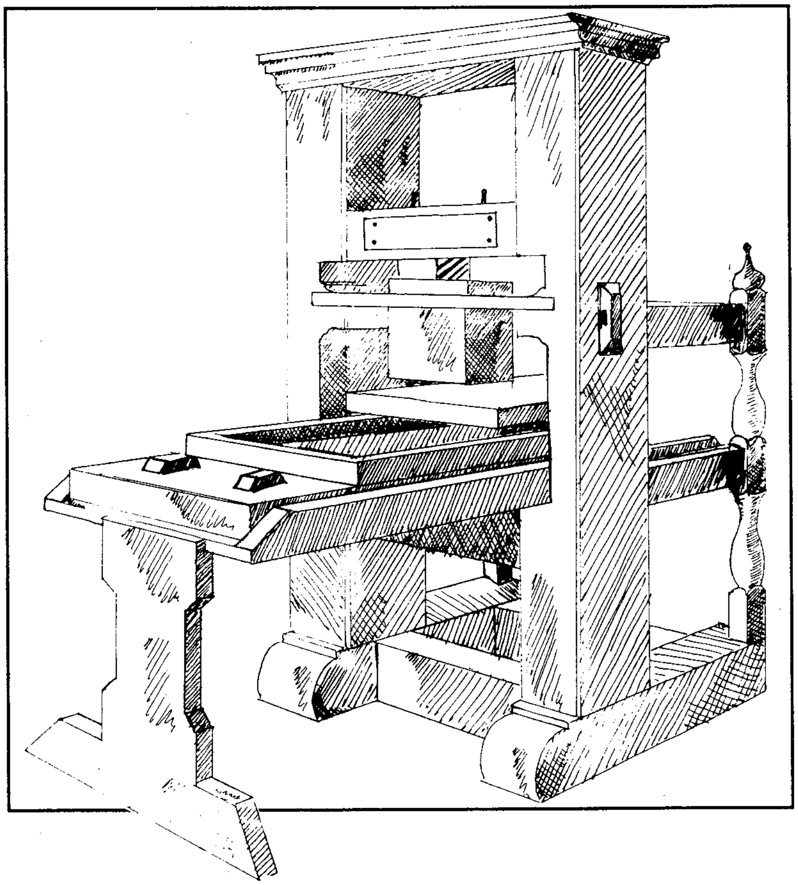
TUGBOAT

The Communications of the TEX Users Group



Volume 15, Number 2, June 1994

TEX Users Group

Memberships and Subscriptions

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

1994 dues for individual members are as follows:

- Ordinary members: $60
- Students: $30

Membership in the TEX Users Group is for the calendar year, and includes all issues of *TUGboat* and *TEX and TUG NEWS* for the year in which membership begins or is renewed. Individual membership is open only to named individuals, and carries with it such rights and responsibilities as voting in the annual election. A membership form is provided on page 155.

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

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

Institutional Membership

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

Printed in U.S.A.

Board of Directors

Addresses
All correspondence, payments, etc.
TEX Users Group
P. O. Box 869
Santa Barbara, CA 93102-0869 USA

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

Telephone
805-963-1338

Fax
805-963-8358

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

TEX is a trademark of the American Mathematical Society.

The question [is] what proportion of the market knowingly accepts sloppy composition and what proportion remains in blissful ignorance of the good manners in typographic presentation? [...] the newer techniques have not dispensed with the need for operatives aware of the finer points of typographic composition.

Lawrence W. Wallis
Irresistible Market Forces?,
in *Electronic Publishing and TypeWorld* (Volume XVIII, Number 6, April 22, 1994)

TUGBOAT

COMMUNICATIONS OF THE TEX USERS GROUP

EDITOR BARBARA BEETON

VOLUME 15, NUMBER 2 • JUNE 1994
SANTA BARBARA • CALIFORNIA • U.S.A.

TUGboat

During 1994, the communications of the TeX Users Group will be published in four issues. One issue (Vol. 15, No. 3) will contain the Proceedings of the 1994 TUG Annual Meeting.

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

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

Submitting Items for Publication

The next regular issue will be Vol. 15, No. 4; deadlines are August 17, 1994, for technical items, and September 14, 1994, for reports and similar items. Mailing is scheduled for December. Deadlines for future issues are listed in the Calendar, page 143.

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. 87).

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

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

Reviewers

Additional reviewers are needed, to assist in checking new articles for completeness, accuracy, and presentation. Volunteers are invited to submit their names and interests for consideration; write to `TUGboat@math.ams.org` or to the Editor, Barbara Beeton (see address on p. 87).

TUGboat Editorial Board

Other TUG Publications

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

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

TUGboat Advertising and Mailing Lists

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

Trademarks

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

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 AMS-TeX are trademarks of the American Mathematical Society.
Textures is a trademark of Blue Sky Research.
UNIX is a registered trademark of UNIX Systems Laboratories, Inc.

Addresses

Note: Unless otherwise specified, network addresses (shown in `typewriter` font) are on the Internet.

TeX Users Group Office
P. O. Box 869
Santa Barbara, CA 93102-0869 U.S.A.
or
Balboa Building, Room 307
735 State Street
Santa Barbara, CA 93101 U.S.A.
805-963-1338
Fax: 805-963-8358
`tug@tug.org`

Paul C. Anagnostopoulos
Windfall Software
433 Rutland Street
Carlisle, MA 01741 U.S.A.
`greek@genome.wi.edu`

Claudio Beccari
Dipartimento di Elettronica
Politecnico di Torino
I-10129 Turin, Italy
`beccari@polito.it`

Barbara Beeton
American Mathematical Society
P. O. Box 6248
Providence, RI 02940 U.S.A.
401-455-4014
`bnb@math.ams.org`
`TUGboat@math.ams.org`

John Berlin
TeX Users Group
P. O. Box 869
Santa Barbara, CA 93102-0869 U.S.A.
805-963-1338
`john@tug.org`

Johannes Braams
PTT Research Neher Laboratories
P. O. Box Box 421
2260 AK Leidschendam
The Netherlands
`J.L.Braams@research.ptt.nl`

Mimi R. Burbank
Supercomputer Computations
Research Institute
B-186, 400 Science Center Library
Florida State University
Tallahassee, FL 32306-4052 U.S.A.
904-644-2440
`mimi@scri.fsu.edu`

Jackie Damrau
P. O. Box 875
Red Oak, TX 75154-0875, U.S.A.
214-617-2323
`damrau@amber.unm.edu`

Luzia Dietsche
Universität Heidelberg
Im Neuenheimer Feld 293
D-69120 Heidelberg, Germany
`x68@vm.urz.uni-heidelberg.de`

Michael Doob
Department of Math & Astronomy
University of Manitoba
341 Machray Hall
Winnipeg R3T 2N2
Manitoba, Canada
204-474-9796
`mdoob@ccu.umanitoba.ca`

Michael Downes
American Mathematical Society
P. O. Box 6248
Providence, RI 02940 U.S.A.
`mjd@math.ams.org`

Victor Eijkhout
Department of Computer Science
107 Ayres Hall
University of Tennessee at Knoxville
Knoxville, TN 37996-1301, U.S.A.
`eijkhout@cs.utk.edu`

Michael J. Ferguson
INRS – Télécommunications
Université du Québec
16 Place du Commerce
Verdun H3E 1H6, Québec Canada
514-765-7834
`mike@inrs-telecom.uquebec.ca`

Peter Flynn
Computer Center
University College
Cork, Ireland
+353 21 276871 x2609
`cbts8001@iruccvax.ucc.ie`

Bernard Gaulle
IDRIS-*CNRS*
BP 167
F–91403 Orsay Cedex, France
`gaulle@idris.fr`

Michel Goossens
CN Division
CERN
CH-1211 Geneva 23, Switzerland
`goossens@cern.ch`

George D. Greenwade
Department of Economics and
Business Analysis
College of Business Administration
P. O. Box 2118
Sam Houston State University
Huntsville, TX 77341-2118 U.S.A.
`bed_gdg@shsu.edu`

Yannis Haralambous
187, rue Nationale
59000 Lille, France
Fax: (33) 20.91.05.64
`yannis@univ-lille1.fr`

Alan Hoenig
17 Bay Avenue
Huntington, NY 11743 U.S.A.
516-385-0736
`ajhjj@cunyvm.cuny.edu`

Donald E. Knuth
Department of Computer Science
Stanford University
Stanford, CA 94305 U.S.A.

Joachim Lammarsch
Computing Center
University of Heidelberg
Im Neuenheimer Feld 293
D-69120 Heidelberg, Germany
`x92@vm.urz.uni-heidelberg.de`

Dag Langmyhr
Department of Informatics
University of Oslo
P. O. Box 1080 Blindern
N-0316 Oslo, Norway
+47-22 85 24 50; fax: +47-22 85 24
01
`dag@ifi.uio.no`

Pat Monohon
TeX Users Group
P. O. Box 869
Santa Barbara, CA 93102-0869 U.S.A.
805-963-1338
`monohon@tug.org`

R.W.D. Nickalls
Department of Anaesthesia
University Hospital
Queen's Medical Centre
Nottingham, NG7 2UH
England, UK
`dick.nickalls@nottingham.ac.uk`
`100115.1010@CompuServe.com`

Nico A. F. M. Poppelier
Elsevier Science Publishers BV
Academic Publishing Division
Information Technology Development
Molenwerf 1
1014 AG Amsterdam,
The Netherlands
`n.poppelier@elsevier.nl`

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

Sebastian Rahtz
ArchaeoInformatica
12 Cygnet Street
York Y02 1AG, U.K.
(44) 904 633904
`Sebastian.Rahtz@cl.cam.ac.uk`

R. Ramasubramanian
Department of Anaesthesia
University Hospital
Queen's Medical Centre
Nottingham, NG7 2UH
England, UK
`100016.3245@CompuServe.com`

M.A. Reed
Cambridge University Press
Technical Applications Group
Printing Division
University Printing House
Shaftesbury Road
Cambridge, CB2 2BS, England, UK
`mreed@cup.cam.ac.uk`

Tomas Rokicki
Box 2081
Stanford, CA 94309 U.S.A.
415-322-6442
`rokicki@cs.stanford.edu`

Chris Rowley
Open University
Walton Hall
Milton Keynes MK7 6AA
United Kingdom
`c.a.rowley@open.ac.uk`

Janet Sullivan
TEX Users Group
P. O. Box 869
Santa Barbara, CA 93102-0869 U.S.A.
805-963-1338
`janet@tug.org`

Philip Taylor
The Computer Centre
Royal Holloway and Bedford New
College
University of London
Egham, Surrey TW20 0EX, England
`P.Taylor@Vax.Rhbnc.Ac.Uk`

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

Hermann Zapf
Seitersweg 35
D-64287 Darmstadt, Germany

General Delivery

Opening words

Christina Thiele
President, TEX Users Group

As we head into the summer months of holidays and hot weather, no-one wants to read a long colums — much less write one! So I'll keep it short.

In my previous column I mentioned a big four-way meeting planned for Darmstadt in April — RIDT'94 and EP'94, along with workshops from two other organisations (PODP'94 and TEP'94. Lucky for us, Michel Goossens, TUG's vice-president, was able to attend, and he's written up a nice report you might find interesting.

And something that's been moving in slow stages over the past several months — ISO recognition of de facto standards. That is, the ISO people are willing to discuss the fact of there being standards in areas where ISO has none, standards which have arisen out of common consent, and where the full specification is published or publicly available. Examples include TCP/IP, PostScript — and TEX. Nothing definite has been set down yet — at this stage, all that's being examined is the structure and procedure for recognising such de facto standards. The issue will be on the board agenda this summer, and we will probably come up with an official TUG statement at that time. In the interim, a number of people have come forward, saying they too support this potential procedure regarding de facto standards, and that it would be most appropriate to consider TEX a good candidate for this status.

I believe it will be of long-term benefit to TEX, which often must be left behind when companies or research institutes are required to submit documentation prepared according to recognised standards. If TEX can become a recognised formatter for electronic documentation, then we are taking a major step forward in ensuring its long life and widespread use.

It's an exciting prospect. If anyone is interested in hearing more, please contact myself, or Barbara Beeton, editor of this publication.

So that's it. Short and sweet. Enjoy your summer. I hope to see many of you at the meeting in Santa Barbara. If not, then we shall meet again in the fall.

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

Editorial Comments

Barbara Beeton

1 "LATEX 2$_\varepsilon$" is now just "LATEX"

The LATEX 3 project team has announced the release of "the new standard version of LATEX". This is the version that, until now, was referred to as "LATEX 2$_\varepsilon$". From now on the "old LATEX" will be referred to as "LATEX 209".

From the announcement:

> [The new LATEX] is upwardly compatible with existing LATEX documents, and contains new features including the long-awaited graphics extensions.
>
> Over the years many extensions of LATEX have been developed. This is, of course, a welcome development, since it shows that the LATEX system is in a healthy state. It has, however, had one unfortunate consequence: there were several incompatible systems all claiming to be LATEX.
>
> The new LATEX puts an end to this unsatisfactory situation — it gives access to all extensions such as SLITEX, AMS-LATEX, and PSLATEX, based on a single format. This will end the proliferation of mutually incompatible dialects of LATEX.
>
> We have also introduced a small number of often-requested features (such as more control over float placement) and an improved interface for writers of document classes and packages.
>
> The new LATEX is described in a new edition of *LATEX: A Document Preparation System* by Leslie Lamport (to appear during 1994) and *The LATEX Companion* by Goossens, Mittelbach and Samarin, both published by Addison-Wesley.

From now on, a completely updated release will be made twice a year, in June and December. In between scheduled releases, bugs will be fixed using a patch technique that has been built into the LATEX system. Users should read the documentation included in the package and check periodically for patch updates. A formal procedure, also described in the documentation, is provided for reporting bugs.

