996: Edited for publication in *TUGboat* by Robin Fairbairns, Barbara Beeton, and Tomas Rokicki.

◇ Tomas G. Rokicki
725B Loma Verde
Palo Alto, CA 94303
USA
Email: `rokicki@cs.stanford.edu`

A Directory Structure for TEX Files (Version 0.999)

TUG Working Group on a TEX Directory Structure (TWG-TDS)

1 Introduction

TEX is a powerful, flexible typesetting system used by thousands of people around the world. It is extremely portable and runs on virtually all operating systems. One unfortunate side effect of TEX's flexibility, however, is that there has been no single "right" way to install it. This has resulted in many sites having different installed arrangements.

The primary purpose of this document is to describe a standard TEX Directory Structure (TDS): a directory hierarchy for macros, fonts, and the other implementation-independent TEX system files. As a matter of practicality, this document also suggests ways to incorporate the rest of the TEX files into a single structure. The TDS has been designed to work on all modern systems. In particular, the Technical Working Group (TWG) believes it is usable under MacOS, MS-DOS, OS/2, Unix, VMS, and Windows NT. We hope that administrators and developers of both free and commercial TEX implementations will adopt this standard.

This document is intended both for the TEX system administrator at a site and for people preparing TEX distributions—everything from a complete runnable system to a single macro or style file. It may also help TEX users find their way around systems organized this way. It is not a tutorial: we necessarily assume knowledge of the many parts of a working TEX system. If you are unfamiliar with any of the programs or file formats we refer to, consult the references in Appendix D.

1.1 The role of the TDS

The role of the TDS is to stabilize the organization of TEX-related software packages that are installed and in use, possibly on multiple platforms simultaneously.

At first glance, it may seem that the Comprehensive TEX Archive Network (CTAN) archives fulfill at least part of this role, but this is not the case. The role of CTAN is to simplify archiving and distribution, not installation and use.

In fact, the roles of the TDS and CTAN are frequently in conflict, as you will see elsewhere in this document. For distribution, many different types of files must be combined into a single unit; for use, it is traditional to segregate files (even similar files) from a single package into separate, occasionally distant, directories.

1.2 Conventions

In this document, "/" is used to separate filename components; for example, `texmf/fonts`. This is the Unix convention but the ideas are in no way Unix-specific.

In this document, "TEX" generally means the TEX system, including METAFONT, `DVI` drivers, utilities, etc., not just the TEX program itself.

The word "package" in this document has its usual meaning: a set of related files distributed, installed, and maintained as a unit. This is *not* a LATEX 2_ε package, which is a style file supplementing a document class.

We use the following typographic conventions:

`literal` Literal text such as `filename` is typeset in typewriter type.

⟨*replaceable*⟩ Replaceable text such as ⟨*package*⟩, identifying a class of things, is typeset in italics inside angle brackets.

2 General

This section describes common properties throughout the TDS tree.

2.1 Subdirectory searching

Many TEX installations store large numbers of related files in single directories, for example, all `TFM` files and/or all TEX input files.

This monolithic arrangement hinders maintenance of a TEX system: it is difficult to determine what files are used by what packages, what files need to be updated when a new version is installed, or what files should be deleted if a package is removed. It is also a source of error if two or more packages happen to have input files with the same name.

Therefore, the TWG felt each package should be in a separate directory. But we recognized that explicitly listing all directories to be searched would be unbearable. A site may wish to install dozens of packages. Aside from anything else, listing that many directories would produce search paths many thousands of characters long, overflowing the available space on some systems.

Also, if all directories are explicitly listed, installing or removing a new package would mean changing a path as well as installing or removing the actual files. This would be a time-consuming and error-prone operation, even with implementations that provide some way to specify the directories to search at runtime. On systems without runtime configuration, it would require recompiling software, an intolerable burden.

As a result, the TWG concluded that a comprehensive TDS requires implementations to support some form of implicit subdirectory searching. More precisely, implementations must make it possible to specify that TEX, METAFONT, and their companion utilities search in both a specified directory and recursively through all subdirectories of that directory when looking for an input file. Other forms of subdirectory searching, for example recursive-to-one-level searches, may also be provided. We encourage implementors to provide subdirectory searching at the option of the installer and user for all paths.

The TDS does not specify a syntax for specifying recursive searching, but we encourage implementors to provide interoperability (see Section B.2).

2.2 Rooting the tree

In this document, we shall designate the root TDS directory by '`texmf`' (for "TEX and METAFONT"). We recommend using that name where possible, but the actual name of the directory is up to the installer. On PC networks, for example, this could map to a logical drive specification such as `T:`.

Similarly, the location of this directory on the system is site-dependent. It may be at the root of the file system; on Unix systems, `/usr/local/share`, `/usr/local`, `/usr/local/lib`, and `/opt` are common choices.

The name `texmf` was chosen for several reasons: it reflects the fact that the directory contains files pertaining to an entire TEX system (including METAFONT, METAPOST, BIBTEX, etc.), not just TEX itself; and it is descriptive of a generic installation rather than a particular implementation.

A site may choose to have more than one TDS hierarchy installed (for example, when installing an upgrade). This is perfectly legitimate.

2.3 Local additions

The TDS cannot specify precisely when a package is or is not a "local addition". Each site must determine this according to their own conventions. At the two extremes, one site might wish to consider "nonlocal" only those files that came with the particular TEX distribution they installed; another site might consider "local" only those files that were actually developed at the local site and not distributed elsewhere.

We recognize two common methods for local additions to a distributed `texmf` tree. Both have their place; in fact, some sites may employ both simultaneously:

1. A completely separate tree which is a TDS structure itself; for example, `/usr/local/umbtex`

at the University of Massachusetts at Boston. This is another example of the multiple `texmf` hierarchies mentioned in the previous section.

2. A directory named '`local`' at any appropriate level, for example, in the ⟨*format*⟩, ⟨*package*⟩, and ⟨*supplier*⟩ directories discussed in the following sections. The TDS reserves the directory name `local` for this purpose.

   We recommend using `local` for site-adapted configuration files, such as `language.dat` for the Babel package or `graphics.cfg` for the graphics package. Unmodified configuration files from a package should remain in the package directory. The intent is to separate locally modified or created files from distribution files, to ease installing new releases.

One common case of local additions is dynamically generated files, e.g., `PK` fonts by the `MakeTeXPK` script originated by *Dvips*. A site may store the generated files directly in any of:

- their standard location in the main TDS tree (if it can be made globally writable);
- an alternative location in the main TDS tree (for example, under `texmf/fonts/tmp`);
- a second complete TDS tree (as outlined above);
- any other convenient directory (perhaps under `/var`, for example `/var/spool/fonts`).

No one solution will be appropriate for all sites.

2.4 Duplicate filenames

Different files by the same name may exist in a TDS tree. The TDS generally leaves unspecified which of two files by the same name in a search path will be found, so generally the only way to reliably find a given file is for it to have a unique name. However, the TDS requires implementations to support the following exceptions:

- Names of TEX input files must be unique within each first-level subdirectory of `texmf/tex` and `texmf/tex/generic`, but not within all of `texmf/tex`; i.e., different TEX formats may have files by the same name. (Section 3.1 discusses this further.) Thus, no single format-independent path specification, such as a recursive search beginning at `texmf/tex` specifying no other directories, suffices. So implementations must provide format-dependent path specifications, for example via wrapper scripts or configuration files.
- Many font files will have the same name (e.g., `cmr10.pk`), as discussed in Section 3.2.2. Implementations must distinguish these files by mode and resolution.

All implementations we know of already have these capabilities.

One place where duplicate names are likely to occur is not an exception:

- Names of METAFONT input files (as opposed to bitmap fonts) must be unique within all of `texmf/fonts`. In practice, this is a problem with those variants of Computer Modern which contain slightly modified files named `punct.mf`, `romanl.mf`, and so on. We believe the only feasible solution here is simply to rename the derivative files to be unique.

3 Top-level directories

The directories under the `texmf` root identify the major components of a TEX system (see Table 1 for a summary). A site may omit any unneeded directories.

Although the TDS by its nature can specify precise locations only for implementation-independent files, we recognize that installers may well wish to place other files under `texmf` to simplify administration of the TEX tree, especially if it is maintained by someone other than the system administrator. Therefore, additional top-level directories may be present.

The top-level directories specified by the TDS are:

`tex` for TEX files (Section 3.1).

`fonts` for font-related files (Section 3.2).

`metafont` for METAFONT files which are not fonts (Section 3.3).

`metapost` for METAPOST files (Section 3.4).

`bibtex` for BIBTEX files (Section 3.5).

`doc` for user documentation (Section 3.6).

`source` for sources. This includes both traditional program sources (for example, *Web2c* sources go in `texmf/source/web2c`) and LATEX `dtx` sources (which go in `texmf/source/latex`).

`source` is intended for files which are not needed at runtime by any TEX program; it should not be included in any search path. For example, `plain.tex` does not belong under `texmf/source`, even though it is a "source file" in the sense of not being derived from another file, but rather in `texmf/tex/plain/base`, as explained in Section 3.1.

⟨*implementation*⟩ for implementations (examples: `emtex`, `web2c`), to be used for whatever purpose deemed suitable by the implementor or TEX administrator. Files that cannot be shared between implementations, such as pool files (`tex.`

`pool`) and memory dump files (`plain.fmt`) go here, in addition to implementation-wide configuration files. See Section B.3 for examples of real ⟨*implementation*⟩ trees.

⟨*program*⟩ for individual configuration files and program-specific input files for TEX-related programs (examples: `mft`, `dvips`). In fact, the `tex`, `metafont`, `metapost`, and `bibtex` directories above may be seen as instances of this case.

3.1 Macros

TEX macro files shall be stored in separate directories, segregated by TEX format and package name (we use 'format' in its traditional TEX sense to mean a usefully `\dump`-able package):

`texmf/tex/`⟨*format*⟩`/`⟨*package*⟩`/`

⟨*format*⟩ is a format name (examples: `amstex`, `latex`, `plain`, `texinfo`).

The TDS allows distributions that can be used as either formats or packages (e.g., Texinfo, Eplain) to be stored at either level, at the option of the format author or TEX administrator. We recommend that packages used as formats at a particular site be stored at the ⟨*format*⟩ level: by adjusting the TEX inputs search path, it will be straightforward to use them as macro packages under another format, whereas placing them in another tree completely obscures their use as a format.

The TDS reserves the following ⟨*format*⟩ names:

- `generic`, for input files that are useful across a wide range of formats (examples: `null.tex`, `path.sty`). Generally, this means any format that uses the category codes of Plain TEX and does not rely on any particular format. This is in contrast to those files which are useful only with Plain TEX (which go under `texmf/tex/plain`), e.g., `testfont.tex` and `plain.tex` itself.
- `local`, for local additions. See Section 2.3.

Thus, for almost every format, it is necessary to search at least the ⟨*format*⟩ directory and then the `generic` directory (in that order). Other directories may need to be searched as well, depending on the format. When using $\mathcal{AMS}$-TEX, for example, the `amstex`, `plain`, and `generic` directories should be searched, because $\mathcal{AMS}$-TEX is compatible with Plain.

⟨*package*⟩ is a TEX package name (examples: `babel`, `texdraw`).

In the case where a format consists of only a single file and has no auxiliary packages, that file can simply be placed in the ⟨*format*⟩ directory, instead of ⟨*format*⟩`/base`. For example, Texinfo goes in `texmf/tex/texinfo/texinfo.tex`, not `texmf/tex/texinfo/base/texinfo.tex`.

The TDS reserves the following ⟨*package*⟩ names:

- `base`, for the base distribution of each format, including files used by `INITEX` when dumping format files. For example, in the standard LATEX distribution, the `ltx` files created during the build process shall be stored in the `base` directory.
- `hyphen`, for hyphenation patterns, including the original American English `hyphen.tex`. These are typically used only by `INITEX`. In most situations, this directory need exist only under the `generic` format.
- `images`, for image input files, such as Encapsulated PostScript figures. Although it is somewhat non-intuitive for these to be under a directory named "`tex`", TEX needs to read these files to glean bounding box or other information. A mechanism for sharing image inputs between TEX and other typesetting programs (e.g., Interleaf, FrameMaker) is beyond the scope of the TDS. In most situations, this directory need exist only under the `generic` format.
- `local`, for local additions and configuration files. See Section 2.3.
- `misc`, for packages that consist of a single file. An administrator or package maintainer may create directories for single-file packages at their discretion, instead of using `misc`.

3.2 Fonts

Font files shall be stored in separate directories, segregated by file type, font supplier, and typeface (`PK` and `GF` files need additional structure, as detailed in the next section):

`texmf/fonts/`⟨*type*⟩`/`⟨*supplier*⟩`/`⟨*typeface*⟩`/`

⟨*type*⟩ is the type of font file. The TDS reserves the following ⟨*type*⟩ names:

- `afm`, for Adobe font metrics.
- `gf`, for generic font bitmap files.
- `pk`, for packed bitmap files.
- `source`, for font sources (METAFONT files, property lists, etc.).

- `tfm`, for TEX font metric files.
- `type1`, for Type 1 fonts (in any format).
- `vf`, for virtual fonts.

As usual, a site may omit any of these directories that are unnecessary (`gf` is a particularly likely candidate for omission).

⟨*supplier*⟩ is a name identifying font source (examples: `adobe`, `ams`, `public`). The TDS reserves the following ⟨*supplier*⟩ names:

- `ams`, for the American Mathematical Society's $\mathcal{AMS}$-fonts collection.
- `local`, for local additions. See Section 2.3.
- `public`, for freely redistributable fonts where the supplier neither (1) requested their own directory (e.g., `ams`), nor (2) also made proprietary fonts (e.g., `adobe`). It does not contain all extant freely distributable fonts, nor are all files therein necessarily strictly public domain.
- `tmp`, for dynamically-generated fonts, as is traditional on some systems. It may be omitted if unnecessary, as usual.

⟨*typeface*⟩ is the name of a typeface family (examples: `cm`, `euler`, `times`). The TDS reserves the following ⟨*typeface*⟩ names:

- `cm` (within `public`), for the 75 fonts defined in *Computers and Typesetting, Volume E*.
- `latex` (within `public`), for those fonts distributed with LATEX in the base distribution.
- `local`, for local additions. See Section 2.3.

Some concrete examples:

```
texmf/fonts/source/public/pandora/pnr10.mf
texmf/fonts/tfm/public/cm/cmr10.tfm
texmf/fonts/type1/adobe/utopia/putr.pfa
```

For complete supplier and typeface name lists, consult *Filenames for TEX fonts* (see Appendix D).

3.2.1 Font bitmaps

Font bitmap files require two characteristics in addition to the above to be uniquely identifiable: (1) the type of device (i.e., mode) for which the font was created; (2) the resolution of the bitmap.

Following common practice, the TDS segregates fonts with different device types into separate directories. See `modes.mf` in Appendix D for recommended mode names.

Some printers operate at more than one resolution (e.g., at 300 dpi and 600 dpi), but each such resolution will necessarily have a different mode name. Nothing further is needed, since implicit in the TEX system is the assumption of a single target resolution.

Two naming strategies are commonly used to identify the resolution of bitmap font files. On systems that allow long filenames (and in the original METAFONT program itself), the resolution is included in the filename (e.g., `cmr10.300pk`). On systems which do not support long filenames, fonts are generally segregated into directories by resolution (e.g., `dpi300/cmr10.pk`).

Because the TDS cannot require long filenames, we must use the latter scheme for naming fonts. So we have two more subdirectory levels under `pk` and `gf`:

`texmf/fonts/pk/`
⟨*mode*⟩`/`⟨*supplier*⟩`/`⟨*typeface*⟩`/dpi`⟨*nnn*⟩`/`
`texmf/fonts/gf/`
⟨*mode*⟩`/`⟨*supplier*⟩`/`⟨*typeface*⟩`/dpi`⟨*nnn*⟩`/`

⟨*mode*⟩ is a name which identifies the device type (examples: `cx`, `gsftopk`, `ljfour`). Usually, this is the name of the METAFONT mode used to build the `PK` file. For fonts rendered as bitmaps by a program that does not distinguish between different output devices, the ⟨*mode*⟩ name shall be that of the program (e.g., `ps2pk`, `gsftopk`).

`dpi`⟨*nnn*⟩ specifies the resolution of the font (examples: `dpi300`, `dpi329`). '`dpi`' stands for dots per inch, i.e., pixels per inch. We recognize that pixels per millimeter is used in many parts of the world, but dpi is too traditional in the TEX world to consider changing now.

The integer ⟨*nnn*⟩ is to be calculated as if using METAFONT arithmetic and then rounded; i.e., it is the integer METAFONT uses in its output `gf` filename. We recognize small differences in the resolution are a common cause of frustration among users, however, and recommend implementors follow the level 0 DVI driver standard (see Appendix D) in bitmap font searches by allowing a fuzz of ±0.2% (with a minimum of 1) in the ⟨*dpi*⟩.

Implementations may provide extensions to the basic naming scheme, such as long filenames and font library files, provided that the basic scheme is also supported.

3.2.2 Valid font bitmaps

The TWG recognizes that the use of short filenames has many disadvantages. The most vexing is that it results in the creation of dozens of different files with the same name. At a typical site, `cmr10.pk` will be the filename for Computer Modern Roman 10 pt at 5–10 magnifications for 2–3 modes. (Section 2.4 discusses duplicate filenames in general.)

To minimize this problem, we strongly recommend that PK files contain enough information to identify precisely how they were created: at least the mode, base resolution, and magnification used to create the font.

This information is easy to supply: a simple addition to the local modes used for building the fonts with METAFONT will automatically provide the required information. If you have been using a local modes file derived from (or that is simply) `modes.mf` (see Appendix D), the required information is already in your PK files. If not, a simple addition based on the code found in `modes.mf` can be made to your local modes file and the PK files rebuilt.

3.3 Non-font METAFONT files

Most METAFONT input files are font programs or parts of font programs and are thus covered by the previous section. However, a few non-font input files do exist. Such files shall be stored in:

`texmf/metafont/`⟨*package*⟩`/`

⟨*package*⟩ is the name of a METAFONT package (for example, `mfpic`).

The TDS reserves the following ⟨*package*⟩ names:

- `base`, for the standard METAFONT macro files as described in *The METAFONTbook*, such as `plain.mf` and `expr.mf`.
- `local`, for local additions. See Section 2.3.
- `misc`, for METAFONT packages consisting of only a single file (for example, `modes.mf`).

3.4 METAPOST

METAPOST is a picture-drawing language developed by John Hobby, derived from Knuth's METAFONT. Its primary purpose is to output Encapsulated PostScript instead of bitmaps.

METAPOST input files and the support files for METAPOST-related utilities shall be stored in:

`texmf/metapost/`⟨*package*⟩`/`

⟨*package*⟩ is the name of a METAPOST package. At the present writing none exist, but the TWG thought it prudent to leave room for contributed packages that might be written in the future.