The new LATEX can be retrieved by anonymous ftp from the CTAN archives; see below for addresses. At all the major sites, the distribution can be found in the area

```
/tex-archive/macros/latex/base
```

The files in these directories require a somewhat long unpacking process. For those who prefer to skip this step, a second distribution is provided in unpacked format in

```
/tex-archive/macros/latex/unpacked
```

The LaTeX 3 group is now returning to work on LaTeX 3. They will continue to consider suggestions for future work; the announcement also pointed out that a number of suggestions already received could be developed into useful LaTeX packages — by you!

Thanks to all the workers on the project:

Johannes Braams
David Carlisle
Michael Downes
Alan Jeffrey
Frank Mittelbach
Chris Rowley
Rainer Schöpf

2 New CTAN features

Just as a reminder, CTAN is the Comprehensive TeX Archive Network, with primary nodes at

ftp.dante.de	(129.206.100.192)
ftp.tex.ac.uk	(128.232.1.87)
ftp.shsu.edu	(192.92.115.10)

2.1 Where are the CTAN sites?

For anyone who has an Internet connection, it is now possible to get a listing of CTAN sites and mirrors along with their root CTAN directories. For those who can use the finger utility:

```
finger ctan_us@ftp.SHSU.edu
```

For those who can't use finger, but have access to Gopher, use the Gopher link information (available in the TeX area on the Gopher server at `Niord.SHSU.edu`):

```
Name=Listing of CTAN sites and mirrors
Type=0
Port=70
Path=exec::gopher_root:[000000]_CTAN.SCRIPT
Host=Niord.SHSU.edu
```

or the URL (this should be all on one line with no spaces):

```
gopher://Niord.SHSU.edu:70/0exec%3a%
  3agopher_root%3a%5b000000%5d_CTAN.SCRIPT
```

or (depending on your World Wide Web viewer; again, one line):

```
gopher://Niord.SHSU.edu:70/0exec::
  gopher_root:[000000]_CTAN.SCRIPT
```

There is also a file, `CTAN.sites`, at the root level of the CTAN hosts which contains this information.

As listings of participating sites change, this file will be updated accordingly.

Another helpful tool compliments of George Greenwade. Thanks, George!

2.2 What's new on CTAN?

A new e-mail list, `CTAN-Ann@SHSU.edu` has been created for the purpose of disseminating announcements to and from the coordinators of CTAN and other interested parties regarding product and utility announcements related to TeX. It is fully archived, completely public for subscription purposes, and is open to posts from anyone. The overriding concern of this list is to ensure an efficient, identifiable, and authoritative single source of information about the evolution of the CTAN collection to it worldwide base of users. This is *not* a discussion list.

To subscribe to the list, send e-mail to

```
LISTSERV@SHSU.edu
```

with the following request in the body:

```
SUBSCRIBE CTAN-Ann "Your Name, in Quotes"
```

2.3 Posting new files to CTAN

If you have a package that should be shared with other TeX users, posting it on CTAN is an excellent way to make it available.

Ftp to `ftp.shsu.edu` and connect to `incoming`. If multiple files are involved, use `mkdir` to create a subdirectory for them. After posting, send a message to `CTAN-Mgr@shsu.edu` asking that it be installed on CTAN; it is always helpful if you can be specific about the location.

Generally, posts to `CTAN-Mgr` should get a response within 72 hours, including weekends. When you learn that your package has been installed, announce it! `info-tex@shsu.edu` has its own list of subscribers and feeds into `comp.text.tex`, and the new list `CTAN-Ann@shsu.edu` (see above) is tailor-made for such announcements.

Please don't send files just to `comp.text.tex` — many people don't have access to it. A package posted on CTAN will get promulgated all over the world, included on CDs, and made available in many more ways than you can even think of.

3 Northwest Computing Support Center closed

A message from Pierre MacKay has announced that the Northwest Computing Support Center was closed by order of the University of Washington as of June 14, 1994. Pierre and Elizabeth Tachikawa will try unofficially to provide some limited services for a few months, but there is no longer a phone connection and support of UNIX TeX from that site has ceased.

The TeX *community will miss this support* greatly. Belatedly, we would like to extend our thanks to Pierre and Elizabeth for their superb service over the years.

4 TeX output in audible form

I spent a few days in May in Princeton, at the offices of Recordings for the Blind, attending a math and sciences working symposium. The goal was to set directions for making available math and science texts in electronic form for the blind and others with reading disabilities. The centerpiece was work by Raman, chair of the TUG Technical Working Group on TeX for the Disabled.

One of the workshop recommendations was to bring Raman's system from its status as a research project on some pretty high-powered (and -priced) equipment to a widely-portable (in the TeX sense) system that can be used by a much wider audience.

RFB's charter is to provide texts for students and other "serious" users (current novels and other popular material for recreational reading are provided by other organizations) and they are supposed to produce "accessible" versions of any book that is requested. These audio or electronic versions are supposed to be identical in all ways to the ordinary printed versions, since they are mostly textbooks and other assigned material on which the students will be tested, and an important objective is to put the disabled student on an equal footing with all other students in the same program.

There are special challenges, of course, with math notation, and anyone who attended the 1992 meeting in Portland or read Raman's paper in the Proceedings may remember the difference between the vocalization generated by his system for a complicated equation by Faa de Bruno and the result of its being read by a trained RFB reader. Although both were rendered in linear form, Raman's captured more of the mathematical structure in a more concise form than did the human reader's — and Raman's system has been extended since then to allow different "views" interactively, with the user permitted to ask for the general structure, then for the details of various parts of the expression. The challenge that the RFB organizers of this symposium set is for all producers of tools and data to work toward the same goal — accurate and economical delivery of mathematical material in a form most easily used by those who have no other option. Raman's system is a very promising start on this path.

There will be challenges for the authors of math books and journals as well. For the system to work, *the math must be input uniformly, with a minimum* of "tinkering" with spacing, positioning, and the like; the matter of user-defined macros created to shorten the keying of complex notation, or to ensure uniform notation, has yet to be addressed. But the principle is important — that a little effort and good tools can make this material much more usable by someone who can't read it on paper.

Raman's system is called "ASTeR", "Audio System for Technical Reading". Anyone who attended the Portland TUG meeting will remember a bright-eyed, energetic black labrador retriever, Raman's guide dog, the eponymous Aster after whom the reading system is named and to whom Raman's dissertation is dedicated. She was there at the symposium too, along with quite a few others of her kind, a most devoted group of working partners; I was glad to renew our acquaintance.

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

WEPT: A Week on Electronic Publishing and Typography

Michel Goossens

From Monday April 11th to Friday April 15th the Lufthansa Training Center in Seeheim, a few kilometers south of Darmstadt, was home to WEPT, a Week on Electronic Publishing and Typography. The Center is situated on a hill overlooking the health resort of Seeheim (pop. 18,000) and lies in the Bergstrasse ('Mountain Road', from the Latin *strata montana*) holiday area, the "German Riviera", well-known for its mild, almost Mediterranean-like climate, in Roman times. The Center is surrounded by a forest, with several scenic footpaths. However, while I was there, both the tight conference schedule and the rainy weather conspired against us, and I had no occasion to find out how quiet and relaxing the surroundings were.

Four events in one

WEPT was in fact a combination of two international conferences — RIDT'94 (Raster imaging and digital typography) and EP'94 (Electronic publishing, document manipulation and typography) — and two international workshops — PODP'94 (Principles of document processing) and TEP'94 (Teaching electronic publishing and digital typography), plus a set of tutorials. WEPT was organized locally by Christoph Hüser of GMD/ISPI Darmstadt. Upon registering, every participant got, together with a printed copy of the proceedings (published as issues 3 and 4 of volume 6 of the journal *EP-ODD — Electronic Publishing-Origination, Dissemination and Design*), an electronic copy in Adobe's PDF format on CD-ROM (more on this later)[1]. As those of us who have been involved in preparing conference proceedings know, it cannot be emphasized too much what a magnificent job the editors have done to be able to distribute the proceedings at the conference; reading their Colophon in both the EP'94 and RIDT'94 issues tells an interesting story about electronic publishing in real life. As several of the activities ran in parallel, I can only describe the ones I attended. Moreover this report will necessarily be biased towards subjects I think I understand best.

Act I: RIDT'94

The formal opening of RIDT'94 (the third international conference on Raster Imaging and Digital Typography), by Jacques André, was followed by a session on font modeling, where we learned about *Infinifont*, a new product for high-performance parametric font generation starting from a single compact set of typographic characteristics. This was followed by a discussion of a font-independent way to describe Kanji that should make it easier to (semi-)automatically design large character sets in a coordinate-independent fashion. Then Yannis Haralambous argued that it is possible to group PostScript fonts into meta font families by defining a set of METAFONT parameters that can be varied to obtain different weights (condensed and extended versions), small caps, and optical scaling. The last talk in the session introduced object-orientation to extend a font-scaler to generate acceptable quality screen fonts on-the-fly.

After coffee there were round table discussions on fonts and piracy, and about the future of type. Two well-known font designers, Hermann Zapf and Charles Bigelow, participated actively in the discussion. Zapf said that there is no future for font designers while their works are freely copied, and in any case, even if a company has the copyright of a set of fonts, it is almost impossible to have it enforced. It was noted that this will be a problem, not only for type, but for all material that is available electronically, so that companies active in the field of publishing or distributing electronic documents (films, music, books, etc.) all face the same problem of how to make the viewer/user pay. The future of typography is closely linked to the development of multimedia techniques, where animation, electronic books, sound, will be all important players. Most speakers seemed to agree that the techniques for rasterizing type at medium and high resolutions are well understood by now, but that a lot of work still needs to be done to improve readability on low-resolution devices, such as computer screens, where reading speed can drop by up to 30%. Grayscale fonts, the active use of color, and a more optimized layout of the information on screen are a few possibilities being studied to improve this situation.

The second day started with a session on raster imaging and fonts. Various optimizations for rasterizing Bezier curves, variable width splines, how to fit a curve to a gray-level image of scanned font characters, digital halftoning and error diffusion, and dynamic regularization of outline fonts, were some keywords I picked up from the titles of the talks, which seemed to indicate that rasterization is at present a well-understood technique. After the break we had two presentations on readability. The first one showed that typography is not only about letterforms, but also about symbolic characters. In a study to classify various symbols by a rule-based

[1] The two issues, as well as the CD-ROM can be obtained separately; please contact the publisher John Wiley & Sons for more details

system, it was found that it is much more difficult to build a semiology for symbols than for typefaces, since symbols are more difficult to identify and discriminate simply. The second talk presented the results of a comparison of possible differences in readability between Times Roman (serif face) and Helvetica (sans serif face). In agreement with other recent work no differences were detected, and it was found that familiarity with a typeface is more important than the presence or otherwise of serifs. These results once more show that the need for serifs in typeface design should be questioned if its main purpose is to make the typeface more legible.

A crash course on hypertext

In the afternoon I could not attend the talks on character recognition, since I wanted to listen to the tutorial on hypertext techniques by George Landow. The speaker first presented the historical evolution of the presentation form, and then went on to describe case studies. He emphasized that one cannot just transpose the printed book typographic model into the electronic world, but that "real" hypertext does not have pages, so that it is not always obvious to talk about equivalents for footnotes, indexes, title pages, etc. Added value is the presence of (hyper)links to navigate freely through a document, and the availability of an easy-to-use find function is a must. Good typographic design based on printed book tradition should certainly be used, but the chunks of information have to be smaller. Information presented to the user/viewer should never annoy or distract, and visual features and gadgets should only be added to the representation window when they serve a well-defined purpose.

The cradle of the printing art

In the evening we visited the Gutenberg Museum, located in Mainz, and founded in 1900 to celebrate the 500th anniversary of Gutenberg's birthday. Gutenberg was above all an engineering genius, who combined various already well established techniques into a practical procedure to print books relatively cheaply using movable metal type. The museum was completely destroyed during an Allied bombing raid on Mainz on February 27th 1945, but thanks to numerous donations from individuals, institutions and companies it was reopened in new buildings in 1962.

The main purpose of this temple of the printing art and technology is to bring together, classify, study, and display everything connected with Gutenberg's invention. The Gutenberg Foundation, which administers the Museum, also considers that one of its main rôles is to disseminate as widely as possible by scientific and other publications all information relating to the history of printing. The museum contains over 24,000 volumes, amongst them more than 2,300 *incunabula* (books printed before the year 1500), but probably its most valuable piece is one of the few remaining complete copies of the original Gutenberg 42-line Bible.

On the several floors of the museum the visitor can also admire many interesting and unique typesetting machines from all over the world. A very interesting "live" experience was a demonstration in Gutenberg's reconstructed workplace of how a page would have been printed on paper manufactured 500 years ago, using "his" printing press. Moreover, a molten metal alloy was taken out of the oven, and the fluid poured into a letter form. To everybody's surprise, after only a few seconds, the letter was cold enough so that it could be handled safely, and used to compose a line of text. As we arrived rather late at the museum (in fact after closing time), we didn't have enough time to spend there. I now know, however, that next time I visit Mainz, I shall put aside at least half a day to go to the Gutenberg Museum and have a good look at all the treasures on display.

After a walk of a few hundred meters we arrived at the restaurant, where the conference banquet was held. On the menu we had the typically German Sauerkraut and, of course, beer flowed freely. As we were seated along long wooden tables, we had ample occasion to "socialize". Amongst less technical discussions about Disney World in California or sailing in Brittany, I was able to get some interesting information about Unicode 16-bit character support in Apple's next update of their operating system (foreseen for the end of 1994).

A tutorial on color

I started the Wednesday morning early with a tutorial on color printing. In less than three hours we received an overview of colorimetry, device calibration, a study of Moiré patterns, halftoning techniques, and a report on ongoing development work among many companies active in the field of color printing and display on a standard describing quantitative color matching techniques for various output devices.