The TDS reserves the following ⟨*package*⟩ names:

- `base`, for the standard METAPOST macro files, such as `plain.mp`, `mfplain.mp`, `boxes.mp`, and `graph.mp`. This includes files used by `INIMP` when dumping mem files containing preloaded macro definitions.
- `local`, for local additions. See Section 2.3.
- `misc`, for METAPOST packages consisting of only a single file.
- `support`, for additional input files required by METAPOST utility programs, including a font map, a character adjustment table, and a subdirectory containing low-level METAPOST programs for rendering some special characters.

3.5 BibTeX

BibTeX-related files shall be stored in:

`texmf/bibtex/bib/`⟨*package*⟩`/`
`texmf/bibtex/bst/`⟨*package*⟩`/`

The `bib` directory is for BibTeX database (`.bib`) files, the `bst` directory for style (`.bst`) files.

⟨*package*⟩ is the name of a BibTeX package. The TDS reserves the following ⟨*package*⟩ names (the same names are reserved under both `bib` and `bst`):

- `base`, for the standard BibTeX databases and styles, such as `xampl.bib`, `plain.bst`.
- `local`, for local additions. See Section 2.3.
- `misc`, for BibTeX packages consisting of only a single file.

3.6 Documentation

Most packages come with some form of documentation: user manuals, example files, programming guides, etc. In addition, many independent files not part of any macro or other package describe various aspects of the TeX system.

The TDS specifies that these additional documentation files shall be stored in a structure that parallels to some extent the `fonts` and `tex` directories, as follows:

`texmf/doc/`⟨*category*⟩`/...`

⟨*category*⟩ identifies the general topic of documentation that resides below it; for example, a TeX format name (`latex`), program name (`bibtex`, `tex`), or other system components (`web`, `fonts`).

The TDS reserves the following categories:

- Within each ⟨*category*⟩ tree for a TeX format, the directory `base` is reserved for base documentation distributed by the format's maintainers.
- `general`, for standalone documents not specific to any particular program (for example, Joachim Schrod's *Components of TeX*).
- `help`, for meta-information, such as FAQ's, David Jones' macro index, etc.
- `html`, for HTML documents.
- `info`, for processed Texinfo documents. (Info files, like anything else, may also be stored outside the TDS, at the installer's option.)
- `local`, for local additions. See Section 2.3.

`bibtex/`	BibTeX input files
`bib/`	BibTeX databases
`base/`	base distribution (e.g., `xampl.bib`)
`misc/`	single-file databases
⟨*package*⟩`/`	name of a package
`bst/`	BibTeX style files
`base/`	base distribution (e.g., `plain.bst`, `acm.bst`)
`misc/`	single-file styles
⟨*package*⟩`/`	name of a package
`doc/`	see Section 3.6 and the summary below
`fonts/`	font-related files
⟨*type*⟩`/`	file type (e.g., `pk`)
⟨*mode*⟩`/`	type of output device (for `pk` and `gf` only)
⟨*supplier*⟩`/`	name of a font supplier (e.g., `public`)
⟨*typeface*⟩`/`	name of a typeface (e.g., `cm`)
`dpi`⟨*nnn*⟩`/`	font resolution (for `pk` and `gf` only)
⟨*implementation*⟩`/`	TeX implementations, by name (e.g., `emtex`)
`metafont/`	METAFONT (non-font) input files
`base/`	base distribution (e.g., `plain.mf`)
`misc/`	single-file packages (e.g., `modes.mf`)
⟨*package*⟩`/`	name of a package (e.g., `mfpic`)
`metapost/`	METAPOST input and support files
`base/`	base distribution (e.g., `plain.mp`)
`misc/`	single-file packages
⟨*package*⟩`/`	name of a package
`support/`	support files for METAPOST-related utilities
`mft/`	`MFT` inputs (e.g., `plain.mft`)
⟨*program*⟩`/`	TeX-related programs, by name (e.g., `dvips`)
`source/`	program source code by name (e.g., `latex`, `web2c`)
`tex/`	TeX input files
⟨*format*⟩`/`	name of a format (e.g., `plain`)
`base/`	base distribution for format (e.g., `plain.tex`)
`misc/`	single-file packages (e.g., `webmac.tex`)
`local/`	local additions to or local configuration files for ⟨*format*⟩
⟨*package*⟩`/`	name of a package (e.g., `graphics`, `mfnfss`)
`generic/`	format-independent packages
`hyphen/`	hyphenation patterns (e.g., `hyphen.tex`)
`images/`	image input files (e.g., Encapsulated PostScript)
`misc/`	single-file format-independent packages (e.g., `null.tex`).
⟨*package*⟩`/`	name of a package (e.g., `babel`)

Table 1: A skeleton of a TDS `texmf` directory tree

The `doc` directory is intended for implementation-independent and operating system-independent documentation files. Implementation-dependent files shall be stored elsewhere, as provided for by the implementation and/or TeX administrator (for example, VMS help files under `texmf/vms/help`).

The documentation directories may contain TeX sources, `DVI` files, PostScript files, text files, example input files, or any other useful documentation format(s). See Table 2 for a summary.

A Unspecified pieces

The TDS cannot address the following aspects of a functioning TeX system:

1. The location of executable programs: this is too site-dependent even to recommend a location, let alone require one. A site may place executables outside the `texmf` tree altogether (e.g., `/usr/local/bin`), in a platform-dependent directory within `texmf`, or elsewhere.

ams/	
amsfonts/	amsfonts.faq, amfndoc
amslatex/	amslatex.faq, amsldoc
amstex/	amsguide, joyerr
bibtex/	BibTeX
base/	btxdoc.tex
fonts/	
fontname/	*Filenames for TeX fonts*
oldgerm/	corkpapr
⟨*format*⟩/	name of a TeX format (e.g., generic, latex)
base/	for the base distribution
misc/	for contributed single-file package documentation
⟨*package*⟩/	for *package*
general/	across programs, generalities
errata/	errata, errata[1-8]
texcomp/	*Components of TeX*
generic/	for non-format-specific TeX packages
babel/	
german/	germdoc
help/	meta-information
ctan/	info about CTAN mirror sites
faq/	FAQs of comp.text.tex, etc.
html/	HTML files
info/	GNU Info files, made from Texinfo sources
latex/	example of ⟨*format*⟩
base/	ltnews*, *guide, etc.
graphics/	grfguide
⟨*program*⟩/	TeX-related programs, by name (examples follow)
metafont/	mfbook.tex, metafont-for-beginners, etc.
metapost/	mpman, manfig, etc.
tex/	texbook.tex, *A Gentle Introduction to TeX*, etc.
web/	webman, cwebman

Table 2: A skeleton of a TDS directory tree under texmf/doc

2. Upgrading packages when new releases are made: we could find no way of introducing version specifiers into texmf that would do more good than harm, or that would be practical for even a plurality of installations.
3. The location of implementation-specific files (e.g., TeX .fmt files): by their nature, these must be left to the implementor or TeX maintainer. See Section B.3.
4. Precisely when a package or file should be considered "local", and where such local files are installed. See Section 2.3 for more discussion.

A.1 Portable filenames

The TDS cannot require any particular restriction on filenames in the tree, since the names of many existing TeX files conform to no particular scheme. For the benefit of people who wish to make a portable TeX distribution or installation, however, we outline here the necessary restrictions. The TDS specifications themselves are compatible with these.

ISO-9660 is the only universally acceptable file system format for CD-ROMs. A subset thereof meets the stringent limitations of all operating systems in use today. It specifies the following:

- File and directory names, not including any directory path or extension part, may not exceed eight characters.
- Filenames may have a single extension. Extensions may not exceed three characters. Directory names may not have an extension.
- Names and extensions may consist of *only* the characters A–Z, 0–9, and underscore. Lowercase letters are excluded. (The only common place where mixed-case names occur in the TeX system is in LaTeX font descriptor files, and LaTeX

does not rely on case alone to distinguish among these files.)

- A period separates the filename from the extension and is always present, even if the name or extension is missing (e.g., `FILENAME.` or `.EXT`).
- A version number, ranging from 1–32767, is appended to the file extension, separated by a semicolon (e.g., `FILENAME.EXT;1`).
- Only eight directory levels are allowed, including the top-level (mounted) directory (see Section 2.2). Thus, the deepest valid ISO-9660 path is:

```
texmf/L2/L3/L4/L5/L6/L7/L8/FOO.BAR;1
1     2  3  4  5  6  7  8
```

The deepest TDS path needs only seven levels:

```
texmf/fonts/pk/cx/public/cm/dpi300/cmr10.pk
1     2     3  4  5      6  7
```

Some systems display a modified format of ISO-9660 names, mapping alphabetic characters to lowercase, removing version numbers and trailing periods, etc.

B Implementation issues

We believe that the TDS can bring a great deal of order to the current anarchic state of many TEX installations. In addition, by providing a common frame of reference, it will ease the burden of documenting administrative tasks. Finally, it is a necessary part of any reasonable system of true "drop-in" distribution packages for TEX.

B.1 Adoption of the TDS

We recognize that adoption of TDS will not be immediate or universal. Most TEX administrators will not be inclined to make the switch until:

- Clear and demonstrable benefits can be shown for the TDS.
- TDS-compliant versions of all key programs are available in ported, well-tested forms.
- A "settling" period has taken place, to flush out problems. The public release of this document is the first step in this process.

Consequently, most of the first trials of the TDS will be made by members of the TDS committee and/or developers of TEX-related software. Indeed, some of this has taken place during the course of our deliberations (see Appendix D for a sample tree available electronically). They will certainly result in the production of a substantial number of TDS-compliant packages.

Once installable forms of key TDS-compliant packages are more widespread, some TEX administrators will set up TDS-compliant trees, possibly in parallel to existing production directories. This testing will likely flush out problems that were not obvious in the confined settings of the developers' sites; for example, it should help to resolve OS and package dependencies, package interdependencies, and other details not addressed by this TDS version.

After most of the dust has settled, hopefully even conservative TEX administrators will begin to adopt the TDS. Eventually, most TEX sites will have adopted the common structure, and most packages will be readily available in TDS-compliant form.

We believe that this process will occur relatively quickly. The TDS committee spans a wide range of interests in the TEX community. Consequently, we believe that most of the key issues involved in defining a workable TDS definition have been covered, often in detail. TEX developers have been consulted about implementation issues, and have been trying out the TDS arrangement. Thus, we hope for few surprises as implementations mature.

Finally, there are several (current or prospective) publishers of TEX CD-ROMs. These publishers are highly motivated to work out details of TDS implementation, and their products will provide inexpensive and convenient ways for experimentally-minded TEX administrators to experiment with the TDS.

Efforts are under way to set up a "TDS Registry" that will coordinate assignment of TDS-compliant directory names and provide a definitive database of TDS-compliant software distributions. (Perhaps this could also serve many sites as the definition of when a package is local.) For now, distribution through CTAN serves as an imprecise registry.

B.2 More on subdirectory searching

Recursive subdirectory searching is the ability to specify a search not only of a specified directory $\langle d \rangle$, but recursively of all directories below $\langle d \rangle$.

Since the TDS specifies precise locations for most files, with no extra levels of subdirectories allowed, true recursive searching is not actually required for a TDS-compliant implementation. We do, however, strongly recommend recursive searching as the most user-friendly and natural approach to the problem, rather than convoluted methods to specify paths without recursion.

This feature is already supported by many implementations of TEX and companion utilities, for example DECUS TEX for VMS, *Dvips(k)*, *emTEX* (and its drivers), *PubliC TEX*, *Web2c*, *Xdvi(k)*, and *Y&YTEX*.

Even if your TEX implementation does not directly support subdirectory searching, you may find it useful to adopt the structure if you do not use many fonts or packages. For instance, if you only use Computer Modern and AMS fonts, it would be feasible to store them in the TDS layout and list the directories individually in configuration files or environment variables.

The TWG recognizes that subdirectory searching places an extra burden on the system and may be the source of performance bottlenecks, particularly on slower machines. Nevertheless, we feel that subdirectory searching is imperative for a well-organized TDS, for the reasons stated in Section 2.1. Implementors are encouraged to provide enhancements to the basic principle of subdirectory searching to avoid performance problems, e.g., the use of a filename cache (this can be as simple as a recursive directory listing) that is consulted before disk searching begins. If a match is found in the database, subdirectory searching is not required, and performance is thus independent of the number of subdirectories present on the system.

Different implementations specify subdirectory searching differently. In the interest of typographic clarity, the examples here do not use the ⟨*replaceable*⟩ font.

Dvips: via a separate `TEXFONTS_SUBDIR` environment variable.

emTEX: `t:\subdir!!`; `t:\subdir!` for a single level of searching.

Kpathsea: `texmf/subdir//`

VMS: `texmf:[subdir...]`

Xdvi: `texmf/subdir/**`; `texmf/subdir/*` for a single level of searching (patchlevel 20 of the program onwards).

Y&Y TEX: `t:/subdir//` or `t:\subdir\\`.

B.3 Example implementation-specific trees

The TDS cannot specify a precise location for implementation-specific files, but for informative purposes, we provide here the default locations for some implementations. Please contact us with additions or corrections. These paths are not definitive, may not match anything at your site, and may change without warning.

We recommend all implementations have default search paths that start with the current directory (e.g., '`.`'). Allowing users to include the parent directory (e.g., '`..`') is also helpful.

B.3.1 Public DECUS TEX

If another VMS implementation besides Public DECUS TEX appears, the top level implementation directory name will be modified to something more specific (e.g., `vms_decus`). Table 3 shows the VMS directory structure.

B.3.2 Web2c 7.0

All implementation-dependent TEX system files (including `.pool`, `.fmt`, `.base`, and `.mem` files) are stored by default directly in `texmf/web2c`. The configuration file `texmf.cnf` and various subsidiary `MakeTeX...` scripts used as subroutines are also stored there.

Non-TEX specific files are stored following the GNU coding standards. Given a root directory ⟨*prefix*⟩ (`/usr/local` by default), we have default locations as follows as shown in Table 4.

See `prep.ai.mit.edu:/pub/gnu/standards.*` for the rationale behind and descriptions of this arrangement. A site may of course override these defaults; for example, it may put everything under a single directory such as `/usr/local/texmf`.

C Is there a better way?

Defining the TDS required many compromises. Both the overall structure and the details of the individual directories were arrived at by finding common ground among many opinions. The driving forces were feasibility (in terms of what could technically be done and what could reasonably be expected from developers) and regularity (files grouped together in an arrangement that "made sense").

Some interesting ideas could not be applied due to implementations lacking the necessary support:

- Path searching control at the TEX level. If documents could restrict subdirectory searching to a subdirectory via some portable syntax in file names, restrictions on uniqueness of filenames could be relaxed considerably (with the cooperation of the formats), and the TEX search path would not need to depend on the format.

- Multiple logical `texmf` trees. For example, a site might have one (read-only) location for stable files, and a different (writable) location for dynamically-created fonts or other files. It would be reasonable for two such trees to be logically merged when searching.

C.1 Macro structure

The TWG settled on the ⟨*format*⟩/⟨*package*⟩ arrangement after long discussion about how best to arrange the files.

`texmf/`	
`vms/`	VMS implementation specific files
`exe/`	end-user commands
`common/`	command procedures, command definition files, etc.
`axp/`	binary executables for Alpha AXP
`vax/`	binary executables for VAX
`formats/`	pool files, formats, bases
`help/`	VMS help library, and miscellaneous help sources
`mgr/`	command procedures, programs, docs, etc., for system management

Table 3: The VMS directory structure

⟨*prefix*⟩`/`	installation root (`/usr/local` by default)
`bin/`	executables
`man/`	man pages
`info/`	info files
`lib/`	libraries (`libkpathsea.*`)
`share/`	
`texmf/`	TDS root
`web2c/`	implementation-dependent files (`.pool`, `.fmt`, `texmf.cnf`, etc.)

Table 4: The VMS directory structure

The primary alternative to this arrangement was a scheme which reversed the order of these directories: ⟨*package*⟩`/`⟨*format*⟩. This reversed arrangement has a strong appeal: it keeps all of the files related to a particular package in a single place. The arrangement actually adopted tends to spread files out into two or three places (macros, documentation, and fonts, for example, are spread into different sections of the tree right at the top level).

Nevertheless, the ⟨*format*⟩`/`⟨*package*⟩ structure won for a couple of reasons:

- It is closer to current practice; in fact, several members of the TWG have already implemented the TDS hierarchy. The alternative is not in use at any known site, and the TWG felt it wrong to mandate something with which there is no practical experience.
- The alternative arrangement increases the number of top-level directories, so the files that must be found using subdirectory searching are spread out in a wide, shallow tree. This could have a profound impact on the efficiency of subdirectory searching.

C.2 Font structure

The TWG struggled more with the font directory structure than anything else. This is not surprising; the need to use the proliferation of PostScript fonts with TEX is what made the previous arrangement with all files in a single directory untenable, and therefore what initiated the TDS effort.

C.2.1 Font file type location

We considered the supplier-first arrangement currently in use at many sites:

`fonts/`⟨*supplier*⟩`/`⟨*typeface*⟩`/`⟨*type*⟩`/`

This improves the maintainability of the font tree, since all files comprising a given typeface are in one place, but unless all the programs that search this tree employ some form of caching, there are serious performance concerns. For example, in order to find a `TFM` file, the simplest implementation would require TEX to search through all the directories that contain `PK` files in all modes and at all resolutions.

In the end, a poll of developers revealed considerable resistance to implementing sufficient caching mechanisms, so this arrangement was abandoned. The TDS arrangement allows the search tree to be restricted to the correct type of file, at least. Concerns about efficiency remain, but there seems to be no more we can do without abandoning subdirectory searching entirely.

We also considered segregating all font-related files strictly by file type, so that METAFONT sources would be in a directory `texmf/fonts/mf`, property list files in `texmf/fonts/pl`, the various forms of Type 1 fonts separated, and so on. Although more blindly consistent, we felt that the drawback of more complicated path constructions outweighed this. The TDS merges file types (`mf` and `pl` under `source`, `pfa` and `pfb` and `gsf` under `type1`) where beneficial.

C.2.2 Mode and resolution location

We considered having the `mode` at the bottom of the font tree:

`fonts/pk/`⟨*supplier*⟩`/`⟨*typeface*⟩`/`⟨*mode*⟩`/`⟨*dpi*⟩`/`

In this case, however, it is difficult to limit subdirectory searching to the mode required for a particular device.

We then considered moving the `dpi`⟨*nnn*⟩ up to below the mode:

`fonts/pk/`⟨*mode*⟩`/`⟨*dpi*⟩`/`⟨*supplier*⟩`/`⟨*typeface*⟩`/`

But then it is not feasible to omit the `dpi`⟨*nnn*⟩ level altogether on systems which can and do choose to use long filenames.

C.2.3 Modeless bitmaps

The TDS specifies using utility names as mode names for those utilities which generate bitmaps, e.g., `texmf/fonts/pk/gsftopk/`. An alternative was to introduce a single directory `modeless/` below which all such directories could be gathered. This has the considerable advantage of not requiring each such directory name to be listed in a search path.