Then we came all together...

Just before lunch the participants of all four concurrent WEPT events were encouraged to enjoy an invited talk by Hermann Zapf on his *hz* program, where he uses micro-typographic algorithms to obtain optical margin compensation at both sides of a column. His program is based on scaling, i.e., applying typographically acceptable expansion or condensing of letters, together with positive and negative kerning. R.F. Bruine, a Director at the E.U., then gave a keynote address on the importance of informatics technology for the future of Europe, where several millions of new jobs have to be created before the year 2000.

The afternoon began with an invited talk by Charles Bigelow, who described his work on the Unicode sans serif Lucida font, that he designed together with Kris Holmes. He started with a few slides showing the evolution of writing over the ages. Then he went on to show why a world-wide character (not glyph) encoding standard is needed to allow an easy exchange of information between various alphabets and more complicated writing systems. He described the work of the Unicode Consortium, who developed the 16-bit Unicode character encoding standard where, in their version one, they define about 30,000 characters, 4,000 of which are alphabetic or mathematical ones, while the others correspond mostly to Japanese/Chinese Han characters or Korean Hangul. The first release of the Bigelow/Holmes Lucida Sans font contains about 1,725 of the Unicode "letters", and it is used by the Windows/NT, Apple GX, and AT&T Plan 9 16-bit operating systems.

Act II: EP'94

The Bigelow talk formally concluded the RIDT'94 Conference, and after a short break Vincent Quint opened EP'94 (the Fifth Conference on Electronic Publishing, Document Manipulation and Typography). The first session was about document manipulation techniques, and the first speaker drew a parallel between document preparation and the processes of compilation and link editing of computer programs, allowing various blocks of documents to be compiled separately. The next presentation discussed formatting of structured documents within the framework of the Grif SGML editor, and the importance of document reuse issues. The final talk described a constraint-built interactive editor for introducing SGML markup in linguistic texts. This editor includes a context-sensitive search mechanism and will be used in the Text Encoding Initiative.

The Thursday program started with a session on structure transformation. The talks concentrated on explaining ways to compare or transform structured documents, formulate queries, or perform context-sensitive pattern matching on the global structural level. After the break we addressed the "hot" topic of multimedia and hypertext. We first got a clear introduction to the syntax of the HyTime standard. This was followed by a description of the Multimedia Forum project, developed at GMD, Darmstadt, which is a working example of a multimedia interactive online journal. The morning ended with a somewhat more theoretical talk about ongoing work at Xerox Japan on techniques for building structured document views on hypertext networks.

Document recognition was the subject of the afternoon session, with a first presentation on recognizing the logical structure of OCR scanned bibliographic references. *MarkItUp*, a system using incremental generation of structure recognition grammars from example structures to detect the logical structure of untagged electronic documents, was described next. The third talk introduced an approach based on fuzzy logic in the matching stage that provides some error tolerance in the parsing process to increase system robustness. Then there was an extended demonstration session, where all conference participants could see most of the systems described during the week at work. The evening was spent in Auerbach Castle, where we attended a middle-age style knights' feast.

The final morning had as its theme document preparation and publishing. An integrated publication environment, based on dynamic DTD management using an object-oriented database, VODAK, was described. It is used in the MultiMedia Forum application. Then came the story of the CAJUN project, or how the articles of all six volumes of the EPODD journal were successfully translated into PDF format, which is the language used by Adobe's Acrobat. These files were made available on a CD-ROM, together with the PDF reader software for Mac, MS-DOS and Windows, and given to each participant to the conference. One of the conclusions of the talk was that it is relatively easy to transform electronic documents into PDF, by adding information in the files, but that the format is at present too page- and contents-based, and links are anchored to physical positions on the page, rather than to logical locations in the document. It is expected that Adobe will enhance Acrobat in an upcoming release to eliminate most of these limitations, and implement a more complex search algorithm than that available at present.

When I started using this CD-ROM on my PC at work, I must say I was quite impressed by the work that the CAJUN people had put into their CD. One is, however, somewhat limited by the amount of information one can display on a screen (to be readable I was using 143% magnification, which is about the minimum; 200% is even better). Also, the lack of links (apart from the ones to the bibliography entries, and the index, which uses the keywords of each article) does not show all the possibilities of hypertext. The absence of backlinks is especially annoying, since you cannot jump back to where you were, before going to the bibliographic reference. Nevertheless, the CAJUN CD already shows some of the possibilities of an information technology that will become ever more evident in the near future.

Large-scale encyclopaedic reference works were discussed in a another presentation by the GMD group. The experience gained while developing efficient tools for the simple construction of the object network of the SGML-tagged corpus of the 34,000-page *Dictionary of Art* (to be published by Macmillan in 1996), was discussed. The last talk before the break was an introduction to the Score TEX (ScEX) system, which is a front-end to Daniel Taupin's MusicTEX. ScEX consists of a Voice Language (SVL), and a Score Language (SSL), and tools exist to move between SVL input and SSL output, and vice versa.

The morning — and the EP'94 Conference — ended with an inspiring invited talk by Roberto Minio. His theme was "Publishing and mathematics — analogous evolutions". He began by saying that publishing is about communicating knowledge, while mathematics is about creating, or discovering knowledge. He presented five parallel areas of evolution.

(a) Today one often talks about the *death of proof*: various articles show that computer proof, using tools such as Mathematica, etc., is not considered acceptable or "mathematically rigorous" by many pure mathematicians, i.e., the use of electronically-assisted tools is not yet socially accepted. Similarly in publishing, for the *paperless office*, tools like Acrobat pose more problems than they solve. Questions about the ownership of the contents of electronic documents, the exploitation of their copyright, and optimal document reuse are being debated at present, with the "classical" printing industry trying to carry the old way of doing things over into the next century.

(b) Although considered "non-rigorous", speculative, intuitive, visual and experimental computer based computation techniques are spreading rapidly in the different fields of mathematics. The world of publishing is actively experimenting with multimedia and network publishing. In both cases one should not be confined to traditional techniques but actively embrace new developments.

(c) Everywhere "marketing hype" is fashionable. Fractals and fuzzy logics are not taken seriously; multimedia and on-demand publishing are victims of their high profile in the press, with expectations often being too high.

(d) Both math and publishing are subject to market forces. The abacus came only into real use when the average man in the street in the late middle ages needed to count beyond a dozen or so. Today multimedia remains a zero-billion dollar business, with many initiatives failing after a few months. To be successful, one needs customer awareness, the willingness and the ability to absorb the technology, and make it feasible. Only then will the market catch on.

(e) Research and innovation must be in synch with the needs. The strength of math is its unity, but that is also its weakness. The strength of electronic publishing is that it can draw on different fields and tap every-day innovation. However, it needs serious quality design, a deep knowledge structure, and above all, an understanding of user needs. A research-driven collaborative effort needs to be set up as a way to develop adequate tools for the user community. Therefore, researchers should remain in permanent contact with the publishers.

For all these reasons, Roberto thought that, at present, and probably in the future, content is and will be more valuable than the application used to capture or deliver it.

A fruitful experience

The possibility of discussion with people attending the various conferences and workshops allowed all participants to widen their horizons beyond their direct field of interest, and to profit from the expertise in these other areas, so I think that the idea of running these conferences in parallel at the same place was an extremely fruitful one, which I hope will be repeated in the future.

As a final remark let me mention that the TEX world was extremely well represented in this cosmopolitan audience (the 87 participants of RIDT'94 and the 105 participants of EP'94 came from about 20 different countries), since I counted at least five active TEX hackers from France, four from Germany, and one each from Great Britain, the Czech Republic, and ... Switzerland.

◇ Michel Goossens
CERN, CN Division
CH1211 Geneva 23, Switzerland
michel.goossens@cern.ch

Dreamboat

Report on the 2nd meeting of the $\mathcal{NTS}$ group, February 1994

Philip Taylor,
Technical Director, $\mathcal{NTS}$ project

The $\mathcal{NTS}$ ('New Typesetting System') group met during February 1994 at Münster in Germany, where they were once again the guests of DANTE. Those present at the meeting included:

Joachim Lammarsch, Executive Director;
Philip Taylor, Technical Director;
Volker Schaa, Minutes Secretary;
Prof. Dr. Peter Breitenlohner;
Bernd Raichle;
Dr. Rainer Schöpf;
Friedhelm Sowa;
Dr. Jiří Zlatuška.

1 Changes in membership: Joachim Schrod had resigned from the group, and Jiří Zlatuška had been invited to join. Also invited to join, but not present at the meeting, were Prof. Dr. Klaus Lagally and Prof. Richard Palais, the latter electing to be a 'corresponding member'.

2 Activities: the group agreed to continue the three-way effort into (a) the specification of the canonical TeX kit (but now in conjunction with TUG), (b) extended TeX (ε-TeX), and (c) $\mathcal{NTS}$ via re-implementation in a rapid prototyping language such as CLOS or Prolog.

3 Division of responsibility: Joachim Lammarsch would continue to take overall responsibility for the financial and political aspects of the project, with Friedhelp Sowa as Treasurer; Philip Taylor would continue to take overall technical responsibility, with (a) Rainer Schöpf and Friedhelm Sowa taking primary and secondary responsibility for the specification of the 'canonical TeX kit'; (b) Peter Breitenlohner and Bernd Raichle taking primary and secondary responsibility for ε-TeX, and (c) Jiří Zlatuška and Bernd Raichle taking primary and secondary responsibility for $\mathcal{NTS}$. Volker Schaa is to take over from Marion Neubauer as minutes secretary.

4 The group agreed that $\mathcal{NTS}$ should be represented at TUG '94, provided that funding can be obtained.

5 The group made a start on discussing the basic details of the 'canonical TeX kit', but far more work is needed in this area; it was interesting to note that there was perhaps greater disagreement within the group in this area than in any other! Clearly close liaison with TUG and with the TeX user community will be needed if the results are to be universally acceptable.

6 The group agreed that Peter Breitenlohner's work on ε-TeX formed an excellent starting point; the ε-TeX sub-group spent much time discussing implementation details, and agreed the following desiderata:

(a) The WEB source should be capable of generating a 100% compatible TeX, as well as ε-TeX;

(b) The Ini-ε-TeX executable should be capable of generating either a 100% compatible TeX format, or an ε-TeX format, if invoked with no format specified; a command-line specifier, analogous to &, would indicate whether a TeX format or an ε-TeX format was to be generated;

(c) Within Vir-ε-TeX, 'harmless' extensions (those which generate new primitives but do not otherwise modify the semantics of TeX) should be permanently enabled; 'potentially harmful' extensions (those which change the semantics of TeX) should be disabled by default, but individually enableable at the document level; such extensions will need to be enabled/disabled globally in the first release, but this restriction may be eliminated in future releases. Initially disabled primitives should be associated with inaccessible primitives; enabling should be defined as equivalent to `\let`ting an accessible primitive equal the corresponding inaccessible primitive. Disabling should be *defined* for all enableable features, but the effect may be within the set {done, implementation-dependent?, disallowed}. Grouped primitives can be enabled/disabled only as a group; an attempt to enable a primitive which has ceased to be undefined should yield an error. Further investigation is needed into `*\ifdefined`, the ability to preserve the current state of enabled features, and the ability to return to that state at a later point.

7 The group agreed that further discussions with Professor Keith Bennet regarding the use of his research system for reverse-engineering TeX would be worthwhile, and this was to be pursued.

8 Some informal discussions took place concerning a disclaimer to accompany ε-TeX, the idea being to ensure that any bugs therein be clearly identifiable as being solely the responsibility of the group and

not of DEK; this disclaimer should also ensure that DEK was given full and proper credit for the vast majority of the algorithms and code used, and for his foresight in developing TEX in a highly literate manner.

9 It was informally proposed that the working logo for the ε-TEX project should be `\varepsilon` (as used by the LATEX2e project), hyphen, canonical TEX logo, with possible improvements to the overall kerning; the working logo for the $\mathcal{NTS}$ project would be `$\cal{NTS}$`, with perhaps additional kerning and/or raising/lowering (it was also suggested that the final form should specify a different colour for each letter).

10 Subject to necessary funding being available, the group next hoped to meet at EuroTEX '94 in Gdansk; work would continue in the meantime with communications via electronic mail.

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

Fonts

{Meta}Font Forum Redux

Alan Hoenig

Georgia Tobin's *Metafont Forum* was a personal favorite of mine for the several years that she moderated it. Fonts are such neat things. It is now time to revivify this column, and I hope you will join with me in this endeavor.

I hope to bring one article — more or less — to these pages in every issue of *TUGboat*. My work will be much easier if you readers take these words to heart and favor your colleagues with articles on fonts.

What kind of articles, exactly? Well, we welcome almost anything of or pertaining to METAFONT. But it is possible to write about fonts in other contexts, and so we welcome those contributions as well. (Of course, articles on any other aspect of TEX are more than welcome to the pages of *TUGboat* if not this particular space.)

There are lots of things to think about. First of all, is METAFONT, with its bitmapped output, all washed up in a world that's increasingly outline-font-oriented? What kinds of tricks can you play with outline fonts (just for fun, of course)? What happens if we add a touch of randomness when we create a meta-font? And much, much more.

If you'd like to contribute, please drop a line, or mail your article to myself or Barbara Beeton. Email *or* snail mail cheerfully accepted.

◇ Alan Hoenig
John Jay College / CUNY
Mail: 17 Bay Avenue
Huntington, NY 11743 USA
(516) 385-0736
ajhjj@cunyvm.cuny.edu

ASCII.sty (version 1) A new style-option and encoded font with IBM graphics control characters for use with TEX & LATEX.

R. Ramasubramanian
R.W.D. Nickalls
M.A. Reed

Abstract

This article describes a new font and style option (ASCII.STY) which includes the IBM graphics characters commonly used by PÇs to represent the control codes (ASCII 1–31). In particular, this article does three things: (a) it announces and describes a pk rendering of the Type 1 Courier font donated to the X Consortium by IBM; (b) a LATEX style option is presented which uses this font; (c) the method used to create the font is described in detail.