But it has disadvantages, too: it would use the last available directory level, preventing any future expansions; and it would put `PK` files at different depths in the tree, possibly hindering search strategies and certainly a likely source of confusion. We decided to stay with existing practice and keep the utility name at the same level as the mode names.

We are making an implicit assumption that METAFONT is the only program producing mode-dependent bitmaps. If this becomes false we could add an abbreviation for the program to mode names, as in `mfcx` vs. `xyzcx` for a hypothetical program *Xyz*, or we could at that time add an additional program name level uniformly to the tree. For our present purposes, it seems more important to concisely represent the current situation than to take account of hypothetical possibilities that may never be realized.

C.3 Documentation structure

We considered placing additional documentation files in the same directory as the source files for the packages, but we felt that users should be able to find documentation separately from sources, since most users have no interest in sources.

We hope that a separate, but parallel, structure for documentation would (1) keep the documentation together and (2) make it as straightforward as possible for users to find the particular documentation they were after.

D Related references

This appendix gives pointers to related files and other documents.

In this document, ⟨*CTAN:*⟩ means the root of an anonymous ftp CTAN tree. This is both a host name and a directory name. For example:

`ftp.dante.de:/tex-archive`
`ftp.shsu.edu:/tex-archive`
`ftp.tex.ac.uk:/tex-archive`

Finger `ctan@ftp.shsu.edu` for a complete list of CTAN sites; there are mirrors worldwide.

Here are the references:

- The TDS mailing list archives can be retrieved via ftp from `shsu.edu:[twg-tds]` and `vms.rhbnc.ac.uk:/archives/twg.tds.*`.
- A sample TDS tree: ⟨*CTAN:*⟩`tds`.
- A collection of BibTeX databases and styles: `ftp.math.utah.edu:/pub/tex/bib`.
- *Components of TeX* by Joachim Schrod: ⟨*CTAN:*⟩`documentation/components-of-TeX`.
- The level 0 DVI driver standard: ⟨*CTAN:*⟩`dviware/driv-standard/level-0`.
- *Filenames for TeX fonts*: ⟨*CTAN:*⟩`documentation/fontname`. This distribution includes recommended supplier and typeface names.
- ISO-9660 CD-ROM file system standard: `http://www.iso.ch/cate/cat.html`.
- A complete set of METAFONT modes: ⟨*CTAN:*⟩`fonts/modes/modes.mf`. This file includes recommended mode names.

E Contributors

The TWG had no formal meetings; electronic mail was the primary communication medium.

Sebastian Rahtz is the TeX Users Group Technical Council liaison. Norman Walsh is the committee chair. For version 0.999 of the draft, Karl Berry took over as editor and coordinated this release.

Original contributors:

Barbara Beeton	Karl Berry
Vicki Brown	David Carlisle
Thomas Esser	Alan Jeffrey
Jörg Knappen	Pierre MacKay
Rich Morin	Sebastian Rahtz
Joachim Schrod	Christian Spieler
Elizabeth Tachikawa	Philip Taylor
Ulrik Vieth	Paul Vojta
Norman Walsh	

Additional contributors:

David Aspinall
Nelson Beebe
Harriet Borton
Bart Childs
Damian Cugley
Alan Dunwell
Michael Ferguson
Erik Frambach
Bernard Gaulle
Jeffrey Gealow
George Greenwade
Thomas Herter
Berthold Horn
Charles Karney
David Kastrup
David Kellerman
Wonkoo Kim
Richard Kinch
Robin Kirkham
Alex Kok
Eberhard Mattes
Bob Morris
Lenny Muellner
Oren Patashnik
David Rhead
Mark Sinke
Andrew Trevorrow
Doug Waud
Chee-Wai Yeung

Hints & Tricks

Whatever is Wrong with my LaTeX File?

Sebastian Rahtz

1 Introduction

Contrary (perhaps) to what many TeX people experience, much of the LaTeX that I have to untangle is not written by me. At Elsevier Science we accept LaTeX files for more or less any of our 1200+ journals, and our production editors have to coerce the submissions into a standard form so that we can apply our journal-specific styles. Along the way, many problems can arise, and we hold in-house training sessions to discuss techniques for finding bugs in other people's LaTeX. These notes arise from those sessions, and are offered as a light-hearted reminder to LaTeX writers and editors alike about some of the ways the agony of using our idiosyncratic system can be lessened. Following a felicitous parallel drawn by the *TUGboat* reviewer of this article, think of this like those posters on the doctor's wall which you read while waiting to see the specialist. It is not a serious guide to TeX debugging, for which I am not qualified, and which would require a very large book indeed...

Perhaps these musings[1] will stimulate others to write about *their* working methods in *TUGboat*.

2 Golden rules

If you do not take the following precautions, you might as well give up writing or editing LaTeX now:

1. *Look* at TeX errors; those messages flashing across the screen are not some kind of screen saver.
2. Be prepared to read the log file too; did you realize it has extra information? Specifically, it will list characters missing from a font.
3. OK, so you ignored those two rules; but at least realize you *have* a log file, and take it with you when you visit the doctor.
4. Lay out the source sensibly; how can you find errors if your input is one long line of mixed macros and text?
5. Use syntax checkers; there are many of these: I use *lacheck*, from the authors of Emacs AUCTeX, and the one built into Eddi4TeX, but there are others. For LaTeX especially, it is a god send to have the missing `\end{enumerate}` spotted for you.
6. LaTeX has several packages to help show you what it is working with: `showkeys` shows you the labels you define; `syntonly` will run a LaTeX file fast, ignoring fancy typesetting; the `listfiles` command lists the macro files that were used at the end (handy for checking versions), and the `draft` option will show overfull boxes and all manner of other things for some packages.
7. If you are a confident macro programmer, be aware of the many TeX primitives that can help you: set `\errorcontextlines` to give more context for help messages, use `\message` to put in diagnostic messages, try `\meaning` to find out what a macro really *is* defined as, rather that what you assumed it was. Don't despair at the amount of verbiage `\tracingall` gives you — there is gold there if you dig deep enough.
8. Remember primitive programmer's debugging techniques; if all else fails in your quest to see why LaTeX dies with that weird error in your 10000 line file, move `\end{document}` gradually back up the file from the end until it *does* work, and then stare at the 10 lines which you know provoke the error, with a wet towel around your

[1] An earlier version was published as part of the editorial in *Baskerville* 5(5), and is used with permission of the UK TeX Users Group.

head. It is faster than reading all 10000 lines over and over again hopelessly...

9. *Do not* mail the LaTeX development team, or other package authors, every time TeX gives you an error prompt; you'll irritate hard-pressed volunteers working in their spare time. If you wait until you have a *good*, well-documented, repeatable, error condition that your friends get too, *then* you can report it, and likely get a friendly reply and a fix.
10. Read before you Write. There are many *excellent* books about TeX and LaTeX that you can buy and read, as well as the freely available 'Frequently Asked Questions' document (make sure you get the UKTUG version, as it is considerably changed and enhanced from the original). You *cannot* use LaTeX without a manual.

3 Examples

3.1 Layout

Did you think I was joking about laying out your text in a readable fashion? Can you easily find the error in this example?

```
\begin
{document}\baselineskip=12pt\newcommand
{\F}{Fig.~}\newcommand {\w}{\omega
}\newcommand {\k}{\xi }\newcommand
{\p}{\phi
}\maketitle\thispagestyle{empty}\centerline
{\bf \underline{Abstract}}\vskip
6ptA probabilisticoptimal design
methodology for complex structures
using the existing probabilistic
optimization techniques.  \vskip
12pt\centerline{\bf
\underline{Nomenclature}\vskip 6pt
\begin{tabbing}\( A
\)\hspace{0.45in} \=:
Transformation matrix\\\( a_i \)
\>: Gradient of performance
function with respect \\$\hskip
1.25in$ to $i^{th}$ random variable
\\\( b \) \>: Design variable
vector\\\( {\it CDF} \) \>:
Cumulative distribution
function\\\( {\it COV} \) \>:
Coefficient of variation \\\( C_x
\) \>: Covariance
```

That is, of course, an artifical example, but one does come across files which look a bit like this. Common sense (and the LaTeX manual) will suggest that replacing code like:

```
\vskip 3pt\noindent{\bf \underline{Safety
Index Interpolation}}\vskip 1pt
```

with

```
\section{Safety Index Interpolation}
```

will considerably aid readability and maintenance. It is a curious fact that some files sent in to Elsevier journals purporting to be LaTeX are little more than plain TeX with `\documentstyle` inserted at the front, and the above is not unusual. It also arises when a frustrated LaTeX user cannot work out how to make the `\section` command do what is required, so brute force is used at the last moment.

Do not stop at simply choosing rational places for line endings; is this

```
\title{Some dull results}
\author{My AlterEgo}
...
and that was the last paragraph.
\section{Another section}
\begin{enumerate}
\item \emph{Look} at \TeX\ errors;
those messages flashing across
the screen are not some kind of
screen saver.
\item Read the  log file too; did
you realize it has extra
information? Specifically, it will
list characters missing
from a font.
\end{enumerate}
```

as easy to read as this?

```
%-------------------------------------
\title
            {Some dull results}

\author
            {My AlterEgo}
%-------------------------------------

....
and that was the last paragraph.

%-----------------------------------------
\section{Another section}

\begin{enumerate}
 \item \emph{Look} at \TeX\ errors; those
       messages flashing across the screen
       are not some kind of screen saver.
 \item Read the  log file too; did you
       realize  it has extra information?
       Specifically, it will list characters
       missing from a font.
\end{enumerate}
```

Again, this seems trivial, but consistent and readable layout of the code is well worth the trouble; some intelligent editors (like Gnu Emacs in AUCTeX mode) can do almost all of it automatically.

3.2 Syntax errors

TEX error messages are not as obscure as we sometimes think; here is an example where the puzzling output is all explained in the log file:

```
{This is not so bad,
\bfseries\ttfamily hello?}
{This is not so bad, \scshape
Hello \bfseries Goodbye?}
{\it\bf\Large byebye}
\end{document}
```

Why do we not see bold typewriter or bold small caps? Because the fonts do not exist, and LATEX tells us it has had to make substitutions as best it can:

```
LaTeX Font Warning: Font shape
 `OT1/cmtt/bx/n' in size <10>
  not available
(Font)        Font shape `OT1/cmtt/m/n'
 tried instead on input line 4.

LaTeX Font Warning: Font shape
 `OT1/cmr/bx/sc '   undefined
(Font)        using `OT1/cmr/bx/n'
 instead on input line 6.
```

What more could you ask? Regular LATEX users must learn to understand these New Font Selection Scheme messages, as they are a crucial part of LATEX 2_ε.

Now let us look at a bad file which is quite easy to understand:

```
\documentclass{article}
something
\begin{document}
hello \( a=
\end{documen
```

LATEX says of this, in an unusually clear way:

```
! Missing $ inserted.
<inserted text>
                $
l.4

?
)
Runaway argument?
{documen
! File ended while scanning use of \end.
<inserted text>
                \par
<*> bad

?
```

though the 'missing $' is a bit confusing when what it meant was 'missing \)'. *lacheck* does a much better job:

```
"bad.tex", line 5:
      <- unmatched "\end{documen}"
"bad.tex", line 3:
      -> unmatched "math begin \("
"bad.tex", line 5:
      <- unmatched "end of file bad.tex"
"bad.tex", line 2:
      -> unmatched "\begin{document}"
```

However, it sees nothing wrong with this:

```
\documentclass{article}
\begin{document}
Funnies: \dag, \AA and \"
\section{Introduction}
\end{document}
```

about which LATEX says:

```
! Argument of \@xdblarg has an extra }.
<inserted text>
                \par
<to be read again>
                   }
l.5 \section
            {Introduction}
?
```

How long did it take you to spot the problem? Can someone suggest a technique other than towel-round-the-head staring to catch it?

3.3 Hyphenation

If hyphenation is your bugbear, do you understand the difference between the following large heavy animals?

```
rhinoceroses
\showhyphens{rhinoceroses}
\hyphenation{rh-ino-cer-os-es}
rhinoceroses
\begin{sloppypar}
rhinoceroses
\end{sloppypar}
rh\"inoceroses
\fontencoding{T1}\selectfont
rh\"inoceroses
\par\hskip\z@skip
rhinoceroses
```

Remember that:

1. TEX may need help hyphenating the word; give it clues;
2. If you want justification at all costs, set the right parameters — `sloppypar` goes too far, using *very* lax settings, but it works;
3. If you put accents in words, hyphenation dies ...
4. ... unless you use T1 encoding, which cleverly transforms `\"i` to an 8-bit character internally

so that TEX proceeds happily (but remember that you need 8-bit hyphenation patterns to do a proper job);

5. The first word of a paragraph will not hyphenate. Insert something harmless to bypass this law.

3.4 Frequently encountered pitfalls

I expect all my readers have written something like this at some time:

```
\begin{figure}
\label{fig1}
\caption{This is a caption}
\end{figure}
```

and wondered why the labels are wrong. It is *not* the figure environment which sets labels, but the `\caption` command; what the example above will do is set the label 'fig1' to the value of the most recent section, equation, list item or whatever.

Do the new LATEX2e packages puzzle you? Why doesn't this work:

```
\usepackage{graphicx}
\begin{document}
This is \rotatebox{75}{hello sunshine}
  at an angle
\end{document}
```

Simply because rotation, colour, scaling, and graphics insertion are all device dependent, and LATEX needs to know what dvi driver you have. You probably meant something like:

```
\usepackage[dvips]{graphicx}
```

Lastly, did your TEX just say 'bufsize exceeded'? Maybe the file it was reading came from a Mac? or a word-processor which stored each paragraph as a single long line? If it is a graphic file, it may have come from a Mac package, and TEX is throwing up while searching for a `%%BoundingBox` line. You should realize that DOS, Unix and Mac treat line-endings differently! If you don't have a dedicated utility to fix this, try using *zip* to package up the files, and then *unzip* them, using the flag to convert text files to the local native format.

4 Conclusions

One could go on listing common problems, and mysterious LATEX errors, for many pages. But the fundamental message is that you cannot treat TEX products like the finite and menu-driven offerings from Microsoft. If you write your documents using a computer programmer's assembly language, you are always going to be exploring strange new worlds. If you think you have better ways of spending your time — don't use TEX directly at all. The excellent *Scientific Word* interface to LATEX will spare you most of the pain described in this article, and others are sure to follow.

Choose LATEX with a light heart: If you can keep your head when all about you Are losing theirs and blaming it on you... If you can wait and not be tired by waiting... if you can meet with Triumph and Disaster, And treat those two imposters just the same; ... If you can bear to hear the truth you've spoken Twisted by knaves to make a trap for fools, Or watch the things you gave your life to, broken, And stoop and build 'em up with worn-out tools ... If you can fill the unforgiving minute With sixty seconds' worth of distance run, Yours is TEX and everything thats in it, And — which is more — you'll be a Man, my son!

◇ Sebastian Rahtz
Production Department,
Elsevier Science Ltd,
The Boulevard, Langford Lane,
Kidlington, Oxford OX5 1GB
UK
Email: s.rahtz@elsevier.co.uk

Macros

New Perspectives on TEX Macros

Jonathan Fine

Abstract

Using the TEX macro language as an example, this article indicates how SGML can be used the specify the source file syntax for literate programming. (This is part of the philosophy behind the author's SIMSIM project, which will allow TEX to typeset SGML documents.) Some of the advantages are shown. The problems of implementation are not discussed.

Introduction

This article is about TEX macros, SGML and literate programming. It also explains some of the philosophy behind the author's SIMSIM package, which will provide a basis for the formatting by TEX of SGML documents. The author hopes for a first release to selected test sites by the end of 1995.

Knuth implemented literate programming by defining the WEB file format, and producing two auxiliary programs, WEAVE and TANGLE, which transform a WEB file into TEX and Pascal files respectively. This was done in the early 1980s. Today it might be better to use an SGML document type definition in the place of the WEB file format. This would allow existing and future SGML tools to process the program source code.

TANGLE can also reorder the code, so that the programmer can present the code the program in an order which suits the programmer (and reader) rather than the compiler. This is considered by its practitioners to be an essential feature of literate programming. (The author thanks the referee for pointing this out.)

Consider now TEX macros. Here is a macro definition, written in an unspecified SGML DTD.

```
<mac n=echo> <par n=Token> writes the
|Token| to the console.
    message { string Token }
</mac>
```

Its meaning should be clear. The intention is to define a macro, whose name is echo. It takes a single parameter, which the author is calling a Token. The replacement text is given by the lines in the <mac> element that begin with a leading space. Notice that there are no backslashes. Instead, the character string message is standing for the control sequence whose name is message (and which is usually referred to by \message). As usual, { and } stand for characters with category code 1 and 2 respectively. Finally, Token stands for #1, as Token was the first parameter to be declared.

Compare this definition to the text

```
\def\echo#1{\message{\string#1}}
```

that would be used in an ordinary macro file to express the same meaning.

Here is another example.

```
<mac n=gobble> <discard> takes a
token (or balanced list) and
throws it away.
</mac>
```

which has empty replacement text. Here

```
\def\gobble#1{}
```

is the ordinary form for this definition.

The SGML form requires more effort to write, but that is because it is more expressive. For complicated macros, it is a great help, to have named rather than numbered macro parameters.

More examples

The benefits of the SGML approach grow, the larger and more complicated the macros are. Here is an example. (The closing </mac> is to be understood. The SGML omitted end tags feature will supply it if the next tag is also a <mac>, or any other element that cannot occur within a <mac> element. In the same way, the short reference feature can recognise a leading blank and within the <mac> element, translate it into <code-line>. Similarly, within <code-line> the carriage return can be translated into the end tag </code-line>.)

```
<mac n=show> <par n=Token> is like
the \show primitive of &TeX except
that it is not like an error message.
    immediate write 16
    {
      > ~~~ string Token =
        meaning Token
    }
```

The ~ stands for an ordinary space character. Dirty tricks are required to get a sequence of such characters into the replacement text of a macro. TEX runs more efficiently if numeric constants such as 16 are replaced by tokens that have been \chardef'd to the appropriate value. It is much easier to write (and read) 16 than it is the control sequence \sixt@@n that is used in the source file for plain and LATEX. The characters > and = stand for themselves, as 'other' characters.

The replacement text of a TEX macro is a sequence of tokens. The syntax and semantics of the source code file format should allow the programmer to specify, perhaps implicitly, the sequence of tokens desired. The analog of TANGLE should produce a file from which TEX can produce (at high speed) the specified macro definition. The technical means to accomplish this are not discussed in this article. Suffice to say that everything described here is known to be possible.

From SGML tags to TEX actions

As an SGML document is parsed, information becomes available to the text processing application. Typesetting (or any other processing) of an SGML document consists of linking actions to the start and end tags, and to other events. Suppose the document to be processed has an element called <tag-name>, with an attribute called text. The code below

```
1. <gi n=tag-name> This tag has a text
2. attribute, whose value will be typeset
3. in a box.
4.     begingroup
5.       //  some code is omitted
6.       let end-element endgroup
7.       hbox { (tag-name*text) }
```

specifies processing for such an element.