1 Introduction

ASCII.STY[1] and the related font files shown in Table 1 which constitute the package 'ASCII', were developed while we were writing a book on *serial interfacing*[2]. We were using LATEX and wanted to include the usual table of ASCII control codes and their equivalent IBM graphics characters as used by PCs. However, in spite of searching all the available archives, we were unable to find a TEX font containing *all* the graphics characters of the 7-bit ASCII code (e.g. ☺ ♪ ♫ ☼ └ ¦ ⌂ etc.).

One of us (Mark Reed) noticed that the same problem was raised in a recent issue of UKTEX Digest[3], in which it was suggested that a possible solution might be to derive the characters from the public domain PostScript IBM COURIER font donated by IBM to the X Consortium.

We therefore downloaded the IBM COURIER font, which consists of two files COUR.AFM and COUR.PFA, and duly got to work extracting the characters and generating the necessary .TFM and bitmapped .PK files.

In practice, however, it was not at all easy owing to a paucity of documentation on just exactly how to do this. This article therefore details what we did and how we did it, and ASCII.STY is the result.

In order to avoid any confusion, and to clearly distinguish the new derivative font from the original IBM COURIER font, this document will refer to the new derivative font as ASCII-10.

We have restricted our efforts to the first 128 characters of the PC character set because the term ASCII, as used by the original ANSI standard[4], refers only to the first 128 characters.

Those who are interested in unmasking the characters 128–255 in the PC character set may wish to examine some of the other .AFM files held in the CTAN archive (e.g. CRB-PC8.AFM; CRI-PC8.AFM).

2 The ASCII font

Table 2 shows the 10-point ASCII-10 font which we derived from the IBM COURIER font. Different sizes can be generated using the TEX magstep procedure.

Each element in the font table can be accessed in the usual way by using the `\char` command. For example, the style option ASCII.STY allocates the command `\CR` for the Carriage Return character as follows.

1. The new font is defined as `\ascii` using `\newfont{\ascii}{ascii-10}`
2. The new command `\CR` is defined using `\newcommand{\CR}{\char'015}`.

Now when the command `{\ascii\CR}` is used in the text it produces the symbol ♪. The brackets are needed to keep the `\ascii` font local.

3 Installing the font files

EmTEX implementation

In order to use ASCII.STY the various font files (.PK and .TFM files) need to be placed in the correct directories, as specified by EmTEX.

ASCII-10.TFM file: This file goes into the usual \EMTEX\TFM directory.

ASCII.FLI library: This library of all ASCII.PK files goes into the usual \TEXFONTS directory.

Non-EmTEX implementations

For those not using EmTEX the separate ASCII.PK files are also provided. Although all the .PK files initially have names which indicate the dpi size (e.g. ASCII300.PK), once correctly located in their respective subdirectories, they must all then be renamed so that they all have the same name ASCII-10.PK.

[1] The complete package of files (ASCII) described in this article (see Table 1) can be found in the CTAN archive (/FONTS/ASCII).

[2] Nickalls RWD and Ramasubramanian R (1994). Interfacing the IBM PC to medical equipment: the art of serial communication. Cambridge University Press (in press).

[3] See note by Sebastian Rahtz on 'AFM files for IBM Courier' in UKTeX Digest, Friday 18 Dec 1992, Vol 92: Issue 47.

[4] American National Standard X3.4 (1986). Available from the American National Standards Institute, 1430 Broadway, New York, 10018.

File name	Size*	Description
ASCII.STY	5805	The ASCII style option.
ASCIISTY.TEX	27437	Source file for ASCII package documentation.
ASCIISTY.DVI	38796	ASCII package documentation .DVI file.
ASCII-10.TFM	772	TeX Font Metric file for ASCII-10 font.
ASCII.ENC	2459	The ASCII-10 PostScript encoding vector used by AFM2TFM and PS2PK to rearrange the font.
ASCII.FLI	55121	Font library containing all the ASCII.PK fonts.
ASCII300.PK	5140	10-point ASCII-10 font file.
ASCII329.PK	5588	11-point ASCII-10 font file.
ASCII360.PK	6016	12-point ASCII-10 font file.
ASCII432.PK	7068	14-point ASCII-10 font file.
ASCII518.PK	8596	17-point ASCII-10 font file.
ASCII622.PK	10188	21-point ASCII-10 font file.
ASCII746.PK	12336	25-point ASCII-10 font file.
UKTEXDIG.DOC	1783	A short note by Sebastian Rahtz.
COUR.AFM	29886	Adobe Font Metric file for IBM COURIER font.
COUR.PFA	128018	PostScript Font ASCII file for IBM COURIER font.
COPYRIGH.DOC	523	The copyright notice by IBM.
A4.STY	11070	Style option needed for typesetting ASCIISTY.TEX.
README.DOC	2987	Information for unpacking ASCII.STY.

Table 1: List of all files contained in the ASCII package (CTAN archive /fonts directory).
* The size given here relates to the file size in bytes on IBM compatible PCs. The point size of the .PK files relates to devices having a 300 dpi resolution.

	´0	´1	´2	´3	´4	´5	´6	´7	
´00x		☺	☻	♥	♦	♣	♠	•	″0x
´01x	◘	○	◙	♂	♀	♪	♫	☼	
´02x	►	◄	↕	‼	¶	§	▬	↨	″1x
´03x	↑	↓	→	←	∟	↔	▲	▼	
´04x		!	"	#	$	%	&	'	″2x
´05x	(	)	*	+	,	-	.	/	
´06x	0	1	2	3	4	5	6	7	″3x
´07x	8	9	:	;	<	=	>	?	
´10x	@	A	B	C	D	E	F	G	″4x
´11x	H	I	J	K	L	M	N	O	
´12x	P	Q	R	S	T	U	V	W	″5x
´13x	X	Y	Z	[	\	]	^	_	
´14x	`	a	b	c	d	e	f	g	″6x
´15x	h	i	j	k	l	m	n	o	
´16x	p	q	r	s	t	u	v	w	″7x
´17x	x	y	z	{	¦	}	~	⌂	
	″8	″9	″A	″B	″C	″D	″E	″F	

Table 2: The ASCII-10 font.

3.1 Adjusting the configuration file (LJ.CNF)

In order that EmTEX can find all fonts, the path for the fonts in the LJ.CNF file (in the \EMTEX directory) may need to be altered to accommodate the directory structure of \TEXFONTS.

The authors use the following library line (/pl) in the LJ.CNF file.

```
/pl=%DVIDRVFONTS%;lj_0;lj_h;lj_1;lj_2;
  lj_3;lj_4;lj_5a;lj_5b;lj_6;ascii;
```

4 Using ASCII.STY

Place ASCII.STY in the \TEXINPUT directory, and then use the style option [ASCII] in the \documentstyle command in the usual way. ASCII.STY includes seven magnifications of the ASCII-10 font as follows: magstep 0, half, 1, 2, 3, 4, 5, which are equivalent to point sizes 10, 11, 12, 14, 17, 21, 25 respectively. Note that the relationship between point size and magstep is such that the point size corresponding to magstep **n** is 10×1.2^n, i.e. magstep 0 is equal to 10-point size.

Using ASCII.STY an ASCII character is most easily printed using the \ascii command followed by the character in the usual way. For example, {\ascii 5} will print the number 5 from the ASCII-10 font table. The ASCII.STY style option will automatically select the font size currently being used in the document (e.g. 10, 11, 12 point). For convenience all the control characters except the four DC_{1-4} have been allocated their usual code name (in *uppercase* letters); thus the carriage return symbol (♪) is called \CR and would be printed using {\ascii\CR}. The four Device Control characters DC1, DC2, DC3, DC4 have been allocated the commands \DCa, \DCb, \DCc, \DCd, since TEX does not allow commands to contain integers.

Two other characters in the ASCII-10 font table which are not usually available for TEX users are

- ⌂ (\DEL) which is printed using {\ascii\DEL}.
- ¦ (\splitvert) which is printed using {\ascii\splitvert}. This name was chosen to be in keeping with Knuth's \vert which is |.

The standard codes and the coding used by ASCII.STY for the control characters are shown in Table 3.

4.1 Using other sizes

When an ASCII character having a different point size to the one being used for the document is required, then a different command without integers has to be used.

In view of this restriction ASCII.STY uses Roman numerals to indicate both point size (e.g. \asciiXII for 12-point size) and magstep number (e.g. \asciimsV for magstep 5), as shown in Table 4. For example, a 21-point heart shape ♥ can be printed using either of the commands {\asciiXXI\ETX} or {\asciimsIV\ETX}.

Examples of the characters and sizes given by the commands are shown below.

The 10 point font: {\asciimsZ\SO} and
{\asciiX\SO} give ♫
The 11 point font: {\asciimsH\SO} and
{\asciiXI\SO} give ♫
The 12 point font: {\asciimsI\SO} and
{\asciiXII\SO} give ♫
The 14 point font: {\asciimsII\SO} and
{\asciiXIV\SO} give ♫
The 17 point font: {\asciimsIII\SO} and
{\asciiXVII\SO} give ♫
The 21 point font: {\asciimsIV\SO} and
{\asciiXXI\SO} give ♫
The 25 point font: {\asciimsV\SO} and
{\asciiXXV\SO} give ♫

5 Generating the ASCII font.

As indicated earlier, the IBM COURIER font files (COUR.AFM and COUR.PFA) contained all the graphics characters needed for typesetting a 7-bit ASCII table. In order to make use of this font in an non-PostScript TEX or LATEX environment, we needed to generate the following:

- A .TFM file from the corresponding PostScript .AFM file.
- An MS-DOS binary .PFB font file from the corresponding .PFA font file (a Unix text file).
- A bit-mapped .PK font file from the corresponding .PFB file.

The first step requires a program called AFM2TFM; the second step requires a program called T1BINARY; and the third step requires the program PS2PK. Fortunately, all the programs required for performing these conversions on a PC are readily available from **ftp.urc.tue.nl [131.155.2.79]** in their /TEX/EMTEX directory. The various steps are shown in Figure 1.

5.1 Generating the .TFM file

Normally this would be straightforward since the AFM2TFM program reads a PostScript font metric information file (i.e. an .AFM file) and produces a .TFM file.

In practice, it was not easy to establish which characters were in the COUR.AFM file since there was no documentation. For example, we could only discover by trial and error that the line

ASCII control codes						
Dec	*Code*	*Hex*		*Name*	*IBM*	ASCII.STY
0	NUL	00h	^@	Null	—	—
1	SOH	01h	^A	Start of heading	☺	\SOH
2	STX	02h	^B	Start of text	☻	\STX
3	ETX	03h	^C	End of text	♥	\ETX
4	EOT	04h	^D	End of transmission	♦	\EOT
5	ENQ	05h	^E	Enquiry	♣	\ENQ
6	ACK	06h	^F	Acknowledge	♠	\ACK
7	BEL	07h	^G	Bell	•	\BEL
8	BS	08h	^H	Back space	◘	\BS
9	HT	09h	^I	Horizontal tab	○	\HT
10	LF	0Ah	^J	Linefeed	◙	\LF
11	VT	0Bh	^K	Vertical tab	♂	\VT
12	FF	0Ch	^L	Formfeed	♀	\FF
13	CR	0Dh	^M	Carriage return	♪	\CR
14	SO	0Eh	^N	Shift out	♫	\SO
15	SI	0Fh	^O	Shift in	☼	\SI
16	DLE	10h	^P	Data link escape	►	\DLE
17	DC1	11h	^Q	Device control 1 (XON)	◄	\DCa
18	DC2	12h	^R	Device control 2	↕	\DCb
19	DC3	13h	^S	Device control 3 (XOFF)	‼	\DCc
20	DC4	14h	^T	Device control 4	¶	\DCd
21	NAK	15h	^U	Negative acknowledge	§	\NAK
22	SYN	16h	^V	Synchronous idle	▬	\SYN
23	ETB	17h	^W	End transmission block	↨	\ETB
24	CAN	18h	^X	Cancel	↑	\CAN
25	EM	19h	^Y	End of medium	↓	\EM
26	SUB	1Ah	^Z	Substitute (EOF)	→	\SUB
27	ESC	1Bh	^[	Escape	←	\ESC
28	FS	1Ch	^\	File separator	∟	\FS
29	GS	1Dh	^]	Group separator	↔	\GS
30	RS	1Eh	^^	Record separator	▲	\RS
31	US	1Fh	^_	Unit separator	▼	\US
124	—	7Ch	—	—	¦	\splitvert
127	DEL	7Fh	—	Delete	⌂	\DEL

Table 3: ASCII table of control characters.

Using point sizes		Using magstep Nos.	
10	\asciiX	Zero (msZ)	\asciimsZ
11	\asciiXI	Half (msH)	\asciimsH
12	\asciiXII	1 (msI)	\asciimsI
14	\asciiXIV	2 (msII)	\asciimsII
17	\asciiXVII	3 (msIII)	\asciimsIII
21	\asciiXXI	4 (msIV)	\asciimsIV
25	\asciiXXV	5 (msV)	\asciimsV

Table 4: Font size commands for point and magstep, as used by ASCII.STY.

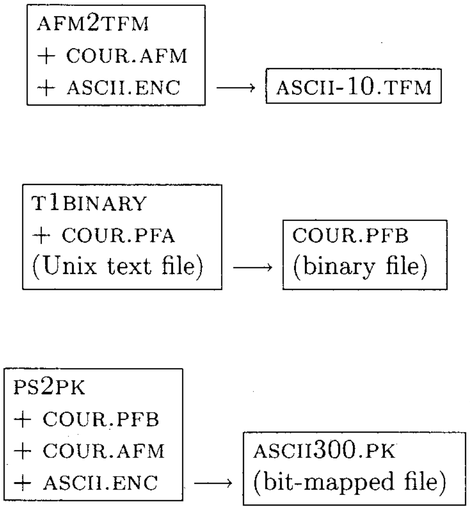


Figure 1: Sequence of file changes to generate ASCII-10.TFM and ASCII300.PK.

```
C -1 ; WX 600 ; N SV040000 ;
  B 102 96 498 467 ;
```

referred to the ASCII character ▾. Note that the C −1 code indicates that the character is *not* 'visible'.

Once we had established the relative positions of the characters in the .AFM file, it was fairly easy to generate an 'encoding vector' file (ASCII.ENC). The following is an extract from the ASCII.ENC file:

```
/asciiEncoding [
  /spacenumeric  /SS000000    /SS010000
  /heart         /diamond     /club
  /spade         /bullet      /SM570001
  :
  :
  /.notdef  /.notdef  /.notdef  /.notdef
] def
```

The ASCII.ENC file contains 128 character names each preceded by /. The order of the names in this file specifies the order of the characters in the TEX font. Any character position which is not defined (i.e. those character positions with a value > 127) are designated /.notdef.

Once we had generated a complete ASCII.ENC file, the ASCII-10.TFM file was generated using the following command.

```
afm2tfm cour.afm -T ascii.enc
  ascii-10.tfm
```

Newer versions of AFM2TFM will automatically print out a line suitable for inclusion in PSFONTS.MAP for use with DVIPS. **Ignore** the automatically generated line and see the section below on using DVIPS for more information.

For further information on AFM2TFM, the reader should refer to the documentation which accompanies this program. Readers who wish to know more about the codes found in .AFM files should consult the PostScript Language Reference Manual [1st edition] published by the Adobe Corporation.

5.2 Conversion of .PFA to .PFB format

When PS2PK is run on PCs with MS-DOS, the program will only read .PFB files and not .PFA files. In view of this, it was necessary to first convert the .PFA font file (which is a Unix text file), into an MS-DOS binary (.PFB) font file using the program T1BINARY, using the following command:

```
t1binary cour.pfa cour.pfb
```

5.3 Generating the .PK bit-mapped file

The .PFB file generated in the previous step, can now be used by PS2PK to produce a .PK font file. This is done with the following command (note that PS2PK requires that COUR.AFM is also in the same subdirectory):

```
ps2pk -e ascii.enc cour.pfb ascii300.pk
```

This produces a 10-point ASCII300.PK file (i.e. ASCII-10) for an output device with a resolution of 300dpi. For further information regarding the command line switches and the use of PostScript fonts with TEX and LATEX, please read the detailed documentation which comes with the PS2PK package.

6 Magnifying the ASCII-10 font

As indicated earlier, the steps described above will produce the font at its optimal 10-point design size i.e. ASCII-10 (since it was derived from the 10-point IBM COURIER font) and for an output device with a resolution of 300dpi.

In order to produce an ASCII table with larger size graphics characters in it, additional .PK files must be generated (e.g. ASCII360.PK) which contain magnified versions of these characters. This is done using PS2PK with the -X flag as follows.

```
ps2pk -e ascii.enc -X360 cour.pfb
  ascii360.pk
```

Using -X360 will produce a 12-point font by magnifying a 10-point design size font. By varying the -X value different magnified sizes can be obtained.

When all the necessary .PK files have been produced, they can be gathered into a single compact font library (ASCII-10.FLI in this case) using a program called FONTLIB. This program and its documentation may be found in the standard EmTEX package in the CTAN archive.

7 Using ASCII-10 with Tomas Rokicki's DVIPS

Using any PostScript font is extremely easy with DVIPS, and ASCII-10 is no exception. Simply copy the COUR.AFM, COUR.PFB and ASCII.ENC files to DVIPS's header directory. Then add the following line to the end of PSFONTS.MAP.

```
ascii-10 Courier <cour.pfb
  " asciiEncoding ReEncodeFont "
  <ascii.enc
```

Don't use the line generated by AFM2TFM as it will not work!

Once DVIPS has been configured in this way, it will automatically use the PostScript outline instead of using a bitmap. This gives much better output quality and is resolution independent.

◇ R. Ramasubramanian *MBBS, FRCA.*
Department of Anaesthesia,
University Hospital,
Queen's Medical Centre,
Nottingham, NG7 2UH,
England, UK.
100016.3245@CompuServe.com

◇ R.W.D. Nickalls *BSc, PhD, MBBS, FRCA.*
Senior Lecturer and Consultant,
Department of Anaesthesia,
University Hospital,
Queen's Medical Centre,
Nottingham, NG7 2UH,
England, UK.
dick.nickalls@nottingham.ac.uk
100115.1010@CompuServe.com

◇ M.A. Reed
Cambridge University Press,
Technical Applications Group,
Printing Division,
University Printing House,
Shaftesbury Road,
Cambridge, CB2 2BS
England, UK.
mreed@cup.cam.ac.uk

Book Reviews

Book review: *The LaTeX Companion*

David M. Jones and David E. Wald

> *In darkness languishes the precious stone.*
> *When will its excellence enchant the world?*
> – Barry Hughart

Michel Goossens, Frank Mittelbach and Alexander Samarin, *The LaTeX Companion*. Addison-Wesley, 1993. xxx + 492 pp. + index (36 pp.) ISBN 0-201-54199-8.

Without question, this book is absolutely necessary for all who use, support, modify, or rue the day that they first heard of LaTeX. You should all run right out and get a copy. We'll wait.

⋮

All set? Good. Now we'll tell you what you've just acquired.

In recent years there has been an explosion of books about TeX and LaTeX, and the majority of LaTeX users have probably gone from gnashing their teeth over the lack of supplemental documentation to gnashing their teeth over which of the numerous books on the market are the best investment for their time and money. Let there be no confusion about the *LaTeX Companion*: the *Companion* belongs next to the *TeXbook* and the LaTeX manual as a book that every LaTeX user should have available.

Since its introduction in the early 80's, LaTeX has been the center of a flurry of grassroots development, with contributors world-wide producing a variety of style files and support packages, many of which have become basic necessities to any well-dressed LaTeX installation. In recent years, this development has reached a minor crisis point, with several slightly incompatible variants of LaTeX coming into common use.

As anyone who has been following the recent development of LaTeX knows, the LaTeX3 team, spearheaded by Frank Mittelbach, Chris Rowley, and Rainer Schöpf, has taken over the development of LaTeX, and is in the process of producing a completely rewritten LaTeX3 [1–3]. Recently, the LaTeX3 team, in an attempt to head off the aforementioned crisis, announced the imminent release of a single "state of the art" version of LaTeX, to be designated LaTeX2$_\varepsilon$.

Along with the "crisis of infinite formats", there has been a corresponding crisis in documentation. Beyond Lamport's manual presenting the original core of LaTeX, there has been no central source that systematically presented the most useful style files, the most necessary support programs and, frankly, the best-known tricks for convincing LaTeX to behave itself. Even when documentation exists, it has tended to circulate in the form of electronic folklore, or unpublished documents distributed with individual macro packages, supplemented by the occasional TUGboat article. The unfortunate result is that many worthy packages have languished in darkness for lack of widely-available documentation.

In other words, there exists a gap between the numerous LaTeX primers now available on the market and the various comprehensive lists and archives of LaTeX styles now available electronically. The authors of this book have taken it upon themselves to fill this gap. They are in a particularly good position to do so, as one of them is a member of the core LaTeX3 team, and all three have extensive experience in supporting LaTeX and initiating novices into its deeper mysteries.

The *Companion* is designed to be used alongside Lamport's LaTeX manual; as such, it assumes that the reader is familiar with the core system as documented in Lamport's book, and gives frequent cross-references to the LaTeX manual. Each chapter focuses on a different aspect of document preparation and presents a carefully-chosen combination of packages, utilities, and less well-known features, which allow the user to transcend the boundaries of Lamport's book.

Since Chapter 2 is in many ways typical of the rest of the book, we will give an extended description of it to illustrate the flavor of documentation provided by the *Companion*. Chapter 2 covers the physical and logical structure of a LaTeX document. The first section covers the package and class interface to LaTeX2ε, which replaces the `\documentstyle` interface of LaTeX 2.09. Succeeding sections cover sectioning commands, tables of contents, and cross-referencing.

The discussion of sectioning commands begins with a brief summary of their syntax and use, covering roughly the material presented in Section 2.2.3 and Appendix C.3 of the LaTeX manual. The authors don't stop here, however; they also present a wealth of information on modifying the behavior of sectioning commands and on defining new ones. (As anyone who has supported LaTeX — or who has read any newsgroup or mailing list where LaTeX is discussed — well knows, this is exactly the sort of question that is asked repeatedly by LaTeX users, and which the determined user has usually had to delve into the LaTeX source code to answer.)

The discussion of tables of contents follows similar lines, and is supplemented by a description of the nonstandard `minitoc` package, which can be used to create a separate table of contents at the beginning of each chapter of a book or report.

The chapter concludes with a description of LaTeX's cross-referencing features, together with descriptions of the `varioref` and `xr` packages, which support alternate styles of referencing and cross-references between documents.

Moving along, Chapter 3 covers the formatting of paragraphs, lists, and other structures; Chapter 4 discusses page layout; and Chapters 5 and 6 cover tables and floats in depth.

Chapter 7 presents the first of several major extensions to LaTeX, the LaTeX2ε font selection scheme. Since the New Font Selection Scheme is the single largest difference between LaTeX 2.09 and LaTeX2ε, this chapter is perhaps the most important in the book. The scheme is indeed described well, with the first sections providing a detailed introduction to the basic vocabulary and features of font selection, and the latter sections providing full details for those wishing to install new fonts.

Chapter 8, entitled "Higher Mathematics", will be welcome reading to anyone to whom mathematical typesetting is of more than passing interest. It has long been a source of amusement — and occasional chagrin — that although TeX's *raison d'être* has always been its superb support for mathematical typesetting, LaTeX hardly represents an advantage over plain TeX in this regard, and has been lagging behind AMS-TeX for some years. Thus, we were delighted when AMS-LaTeX was released a few years ago, and are now pleased that AMS-LaTeX has been fully legitimized by Chapter 8, which provides a detailed introduction to and description of the various features of this excellent package. Armed with the *LaTeX Companion*, we will now continue with renewed vigor our campaign to see that every LaTeX site installs the AMS-LaTeX package and brings LaTeX out of the dark ages of mathematical typesetting.[1]

Chapter 9 addresses another issue that has been of burning interest in the LaTeX community for several years — support for typesetting in languages other than American English. The authors briefly

[1] While approving of the sentiment, one of the authors of this review would like to distance himself from the evangelical tone of this paragraph, and wishes to remind the other author that there are several full-flavored decaffeinated coffees on the market.

discuss the handling of alternate character sets and encodings such as the Cork standard, and then discuss the Babel package of Johannes Braams, as well as an alternate package for French typography.

Chapter 10 discusses several extensions to LaTeX's picture environment, a subject which leads naturally to Chapter 11, which tackles the complex issues surrounding the use of PostScript in LaTeX documents. This chapter covers such diverse topics as choice of DVI device driver, use of PostScript graphics, installation of PostScript fonts, and various PostScript `\special` effects.

Chapters 12 and 13 discuss two auxiliary programs that, although they are an essential part of any LaTeX installation, have long suffered from a dearth of documentation. Chapter 12 describes `makeindex` in detail, bringing into print for essentially the first time a comprehensive discussion not only of the LaTeX interface for creating indexes and the use of the `makeindex` program itself, but also of `makeindex` style files and packages for producing documents with multiple indexes. Chapter 13 is a similarly detailed description of BibTeX. This chapter, in fact, seems capable of replacing not only Appendix B of Lamport's book, but also much of the documentation supplied with BibTeX itself, since it not only subsumes them, but also offers an extensive list of non-standard BibTeX style files.

Chapter 14 describes the `doc` system for writing self-documenting LaTeX code and perhaps should have been merged with Appendix A, which is an overview of LaTeX for writers of packages and classes. Appendix B rounds the book out with a brief introduction to the major resources for TeX users. The first part of this appendix describes the major electronic sites that serve as sources for TeX software (which will surely be appreciated by those whose curiosity has been piqued by the rest of the book), and the second gives a list of TeX user groups throughout the world. Regrettably missing is a separate list of the major mailing lists and newsgroups devoted to TeX.

The book ends with a well-annotated bibliography that should serve as a valuable introduction to the literature, a well-constructed — though occasionally incomplete — index, and a combination colophon and empirical software study that should prove enlightening to anyone who has used or is contemplating using LaTeX to produce a book-length document.

Although we recommend the *Companion* without hesitation, there are a few caveats worth mentioning. The major one is that this book, like LaTeX2ε, is a snapshot of a rapidly changing system. As such, there are aspects of the book that are sure to be confusing to anyone who has not been following the development of LaTeX over the past several years, and there are other aspects that are likely to become obsolete in the near future. For example, although the discussion of BibTeX in Chapter 13 is comprehensive as of this writing, this may cease to be true with the eventual release of BibTeX 1.0. Similarly, all of the cross-references to the LaTeX manual refer to the first edition of the book, which will soon be superseded.[2]

Another problem is that the very thoroughness of the book may occasionally present obstacles for readers who merely want to use the new features documented in the book without becoming intimately acquainted with all aspects of the new system. For example, although the description of the font system in Chapter 7 is very detailed, this means that the reader is ten pages into the chapter before being told the now-preferred method for changing to an italic font. Similarly, there is no place where the reader can turn to find a brief summary of the most important differences between LaTeX 2.09 and LaTeX2ε.[3] We hope that with the release of a new edition of Lamport's LaTeX manual, beginners will be able to find the answers to their questions with less difficulty.