Line 4 tells us that once the tag has been parsed, a group is begun. Line 5 is a comment. Line 6 says that `\endgoup` is the action to be performed when the element comes to an end. Note that because SGML allows hyphens, periods and digits to occur in a name, it is convenient to allow the same for control sequences. Incidentally, it is much easier to type and to read a hyphen, than it is an underscore.

It is line 7 that sets the value of the text attribute in a horizontal box. The sequence of characters

```
(tag-name*text)
```

stands for a single token, whose expansion will be the current value of the `text` attribute of the `<tag-name>` element. Thus, the text

```
<tag-name text="This and that">
```

will cause the SGML parser to define the token referred to by

```
(tag-name*text)
```

to be a macro whose expansion is the sequence

```
This and that
```

of letters. Just quite what that token is, should be of no concern to the programmer. Indeed, it should not be possible for the programmer to access this token, except throught the `(tag*att)` construct.

In the same way

```
<ent n=TeX> typesets the &TeX logo.
    'T kern <dim v=-.1667em>
    lower <dim v=.5ex> hbox { 'E }
    kern <dim v=-.125em> 'X
```

specifies the action to be linked to the SGML entity `&TeX`. By way of explanation, the right quote `'` is an escape character. Thus, `'T` stands for a letter `T` with (for technical reasons) category code 'other'. The `<dim>` element in the macro definition should be translated, by the `TANGLE` equivalent, to an appropriate quantity. So long as the semantics are well defined, the translation can be made.

Conclusion

In the humanities, it is becoming more widely accepted that structured documents should be stored with a rigorous syntax, and that SGML provides a means of specifying that syntax. In addition, a growing collection of SGML software tools are becoming available.

Literate programming (which if not in the humanities is at least an art) also requires a rigorous document syntax. There is a strong case for using SGML in this context also. For this to succeed, there must be available suitable typesetting tools, that will accept SGML documents. The author's SIMSIM project is intended to provide such.

◇ Jonathan Fine
203 Coldhams Lane
Cambridge CB1 3HY
UK
Email: `J.Fine@pmms.cam.ac.uk`

LaTeX

Never again active characters! Ω-Babel

Yannis Haralambous, John Plaice and Johannes Braams

»Weißt Du, wo ich das Wasser
des Lebens finden kann?«
»An der Grenze Phantásiens«,
sagte Dame Aiuóla.
»Aber Phantásien hat keine Grenzen«,
antwortete er.
»Doch, aber sie liegen nicht außen,
sondern innen.«

M. Ende, Die unendliche Geschichte

Abstract

This progress report of the Ω development team (the first two authors) presents the first major application of Ω: an adaptation of Babel, the well-known multilingual LaTeX package, developed by the third author. We discuss problems related to multilingual typesetting, and show their solutions in the Ω-Babel system.

The paper is roughly divided into two parts: the first one (sections 1–4) is intended for average LaTeX users, especially those typesetting in languages other than American English; the second part (section 5) is more technical and will be of more interest to developers of multilingual LaTeX software.

1 Introduction

Ω-Babel is the first real-world application of Ω: we are in the process of adapting the multilingual LaTeX package Babel by Johannes Braams (1991a, 1991b) to take advantage of the functionality of Ω. This allows safer and more complete LaTeX typesetting of languages other than American English. Problems due to technical limitations of TeX are solved; for example, the LaTeX macros `\MakeUppercase` and `\MakeLowercase` have been replaced by Ω filtering processes. The whole process is simpler and more natural.

In this progress report we present the first step in adapting Babel to Ω. There will be (at least) two more steps which we describe below; for more information, see section 5 (the technicalities).

1. Babel adapted to Ω; we use DC font output. `\MakeUppercase` and `\MakeLowercase` macros are being replaced by macros launching translation processes. Various input encoding translation processes are being written. The `inputenc` and `fontenc` packages are being adapted (their Ω counterparts are called `inpenc` and `fntenc`). This step has been completed.
2. UC (Unicode Computer Modern) fonts will be released by spring 1996. These fonts contain Latin, Greek and Cyrillic characters, and a certain number of dingbats and graphical characters. Ω-Babel uses UC fonts for output: new languages will be dealt with (Bulgarian, Esperanto, Greek, Latvian, Lithuanian, Maltese, Russian, Vietnamese, Welsh, etc.). Alan Jeffrey's `fontinst` will be adapted to make extended virtual fonts in UC encoding.

   Latin letters with dieresis will be provided in two versions: with high or low accent: Frenchmen like dieresis (→ tréma) to be high, Germans prefer to have it (→ Umlaut) a little lower.
3. Arabic alphabet languages, Hebrew and Yiddish will be added at some point during 1996.
4. Soft hyphenation will be done through Ω translation processes. This allows dynamic loading of hyphenation algorithms, independently of the process of format creation. There will be no need to recompile formats when adding or changing hyphenation patterns. It will also be possible to add new features to hyphenation (automatic processing of German „ck→k-k“, preferential hyphenations, etc.).

2 Practically, what does this mean for me?

It means that if you are typesetting in some non-American English language covered by current Babel (Bahasa, Breton, Catalan, Croatian, Czech, Danish, Dutch, English, Estonian, Finnish, French, Galician, German, Hungarian, Irish, Italian, Lower or upper Sorbian, Norwegian, Polish, Romanian, Scottish, Slovakian, Slovenian, Spanish, Swedish, Portuguese or Turkish), then Ω can make it easier for you and give you better results. All you have to do is download the Ω implementation for your machine from `ftp://ftp.ens.fr/pub/tex/yannis/omega/systems` or the top-level `omega` directory[1]; if you use a TeX implementation not already covered, then ⁂kindly⁂ suggest to the implementor that they consider including Ω support). Then download the Ω-Babel package from `ftp://ftp.ens.fr/pub/tex/yannis/omega/macros/obabel` or , install it on your machine and use it! The basic syntax is the same as in Babel, but we will discuss new options and functionality in section 4.1.

Figure 1: Allegory: input (on the left) and output (on the right) encodings, too close together.

3 General philosophy of combined Ω and Babel

In Fig. 1, the reader can see an allegory of how TeX works: input (the worker on the left) and output

[1] At press time Ω has been ported to several UNIX machines (Sun, Silicon Graphics and others), DOS (by Kraus Kalle) and Macintosh (by Tom Kiffe).

(the one on the right) are close together. For example, consider the fact that hyphenation patterns—which are a language-intrinsic feature—are described in the output font encoding. In Fig. 2 you can see how Ω remedies this situation: input and output are clearly separated and, in between, there is a big container (imagine a huge barrel), the Unicode encoding.

Whatever you type is first of all converted into Unicode. This conversion is language dependent; for example, the ASCII character 'i' does not mean the same thing in Turkish and in the other Latin-alphabet languages. Unicode is very big; so you will hardly ever ask for something not available; even if that happens there is a "private zone" where we can temporarily store characters of our own choice.

Once inside Unicode we deal with pure device-independent information: we can process it in many different ways. In Fig. 2 we list five possible transforms, all pertaining to Ω-Babel, and we will discuss these in turn. The Ω translation processes needed for every language are activated when you enter the corresponding Ω-Babel environment. Our goal is to keep the technical aspects hidden: the average user typesetting in a given language does not need to be aware of the different transformations we have just described.

Aliasing

We call "aliasing" the making of aliases. An *alias* is the expression of a character in some convenient way: for example in 7-bit ASCII. In German Babel, when you write `"s` instead of `\ss{}`, this is an alias: (a) it is 7-bit, (b) you can very well avoid it if your keyboard and screen support 8-bit characters, and (c) in TEX, it used to be handled using active characters.

Inherent transforms

These go deeper than aliases; they are:

1. either transforms that traditionally belong to TEX syntax, like `---` for "—", or `‘‘` for the English opening double quote, or `?‘` for the Spanish inverted question mark "¿";
2. or transforms that historically derive from dactylographical keyboard traditions (for example, the French `<<` that gets converted into "«" with the appropriate spacing, or the Catalan `l.l` which produces "l·l");
3. or things that are supposed to be hidden from the user: for example, the Dutch `ij` which always produces the "ij" ligature, unless the user places something invisible in between: `bewijs` vs. `bi{}jektie`; or, in Turkish and Portuguese, the fact that there should be no 'ff', 'fi', ... ligatures, etc.

See 4.4 for a good example of the difference and combined use of aliases and inherent transforms.

Hyphenation

Hyphenation must be done on the level where the information is most device-independent: that is, the Unicode level. Not only can you define hyphenation algorithms using all possible 16-bit characters (for example to hyphenate Welsh, which uses letters such as "ŵ"), but your algorithm is automatically valid for any input or output encoding. We'll come back to this issue later in 1996, when we reach step 3 of Ω-Babel development.

Upper/lowercasing

This is certainly not a trivial process: different letters may share the same glyph for their upper or lower forms (example: both the Icelandic eth "ð" and the Croatian dz "đ" share the glyph "Đ" for their upper form); letters may have different upper forms depending on the semantics of the word (example: one possible upper form of the German "ß" is "SS", another one is "SZ"), or on local traditions (in bad French typography, upper forms of accented letters are not always accented), or on the language itself (example: the upper form of the Turkish letter "i" is "İ" and not "I"—the lower form of "I" is "ı" and not "i") these are major incompatibilities, and hence we should be able to dynamically change the upper/lowercasing process.

Finally, as Martin Dürst pointed out on the `omega` list[2], there are six kinds of letter cases:

1. regular lowercase (glyph becomes uppercase when we apply the uppercasing process);
2. regular uppercase (glyph becomes lowercase when we apply the lowercasing process);
3. fixed lowercase (glyph is invariant under the uppercasing process), for example the "m" and "b" in the German „GmbH“;
4. fixed uppercase (glyph is invariant under the lowercasing process), for example the first letter of a name;
5. fake lowercase (the glyph is uppercase, it becomes lowercase when we apply the uppercasing process), for example the "i" in the modern German word "STUDENTiNNEN" (=politically correct male and female students);

[2] Join us on the Ω e-mail discussion forum `omega@ens.fr`, by sending the usual subscription message to `listserv@ens.fr`.

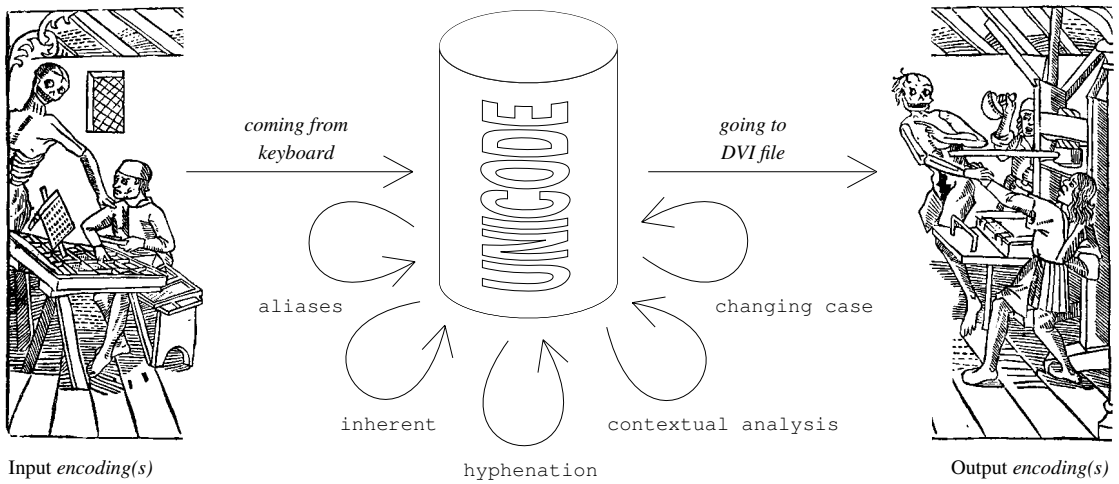


Figure 2: The Ω way: inserting Unicode between input and output encodings.

6. fake uppercase (the glyph is lowercase, it becomes uppercase when we apply the lowercasing process), for example the "I" in the the same modern German word "StudentInnen", written in lowercase type.

Ω introduces two new commands to handle "fixed" case (`\fixedcase`) and "fake" case (`\fakecase`). Using these commands, you can write

```
\fixedcase{T}ruth
```

to be sure that your word will always be "Truth, with a big T" or

```
\fixedcase{S}tudent\fakecase{I}nnen
```

to obtain correctly spelled politically correct (fe)male students in German. The argument of `\fixedcase` or `\fakecase` must be a single letter.

In Table 1 we present some examples of the use of the different case-related commands. Note in the first example, that—contrary to the usual TEX uppercasing commands—math mode is not affected.

Contextual analysis

This transform is perhaps less important for Latin-alphabet languages (although it becomes important when you want to use a "long s" at the beginning of a word). It is extremely important for Greek (final or medial sigma), Hebrew (there are five letters with medial and final form), and especially for the Arabic alphabet.

4 Details on each language

Before we start considering languages one by one, a brief description of Ω-Babel syntax.

4.1 Ω-Babel syntax

To be able to use Ω-Babel you have to load the `omega` package: this is done automatically if you use either `inpenc` (the input encoding package) or `fntenc` (the output font encoding package). The former can take as an option an input encoding (the reader can find a list of such options on Table 3), the default value is `lat1` (ISO Latin-1). The latter can take only one option for the moment: `T1`. As of step 2 of Ω-Babel development a new encoding will be added: `UT1`. Here is an example of the loading of these packages:

```
\usepackage[stmac]{inpenc}
\usepackage[T1]{fntenc}
```

As in the original Babel package, you load Ω-Babel by using the command `\usepackage`, and by giving the names of languages you are going to use as an optional argument. These names are separated by commas, and the last one becomes the "default" language, as in

```
\usepackage[turkish,german,francais]{obabel}
```

The names for languages are the same as in original Babel.

Once Ω-Babel is loaded, you are in the "default language": captions, date, hyphenation, 7-bit input and typographical specifications are adapted.

Contrary to the original Babel where the command `\selectlanguage` was used to switch between languages, in Ω-Babel you have to use an environment, called `lang`. Here is an example:

```
\documentclass{article}
\usepackage[stmac]{inpenc}
\usepackage[T1]{fntenc}
\usepackage[germanb,turkish,%
```

`\MakeUppercase{is it an $a$ or a $b$?}` → IS IT AN a OR A b
`\MakeUppercase{mon \oe il}` → MON ŒIL
(in German) `\MakeUppercase{ma"se oder ma"ze?}` → MASSE ODER MASZE?
`\MakeLowercase{\MakeUppercase{ma"ze}}` → maße
(in Turkish) `\MakeUppercase{kap\i y\i{} i\c ceri}` → KAPIYI İÇERİ
`\MakeLowercase{BROWN}` → brown
`\MakeLowercase{\fixedcase{B}ROWN}` → Brown
`\MakeLowercase{\fixedcase{S}TUDENT\fakecase{i}NNEN}` → StudentInnen

Table 1: Some examples of upper/lowercasing.

```
francais]obabel}
\begin{document}

Histoires d'oiseaux en deux langues :

\begin{lang}{german}
"Uber allen Gipfeln ist Ruh,
die V"oglein schweigen im Walde...
\end{lang}

\begin{lang}{turkish}
Kedinin yakalad\i\u g\i{} ku\c sun
t\"uyleri havada u\c cu\c suyordu...
\end{lang}

\end{document}
```

This is because Ω uses a stack for translation processes: every time you enter a `lang` environment, Ω pushes a set of translation processes on the stack (see also section 5); every time you leave it, Ω pops a set of translation processes from the stack.

The environment approach is also essential for typesetting reasons: in right-to-left languages (Arabic, Hebrew, etc.), line breaking is done differently depending on whether we are in "global" or in "encapsulated" right-to-left mode.

As in the original Babel, the `\languagename` macro contains the name of the language.

Finally you have the following additional Ω commands: `\fixedcase` and `\fakecase` as described above, and a command `\omegaversion` which returns the version of your Ω implementation.

Let us now take one by one the languages covered by Ω-Babel to see what has changed.

4.2 French

French features are loaded by the `francais` option. Besides the usual hyphenation, caption and date changes, Ω-Babel transforms the punctuation you type, whether you include blank spaces or not.

For example, whether we write `<< Ciel, mon mari ! >>` or `<<Ciel, mon mari!>>` (or even ``` ``Ciel, mon mari!'' ``` if the `clever` option is on), the result will be the same: « Ciel, mon mari ! ». Not a single character is active, so there can be no possible interference with other TEX or LATEX macros.

When we pass to step 2 of Ω-Babel development, `francais` will also act on the "Unicode → output font" level and switch to low dieresis letters[3].

4.3 German

The German features are loaded by the `germanb` option. All original Babel aliases have been kept:

- `"a` `\"a`, also implemented for the other lowercase and uppercase vowels.
- `"s` to produce the German ß (like `\ss{}`).
- `"z` to produce the German ß (like `\ss{}`).
- `"ck` for `ck` to be hyphenated as `k-k`.
- `"ff` for `ff` to be hyphenated as `ff-f`, this is also implemented for l, m, n, p, r and t
- `"S` for `SS` to be `\MakeUppercase{"s}`.
- `"Z` for `SZ` to be `\MakeUppercase{"z}`.
- `"|` disable ligature at this position.
- `"-` an explicit hyphen sign, allowing hyphenation in the rest of the word.
- `""` like `"-`, but producing no hyphen sign (for compound words with hyphen, e.g. `x-""y`).
- `"~` for a compound word mark without a breakpoint.
- `"=` for a compound word mark with a breakpoint, allowing hyphenation in the composing words.
- `"‘` for German left double quotes (looks like „).
- `"’` for German right double quotes.
- `"<` for French left double quotes (similar to <<).
- `">` for French right double quotes (similar to >>).

[3] Default letters with dieresis will carry a "low" accent, French "higher" accented letters will be the exception, introduced by an additional translation process: this decision of the Ω team has no political connotation ☺, it comes just from the fact that French letters with tréma are far rarer than German letters with Umlaut.

(description taken from the source file, `germanb.dtx`). Maybe it is not very clear from the description above, when you should use `"z` instead of `"s`: in fact, they both produce exactly the same result, but when they are uppercased, the first becomes "SS" and the second "SZ" (for example, „MASSE“ comes from „Masse“ and „MASZE“ from „Maße“).

It should be noted also that although "French double quotes" are called "French", they are not typeset with the proper French spacing, as in the French Ω-Babel style.

The UC fonts will also contain an "ft" ligature and closely kerned versions of "ck" and "ch" (as requested by Frank Mittelbach in 1991), and the possibility of typesetting with a "long s" as in old German (the "long s" *is* in fact a Unicode character, and the UC fonts will contain all kind of ligatures "long s + i", "long s + l", ...).

4.4 Dutch

Dutch features are loaded by the `dutch` option. Once again, all original Babel aliases have been kept. We have included one automatic transformation: `ij` and `IJ` produce the "ij" ligature, in lower and upper form. To avoid this ligature, it suffices to introduce an empty group between the letters, or any other "invisible" command.

Dutch is a fine example of separation of *aliases* and *inherent transforms*. Consider for example the case of the letter "ı" in the word „ongeïnteresseerd". When this word is hyphenated as „onge-interesseerd", the letter "i" loses the dieresis (the dieresis is there to indicate that "ei" is not a diphthong; by hyphenating at that location there is no doubt any more that this is the case, so the dieresis is useless).

On the TEX level, Babel solves this problem by using a `\discretionary` command. This is an essential transformation, inherent to the Dutch language. Hence, we have both an alias and an inherent transform:

```
"a --alias--> 0x00e4 --inherent--> \allowhyphens%
   \discretionary{-}{U}{^^^^00dc}%
   \allowhyphens
```

4.5 Portuguese

Portuguese features are loaded by the `portugues` option. Portuguese has just a few aliases, similar to those of the German style. The important fact is that Portuguese has a special inherent transform to avoid 'ff', 'fi', 'fl', 'ffi' and 'ffl' ligatures. We obtain this by inserting between these letters the character ZERO WIDTH SPACE, which—when going to the DC output fonts—becomes a `\kern0pt` command.

This is not the best way of solving this problem: we are forced to use DC fonts (until we reach step 2 of Ω-Babel development), and these fonts have an automatic ligaturing mechanism to produce the ligatures: by inserting a zero-length skip we avoid the ligature, but lose a possible kern between the letters.

The forthcoming UC fonts will have no internal ligatures: all ligatures will be provided by translation processes, so that we can activate and deactivate them *ad libitum*.

4.6 Catalan

The Catalan features are loaded by the `catalan` option. We provide the following inherent transforms: `l.l` and `L.L` which produce "l·l" and "L·L" respectively. Of course, these can also be typeset by using the aliases `"ll` and `"LL`, as in the original Babel style.

We must point out that for the moment this character is produced in a very unorthodox way (28 lines of code!!). We will obtain *real* typesetting of Catalan only after step 2 of Ω-Babel development, since "l·l" and "L·L" are characters of the UC fonts (this is not the case for DC fonts: the dots have to be dragged to the right place...).

4.7 Spanish

The Spanish features are loaded by the `spanish` option. All aliases requested by the author of the original Babel style have been included in the Ω-Babel adaptation, *except one*: `~n ~N` for ñ, Ñ. In TEX, the character `~` is traditionally used to obtain a non-breakable space, and this should be valid for all languages.

In some languages (like in Greek) one can argue that letters carrying the tilde accent do not appear at the beginning of a word. This is unfortunately not the case of Spanish (we found four such words in a pocket dictionary: *ñandú*, *ñoño*, *ñudo*, *ñudoso*, there might be more...). That's why we decided not to retain this alias for Spanish (fortunately, the author of the Spanish style also has a second alias for the same letter: `'n 'N`).

4.8 Turkish

The Turkish features are loaded by the `turkish` option. This language has two versions of letter "i": with and without dot. The glyph "I" is the uppercase form of letter "ı" and the glyph "İ" is the uppercase form of letter "i". The distinction is essential, both for hyphenation and for upper/lowercasing. To solve this problem, we have defined new codes in (the private zone of) Unicode, for the *Turkish uppercase form of 'i'* and the *Turkish lowercase form*

of 'I'. Whenever you switch to Turkish, the inherent translation process sends all of your (otherwise innocent) 'i's and 'I's to these far away locations in the code (`0xe083` and `0xe084`, that is decimal 57475 and 57476!!), so that Ω has no doubt on the information it is processing. Of course, after processing, we return to the usual 'i' and 'I' glyphs of the DC fonts.

Like French, Turkish needs special punctuation spacing for the colon, exclamation mark and equal sign. This is also done through the inherent translation process.

Finally, like Portuguese, Turkish avoids 'ff'-like ligatures: the same methods are applied.

4.9 Breton, Danish, Estonian, Finnish, Galician, Polish, Slovene, Upper Sorbian

These language styles use aliases, mostly similar to those of the German style (Breton aliases and inherent transforms are similar to the French ones, a coincidence?). We have adapted these aliases; there have been no further changes.

4.10 Bahasa, Croatian, English, Czech, Irish, Italian, Lower Sorbian, Hungarian, Norwegian, Romanian, Scottish, Slovakian, Swedish

Last, but not least, these language styles use no aliases at all (either due to their simplicity, or to the wishes (or keyboard facilities?) of the respective authors). They all have the same trivial alias and inherent translation processes.

4.11 Forthcoming language styles

To prepare a Babel or Ω-Babel language style the most difficult task is to create the hyphenation patterns. We already have Welsh, Esperanto and Lithuanian hyphenation patterns. As for non-Latin alphabet languages, we have hyphenation patterns and fonts for Greek, Russian, Bulgarian and Serbian: all these languages will be covered by Ω-Babel, in step 2 of Ω-Babel development.

5 Let's get technical!

There are two main differences between TEX and Ω: (a) the latter works with bigger numbers (more code positions, more fonts, more boxes, more registers, etc.), (b) it uses *translation processes*. Translation processes are organized in *lists*, which are pushed on a *stack*. Inside a list, a translation process has an ID, a (not necessarily integer) number between 1 and 4095. When you push a stack on a list, translation processes with the same IDs replace each other; when there is a process of a given ID in a given list, and you push upon it a list which has no process with the same ID, then that process remains active.

Input encoding (for example, Macintosh Standard Roman → Unicode, or codepage 437 → Unicode, etc.)	500
Aliases (for example, `"a` → ä, `"z` → "special" ß, etc.) and inherent transforms (for example, French guillemets, Dutch 'ij' ligature, avoiding 'ff' ligatures in Turkish and Portuguese, etc.)	1000
Hyphenation (not available yet)	2000
Contextual analysis (Greek final sigma, Hebrew final letters, Arabic alphabet contextual forms, etc.)	2500
Case change (lower to uppercase, case inversion, etc.)	3000
Output encoding (for example, Unicode to DC, or Unicode to UC, etc.)	3500

Table 2: IDs for usual translation tasks.

So, for example, if you attribute ID 2500 to a contextual analysis process, and ID 1000 to an aliasing and inherent transform, then you can push the latter upon the former: both will still remain active. But if you wish to change the language, you will push a process of ID 1000 (for the new language), and it will replace the previous one.

This means that we have to be consistent in our choices of translation process IDs: we can always insert additional processes between the existing ones, but to be able to push them away automatically when conditions change (for example when we switch languages), we must keep the same IDs. In Table 2 we present some IDs we have chosen for the tasks described in this paper.

So, in each Ω-Babel language style a CTP list is defined (CTP stands for "compiled translation processes"), consisting of a CTP of ID 1000 which contains both aliases and inherent transforms.[4] Each time we open a new `lang` environment, we push new CTPs over it; when we leave the environment it is automatically popped off the stack (CTP loading obeys TEX's grouping rules).

5.1 Description of available CTPs

5.1.1 Input encoding CTPs

In Table 3 we list the input encoding CTPs we have prepared.

[4] The original idea was to split this CTP into one for aliases and one for inherent transforms, of IDs 1000 and 1500; but for speed reasons we have decided to bundle these two into a single CTP.

ISO Latin-1	`lat1uni.ctp`
ISO Latin-2	`lat2uni.ctp`
ISO Latin-3	`lat3uni.ctp`
ISO Latin-4	`lat4uni.ctp`
ISO Latin-5	`lat5uni.ctp`
ISO Latin-6	`lat6uni.ctp`
Macintosh Standard Roman	`stmacuni.ctp`
Macintosh Central Europe	`cemacuni.ctp`
Macintosh Croatian	`hrmacuni.ctp`
Macintosh Icelandic	`ismacuni.ctp`
Macintosh Turkish	`trmacuni.ctp`
Windows ANSI	`ansiuni.ctp`
IBM codepage 437	`cp437uni.ctp`
IBM codepage 850	`cp850uni.ctp`
IBM codepage 852	`cp852uni.ctp`
IBM codepage 857	`cp857uni.ctp`
IBM codepage 860	`cp860uni.ctp`
IBM codepage 861	`cp861uni.ctp`

Table 3: Available input encoding CTPs.

Any suggestions for further enhancement of this list will be welcome (for the moment we are dealing only with Latin alphabet encodings, Greek and Cyrillic will follow, the rest is for... later ☺).

There is not much to say about these translations. In the Macintosh encodings we have chosen to send π and Ω to the corresponding Greek letters in Unicode (although they are supposed to be math symbols), Δ to the math symbol INCREMENT and *not* to the Greek letter Delta, $\Diamond$ to the geometric shape LOZENGE, and the Macintosh Apple to the (private) Unicode character `0xe090` (yes, we have reserved a slot for the Apple logo: after all Yannis owes that to those little machines "for the rest of us"[5]).

In the codepage 437 to Unicode translation, we send `0xd5` (a small vertical stroke) to Unicode `0x02c8` MODIFIER LETTER VERTICAL LINE. Once again, any suggestions will be welcome.

5.1.2 CTPs for aliases and inherent transforms

There is one CTP for every Ω-Babel language style, except for Bahasa, Croatian, English, Czech, Irish, Italian, Lower Sorbian, Hungarian, Norwegian, Romanian, Scottish, Slovakian and Swedish which share a minimal CTP called `minalias.ctp`. The name of each CTP is formed by the two-letter ISO code for the corresponding language, as described in Haralambous (1992a), and the extension `.ctp`: for example: `fr.ctp`, `de.ctp`, etc.

[5] No, we have not reserved any slot for the Windows logo ☺.

`0xe000` ↓ `0xe0ff`	"Direct-to-Unicode" 8-bit table (see below)
`0xe100` ↓ `0xe17f`	Fake ASCII (see below)
`0xe180`	Uppercase German ß (glyph: SS)
`0xe181`	Lowercase special German ß (`"z`)
`0xe182`	Uppercase special German ß (glyph: SZ)
`0xe183`	Uppercase Turkish ı (glyph: I)
`0xe184`	Lowercase Turkish İ (glyph: i)
`0xe185`	Uppercase letter kra (glyph: K)
`0xe186`	Uppercase Afrikaans 'n (glyph: 'N)
`0xe187`	Uppercase long s (glyph: S)
`0xe188`	Uppercase h̲ (glyph: H̲)
`0xe189`	Uppercase ẗ (glyph: T̈)
`0xe18a`	Uppercase ẘ (glyph: W̊)
`0xe18b`	Uppercase ẙ (glyph: Y̊)
`0xe18c`	Uppercase a᾽ (glyph: A᾽)
`0xe18d`	Uppercase ȷ (glyph: J)
`0xe18e`	TEX character ȷ
`0xe18f`	Uppercase ǰ (glyph: J̌)
`0xe190`	Apple Macintosh logo
`0xe191`	Fixed case modifier
`0xe192`	Fake case modifier
`0xe1a0` ↓ `0xe1cf`	Needed for Greek and Armenian

Table 4: How Ω uses the Unicode private zone.

Language styles with no inherent transforms (other than '`ff`' ligatures, `---` punctuation etc.) use the common CTP `minilang.ctp`.

5.1.3 CTPs for hyphenation

These are under development.

5.1.4 CTPs for contextual analysis

These are also under development; the Arabic one is up and running, but we have to bundle the complete package.

5.1.5 Case change CTPs

We have prepared CTPs for uppercasing and lowercasing, called `uppercas.ctp` and `lowercas.ctp`. These are symmetric: both uppercase followed by lowercase, as lowercase followed by uppercase are equal to the identity. This has been achieved by attributing a few special code positions in the private zone of Unicode. In Table 4 the reader can find an overview of the private zone of Unicode as we are using it at the present time.

This table will certainly change in the future, but we promise to keep the changes upwards compatible.

What are the "Direct-to-Unicode" slots for? Suppose you are writing a macro which will produce a character in the 8-bit range of Unicode (which is similar to the ISO Latin-1 encoding), for example Ç, which has code `^^c7`. If you write `^^c7` or `\char"C7` (or `Ç`, using an ISO Latin-1 screen font), then Ω will pass this character through the input encoding filter: if your macro is used on a machine with a different input encoding, the output will not be the same. One needs a way to produce a Unicode character `^^^^00c7` independently of the input encoding. This can always be done by temporarily deactivating the input encoding: a more simple solution is to apply an offset of `0xe000` to the codes that must remain invariant by the input re-encoding process: use `^^^^e0c7` instead of `^^^^00c7` in your macro, and the result will always be Ç (of course, the end user will be able to utilize macros like `\c{C}`: this is how these macros are defined in our system).

And what is this "fake ASCII" all about? It is a trick similar to the previous one, but it sends the information further down the chain of translation processes. It is used to send information to one of the last translation processes (for example the output encoding translation process) which should not be modified at all by the intermediate processes, *while staying in the same buffer*. A buffer is a sequence of bytes read by Ω and processed by translation processes: Ω will stop reading a buffer as soon as it encounters a character that is not of catcode 11 or 12. Why do we need that?

Suppose you are writing Arabic. Contextual analysis is entirely done *inside* a buffer.[6] We may also want the central letter of a word to be typeset in red[7] (this can be of great help in an Arabic grammar book). Inserting a `\red` command inside the word will completely break the contextual analysis. We solve the problem by transposing the characters `\`, `r`, `e`, `d` to the private Unicode zone (by a fixed offset of `0xe000`). The contextual analysis translation process is aware of the fact that characters in that range shall not interfere in the process of performing contextual analysis. Once the contextual analysis is done, and our candidate for being red has the right contextual form, we perform the opposite offset and our string becomes once again a regular TEX command. We will return to this in more detail when Arabic Ω is released.

The fixed and fake case modifiers affect the character following them: they are introduced by the `\fixedcase` and `\fakecase` commands, and are only considered by the `uppercas` and `lowercas` translation processes, in the following way: a character following the fixed case modifier is not affected by either `uppercas` or `lowercas`; a character following the fake case modifier (like the 'I' in StudentInnen) is converted "the other way around": `uppercas` will call `lowercas`, and `lowercas` will call `uppercas`, for the specific character only.

The various strange characters you see on the lists are not included in Unicode for various reasons, we need them in particular to keep `uppercas` and `lowercas` symetric.

5.1.6 Output font CTPs

We have already written the CTP for the Unicode → DC font encoding: all characters included in the DC font table are used as is, others are constructed by using the `\accent` primitive. For step 2 of Ω-Babel development, we will prepare a Unicode → UC translation process. We expect users to write their own translation processes for other output font encodings.

6 Conclusion

We want TEX implementors to join us in the Ω adventure and consider including Ω in their TEX distributions; those based in Web2C should find it very easy. There will be virtual UC fonts, based upon real fonts in the `0x20-0xff` range.

We want DVIware developers to consider making their DVI software compatible with our XVF and XFM files (extended VF and extended TFM), which are 16-bit. For the moment we have extended Peter Breitenlohner's *dvicopy* into a 16-bit version (which we call *xdvicopy*); using this utility we are able to de-virtualize extended virtual fonts into their underlying real fonts, which for the moment are all only 8-bit. De-virtualized DVI files are then compatible with every DVI software in the market. But it would be quicker and more practical for the user if software recognized XVF and XFM files directly (after all, the DVI standard allows up to 32-bit characters: our DVI files conform to the standard).

Finally, we want users to give Ω and Ω-Babel a hard try, and us a hard time in debugging the software, the macros and (later on) the fonts. Ω will soon grow a lot: several Oriental TEX packages are

[6] Otherwise we would have to record somewhere the form of the last-read letter: this is possible of course, but we won't know *why* the buffer was terminated: was it something harmless, like an empty group (in which case the contextuality is kept), or was it something as horrible as a `\newpage` command? In which case we had better finish our word before going to the next page.

[7] This is rendered as a grey scale in *TUGboat*.

already ready and waiting to be adapted to Ω (see references below), but before this happens, we want to be sure that the Latin-Greek-Cyrillic part of it is clean and robust.

References

Andulem (Amnulehe), A. "The road to Ethiopic TEX". *TUGboat* **10**(3), 352–354, 1989.

Braams, J. "Babel, a multilingual style-option system for use with LATEX's standard document styles". *TUGboat* **12**(2), 1991a.

Braams, J. "An update on the Babel system". *TUGboat* **14**(1), 1991b.

Haralambous, Y. "TEX and those other languages ... ". *TUGboat* **12**(4), 539–548, 1991.

Haralambous, Y. "TEX Conventions Concerning Languages". *TTN* **1**(4), 3–10, 1992a.

Haralambous, Y. *TEX et les Langues Orientales*. Paris, 1992b.

Haralambous, Y. "Typesetting the Holy Qur'ān with TEX". In *Proceedings of the 2nd International Conference on Multilingual Computing—Arabic and Latin script (Durham)*. 1992c.

Haralambous, Y. "The Khmer Script tamed by the Lion (of TEX)". In *Proceedings of the 14th TEX Users Groups Annual Meeting (Aston, Birmingham)*. 1993a.

Haralambous, Y. "Un système TEX berbère". In *Actes de la table ronde internationale « Phonologie et notation usuelle dans le domaine berbère », INALCO*. 1993b.

Haralambous, Y. "*Indica*, a Preprocessor for Indic Languages—Sinhalese TEX". In *Proceedings of the 15th TEX Users Groups Annual Meeting (Santa Barbara)*. 1994a.

Haralambous, Y. "Tiqwah: a Typesetting System for Biblical Hebrew, based on TEX". In *Proceedings of the Fourth International Colloquium "Bible and Computer: Desk and Discipline, The impact of computers on Bible Studies" (Amsterdam)*. 1994b.

Haralambous, Y. "Sabra: a Syriac TEX system". In *Proceedings of the First International Forum on Syriac Computing (Washington, D.C.)*. 1995.

Haralambous, Y. and J. Plaice. "First Applications of Ω: Greek, Arabic, Khmer, Poetica, ISO 10646/UNICODE, etc.". In *Proceedings of the 15th TEX Users Groups Annual Meeting (Santa Barbara)*. 1994.

Mittelbach, F. "E-TEX: Guidelines for Future TEX Extensions". *TUGboat* **11**(3), 1991.

Plaice, J. "Progress in the Ω Project". In *Proceedings of the 15th TEX Users Groups Annual Meeting (Santa Barbara)*. 1994.

Velthuis, F. J. *Devanagari for TEX*, 1991.

Haralambous, Y. and J. Plaice. "Ω, a TEX Extension Including Unicode and Featuring Lex-like Filtering Processes". In *Proceedings of the 1994 EuroTEX Conference (Gdansk)*. 1994.

◇ Yannis Haralambous
187, rue Nationale
59800 Lille, France
Email: haralambous@univ-lille1.fr
URL: http://www.ens.fr/~yannis

◇ John Plaice
Université Laval
Québec, Canada
Email: plaice@ift.ulaval.ca

◇ Johannes Braams
TEXniek
Kooienswater 62
The Netherlands
Email: JLBraams@cistron.nl

Calendar

1996

Jan 11 — UK TEX Users Group, School of Oriental and African Studies, London. Structured Documentation (with BCS electronic publishing specialist group). For information, e-mail `uktug-enquiries@ftp.tex.ac.uk`

Feb 6 — DANTE TEX–Stammtisch at the Universität Bremen, Germany. For information, contact Martin Schröder (`MS@Dream.HB.North.de`; telephone 0421/628813). First Tuesday (if not a holiday), 18:00, Universität Bremen MZH, 28359 Bremen, 4th floor, across from the elevator.