A final difficulty with the book results from LaTeX itself. Experience with LaTeX has taught us that there are many perfectly reasonable things a user wishes to do that are impossible in LaTeX with the published interface. Such tasks often require a use of LaTeX's internal commands, and the *Companion* has to deal with this. To their credit, the authors have handled this delicate situation with tact, presenting a careful selection of internals without requiring the user to understand the entire composition of the program. Naturally, it is our hope that with the eventual release of LaTeX3, this last difficulty (and, dare we hope, the need for two separate manuals) will disappear.

[2] A related problem is that LaTeX2ε itself is still under development. Although a stable distribution version of LaTeX2ε is expected soon, many of the packages described in the *Companion* have not yet been converted into LaTeX2ε packages, which is likely to cause discrepancies between the actual behavior of some packages and the behavior described in the *Companion*.

[3] Most likely, this reflects the history of the book: When the book was begun, there were no plans to release a new version of LaTeX, and indeed, production of the book was well underway before the necessity of LaTeX2ε became evident.

References

[1] Frank Mittelbach and Chris Rowley. LaTeX 2.09 ↪ LaTeX3. *TUGboat*, 13(1):96–101, April 1992.

[2] Frank Mittelbach and Rainer Schöpf. With LaTeX into the Nineties. *TUGboat*, 10(4):681–690, December 1989.

[3] Frank Mittelbach and Rainer Schöpf. Towards LaTeX 3.0. *TUGboat*, 12(1):74–79, March 1991.

◇ David M. Jones
MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139
dmjones@theory.lcs.mit.edu

◇ David E. Wald
MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139
wald@theory.lcs.mit.edu

Book review: *Making TeX Work*, by Norman Walsh

Victor Eijkhout

Norman Walsh, *Making TeX Work*. O'Reilly & Associates, Inc., 1994, 483 pages, ISBN 1-56592-051-1.

Most TeX books limit themselves to the TeX language proper, and when the subject of actually printing something comes up, mumble something about a mythical object called the *Local Guide*. Not so *Making TeX Work*. Arguing that the TeX environment is mature and stable enough to warrant a 'global guide', this book sets out to be such a document. It lists many macro collections, utilities, and matters related to fonts.

The book is divided into three parts and a couple of appendices. Part 1, 'An Introduction to TeX', has chapters on TeX itself, editing, running TeX, and macro packages. Part 2, 'Elements of a Complex Document', is the most interesting. Its chapters concern themselves with fonts, pictures and figures, international considerations, printing, previewing, online documentation, Metafont, and Bibliographies, Indexes, and Glossaries. Part 3, 'A Tools Overview', concerns itself with commercial and non-commercial TeX environments, TeX on the Macintosh, and TeX utilities. The appendices are on filename extensions, font samples, resources, and longer examples (mostly shell scripts).

Some of the topics discussed in this book are by definition system-dependent. Therefore this book will be of most use to you if your platform is Unix, MSDOS, OS/2, or the Apple Macintosh. Otherwise, the chapters on editing or previewing will not be particularly relevant in your case. On the other hand, if you have Gnu Emacs, emTeX, or a PostScript printer, you will find a lot of detailed information here. There is a slight bias in the book towards these common tools, but not to an extent that it would be de facto an MSDOS book or a Unix book.

This book is concerned largely with those aspects of TeX that are system-dependent, with the exception of a chapter on macro packages and a token chapter on the workings of TeX. Thus, there is no mention of the various LaTeX styles written by third parties, and the list of literature does not mention books such as the *LaTeX Companion* which does mention these, nor any books (other than *The TeXbook*) that discuss TeX programming.

On the other hand, there is extensive discussion of drawing packages, BibTeX and related utilities, editors, previewers, and matters pertaining to fonts. In particular, this last topic is treated in great detail, covering such aspects as encoding vectors which I have never seen discussed before.

The amount of detail in which resources are discussed varies. Several pages are devoted to emTeX and how to install it, while simple BibTeX-related utilities are given just a few lines. While this is justified, I find that some of the shorter entries are too short: often nothing is mentioned on what platform they run, or in what scripting language they are written. I suspect a Unix bias.

The discussion of various TeX implementations can be of use to someone just piecing together a TeX system, but I think the main value of the book is to alert existing TeX users to various utilities and tools that can make life easier. From this point of view the book contains a mass of information on things you may never have suspected existed.

One of the reasons a book like this can exist is that many TeX resources are reachable by a large segment of the TeX users community: there are well-established archive sites, in particular those of the Comprehensive TeX Archive Network, and many people can ftp to them directly, or use ftp-mail (the book contains an explanation of ftp and ftp-mail, as well as, several hundred pages later, a brief mention of the World Wide Web). Still, not everyone is electronically connected, and I consider it an omission that the book does not mention the archives that are available on CD-Rom, or user groups that can be of assistance in obtaining materials.

Finally, it is customary to mention briefly the layout and design of any book about TEX. Although the author and publisher consider this book an 'excellent example' of a beautiful book typeset with TEX, I do not entirely agree. With the many white spaces around illustrations and headings, it is a less than felicitous idea to separate paragraphs by white space, instead of simply indenting them. Also, the only distinction between levels of headings is through font sizes. Any real size difference, however, is dwarfed by the difference between headings in all lowercase such as **xv**, and ones in the same font size but mostly uppercase such as **PBMPlus**.

In conclusion I can safely state that this book stands alone among TEX books. It contains much that is not in print anywhere else, and its usefulness spans all types of TEX users. Victor sez check it out.

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

Book review: *LATEX guide pratique*

Jacques André

Christian ROLLAND, *LATEX guide pratique*. Éditions Addison-Wesley-France, Paris, 1993, 280 pages; ISBN 2-87908-025-8; in French.

Up to now, only two books have been available in French on TEX or LATEX: the famous *Petit livre de TEX* by Seroul, and a translation, by Éric Cornelis, of Michael Urban's *An introduction to LATEX* published as *Premier pas en LATEX* by *Cahiers GUTenberg*. The gap between these two books is now filled.

Although it follows more or less the same road as Lamport's *LATEX user's guide*, this French guide is not just a translation of it. It contains a lot of useful macros, examples, tables, index, etc. Furthermore, this book contains details on extensions that are not described in Lamport's book (such as on Makeindex, BIBTEX, Picture, and, an important point, `french.sty`, etc.).

I used to have Lamport's book near my station. Now, as a French speaker, I have Rolland's: this is a good book!

However... I disagree with the publishers: I do not understand why they have published this book. I mean, why so late, or why they have not waited for Goossens and Mittelbach's *LATEX Companion* translation? This book doesn't have exactly the same audience (Rolland's is more directed to beginners, not low level ones but rather, let us say, regular users, while the *Companion* is instead for advanced LATEXers). No, the problem is that Rolland's book is rather old: nothing about multicolumns, nothing (or almost) about NFSS, nothing about the use of PostScript files (through psfig, pstricks, etc.). Not one Frequently Asked Question is answered here. The author is a beginner and he wrote the book he would have liked to have when learning LATEX. On the other hand, the publisher should have asked (for example to the French TUG: Association GUTenberg) for opinions on this book. It would have been said that chapters should have been added (LATEX in 1994 is not the same as in 1986...). At least, let us hope that a French translation of *The Companion* will be published as soon as possible.

◇ Jacques André
IRISA University of Rennes
Campus de Beaulieu
F-35042 Rennes Cedex, France
Jacques.Andre@irisa.fr

Typesetting on Personal Computers

NextTEX: TEX plus the NextStep Operating System

Alan Hoenig

I recently upgraded my computer system and now use the NextStep operating system on a 486 Intel box. I can't imagine using any other operating system. One of the many pleasures of this computer environment is the implementation of TEX (plus METAFONT plus all other TEX- and METAFONTware) developed by Tom Rokicki of Radical Eye Software for NextStep. The purpose of this article is to describe the unique features of *TEXView*, the name for this system. I will not spend much time on the standard features common to all implementations. As far as I have so far been able to determine, *TEXView* contains within it every standard feature you expect to be there.

Since many readers may not be familiar with NextStep, I include some comments at the end of the article. *TEXView* is a standard component of NextStep. NextStep consists of a graphical user interface on top of Unix (BSD 4.3). Windows can be opened in which different tasks may be launched. When you purchase NextStep, you get Unix and the GUI, together with *TEXView* and other packages bundled together.

1 NextTEX

It's possible to bring up a window in NextStep which accepts commands in a familiar terminal mode. All of the component programs of *TEXView* may be invoked using familiar command line options. The TEX that is the core of *TEXView* is a "big" TEX which contains two non-standard enhancements. If the first line of a document file is of the form

```
%&foo
```

then the vanilla command `tex myfile` automatically invokes the format file `foo.fmt`. That is, it is equivalent to the command line

```
tex &foo myfile
```

With this convention in place, a single TEX invocation works for plain files, LATEX files, AMS-TEX files, LAMS-TEX files, and so on.

A second enhancement makes it possible to interrupt a TEX run to execute another command. We can do so by simply writing the commands to output stream 18. Example: If the first part of a TEX run prepares a raw index file `idx.r`, we can sort and typeset it in the same pass by issuing commands like

```
\immediate\write18{mysort <idx.r >idx.s}
\input idx.s
```

2 An Integrated System

But the most useful way to use NextTEX is not in the traditional command line mode but as part of the integrated *TEXView* environment. I begin by using the Emacs editor to create `myfile.tex`, say. When finished, I save the file to disk, but I don't exit Emacs. Since Unix is multi-tasking, I can keep Emacs 'up' so it is easy to re-enter it to make the inevitable fixes to my source file. If you prefer, just enter `e` in response to the TEX error prompt, and this TEX turns you and the document over to the system editor. To change the choice of editors, it is necessary only to set an environment variable.

The File Viewer is the main NextStep window which lists in icon form the files I am currently interested in. After making sure that `myfile.tex` is one of these files, I double click on it to launch *TEXView*. (I make sure that the first line of my file is of the form `%&foo`, but purists can ensure that the default version of TEX invokes their favorite format.)

In a moment, a small TEX Command Window opens up; this acts as the console which displays the contents of the `log` file and prompts for corrections when necessary. Just as soon as TEX ships out the first page, the *TEXView* previewer displays this page in a new, large window. It's not necessary to wait for the rest of the job to finish—you can begin scrutiny of your document right away. This preview window is so central to this implementation that it's no wonder Tom calls it *TEXView*.

3 Previewing

NextStep incorporates Display PostScript technology, and this is fully integrated into *TEXView*. So, if your document contains references to outline fonts, or encapsulated PostScript files, they will be fully visible in the preview window. For me, this feature alone is worth the price of admission. (I have been reminded, though, that AmigaTEX has possessed this capability since 1990.)

You can use the mouse to scroll or drag the preview display, and the size of the preview window is itself easily adjusted. A single click of the mouse zooms and unzooms the image, and you have access to a huge range of zoom magnifications as part of options to a Window Command.

A single click on the preview image reports the current position of the mouse. If you click on two different points of the previewed document, *TEXView* will report on the real distance between these points in units either of inches, centimeters, real points, PostScript (big) points, or pico-light-seconds. (I learned from this that a pico-light-second is about 17% larger than a point.) You can improve the accuracy of the click by zooming the preview. I was surprised at how quickly I came to rely on this feature. Whenever a printed element doesn't appear quite where I intended—a frequent occurrence—the double click gives a good idea of the magnitude of the displacement. Having this magnitude in hand often provides a vital clue for correcting the problem, and it's nice to be able to get this hint without printing the document.

TEXView does color. Courtesy of PostScript and of the latest version of `dvips` (which is of course part of the package) it's possible to include color in your document. With a color monitor, you can preview in color. (Otherwise, the colored regions appear in a suitable shade of gray.)

4 Printing the Document

To print the document, I simply click the Print button in the *TEXView* menu. Both *TEXView* (and NextStep, for that matter) expect to print to a PostScript printer. (If your printer is not color, any colored regions appear as a shade of grey on the printed document.) You can also 'print' to fax by pressing a 'fax' button.

5 The Integrated Product

A description of any integrated software system, even when scrupulously accurate, may fail to convey a feel for the success of the implementor in creating an integrated environment which feels right, like an old baseball glove. I don't quite know how such a concept could possibly be quantified, but it is my opinion that *TEXView* succeeds in this endeavor, and succeeds admirably. I have become dependent on *TEXView*'s special features, and my hand has begun to creep naturally toward the mouse at appropriate points in the *TEXView* cycle. It is largely for this reason that I refrain from presenting a table of performance statistics for *TEXView*. What's the point? Normally, it's nice to know speed stats so you know how long you have to wait before you can print or preview. But since *TEXView*'s preview begins long before the TEX compilation is complete, such a consideration is irrelevant. (Subjectively, though, NextTEX seems speedy to me.)

Bringing up NextStep and *TEXView* demands a little more in terms of CPU power and expense than the typical PC user may be used to (see below). Nevertheless, if TEX is a critical part of your computer operations, you might well consider making the switch if only to have access to *TEXView*.

A The NextStep Operating Environment

NextStep is a persnickety operating system. Not just any 486 will do, but only those for whom NextStep has been appropriately tweaked. To find out which computers will work, you will need to call NeXT Computer and ask for their hardware compatibility guide ([800] TRY-NEXT or [800] 848-NEXT).

I am running NextStep on a Logisys computer whose chip is a 486 running at 66 mhz. I have 32 meg of ram and an 820 meg hard disk. NextStep systems don't have to be quite this powerful, but I wanted to indulge myself. In addition, you will need a SCSI CD-ROM drive (the operating system is distributed on a CD-ROM). My system, which includes a Nanao 17 inch SVGA monitor, costs about $5500 all told. At the time of the purchase, the Logisys (with which I am very pleased) was the cheapest NextStep desktop system by quite a bit. It wouldn't surprise me if the situation has changed. In addition to NeXT's hardware compatibility guide, you should check the ads in the journal *NextWorld* (which I can find at the larger newsstands in my home town) for information about competitively priced systems. (Or just wait. Computer power continues to get ever cheaper, and NextStep-able computers will in a year or two surely cost a fraction of their current price.)