Mar 5 — DANTE TEX–Stammtisch at the Universität Bremen, Germany. (For contact information, see Feb 6.)

Feb 20 — ***TUGboat* Volume 17, 2nd regular issue:** Deadline for receipt of *technical* manuscripts (tentative).

Mar 20 — UK TEX Users Group, University of Warwick. TEX and the Internet. Local organizer, Malcolm Clark. For information, e-mail `uktug-enquiries@ftp.tex.ac.uk`

Mar 27 – 29 — DANTE'96 and 14th general meeting of DANTE e.V., Universität Augsburg, Germany. For information, contact Gerhard Wilhelms (`dante96@Uni-Augsburg.de`).

Apr 2 — DANTE TEX–Stammtisch at the Universität Bremen, Germany. (For contact information, see Feb 6.)

Apr 23 — ***TUGboat* Volume 17, 2nd regular issue:** Deadline for receipt of news items, reports (tentative).

May 2 – 4 — BachoTEX'96: GUST 4th Annual Meeting in Bachotek, Poland, "The World around TEX". For information, contact Jola Szelatyńska (`mjsz@cc.uni.torun.pl`).

May 7 — DANTE TEX–Stammtisch at the Universität Bremen, Germany. (For contact information, see Feb 6.)

May 29 — GUTenberg'96, on TEX distributions, and GUTenberg Annual Meeting, Paris, France.

Jul 18 – 21 — SHARP 1996: Society for the History of Authorship, Reading and Publishing, Fourth Annual Conference, Worcester, Massachusetts. For information, contact the American Antiquarian Society, `cfs@mark.mwa.org`.

Jul 28 – Aug 2 — **TUG 17th Annual Meeting**: "Poly-TEX", Dubna, Russia. For information, send e-mail to `TUG96@pds.jinr.ru`. See the Call for Papers, next page.

Aug 20 — ***TUGboat* Volume 17, No. 4:** Deadline for receipt of *technical* manuscripts (tentative). Theme issue, contributions by invitation.

Sep 23 — PODP, Workshop on Principles of Document Processing, Xerox Palo Alto Research Center, Palo Alto, California. For information, visit the PODP Web page at `http://www.cs.umbc.edu/conferences/podp/`.

Sep 24 – 26 — EP96, the International Conference on Electronic Documents, Document Manipulation and Document Dissemination, Xerox Palo Alto Research Center, Palo Alto, California. *Deadline for submission of papers: 1 April 1996.* For information, contact `ep96@xsoft.xerox.com` or visit `http://www.xsoft.com/XSoft/ep96.html`.

Oct 21 — ***TUGboat* Volume 17, No. 4:** Deadline for receipt of news items, reports (tentative).

For additional information on the events listed above, contact the TUG office (415-982-8449, fax: 415-982-8559, e-mail: `tug@tug.org`) unless otherwise noted.

Status as of 31 December 1995

The 17th Annual TEX Users Group Meeting

ПОЛУТЕХ

July 28 – August 2, 1996
Organization: TUG, *CyrTUG*, JINR

CALL FOR PAPERS

Дорогие друзья!

Вот и настало время, когда TEX-полиглот, уже бойко говорящий на многих языках с латиницей, начинает углублять свои познания и в других алфавитах. Летом 1996 года он собирается посетить Россию и провести TEX-практикум на кириллице.

РОССИЯ ... Страна необъятных просторов, которую населяют такие загадочные русские, в начале века совершившие Революцию, а в конце века — Перестройку. Страна, создавшая блестящую математическую школу, освоившая Космос, покорившая мир своей литературой, музыкой, балетом.

Что же ждет TEXuser'а, решившего принять участие в TUG'96?

В Аэропорту Шереметьево-2 участников конференции встретят на автобусе и доставят в город Дубну, где с 28 июля по 2 августа и будет проходить конференция (подробнее о г. Дубне см. ниже). У кого-то, возможно, возникают такие ассоциации: мороз, тайга, медведи, ... Спешим обрадовать: это центр Европейской части России и до северных льдов отсюда значительно дальше, чем до субтропиков. Сосновый бор, в котором расположился этот уютный городок, похож скорее на парк, чем на непроходимые лесные дебри. Погода же в конце июля обычно жаркая и солнечная: средняя температура +28C°. Гостей ждет комфортабельная гостиница с одноместными и двухместными номерами (горячая вода, душ, телефон, телевизор). Русская кухня отличается изобилием, так что похудеть вряд ли удастся.

Среди культурных мероприятий запланирован пикник на живописном берегу Волги, куда учатников доставят на катере, автобусная экскурсия в г. Сергиев-Посад (центр Русской православной церкви) и в последний день — автобусная экскурсия в Москву, по окончании которой желающих доставят в аэропорт Шереметьево 2.

Dear Friends!

So the time has arrived when TEX, the Polyglot, who has for many years happily "spoken" many different languages written in the Latin alphabet, extends its knowledge in the field of other alphabets. During the Summer of 1996, a visit is planned to Russia where TEX will have its first practical session in Cyrillic.

Russia is that large country, with its enormous spaces, inhabited by those enigmatic Russians, who started the century with a Revolution, and ended it with "Perestroyka". A country who founded a brilliant mathematical school, colonized the Cosmos, and conquered the world with its literature, music, and ballet.

So what awaits the TEX user who plans to attend TUG'96?

At Moscow's international Sheremetevo-2 Airport, conference participants will be met by a member of the organizing committee and escorted by bus to Dubna, where from July 28th to August 2nd the 17th TUG conference will take place (see below for more about Dubna). If, for some reason or another, you envision cold, the taiga, and white bears, then we can reassure you, Dubna is in the European part of Russia, and if you want to see the ice on the nordic oceans, you will have to travel further than you would to reach the subtropics of the Black Sea shore. Indeed, you will soon find that the pine forest in which the comfortable town of Dubna is situated will remind you of a park. In July and August, the weather is mostly warm and sunny, with an average temperature of 28C°. Guests will be housed in a comfortable hotel on the banks of the Volga river in single or double rooms (hot water, shower, telephone, and television). The Russian "cuisine" is characterized by its abundance, so one can forget about slimming.

The social program includes a picnic on the picturesque banks of the Volga, where we will be taken by boat, a bus excursion to Sergiev Posad (the center of the Russian Orthodox Church, where Andrey Slephkhin works — see his article on page 373), and, on the last day a visit to Moscow, following which the participants can be dropped of at the airport to fly home, or at one of the railway stations, if they want to prolong their visit.

Что касается научной программы конференции, то эту информацию мы Вам предоставим, как только получим от вас предложения с аннотациями докладов, заявки на проведение курсов и участие в них, пожелания осветить глубже ту или иную проблему. Адрес конференции: `TUG96@pds.jinr.ru`.

Пишите нам — все ваши предложения впишутся в наш девиз:

The conference's preliminary program will be announced as soon as we receive from you, dear readers, proposals for presentations, courses that you would like to teach or attend, poster sessions, or any problem(s) or subject(s) that interest you. Please send your suggestions to the conference electronic address `TUG96@pds.jinr.ru`.

Write to us — all your proposals will enrich the theme of our conference:

$$\text{ПОЛ}\Upsilon\text{TEX}= \begin{cases} \text{Polytechnic} \\ \text{Polymath} \\ \text{Polyglot} \end{cases}$$

Practical Information

Conference costs

The Conference Committee foresees a cost in the range 550–600 USD. This sum includes the complete cost of the conference, namely the registration fee, lodging (6 nights with six breakfasts, lunches, and dinners), coffee/tea breaks, social events, and transport from Sheremetevo Airport to Dubna. The payment should be made in the following way: a *non-refundable* sum of $100 per person should be transferred to a bank account (to be announced) before June 1st. After receipt of that sum an official invitation, necessary for obtaining a visa (see below), will be faxed to the participant. The rest will be payable in cash upon arrival at the Conference (no credit cards or cheques can be used in Dubna). We hope to arrange bursary funds for support of students and those participants who demonstrate need.

Visas

Most of the visitors from outside Russia will need a visa to attend the Conference. Therefore, for arranging a visa into Russia, participants should inform the Conference Secretariat (Mrs. N. Dokalenko, E-mail `nataly@ypr.jinr.dubna.su`; Fax 7 095 975 2381 or 7 09621 65 891) of their and (possibly) the accompanying person(s)'s full name, date of birth, citizenship, passport number, arrival and departure dates. The Secretariat will forward by fax the visa support message to the participants with which they should apply for visas to the nearest Russian Embassy or Consulate. Please note that you should apply for a visa valid for Dubna, Moscow, and Sergiev Posad.[1]

Transportation

The Organizing Committee will arrange direct transportation by bus from the Sheremetevo-2 Airport to Dubna (130 km north of Moscow). The Secretariat should be informed of the flight number, precise date and time of arrival (Moscow time) *no later than* four working days before a participant wishes to be met at the airport. It is our intention to have each participant met by a member of Organizing Committee. Details will be available later.

Organizing Committee

Кореньков Владимир Васильевич
Vladimir Vasilievitch Korenkov
Дубна, Dubna
`korenkov@cv.jinr.ru`

Маховая Ирина Анатольевна
Irina Anatolievna Makhovaya
Москва, Moscow
`irina@mir.msk.su`

Sebastian Rahtz, Oxford, UK
`s.rahtz@elsevier.co.uk`

Program Committee

Панкратьев Евгений Васильевич
Evgeniy Vasilievitch Pankratiev
Москва, Moscow
`pankrat@shade.msu.ru`

Michel Goossens, Geneva, Switzerland
`goossens@cern.ch`

Mimi Burbank, Florida, USA
`mimi@scri.fsu.edu`

[1] If you plan to visit other cities in Russia you should obtain the relevant documents and join them to the visa application, so that the names of all places to be visited can be entered on the visa form as required.

Deadlines

Submission of abstracts	February 20
Acceptance signified to authors	February 29
Preliminary articles	March 31
Proposals for workshops, demos, poster sessions	April 20
Registration and transfer of a non-refundable sum of \$100 *per person* to a Dubna bank	May 31
Visa supporting information	June 5
Revised articles	June 10
Start of Conference	July 28

Welcome to Dubna!

Dubna was founded in 1956 when the Convention establishing the Joint Institute for Nuclear Research was signed. The town is situated on the picturesque banks of the Volga river and the Moscow sea 120 km to the north of Moscow. One can reach Dubna from Moscow within 2 hours going by car, by bus or by express train. It will take you 1.5 hours to go to Dubna from the Sheremetevo-2 International Airport. Waterways connect Dubna not only to the Russian Volga cities, but also to the waters of the Black, the Caspian, the Baltic and the White seas.

There is no harmful environmental impact from the industrial plants; this together with the large tracts of forest in the environs of Dubna, and the vast water area dotted with small islands, makes the area quite attractive for tourism and rest. The Volga embankment is one of the prettiest parts of the town. In springtime, the streets of Dubna are full of the odour of lilacs, the apple trees are pink-white; in summer, lime trees, maples, birch trees and poplars make the town seem totally green; in autumn the town is all golden excepting the evergreen of old pine trees. The town's modern look harmonizes with the quietness of the surrounding forest. The town was built in the midst of a forest. There are separate patches of trees in the town itself, and the town park is just a part of the forest. It takes just a few minutes to get to the forest from the shopping centre on foot. A few minutes' walk and you are outside the city limits!

Small as it is, Dubna is a real metropolis. It is a scientific metropolis. It is a *big little city*, as a visiting American scientist called it many years ago. Since the foundation of the Joint Institute for Nuclear Research (JINR), the name of Dubna has constantly been in the pages of the world's newspapers and journals. Dubna is one of the world centres for fundamental research in nuclear physics. The Joint Institute plays an important role as a coordinator of investigations of the scientists from 18 JINR member-state institutes. Wide international scientific and technical cooperation is one of the fundamental concepts of the JINR.

Dubna is indeed a town of international friendship. Foreign speech can be heard everywhere. But the words, no matter in which language they are pronounced, are clear to everybody: friendly cooperation and fraternity unite all the physicists and mathematicians living and working in Dubna into an international scientific community.

On October 3, 1994, Dubna opened its doors to its first university, "International University of Dubna: Nature, Society and Man". The university is composed of five "cathedra" or faculties, including socioeconomic sciences, ecology and earth science, computer education, linguistics, and health and physical education. Two more faculties — law and government and technology — are also being contemplated.

The town has great experience in holding international conferences, and exchanges of delegations between countries in the sphere of science, education and culture. Dubna and La Crosse, Wisconsin, USA, are sister cities. Dubna is famous for its hospitality. Famous scientists, public figures and statesmen from different countries visit Dubna. They are always impressed by the gracious welcome they receive in Dubna and warm generosity which Dubna residents demonstrate.

A few words on Moscow

Moscow, the capital of Russia, has a population of some nine million people. It is a city rich in cultural, architectural and historical monuments, and, at the same time, boasts a rapidly developing modern urban community with brand new blocks of flats, long, straight and broad avenues, parks, gardens, stadiums, schools, cinemas, department stores, recreation centres, bridges and highways. Though forward-looking, it cherishes the memory of its past, and its old sections lend it a special charm.

See page 352 for views of some local scenery!

Look for current information on the WWW at

`http://www.scri.fsu.edu/~mimi/tug96/tug96.html`

Attending EuroTEX'95 in Papendal

Michel Goossens

I arrived with the night train from Basel at 8:29 precisely in Arnhem Railway station on Monday morning September 4th. I decided to take a cab to Papendal, the Sports Center a few km outside of the town, where I arrived just before nine. In no time I was registered and given two keys to my room, where, as expected, I would spend only a minimal amount of time during the four next nights...

Indeed, while walking the fifty meters or so to the breakfast area, I met many known faces, and I had to promise each one of them that we would have a chat later during the conference.

A Truly International Gathering

EuroTEX conferences are quite different from the TUG conferences, in that many more participants from Russia and Central Europe participate, and this year was no exception. In all 103 TEX users from 18 different countries (4 from Belgium, 1 from Canada, 7 from the Czech Republic, 2 from France, 17 from Germany, 1 from Hungary, 2 from Lithuania, 18 from the Netherlands, 1 from Norway, 20 from Poland, 15 from Russia, 1 from South Africa, 2 from Spain, 1 from Sweden, 3 from Switzerland, 1 from Turkey, 6 from the United Kingdom, and 1 from the United States of America) came to EuroTEX'95, where our hosts, the Dutch-speaking TEX Users Group NTG, had prepared a nice, interesting, and extremely dense program of TEX and text-processing related presentations given by contributors who are active in the various areas mentioned.

After breakfast the members of the Program Committee, Johannes Braams, Chris Rowley and myself, met to make final adjustments to the schedule of the various talks and tutorials, taking into account the availability of the speakers, so that the program could be communicated to the participants of the conference, who now started to arrive.

The EuroTEX Bus

But before going any further let me explain how such a large group from Russia and Central Europe was able to come to Papendal. It was mainly thanks to the generous contributions of the NTG, who donated a sizable sum of money they had collected by selling the 4AllTEX CD-ROM(s), plus over $1800 from the book auction that took place at TUG'95 in St. Petersburg Beach (Florida, USA) that a Polish bus was hired. In Warsaw the Russian, Lithuanian and part of the Polish group boarded, while other (Polish and Czech) participants were picked up later on the way to Papendal. In this way the travel expenses of about forty people were kept to a minimum. This great initiative, which, I hope, will be repeated in the future, shows the enormous solidarity in the TEX community. It can serve as a proof that TEX is not only a (or "the") utility to compose beautiful documents in most of the languages written by mankind, but it also brings people of various nations, cultures, and backgrounds together, and thus fosters communication, which is the basis of all human success.

The Conference Starts

A few minutes before 2 p.m., the conference was formally opened by Erik Frambach, the President of NTG. He welcomed all participants, and then passed things over to me. In my role as President of TUG, I congratulated NTG for the magnificent work they had done for allowing so many participants from different parts of Europe to take part in the EuroTEX meeting, and I underlined that TUG fully supports all such initiatives and is doing its best to help all TEX users in the world. I therefore invited everybody who wanted to express constructive ideas about how to develop or improve the actions of TUG in the area of supporting the international community better, to find me during the week for a discussion. Finally Johannes Braams, on behalf of the Program Committee, explained the few changes that had to be introduced in the program in order to accommodate a maximum of the wishes of speakers and participants, and he gave a few details on logistics for the rest of the week.

Monday p.m. — Font Developments

Thus, well informed, the conference could start with the talk on VFComb, a program for designing virtual fonts by Alexander Berdnikov and Sergey Tirtia from St. Peterburg (Russia). The main aim of their MS-DOS program is to facilitate setting up virtual fonts for use with CM and the Cyrillic LL-fonts. This program is especially interesting in that it constructs font and user-defined ligature tables extracted from various fonts and combines it with metric information from various `tfm`-files. The VFComp program not only supports the full syntax of `.pl` and `.vpl` files, but also adds some symbolic variables and conditional operators to ease the production and debugging of virtual fonts.

The next speaker, Jörg Knappen of Mainz (Germany), described his work on version 1.2 of the DC fonts and the text companion symbol fonts. He first gave an overview of the improvements introduced into version 1.2 with respect to the previous release

in order to cope with criticism in the areas of the placement of accents, the design of the quotation marks, plus the outlines for various Polish, Czech, Slovak diacritics, and the height of the umlauts. The *text companion* TC fonts try to address the problem of more clearly separating the symbols present in the CM math fonts from the really mathematical characters, and to make uniformly available to TEX users several custom signs (currencies, trademarks, some arrows, a musical note, etc.) that are present in the ISO standards 8859-1, 8859-2 and 6937. For practical reasons Jörg proposes to put all those characters into two different fonts, TS1 and TSA. Last but not least, the speaker underlined the importance of design size in fonts, since the linear scaling of a font over a large range gives the wrong results. He therefore proposed encoding the design size of the font in the font name in each case by using the trailing four digits of the name to represent the design in points (multiplied by a factor of 100). He therefore has to rename a few of the present DC font names, since one has only two characters left for weight and shape. For instance the present `dcssi10` would become `dcsi1000`, and `dcbxti10` is `dcbi1000` in the new scheme. Thanks to Jörg's work, and the work done earlier by the LATEX3 team on math font encodings, it is now hoped that a proposal for the two math fonts will be reached soon, so that by next summer the DC fonts will be promoted to the EC fonts, and be accompanied by a useful and standardized set of 256-character math fonts.

Just before the tea break, Jiří Zlatuška from Brno (Czech Republic) gave a talk on how to use METAFONT and TEX together for typesetting text and graphics.[1] By using TEX's extended ligature mechanism, label placement on diagrams generated by METAFONT, as well as the generation of curvilinear texts, can be accomplished in one METAFONT pass and only requires a simple TEX interface. Institutional seals and other logos can thus be conveniently generated.

Multiple Languages

After the break Andrey Slepukhin (Sergiev Posad, Russia) introduced the audience to the magnificent world of old Church Slavonic typesetting.[2] Andrey works for the Holy Trinity St. Sergius Lavra publishing house, printers of Bibles and other texts in Church Slavonic. For his project of printing such texts he has to develop high quality fonts, tools for simplifying text input, and put together hyphenation tables. Church Slavonic has a lot of characters and they have very few common parts, so that designing them takes a lot of time. The main problem, however, is that almost every word in Church Slavonic texts has a diacritical mark. Since the placement of the diacritical mark is not fixed, and the use of the `\accent` primitive has a lot of unwanted side-effects, Andrey developed the various letter-accent combinations as separate glyphs. By carefully studying Church Slavonic texts it was found that some letter-accent combination could only occur in certain locations in a word. In this way, with the help of a set of special macros and using active characters, the number of needed symbols could be kept below 256, the limit of symbols in a single font. Other problems he had to solve were coloring symbols (a practice occurring extremely frequently in such texts), numbering (numbers are represented by letter combinations), and encoding (several different encodings are actively used in the Cyrillic world). To help solve the last problem he proposes using symbolic names to map to character codes and has implemented these ideas in the latest version of his package. He is also working on Type1 variants of his fonts, and on extending his character set to include older Slavonic letters, initial caps, and a special font for headings. Finally, he has expressed a wish that at some time in the not too distant future it might be possible to typeset a multilingual Bible with parallel texts in Church Slavonic, Greek, Latin, Hebrew, and any other language.

Olga Lapko of Mir Publishers (Moscow, Russia) then presented work she has been doing together with Irina Makhovaya and other collaborators of CyrTUG, on developing a Russian style for the `babel` system that implements in a user-friendly way the typographics and national peculiarities of the Russian language, such as correct names for some math operators, repeating signs in broken math formulae, and Russian-character enumeration lists. Several font encodings are supported and are interfaced to 256-glyph Russian fonts.[3]

In the next talk Johannes Braams reviewed his work on the `babel` system. The present release 3.5 has a completely rewritten interface to deal with shortcuts, introduces new ways to switch languages, has been made compatible with LATEX 2_ε's input and font encoding packages, has added support for a configuration file and many more different languages,

[1] He presented this talk also at TUG'95, see *TUGboat* **16** (3), p. 223–228, 1995.

[2] His article is published in both Russian and English in this issue of *TUGboat*, p. 373.

[3] Their article is published in both Russian and English in this issue of *TUGboat*, p. 364.

and offers an extended syntax for specifying hyphenation patterns.

The final talk of the afternoon was by Petr Sojka from Brno (Czech Republic), who presented his work on hyphenation for compound words. His paper was chosen at TUG'95[4] as the best technical paper by Knuth for pointing out problems with hyphenation that he himself had not considered. The problem is with long compound words that occur frequently in German, Dutch and Slavic languages. In these languages constituent word parts are not signaled by a hyphen or other fill character, thus making it difficult to find the correct hyphenation in some cases. Petr proposed extensions to the hyphenation algorithm and primitives which would make it possible to deal with these cases. Perhaps an interesting suggestion to be implemented in e-TEX or Ω?

After dinner Sebastian Rahtz headed a workshop on Acrobat and electronic publishing. He presented tools that take advantage of LATEX to generate hypertext views of a document. In his presentation he made it clear that LATEX, HTML, and PDF should be viewed as complementary representations, that each has its advantages and is suited for addressing the specific needs of some applications.

Tuesday a.m. — Graphics and Packages

The Tuesday morning started with a presentation by Andrey Astrelin of Moscow (Russia), who talked about a new implementation of graphics in TEX. Of the three basic ways of introducing graphics inside TEX (special fonts, using rules, or exploiting the `\special` command) his implementation uses the approach of extending TEX's graphics capabilities by writing graphics commands via `\special` commands into the `.dvi` file, which is then postprocessed to realize the required functionality. Each command is represented by a character and zero or more numeric arguments. There are general graphics commands, complemented by path and area commands. The `.dvi` file is processed by a special MS-DOS program, where the graphics `\special` commands are replaced by emTEX `\specials` pointing to `.pcx` pictures. These facilities are complemented by macros to place text and graphics on the page and a library of graphics macros that is still under development. It is planned that future releases will offer support for generating `.pk` fonts and PostScript output.

TEX Plotter, a program to create two- and three-dimensional pictures developed by Alexander Berdnikov and Sergey Turtia of St. Petersburg (Russia), was presented next. The program is written in Pascal for MS-DOS, and plots functions depending on two variables. It allows one to obtain equiline and surface representations of complex functions without running into overflow problems with TEX's memory. The program has a menu-driven user interface which allows the creation and viewing of pictures in a variety of forms, including LATEX's `picture` environment, the `epic`/`eepic` package commands or emTEX `\specials`. Future releases will also support `mfpic`, PICTEX and EPS files.

Gabriel Valiente Feruglio of Palma de Mallorca (Spain) then gave a detailed overview of the various commutative diagram packages that are available. He compared eleven of them in terms of their ability to generate complex diagrams, quality of documentation and generated output, ease of installation, user interface, resource requirements, and portability. The way a diagram is structured to increase readability is a somewhat subjective matter, often related to points of aesthetics, so that it is difficult to say which package is the best in applications. In his estimation, the familiarity of an author with the syntax of one package is often more important than the intrinsic power of other packages.

After coffee Hans Hagen of Pragma (Zwolle, the Netherlands) presented a paper on typesetting chemical formulae with the PPCHTEX system he wrote together with A.F. Otten.[5] The system is based on PICTEX. Basic structures for often used chemical structure diagrams are available as macros, and can be combined and "dressed" with radicals in various way to obtain complex formulae. The syntax is straightforward and logical. It also allows typesetting reaction equations easily and provides for several special features, like fine-tuning the position of radicals, placing small formulae inside running text, and optimizing the depth of subscripts.

The last talk of the morning was by Daniel Taupin (Orsay, France) who introduced MusiXTEX, an improved version of his earlier MusicTEX system for typesetting polyphonic music with TEX. MusiXTEX was developed together with Werner Icking and Andreas Egler. It uses three passes (MusicTEX was a one-pass system), where during the second pass a program `muflex` is used to compute optimal spacing to position the notes and determine the length

[4] His article is published in *TUGboat* **16** (3), p. 302–305, 1995.

[5] Their article will be published in an upcoming issue of *TUGboat*. Their approach can be compared to the one of XyMTEX described in *TUGboat* **16** (1), p. 81 – 88.

for slurs and ties. This makes for more aesthetically pleasing scores. A planned future improvement is cleaner lyrics insertions. Some issues remain between Andreas Egler and the other two developers about future developments and the compatibility between MusicTEX and MusiXTEX.

Tuesday p.m. — Electronic Documents

The theme of the Tuesday afternoon was electronic documents, and Wiegert Tierie of Adobe Systems Benelux (Amsterdam, the Netherlands) started off with a detailed overview of Abode's Acrobat series of utilities to create, use, store, annotate, send, view and print electronic documents. To do this Adobe developed the "Portable Document Format" (PDF), a language that is based on PostScript and includes commands to implement a hypertext functionality. He introduced the (freely available) Reader — for reading PDF documents —, Exchange — for introducing annotations —, Writer — for directly creating PDF files from applications —, Distiller — for translating PostScript documents into PDF —, Catalog — for creating full-text indexes —, and finally Search — for full-text searches. Via Application Programming Interfaces (API's) developers can easily plug in their own extensions to take full advantage of the capabilities of the Acrobat system, thus allowing an optimal integration of Acrobat into custom products.

Hans Hagen of Pragma (Zwolle, the Netherlands) then told us about his experience using TEX to generate hypertext documents in PDF. Hans expects that in a few years PDF will be as stable and standard as TEX is today, so that at that time PDF might well become an important format for distribution of documents. Because PDF is based on PostScript it can combine high typographic quality with hypertext capabilities offered via PDF's `pdfmark` operator. Building a utility able to exploit both TEX's and PDF's strong points relies on communicating information from TEX to PDF via `\special` commands that will contain `pdfmark` instructions. Hans emphasized, however, that writing hypertext documents for viewing on a screen requires a somewhat different approach from optimizing them for paper. He presented a set of points that have to be dealt with, in particular different aspect ratios, a more complex pagebody to optimally place navigation aids, synchronization of screen and printed versions of documents, the design of the typographic interface, the generation of tables of contents, multiple indexes, cross-references. Other points to be considered are how to handle multiple documents, shared data, status bars, active figures, color, and version control. The speaker thought that TEX was an ideal tool to explore the advantages and limits of electronic documents and that TEX and Acrobat thus formed a nearly perfect match, combining the advantages of both a flexible programming language and a powerful document delivery system.

After the tea break Sebastian Rahtz (Elsevier Science, Oxford, United Kingdom) talked about the experience he had gained working on the `hyperref` package, developed together with Yannis Haralambous of Lille (France). This LATEX package uses LATEX's cross-reference commands to transmit the necessary information to PDF and thus provides a rather straightforward and effortless way for LATEX users to transform their documents into hypertext. Sebastian pointed out some difficulties of this approach, namely the successful handling of fonts, and the generation of back-references from the bibliography and index. Although it thus is easy to generate PDF from LATEX documents in a way that they are faithful representations of each other, all problems of optimizing the presentation for screen viewing, as explained by Hagen in the previous talk, still remain to be addressed.

During the next half hour or so I gave an introduction to SGML, using HTML as an example of a DTD. I emphasized that using and understanding SGML is straightforward by using examples drawn from HTML. I outlined the structure of a DTD and how it describes a document, its elements, their attributes and the entities that are available. The mathematics and table extensions of HTML3 were presented briefly, but it was emphasized that HTML's main aim is describing the document's structure, not its perfect typesetting, where it is better to consider the PDF solution, so that HTML and PDF are complementary rather than competing technologies.

My second talk presented the LATEX2HTML tool of Nikos Drakos, which translates LATEX documents into HTML using a perl program. All standard LATEX commands are dealt with. Tabular and mathematics constructs are turned into images since they cannot yet be treated by HTML, and the same strategy is applied to user-defined commands or environments. I emphasized that with a little work by the user such user-defined extensions can be dealt with by providing a perl routine for the extension in question, thus increasing substantially the usabilitity of the generated HTML document, decreasing at the same time its size and fragmentation. The hypertext extension package `html` was described and it was shown how it allows the introduction of hypertext elements into a LATEX document, so that it can

be optimized both for printing and hypertext viewing on screen. Finally, work on providing support for the automatic generation of HTML3 and the implementation of HTML3 capabilities in the experimental `arena` browser were briefly mentioned.

The afternoon was concluded by a panel discussion on electronic documents, chaired by Joachim Schrod of Darmstadt (Germany), with all speakers of the afternoon as panel members. Each of the speakers again emphasized the complementarity of the various approaches to enrich the LaTeX source code with markup that can be used to exploit the hypertext capabilities of the various output media targeted. However, hypertext viewing on a computer screen is quite different from writing books, and therefore care must be taken to design and mark up documents with enough generality to optimize their re-use in the various circumstances. Another interesting question which was raised was the level of control that should be given to the viewer to customize the appearance of the viewed document. Should one allow complete freedom to the receiver to represent the document (font type and size, setting of colors, preferences for lists, page size, etc.) or should the author be able to freeze some or all of the document's appearance (for instance for legal documents). It became soon evident that a general answer to this question was impossible, and that as much flexibility as needed should be given to the receiver. As the hypertext medium, and the freedom to "view as one likes" are still quite new, the plethora of browsers, and possibilities to (over)dress one's documents will soon die out as the technology matures, and some studies in readability and efficiency in communication information via the screen on the Internet come up with some guidelines that will eliminate the more extreme excesses in the area, just as fontitis, or the disease to use as many fonts a possible in one's documents, died out after a few years after people realized that the documents became unreadable.

After dinner a long presentation/discussion session on the achievements of the e-TeX project was organized by the e-TeX team, with Phil Taylor, Jiří Zlatuška, Bernd Raichle, and Friedholm Sowa, who came specially for that day, present. It was an extremely useful discussion, that lasted until midnight, and I am sure that both the members of the e-TeX team and the participants benefited greatly from the exchange of ideas.

Wednesday a.m. — Tools I

The tools section started on Wednesday morning with the presentation by Andries Lenstra (Uitgeest, the Netherlands) of the $\delta\alpha$TeX package that he developed with his colleagues Seven Kliffen and Ruud Koning. The $\delta\alpha$TeX system is a series of TeX macros for storing and retrieving data that work both with LaTeX and plain TeX. The system makes it possible to keep source texts short and guarantees data integrity and uniformity in typesetting. Data are stored in $\delta\alpha$TeX format in a $\delta\alpha$T file. The format is simple, versatile, easy to learn, and as portable as a TeX document. Conversion utilities to generate $\delta\alpha$ from ASCII files are available. $\delta\alpha$TeX has sorting-out facilities, supports default fields, conditionals, and wrapping.

Kees van der Laan (Garnwerd, the Netherlands) explained how to use his BLUe's format to manage a database. In particular, TeXnical details concerning storage and access of the material are hidden from the user, and since the database code is written in plain TeX, it can in fact be used with any TeX format. High-level user commands allows non-TeXnical users to take full advantage of this tool, which moreover allows one to include pictures and cross-references. Facilities for preparing letters to be sent to multiple addressees are also included.

Philip Taylor (London, United Kingdom) then gave one of his famous pedagogical presentations, explaining how one can use `\csname`...`\endcsname` constructs advantageously in various often-occurring programming paradigms in TeX. He showed how this construct helps to clarify the programming logic in many TeX macros, and thus makes documenting and maintaining such macros more straightforward and more robust. As an interesting side-remark he mentioned the fact that an undefined `\csname`-construct has the value `\relax` can be a disadvantage in some cases. All in all a well-prepared, stimulating, and thought-out talk, for which Phil rightly got the prize for the most pedagogical presentation of the conference. Congratulations Phil!

Just after the coffee break Laurent Siebenmann (Paris, France) introduced his TeX utility *Occam* that is useful for managing macros. The *Occam* utility eliminates from a set of TeX macros all those that are not referenced (and thus not necessary), thereby reducing (sometimes significantly) the size of a TeX source document, at the same time making the maintenance and comprehension of the supporting macro collection easier.

The final talk of the morning was a presentation of *pascal*, a TeX macro package for typesetting Pascal programs, based on work by Pedro Palao Gostanza and Manual Núñez García (Madrid, Spain), and presented by the former. This utility allows the user to clearly indicate the structure of a

program by using special typesetting conventions. Technically speaking, all reserved words and identifiers become TEX control sequences, so that no external parser is necessary and layout conventions can be easily enforced. Extensions of these ideas to other structural programming languages such as Modula-2, Ada, or ML should be straightforward.

Wednesday p.m. — General Developments in TEX and LATEX

The afternoon was reserved for presentations related to recent general developments in TEX and LATEX. Chris Rowley (Open University, United Kingdom), as a representative of the LATEX3 Team, gave an overview of the activities of the team. Having devoted the Group's resources over the past two years on the development and consolidation of LATEX 2_ε, the Group will now concentrate on further studies on the way to LATEX3. During his talk Chris also explained the Group's policy on modifying files in the LATEX 2_ε distribution.

Next Philip Taylor, speaking for the NTS and ε-TEX teams, announced that version 1 of ε-TEX was now available, and gave a list of the extensions which have been introduced. He emphasized that ε-TEX can be used in a 100% compatibility mode with TEX and he expressed the hope that with time and after a lot of tests, most TEX users will have enough confidence to use ε-TEX as a valid replacement for TEX. In fact, when generating the format file with `initex`, one can configure ε-TEX so that it is completely compatible with TEX, and the extensions and enhancements are disallowed. The new features implemented in this first version are in the areas of additional control over expansions, re-scanning tokens, environmental enquiries, additional marks and debugging facilities, bi-directional typesetting, and a few supplementary primitives.

John Plaice (Laval University, Ste-Foy, Québec, Canada) of the Ω team then gave an update of work done since the summer of 1994 on this 16-bit extension to TEX. In particular, he described the additional collaboration with a graduate student in Estonia in the area of generalized multi-dimensional typesetting, and recent progress in the area of Arabic. A beta-version now runs on a PC. Dynamic memory allocation is next on the list of features to be implemented in order to reduce the memory requirements of the system when static arrays are used.

It was then Joachim Schrod's turn to introduce the audience to the proposals of the "tds" (TEX Directory Structure) team, whose members have been working for many months on the definition of an ISO 9660-compliant directory structure that can be used as a plug-and-play run-time system by all operating systems.[6]

A panel discussion on the theme *TEX quo vadis?* concluded the afternoon. Several members in audience thanked the various teams who are working so hard to improve the functionality of TEX, LATEX, and their user production environments and gave valuable input to the respective development teams, who promised to look at ways of implementing the suggestions put forward.

Wednesday Evening — The Social Event

At 16:30 two buses took all participants to the center of Arnhem, where we had two hours for a walk through the streets of this historic town, perhaps best known from the film "A bridge too far", which described the battle for the bridges over the Rhine in the vicinity of Arnhem. The inner city, which was almost completely devastated in 1944, has since been rebuilt, and has become a modern local center where the importance of the river Rhine is omnipresent. It was thus also appropriate that our whole company boarded the ship "MPS Graaf van Bylant", named after a wealthy count who in the 18th century owned large parts in the neighbourhood of Arnhem. It was already getting dark when we left the embankment, and set off for a three-hour tour of the Rhine and the IJssel, taking us along the quays with its wharfs, warehouses, and the apartment building in the background, with their thousands of little lighted windows, each corresponding to a little cell of present-day society, people eating, drinking, talking, watching television or just enjoying each other's company: yellow-orange rectangular reflections of human joy and grief, micro-windows on the life of a modern city. But I am sure that these metaphysical considerations were absent from the heads of most EuroTEXers that evening, since our NTG hosts did their utmost best to make the outing as enjoyable as possible. Apart from the excellent food and drinks (including quite a few bottles of Polish and Russian Vodka, contributed by GUST and CyrTUG), there were also the MAPS awards for the best papers in the proceedings. They went to Gabriel Feruglio for his overview article of commutative diagrams and Bogusław Jackowski for his beautiful article on the use of EPS and METAFONT (this was, in fact, already his second MAPS award!). On top of that the MAPS editing team decided to award two "special" prizes and it was quite a nice surprise that MAPS chose to

[6] More about tds can be found in this issue of *TUGboat*, p. 401.

honor the "*TUGboat* Production Team", for their efforts to publish *TUGboat* on time again. With great pleasure Barbara Beeton, Sebastian Rahtz and I myself received the now-traditional MAPS-sweetscone (a collections of 256 sweets of many different colors, with the necessary "glue" and all other necessary attributes...). And I found it very appropriate that MAPS gave also a special prize to Erik Frambach, who had been doing an enormous amount of visible (and even more invisible) work to make the conference the success it has become. Apart from these formal parts of the evening, there was of course ample occasion for direct person-to-person discussions and other forms of socializing, making contacts, making new acquaintances and friends, and around midnight, when we arrived back at the Papendal Sport Center, we could all go to bed trying to assimilate all those new and enriching impressions and contacts of this memorable Wednesday Evening EuroTEX'95 cruise.