In addition to the hardware, you'll need to purchase NextStep. The full package, including developer's version, runs about $1700, which may make DOS users gasp, but appears to be quite competitive with other versions of Unix for the PC. It is possible to get the regular version of NextStep for about $700. If your timing is right and special promotional sales are under way, this might reduce the price further. I purchased NextStep in the fall of 1993, at which time there was a very attractive introductory offer that I was able to take advantage of. I do not know what (if any) special offers might still apply, but you should ask the folks at NeXT when you request the hardware guide.

There are two major FTP sites for NextStep software, although much of this material runs on the now-discontinued Motorola NeXT computers. There are at least half-a-dozen active news groups devoted to NextStep, and the Internet community has proven to be unfailingly courteous and helpful the many times I bugged total strangers while setting up my system.

Any discussion of NextStep, no matter how brief, would be remiss if it did not include some mention of NextStep's graphics. They are stunning. NextStep introduces and demands new æsthetic standards for graphic user interfaces. Graphics aside, the interface itself is far more useful than that of the Macintosh or OS/2, and substantially more so than Microsoft *Windows*. (A third-party *Windows* emulator runs under NextStep so your mountains of *Windows* and DOS software are still usable under NextStep.)

It is too much to hope that this upstart operating system will make much headway against *Windows*. I am having so much fun with it, that I can't help rooting for it anyway.

◇ Alan Hoenig
17 Bay Avenue
Huntington, NY 11743 USA
`ajhjj@cunyvm.cuny.edu`

Macros

Interaction tools: dialog.sty and menus.sty

Michael Downes

Introduction

This article describes dialog.sty and menus.sty, which provide functions for printing messages or menus on screen and reading users' responses. The file dialog.sty contains basic message and input-reading functions; menus.sty takes dialog.sty for its base and uses some of its functions in defining more complex menu construction functions. These two files are set up in the form of LaTeX documentstyle option files, but in writing them I spent some extra effort to try to make them usable with PLAINTeX or other common macro packages that include PLAINTeX in their base, such as AMS-TeX or EPLAIN.

The appendix describes grabhedr.sty, required by dialog.sty, which provides two important file-handling features: (1) Functions \localcatcodes and \restorecatcodes that make it possible for dialog.sty (or any file) to manage internal catcode changes properly regardless of the surrounding context. And (2) a command \inputfwh that when substituted for \input makes it possible to grab information such as file name, version, and date from standardized file headers in the style promoted by Nelson Beebe — and to grab it in the process of first inputting the file, as opposed to inputting the file twice, or \reading the information separately (unreliable due to system-dependent differences in the equivalence of TeX's \input search path and \openin search path).

These files and a few others (notably listout.tex) are combined in a package[1] which should, by the time this article appears, be available on the Internet by anonymous ftp from nodes of CTAN, the Comprehensive TeX Archive Network, e.g., ftp.shsu.edu (USA), or ftp.dante.de (Europe). The file listout.tex is a utility for printing out plain text files, with reasonably good handling of overlong lines, tab characters, other nonprinting characters, etc. It uses menus.sty to present an elaborate menu system for changing options (like font size, line spacing, or how many spaces should be printed for a tab character).

Here's an example from the menu system of listout.tex to show how features from dialog.sty and menus.sty can be put to use. First, the menu that you would see if you wanted to change the font or line spacing:

```
==============================================
 F  Change font
 S  Change font size
 L  Change line spacing

Current settings: typewriter 8 / 10.0pt.

    Q  Quit         X  Exit        ?  Help
==============================================


Your choice?
```

Suppose you wanted to change line spacing to 9 points, so you entered l and then 9pt, except that on your first attempt you accidentally mistyped 9pe instead of 9pt. Here's what you would see on screen:

```
Your choice? l
Desired line spacing [TeX units] ? 9pe
?---I don't understand "9pe".
Desired line spacing [TeX units] ? 9pt

* New line spacing: 9.0pt
```

Both lowercase l and capital L are acceptable responses, and the value given for line spacing is checked to make sure it's a valid TeX dimension. Before continuing, the internalized version of the user's value is echoed on screen to confirm that the entered value was read correctly.

Now here's how the above menu is programmed in listout.tex. A function \menuF is constructed using \fxmenu:

```
\fxmenu\menuF{}{
F  Change font
S  Change font size
L  Change line spacing
}{
Current settings: &\mainfont &\mainfontsize / %
&\the&\normalbaselineskip.
}

\def\moptionF{\lettermenu F}
```

In the definition of \moptionF, \lettermenu is a high-level function from menus.sty that calls \menuF (given the argument F) to print the given menu on screen, reads a line of input from the user, extracts the first character and forces it to uppercase, then branches to the next menu as determined by that character. The response of l causes a branch to the function \moptionFL:

```
\def\moptionFL{%
```

[1] Or 'bundle', using more recently established LaTeX terminology for a group of related files, since 'package' now means a LaTeX extension suitable for use with \usepackage.

```
  \promptmesj{%
    Desired line spacing [TeX units] ? }%
  \readline{Q}\reply
```

If Q, X, or ? was entered, the test \xoptiontest will return 'true'; then we should skip the dimension checking and go directly to \optionexec, which knows what to do with those responses:

```
  \if\xoptiontest\reply
  \else
```

Otherwise we check the given dimension to make sure it's usable. If so, echo the new value as confirmation.

```
    \checkdimen\reply\dimen@
    \ifdim\dimen@>\z@
      \normalbaselineskip\dimen@\relax
      \normalbaselines
      \confirm{New line spacing:
        \the\normalbaselineskip}%
      \def\reply{Q}%
    \fi
```

If \reply was changed to Q during the above step, \optionexec will pop back up to the previous menu level (normal continuation); otherwise \reply retains its prior definition — e.g., 9pe — to which \optionexec will simply say "I don't understand that" and repeat the current prompt.

```
  \fi
  \optionexec\reply
}
```

For maximum portability, listout.tex uses in its menus only lowest-common-denominator ordinary printable ASCII characters in the range 32–126. Fancier menus can be obtained at a cost of forgoing system independence, for instance by using emTeX's /o option to output the box-drawing characters in the standard PC DOS character set.

Notation

Double-hat notation such as ^^J is used in this article for control characters, as in *The TeXbook*, although occasionally the alternate form 'CONTROL-J' is used when the emphasis is away from the character's tokenized state inside TeX. A couple of abbreviations from grabhedr.sty are used frequently in the macro code: \xp@ = \expandafter, and \nx@ = \noexpand. Standard abbreviations from plain.tex such as \z@ or \toks@ are used without special comment.

Part 1
Basic dialog functions: dialog.sty

1.1 History

This file, dialog.sty, was born out of a utility called listout.tex that I wrote for my personal use. The purpose of listout.tex was to facilitate printing out plain text files — electronic mail, program source files in various programming languages, and, foremost, TeX macro files and log files. An important part of my TeX programming practice is to print out a macro file on paper and read it through, marking corrections along the way, then use the marked copy as a script for editing the file. (For one thing, this allows me to analyze and mark corrections while riding the bus to work, or sitting out in the back yard to supervise the kids.) The output I normally desired was two 'pages' per sheet of U.S. letter-size paper printed landscape, in order to conserve paper.

Once created, listout.tex quickly became my favorite means of printing out plain text files, not to mention an indispensable tool in my debugging toolbox: I turn on \tracingmacros and \tracingcommands, then print out the resulting log file so that I can see several hundred lines of the log at once (by spreading out two or three pages on my desk with 100+ lines per page); then I trace through, cross things out, label other things, draw arrows, and so forth.

I soon added a filename prompting loop to make it convenient to print multiple files in a single run. In the process of perfecting this simple prompting routine — over two or three years — and adding the ability to optionally specify things like number of columns at run time, eventually I wrote so much dialog-related macro code that it became clear this code should be moved out of listout.tex into its own module. The result was dialog.sty.

Before getting into the macro definitions and technical commentary, here are descriptions from the user's perspective of the functions defined in this file.

1.2 Message-sending functions

\mesj{⟨*text*⟩}

Sends the message verbatim: category 12 for all special characters except braces, tab characters, and carriage returns:

```
{ } ^^I ^^M
```

Naturally, the catcode changes are effective only if \mesj is used directly, not inside a macro argument or definition replacement text.

Multiple spaces in the argument of \mesj print as multiple spaces on screen. A tab character produces always eight spaces; 'smart' handling of tabs is more complicated than I would care to attempt.

Line breaks in the argument of \mesj will produce line breaks on screen. That is, you don't need to enter a special sequence such as ^^J% to get line breaks. See the technical commentary for \mesjsetup for details. Even though curly braces are left with their normal catcodes, they can be printed in a message without any problem, if they occur in balanced pairs. If not, the message should be sent using \xmesj instead of \mesj.

Because of its careful handling of the message text, \mesj is extremely easy to use. The only thing you have to worry about is having properly matched braces. Beyond that, you simply type everything exactly as you want it to appear on screen.

\xmesj{⟨*text*⟩}

This is like \mesj but expands embedded control sequences instead of printing them verbatim. All special characters have category 12 except backslash, percent, braces, tab, return, and ampersand:

```
\ % { } ^^I ^^M &
```

The first four have normal TeX catcodes to make it possible to use most normal TeX commands, and comments, in the message text. ^^I and ^^M are catcode 13 and behave as described for \mesj. The & is a special convenience, an abbreviation for \noexpand, to use in controlling expansion inside the message text.

Doubled backslash \\ in the argument will produce a single category 12 backslash character — thus, \\xxx can be used instead of \string\xxx or \noexpand\xxx (notice that this works even for outer things like \bye or \newif). Similarly \%, \{, \} and \& produce the corresponding single characters.

Category 12 space means that you cannot write something like

```
\ifvmode h\else v\fi rule
```

in the argument of \xmesj without getting a space after the \ifvmode, \else, and \fi.[2] Since occasionally this may be troublesome, \. is defined inside the argument of \xmesj to be a 'control word terminator': If the expansion of \foo is abc, then \foo\.xyz produces abcxyz on screen (as opposed to \foo xyz which would produce abc xyz). Thus the above conditional could be written as

```
\ifvmode\.h\else\.v\fi\.rule
```

Even though the catcode changes done by \xmesj setup have no effect if \xmesj is used inside an argument or definition replacement text, I find it convenient occasionally to use \xmesj in those contexts, in order to get other aspects of the \xmesj setup. For instance, if you need to embed a message that contains a percent sign inside a definition, you can write

```
\def\foo{...
  \xmesj{... this is a percent
    sign: \% (sans backslash) ...}
...}
```

To further support such uses of \xmesj, the following changes are also done by \xmesj setup: the backslash-space control symbol \␣ is made equivalent to \space; \^^J and \^^M are defined to produce a \newlinechar; and active tilde ~ will produce a category-12 tilde.

Among other things, this setup makes it easier to obtain newlines and multiple spaces in an embedded message. For example, in the following definition the message will have a line break on screen for each backslash at the end of a line, and the third line will be indented four spaces.

```
\def\bar{...
  \xmesj{First line\
    Second line\
    \ \ \ \ Indented line\
    Last line}%
...}
```

The alternative of defining a separate message function \barfoo with \f[x]mesj and calling \barfoo inside of \bar would allow more natural entry of the newlines and the multiple spaces, but would be slightly more expensive in string pool and hash table usage.

\promptmesj{⟨*text*⟩}
\promptxmesj{⟨*text*⟩}

These are like \mesj, \xmesj but use \message rather than \immediate\write16 internally, thus if the following operation is a \read, the user will see the cursor on screen at the end of the last line, as may be desired when prompting for a short reply, rather than at the beginning of the next line. The character ! is preempted internally for newlinechar, which means that it cannot actually be printed in the message text. Use of a visible character such as !, rather than the normal \newlinechar ^^J, is

[2] Well, actually, you could replace each space by %⟨*newline*⟩ to get rid of it. But that makes the message text harder to read for the programmer.

necessary for robustness because of the fact that the `\message` primitive was unable to use an 'invisible' character (outside the range 32–126) for newlines up until TEX version 3.1415, which some users do not yet have (at the time of this writing — July 1994).

```
\storemesj\foo{⟨text⟩}
\storexmesj\foo{⟨text⟩}
```

These functions are similar to `\mesj`, `\xmesj` but store the given text in the control sequence `\foo` instead of immediately sending the message. Standard TEX parameter syntax can be used to make `\foo` a function with arguments, e.g. after

```
\storemesj\foo#1{...#1...}
```

then you can later write

```
\message{\foo{\the\hsize}}
```

and get the current value of hsize into the middle of the message text. Consequently also in the x-version `\storexmesj` a category-12 `#` character can be obtained with `\#`.

```
\fmesj\foobar#1#2...{...#1...#2...}
```

Defines `\foobar` as a function that will take the given arguments, sow them into the message text `{...}`, and send the message. In the message text all special characters are category 12 except for braces, `#`, and carriage return.

If an unmatched brace or a `#` must be printed in the message text `\fxmesj` must be used instead. (`##` could be used to insert a single category-6 `#` token into the message text, and TEX would print it without an error, but both `\message` and `\write` would print it as two `##` characters, even though it's only a single token internally.)

```
\fxmesj\foobar#1#2...{...#1...#2...}
```

Combination of `\xmesj` and `\fmesj`. Defines `\foobar` like `\fmesj`, but with full expansion of the replacement text and with normal category codes for backslash, percent, braces, and hash `#`. The control symbols `\\` `\%` `\{` `\}` `\&` and `\.` can be used as in `\xmesj`, with also `\#` for printing a `#` character of category 12.