Thursday a.m. — Tools II

The last formal session of the conference on the Thursday morning started with a talk by Philip Taylor who tried to convince the audience that TEX is quite an unsuitable language for marking up documents. He came to this conclusion after he had to deal with the production of a book in the field of linguistics, and had to communicate with an author who did not know TEX. Phil therefore developed a syntax, `<ATML>`, for "A TEX Markup Language", where all markup is enclosed between the triangular brackets < and >, thus disallowing all direct TEX markup. Phil explained the precautions that he had to take to implement this scheme, and discussed the advantages and drawbacks. It seemed clear that for non-TEX-aware authors this approach can certainly maximize the efficiency for preparing documents where TEX should be used as the typesetting engine, but where otherwise consistency and structural markup can be checked at a level of `<ATML>`.

Similar ideas were supported by Antonin Strejc (Prague, Czech Republic) who described the W95 environment, where contributors to a conference can use an MS-DOS based authoring system, which is a menu-driven interface to LATEX. The authors can specify the information needed for preparing their article (title, author, affiliation, abstract, etc.) via these menus, and also have the possibility of entering other text elements, like lists, equations, tables, and figures. Thanks to this approach, which guarantees consistent and correct markup, the editors of the proceedings were able to typeset almost 1000 pages by over 430 contributors in about two weeks. The speaker emphasized that such a system needs close cooperation between the organizing committee and the typesetter, and it presupposes that the instructions have been announced to the contributors about six month before the conference. To minimize the need for direct hotline-type support (it was found that with the W95 system only 1–2% of the authors needed personal help) a sufficiently self-documenting help facility should be provided.

Various ways of improving the archiving of scientific documents by optimizing the ways external material can be included by standardized keywords for TEX's `\special` commands were discussed by Laurent Siebenmann, the next speaker. Although his ideas were quite interesting, it was not completely evident that his demands were not already dealt with by the work of the `dvi`-standard committee, that had a few meetings at TUG'95 in St. Petersburg, and Laurent was invited to contact the members of that working group. The speaker also said a few words about the effort concerning atomic fonts...

After the coffee break Kees van der Laan showed how one can implement indexing in a one-pass TEX run. Although only moderate indexes can be dealt with in this way, the approach is nevertheless quite useful for smaller documents, or for proofing indexes on a chapter-by-chapter basis. For the markup Kees followed closely the circumflex (^) notation proposed by Knuth in the TEXbook. Sorting is possible and several methods of ordering (lowercase/uppercase, accented/non-accented/word ordering) are available. This indexing system is part of the BLUeTEX system.

Stanislav Brabec (Prague, Czech Republic) discussed his typesetting system, based on plain TEX. He uses TEX's powerful programming facilities to generate macros for managing references, contents tables, defining page layout, with a flexible facility for adding cropmarks, specifying margins, preparation of booklets, and typesetting in landscape mode. His systems `upages.tex` allows one to interpret the input stream on a token-by-token basis, is able to prepare output for PostScript devices, with support for rotation and scaling; it generates device-independent color, with provision for color signatures and separations, and has primitives for line-drawings.

The last formal presentation of the conference was by Bogusław Jackowski (BOP, Gdańsk, Poland), who presented his METAFONT-EPS interface.[7] The heart of this interface is the `MFTOEPS` METAFONT package which provides the necessary definitions for

[7] His article is published in this issue of *TUGboat*, p. 388.

translating the descriptions of graphics objects from METAFONT to PostScript. The PostScript code is written to a log file, that can be post-processed to generate EPS files that can be read by popular PostScript drawing tools like Adobe Illustrator, CoralDraw, or Fontographer. The package comes with two such utilities, one written in awk, the other, more general, but slower, in TEX. This system is thus merely useful for generating PostScript code that should be edited further by the tools mentioned above. For generating end-user PostScript, John Hobby's `METAPOST` program is probably more suited. The author is also working on a program `PSTOMF`, that would translate EPS files (via the `ghostview` program) into METAFONT, thus completing the link between METAFONT and PostScript and allowing everybody to make use of the advantages of both languages.

It was at about half past twelve when Johannes Braams, in the name of the EuroTEX'95 Program Committee, thanked all speakers for their contributions, and Wietse Dol for his work on preparing the Proceedings.[8] He then invited Andrey Slepukhin to step forward to receive the prize for "Best Presentation of EuroTEX'95", not only for his seminal work in the area of font creation of beautiful Church Slavonic letters, but also for developing solutions for the many technical problems relating to typesetting complex documents. A prize for the most pedagogical presentations (not only at this conference, but also on other occasions) was given to Philip Taylor. Sincere congratulations to both recipients of these prizes.

I then, in the name of the international TEX Organization, TUG, thanked the NTG organizers for their dedication and hard work in making EuroTEX'95 possible. I also thanked the many contributors, and last but not least the participants to the conference, and I invited everybody to the joint TUG-EuroTEX meeting that will take place next year 1996 in Dubna (Russia) from Sunday July 28th to Thursday August 2nd, 1996. This will be the first time a conference will take place in a country whose major language does not use the Latin alphabet. A unique occasion to remain up-to-date on what is happening in the world of TEX and friends, and to become acquainted with the rich Russian culture thanks to direct contact with Russian people and visits to famous monuments.

[8] Copies of the EuroTEX'95 Proceedings — counting more than 440 pages — are available at the price of 50 Dutch Guilders (postage included) from NTG, P.O. Box 394, NL-1740 AJ Schagen, The Netherlands.

It was NTG's President, Erik Frambach, who had the honor of having the last word. He gave a complete list of the people who had spent many months since the beginning of 1995 to make the conference a success: yes it takes many people to do all the little (and not-so-little) tasks to have everything ready on time. Then he announced that the first 4AllTEX prize had been awarded to Eberhard Mattes, the father of `emtex`, the TEX engine that lies without doubt at the heart of most TEX installations in the world. A cheque of 3,141.59 German Marks will be sent to Eberhard, and the applause that accompanied the announcement underlined how much all TEX users worldwide, especially those using PC's, appreciate his extremely valuable and continuous contributions. With this Erik formally closed the conference in a grandiose way, and hoped to see everybody again next year on the Volga in Dubna.

Thursday p.m. and Friday a.m. — Tutorials

The Thursday afternoon two tutorials ran in parallel, namely one by Bogusław Jackowski on METAFONT and another by Piet van Oostrum on page layout with LATEX. Both teachers gracefully agreed to repeat their respective tutorials the next morning to give a maximum number of participants the chance to attend both and to give those who had to leave already the Thursday evening the chance to attend the one of their choice. As I had meetings scheduled with various people during the time of the tutorials I was unable to go to either of them, but listening to those fortunate enough to have attended I learned that they were both an enormous success. Congratulations to both Bogusław and Piet for their careful preparations and pedagogical presentations.

Finally, on the Friday afternoon, those interested were able to join our friends from Central and Eastern Europe in a visit of Amsterdam with the EuroTEX bus, a second (and especially appreciated) "social event".

The Saturday morning with the departure of the EuroTEX bus for Warsaw via Berlin, the last EuroTEXies left the Papendal Sports Center. Thus EuroTEX'95 became history and will henceforth be remembered as the latest (in fact the ninth) in the series of European TEX Conferences. Thank you NTG, thank you Erik, and Wietse, and Gerard, and all your colleagues for that week of intense TEX happiness, and see you all next July in Russia!

◇ Michel Goossens
CERN, Geneva, Switzerland
Email: `goossens@cern.ch`

Late-Breaking News

Production Notes

Mimi Burbank and Michel Goossens

Production Notes

Well, for those of you who are relative novices at some of the more arcane uses of TEX, I'd love to spend some time explaining some of the "on-the-job training" I get with each issue produced by the production team.

First of all, I must say that I have no idea how one person *ever* did the job alone! Now that the technology has become so advanced, and the world of TEX and METAFONT has grown so much, it has become a delight to look at output. This production team, though there are still administrative obstacles to be worked out, is a dynamic group, and TEXnical problems are solved rather easily and quickly. The greatest source of frustration now seems to be networks, and devices.

This issue heralds in a new era in typesetting *TUGboat* and the first truly bilingual issue. For a more detailed description of how the two Russian articles were prepared, see Michel's production notes at the end of Andrey Slepukhin's article on page 379.

All files were received electronically. This issue is chock full of T1 encoding, use of the new (and old) `dc` fonts, multiple uses of METAFONT, considerable use of Babel, and an additional two Russian formats. Over 70 files were used to produce camera copy for Slepukhin's article, while only 13 files were needed to produce camera copy for Lapko and Makhovaya's article and the TUG'96 announcement on page 429. Some 50 files were necessary to run Jackowski's article, 29 of which were `.eps` figures.

Macros

80% of the articles received were in LATEX 2_ε. While some articles were received tagged for another or for no publication, most articles were received as fully tagged for *TUGboat*, using either `plain`-based or LATEX conventions described in the Authors' Guide (see *TUGboat* 10, no. 3, pages 378–385). Several macros defined by authors or from LATEX packages clashed with those of `ltugboat.cls` (for instance `multicol.sty` and `ltugboat`'s use of the `twocolumn` option), and some time was spent trying to get them to "cohabit" in the same file.

Fonts

Most of this issue has been set in Computer Modern (or DC, version 1.1) fonts — in Malyshev's BaKoMa PostScript Type 1 versions. Exceptions are, obviously, the Russian articles by Slepukhin, and Lapko and Makhovaya. Knappen's article was typeset using the T1 encoding of Computer Modern fonts, in addition to the DC, version 1.2 fonts.

Output

The final camera copy was prepared at SCRI on an IBM RS6000 running AIX, using the *Web2C* implementation of TEX. Output was printed on a QMS 680 print system at 600 dpi.

Future Issues

The next issue will contain *TTN* material, as well as abstracts from other LUG publications, a summary of the CyrTUG'95 meeting, and articles on math in LATEX, the first of a two-part tutorial on typography by Phil Taylor, and articles on graphics and METAFONT.

Plans are underway for *TUGboat* 17(2) to be the Proceedings Issue for 1996, so that copies will be available for the meeting in Dubna, Russia. This will be a change in the usual order of issues; 17(3) will be a regular issue, and 17(4) will be a multilingual "theme" issue, with a guest editor. Suggestions are welcome for prospective topics and guest editors. Send them to the Editor, Barbara Beeton (see address on page 351), or via electronic mail to `TUGboat@ams.org`.

◇ Mimi Burbank
SCRI, Florida State University,
Tallahassee, FL 32306–4052
Email: `mimi@scri.fsu.edu`

◇ Michel Goossens
CERN, Geneva, Switzerland
Email: `goossens@cern.ch`

TUG Business

1996 TEX Users Group Election

Barbara Beeton
for the Elections Committee

The terms of 6 members of the Board of Directors will expire as of the 1996 Annual Business Meeting, which will take place in conjunction with the 17th Annual TUG Meeting in July 1996 in Dubna, Russia. The directors whose terms expire in 1996 are Mimi Burbank, Michael Ferguson, Peter Flynn, Mimi Jett, Tom Rokicki and Norm Walsh. There is also one additional vacancy due to a resignation. The election will be held this Spring, and nominations for these seven openings are now being invited.

The Bylaws provide that "Any member may be nominated for election to the Board by submitting a nomination petition in accordance with the TUG Election Procedures. Election of the directors shall be by written mail ballot of the entire membership, carried out in accordance with those same Procedures." The term of office of a director is three (3) years. Incumbent officers may be nominated for successive terms.

The name of any member may be placed in nomination for election to the board by submission of a petition, signed by two other current (1996) members, to the TUG office at least two weeks (14 days) prior to the mailing of ballots. (A candidate's membership dues for 1996 will be expected to be paid by the nomination deadline.) A petition form follows this announcement; forms may also be obtained from the TUG office, and electronically from the `usergrps/tug` area of CTAN.

Along with a petition form, each candidate is asked to supply a passport-size photograph, a short biography, and a statement of intent to be included with the ballot; the biography and statement of intent together may not exceed 400 words.

The deadline for receipt at the TUG office of petitions and ballot information is **March 15, 1996**.

Ballots will be mailed to all members in early **April**. Marked ballots must be postmarked no later than **May 17**, and received no later than **May 31**. These deadlines will be noted on the ballots.

Ballots will be counted by a disinterested party not part of the TUG organization. The results of the election should be available by early June, and will be announced in a future issue of *TUGboat* as well as through various TEX-related electronic lists.

1996 TUG Election — Nomination Form

Only current (1996) TUG members are eligible to participate. The signatures of two (2) members are required in addition to that of the nominee. **Type or print** names clearly, exactly as they appear in the most recent TUG membership list or on a TUG mailing label; new members should enter the name which they used on their membership application form. Names that do not exactly match the TUG records will not be accepted as valid.

The undersigned TUG members propose to nominate the following TUG member for the position of **member of the TUG Board of Directors**, for a term beginning with the July 1996 Annual Meeting:

Name of Nominee: _____

Signature: _____

Date: _____

Members supporting this nomination:

1. _____ (please print)

   _____ (signature) _____ (date)

2. _____ (please print)

   _____ (signature) _____ (date)

Return this petition form to the TUG office (FAXed petition forms will be accepted). Petitions and all required supplementary material (photograph, biography and personal statement for inclusion on the ballot) must be received in the TUG office no later than **March 15, 1996**.[1] It is the responsibility of the candidate to ensure that this deadline is met. Under no circumstances will incomplete applications be accepted.

- ☐ nomination form
- ☐ photograph
- ☐ biography/personal statement

TEX Users Group **FAX:** 415-982-8559
Nominations for 1996 Election
1850 Union Street, #1637
San Francisco, CA 94123
U.S.A.

[1] Supplementary material may be sent separately from the form, and supporting signatures need not all appear on one form.

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*

Brookhaven National Laboratory,
Upton, *New York*

CNRS - IDRIS,
Orsay, *France*

CERN, *Geneva*, *Switzerland*

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

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

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

*CS*TUG, *Praha*, *Czech Republic*

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

Fermi National Accelerator Laboratory, *Batavia*, *Illinois*

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

Grinnell College,
Noyce Computer Center,
Grinnell, *Iowa*

Hong Kong University of Science and Technology,
Department of Computer Science,
Hong Kong

Institute for Advanced Study,
Princeton, *New Jersey*

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

Iowa State University,
Ames, *Iowa*

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

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

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

Max Planck Institut für Mathematik,
Bonn, *Germany*

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

Nippon Telegraph & Telephone Corporation,
Basic Research Laboratories,
Tokyo, *Japan*

Personal TEX, Incorporated,
Mill Valley, *California*

Princeton University,
Princeton, *New Jersey*

Smithsonian Astrophysical Observatory, *Cambridge*, *Massachusetts*

Space Telescope Science Institute,
Baltimore, *Maryland*

Springer-Verlag,
Heidelberg, *Germany*

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

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

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

United States Naval Postgraduate School,
Monterey, *California*

United States Naval Observatory,
Washington DC

University of California, Berkeley,
Center for EUV Astrophysics,
Berkeley, *California*

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

University of Canterbury,
Christchurch, *New Zealand*

University College,
Cork, *Ireland*

University of Delaware,
Newark, *Delaware*

University of Groningen,
Groningen, *The Netherlands*

Universität Koblenz–Landau,
Koblenz, *Germany*

University of Manitoba,
Winnipeg, *Manitoba*

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

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*

Università degli Studi di Trieste,
Trieste, *Italy*

Uppsala University,
Uppsala, *Sweden*

Vrije Universiteit,
Amsterdam, *The Netherlands*

Wolters Kluwer,
Dordrecht, *The Netherlands*

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

TEX Consulting & Production Services

Information about these services can be obtained from:

TEX Users Group
1850 Union Street, #1637
San Francisco, CA 94123, U.S.A.
Phone: +1 415 982-8449
Fax: +1 415 982-8559
Email: tug@tug.org

North America

Anagnostopoulos, Paul C.
Windfall Software,
433 Rutland Street, Carlisle, MA 01741;
(508) 371-2316; greek@windfall.com
We have been typesetting and composing high-quality books and technical Publications since 1989. Most of the books are produced with our own public-domain macro package, ZzTEX, but we consult on all aspects of TEX and book production. We can convert almost any electronic manuscript to TEX. We also develop book and electronic publishing software for DOS and Windows. I am a computer analyst with a Computer Science degree.

Cowan, Dr. Ray F.
141 Del Medio Ave. #134, Mountain View, CA 94040;
(415) 949-4911; rfc@netcom.com
Twelve Years of TEX and Related Software Consulting: Books, Documentation, Journals, and Newsletters
TEX & LATEX macropackages, graphics; PostScript language applications; device drivers; fonts; systems.

Hoenig, Alan
17 Bay Avenue, Huntington, NY 11743; (516) 385-0736
TEX typesetting services including complete book production; macro writing; individual and group TEX instruction.

NAR Associates
817 Holly Drive E. Rt. 10, Annapolis, MD 21401;
(410) 757-5724
Extensive long term experience in TEX book publishing with major publishers, working with authors or publishers to turn electronic copy into attractive books. We offer complete free lance production services, including design, copy editing, art sizing and layout, typesetting and repro production. We specialize in engineering, science, computers, computer graphics, aviation and medicine.

Ogawa, Arthur
40453 Cherokee Oaks Drive,
Three Rivers, CA 93271-9743;
(209) 561-4585
Experienced in book production, macro packages, programming, and consultation. Complete book production from computer-readable copy to camera-ready copy.

Quixote Digital Typography, Don Hosek
555 Guilford, Claremont, CA 91711;
(909) 621-1291; Fax: (909) 625-1342;
dhosek@quixote.com
Complete line of TEX, LATEX, and METAFONT services including custom LATEX style files, complete book production from manuscript to camera-ready copy; custom font and logo design; installation of customized TEX environments; phone consulting service; database applications and more. Call for a free estimate.

Richert, Norman
1614 Loch Lake Drive, El Lago, TX 77586;
(713) 326-2583
TEX macro consulting.

Type 2000
16 Madrona Avenue, Mill Valley, CA 94941;
(415) 388-8873; Fax: (415) 388-8865
pti@crl.com
$2.50 per page for 2000 DPI TEX and PostScript camera ready output! We provide high quality and fast turnaround to dozens of publishers, journals, authors and consultants who use TEX. Computer Modern, PostScript and METAFONT fonts available. We accept DVI and PostScript files only and output on RC paper. $2.25 per page for 100+ pages, $2.00 per page for 500+ pages; add $.50 per page for PostScript.

Outside North America

TypoTEX Ltd.
Electronical Publishing, Battyány u. 14. Budapest,
Hungary H-1015; (036) 11152 337
Editing and typesetting technical journals and books with TEX from manuscript to camera ready copy. Macro writing, font designing, TEX consulting and teaching.