1.3 Reading functions

```
\readline{⟨default⟩}\answer
```

This reads a line of input from the user into the macro `\answer`. (The macro name can be anything chosen by the programmer, not just `\answer`.) Before reading, all special characters are deactivated, so that the primitive `\read` will not choke if the user happens to enter something like `\newif` or CONTROL-L or `}`. Depending on the operating system, certain characters — e.g., CONTROL-C, CONTROL-Z, CONTROL-D, CONTROL-H — might have special effects instead of being entered into the replacement text of `\answer`, regardless of the catcode changes. To take the most obvious example, under most operating systems, typing CONTROL-H (the Rubout or Backward-Delete key) will delete the previous character from the user's response, instead of entering an ASCII character 8 into `\answer`.

There is one significant exception from the catcode changes that are done for `\readline`: spaces and tabs retain their normal catcode of 10, so that multiple spaces in an answer will be reduced to a single space, and macros with normal space-delimited arguments will work when applied to the answer. (I can't think of any likely scenario where category 12 for spaces would be useful.) Also, the catcode of `^^M` is set to 9 (ignore) so that an empty line — meaning that the user just pressed the carriage return/enter key — will result in an empty `\answer`. If the answer is empty, the given default string will be substituted. The default string can be empty.

```
\xreadline{⟨default⟩}\answer
```

Like `\readline` but the answer is read as executable tokens; the usual catcodes of the TEX special characters remain in effect while reading the answer. A few common outer things (`\bye`, `\+`, `\newif`, `^^L`, among others) are neutralized before the `\read` is done, but the user can still cause problems by entering some other outer control sequence or unbalanced braces. I doubt there's any bulletproof solution, if the tokens are to remain executable, short of the usual last resort: reading the answer using `\readline`, writing it to a file, then inputting the file.

```
\readchar{⟨default⟩}\answer
```

This is like `\readline` but it reduces the answer to its first character. ⟨*default*⟩ is either a single character or empty.

```
\readChar{⟨default⟩}\answer
```

This is like `\readchar` and also uppercases the answer.

```
\changecase\uppercase\answer
```

The function `\changecase` redefines its second argument, which must be a macro, to contain the same text as before, but uppercased or lowercased according to the first argument. Thus `\readChar{Q}\answer` is equivalent to

```
\readchar{q}\answer
\changecase\uppercase\answer
```

It might sometimes be desirable to force lower case before using a file name given by the user, for example.

1.4 Checking functions

```
\checkinteger\reply\tempcount
```

To read in and check an answer that is supposed to be an integer, use \readline\reply and then apply \checkinteger to the \reply, supplying a count register wherein \checkinteger will leave the validated integer. If \reply does not contain a valid integer the returned value will be -\maxdimen.

At the present time only decimal digits are handled; some valid TeX numbers such as "AB, `\@, \number\prevgraf, or a count register name, will not be recognized by \checkinteger. There seems to be no bulletproof way to allow these possibilities.

Tests that hide \checkinteger under the hood, such as a \nonnegativeinteger test, are not provided because as often as not the number being prompted for will have to be tested to see if it falls inside a more specific range, such as 0–255 for an 8-bit number or 1–31 for a date, and it seems common sense to omit overhead if it would usually be redundant. It's easy enough to define such a test for yourself, if you want one.

```
\checkdimen\reply\tempdim
```

Analog of \checkinteger for dimension values. If \reply does not contain a valid dimension the value returned in \tempdim will be -\maxdimen.

Only explicit dimensions with decimal digits, optional decimal point and more decimal digits, followed by explicit units pt cm in or whatever are checked for; some valid TeX dimensions such as \parindent, .3\baselineskip, or \fontdimen5 \font will not be recognized by \checkdimen.

What good is all this?

What good is all this stuff, practically speaking? — you may ask. Well, a typical application might be something like: At the beginning of a document, prompt interactively to find out if the user wants to print on A4 or U.S. letter-size paper, or change the top or left margin. Such a query could be done like this:

```
\promptxmesj{
Do you want to print on A4 or US letter paper?
Enter u or U for US letter, %
anything else for A4: }
\readChar{A}\reply % default = A4
\if U\reply \textheight=11in \textwidth=8.5in
\else \textheight=297mm \textwidth=210mm \fi
%    Subtract space for 1-inch margins
\addtolength{\textheight}{-2in}
\addtolength{\textwidth}{-2in}

\promptxmesj{
Left margin setting? %
[Return = keep current value,
\the\oddsidemargin]: }
\readline{\the\oddsidemargin}\reply
\checkdimen\reply{\dimen0}
\ifdim\dimen0>-\maxdimen
  \setlength\oddsidemargin{\dimen0}%
  \xmesj{OK, using new left margin of %
\the\oddsidemargin.}
\else
  \xmesj{Sorry, I don't understand %
that reply: `\reply'.\
Using default value: \the\oddsidemargin.}
\fi
```

Although LaTeX's \typeout and \typein functions can be used for this same task, they are rather more awkward, and checking the margin value for validity would be quite difficult.

1.5 Preliminaries

If grabhedr.sty is not already loaded, load it now. The \trap.input function is explained in grabhedr.doc.

```
\csname trap.input\endcsname
\input grabhedr.sty \relax
\fileversiondate{dialog.sty}{0.9v}{6-Jul-1994}%
```

The functions \localcatcodes and \restorecatcodes are defined in grabhedr.sty. We use them to save and restore catcodes of any special characters needed in this file whose current catcodes might not be what we want them to be. Saving and restoring catcode of at-sign @ makes this file work equally well as a LaTeX documentstyle option or as a simple input file in other contexts. The double quote character " might be active for German and other languages. Saving and restoring tilde ~, hash #, caret ^, and left quote ` catcodes is normally redundant but reduces the number of assumptions we must rely on. (The following catcodes are assumed: \ 0, { 1, } 2, \endlinechar 5, ␣ 10, % 14, a–z A–Z 11, 0–9 . - 12.)

```
\localcatcodes{\@{11}%
  \~{13}\"{12}\#{6}\^{7}\`{12}}
```

1.6 Definitions

For deactivating characters with special catcodes during \readline we use, instead of \dospecials, a more bulletproof (albeit slower) combination of \otherchars, \controlchars, and \highchars that covers all characters in the range 0–255 except letters and digits. Handling the characters above 127 triples the overhead done for each read operation or message definition but seems mandatory for maximum robustness.[3]

\otherchars includes the thirty-three nonalphanumeric visible characters (counting space as visible). It is intended as an executable list like \dospecials but, as an exercise in memory conservation, it is constructed without the \dos. For the usual application of changing catcodes, the list can still be executed nicely as shown below. Also, if we arrange to make sure that each character token gets category 12, it's not necessary to use control symbols such as \% in place of those few special characters that would otherwise be difficult to place inside of a definition. This avoids a problem that would otherwise arise if we included \+ in the list and tried to process the list with a typical definition of do: \+ is 'outer' in plain TEX and would cause an error message when \do attempted to read it as an argument. (As a matter of fact the catcode changes below show a different way around that problem, but a list of category-12 character tokens is a fun thing to have around anyway.)

```
\begingroup
```

First we start a group to localize \catcode changes. Then we change all relevant catcodes to 12 except for backslash, open brace, and close brace, which can be handled by judicious application of \escapechar, \string, \edef, and \xdef. By defining \do in a slightly backward way, so that it doesn't take an argument, we don't need to worry about the presence of \+ in the list of control symbols. Notice the absence of \‘ from the list of control symbols; it was already catcoded to 12 in the \localcatcodes declaration at the beginning of this file — otherwise it would be troublesome to make the definition of \do bulletproof (consider the possibilities that ‘ might have catcode 0, 5, 9, or 14).

```
\def\do{12 \catcode‘}
\catcode‘\~\do\!\do\@\do\#\do\$\do\^\do\&
\do\*\do\(\do\)\do\-\do\_\do\=\do\[\do\]
\do\;\do\:\do\’\do\"\do\<\do\>\do\,\do\.
\do\/\do\?\do\|12\relax
```

To handle backslash and braces, we define \\, \{, and \} to produce the corresponding category-12 character tokens. Setting \escapechar to −1 means that \string will omit the leading backslash that it would otherwise produce when applied to a control sequence.

```
\escapechar -1
\edef\\{\string\\}
\edef\{{\string\{}\edef\}{\string\}}
```

Space and percent are done last. Then, with almost all the special characters now category 12, it's rather easy to define \otherchars.

```
\catcode‘\ =12\catcode‘\%=12
\xdef\otherchars
{ !"#$%&’()*+,-./:;<=>?[\\]^_‘\{|\}~}
\endgroup
```

\controlchars is another list for the control characters ASCII 0–31 and 127. The construction of this list is similar to the construction of \otherchars. We need to turn off \endlinechar because the catcode of ^^M is going to be changed. The ^^L inside the \gdef is not a problem (as it might have been, due to the usual outerness of ^^L) because the catcode is changed from 13 to 12 before that point.

```
\begingroup
\endlinechar = -1
\def\do{12 \catcode‘}
\catcode‘\^^@\do\^^A\do\^^B\do\^^C
\do\^^D\do\^^E\do\^^F\do\^^G\do\^^H\do\^^I
\do\^^J\do\^^K\do\^^L\do\^^M\do\^^N\do\^^O
\do\^^P\do\^^Q\do\^^R\do\^^S\do\^^T\do\^^U
\do\^^V\do\^^W\do\^^X\do\^^Y\do\^^Z\do\^^[
\do\^^\\do\^^]\do\^^^\do\^^_\do\^^? 12\relax
%
\gdef\controlchars{^^@^^A^^B^^C^^D^^E^^F^^G
  ^^H^^I^^J^^K^^L^^M^^N^^O^^P^^Q^^R^^S^^T
  ^^U^^V^^W^^X^^Y^^Z^^[^^\^^]^^^^^_^^?}
\endgroup
```

And finally, the list \highchars contains characters 128–255, the ones that have the eighth bit set.

```
\begingroup
\def\do{12 \catcode‘}
\catcode‘\^^80\do\^^81\do\^^82\do\^^83\do\^^84
\do\^^85\do\^^86\do\^^87\do\^^88\do\^^89\do\^^8a
\do\^^8b\do\^^8c\do\^^8d\do\^^8e\do\^^8f
\do\^^90\do\^^91\do\^^92\do\^^93\do\^^94\do\^^95

...

\do\^^fc\do\^^fd\do\^^fe\do\^^ff 12\relax
%
\gdef\highchars{%
^^80^^81^^82^^83^^84^^85^^86^^87^^88%
^^89^^8a^^8b^^8c^^8d^^8e^^8f%
^^90^^91^^92^^93^^94^^95^^96^^97^^98%

...

^^f9^^fa^^fb^^fc^^fd^^fe^^ff}
```

[3] If you are using dialog.sty functions on a slow computer, you might want to try setting \highchars = empty to see if that helps the speed.

```
\endgroup
```

The function \actively makes a given character active and carries out the assignment given as the first argument. The assignment can be embedded inside other definitions without requiring any special setup to produce an active character in the replacement text. The argument should be a control symbol, e.g. \@ or \# or \^^M, rather than a single character. (Except that + is safer than \+ in PLAINTEX.) If the assignment is a definition (\def, \edef, \gdef, \xdef) it is allowed to take arguments in the normal TEX way. Prefixes such as \global, \long, or \outer must go inside the first argument rather than before \actively.

Usage:

```
\actively\def\?{⟨replacement text⟩}
\actively\def\%#1#2{⟨replacement text⟩}
\actively{\global\let}\^^@=\space
```

doesn't work! because of lowercase

One place where this function can be put to good use is in making ^^M active in order to get special action at the end of each line of input. The usual way of going about this would be to write

```
\def\par{something}\obeylines
```

which is a puzzling construction to the TEX novice who doesn't know what \obeylines does with \par. The same effect could be gotten a little more transparently with

```
\actively\def\^^M{something}
```

In the definition of \actively we use the unique properties of \lowercase to create an active character with the right character code, overlapping with a \begingroup \endgroup structure that localizes the necessary lc-code change.

```
\def\actively#1#2{\catcode`#2\active
  \begingroup \lccode`\~=`#2\relax
  \lowercase{\endgroup#1~}}
```

The \mesjsetup function starts a group to localize catcode changes. The group will be closed eventually by a separate function that does the actual sending or stores the message text for later retrieval.

We want to change the catcode of each character in the three lists \otherchars, \controlchars, and \highchars to 12. After giving \do a recursive definition, we apply it to each of the three lists, adding a suitable element at the end of the list to make the recursion stop there. This allows leaving out the \do tokens from the character lists, without incurring the cost of an if test at each recursion step.

```
\def\mesjsetup{\begingroup \count@=12
  \def\do##1{\catcode`##1\count@ \do}%
```

The abbreviation \xp@ = \expandafter is from grabhedr.sty.

```
  \xp@\do\otherchars{all \@gobbletwo}%
  \xp@\do\controlchars{all \@gobbletwo}%
  \xp@\do\highchars{all \@gobbletwo}%
  \actively\edef\^^I{ \space\space\space
    \space\space\space\space}%
```

The convenient treatment of newlines in the argument of \mesj (every line break produces a line break on screen) is achieved by making the ^^M character active and defining it to produce a category-12 ^^J character. Although for \mesj it would have sufficed to make ^^M category 12 and locally set \newlinechar = ^^M while sending the message, it turns out to be useful for other functions to have the ^^M character active, so that it can be remapped to an arbitrary function for handling new lines (e.g., perhaps adding extra spaces at the beginning of each line). And if \mesj treats ^^M the same, we can arrange for it to share the setup routines needed for the other functions.

```
  \endlinechar=`\^^M\actively\let\^^M=\relax
  \catcode`\{=1 \catcode`\}=2 }
```

In \sendmesj we go to a little extra trouble to make sure ^^M produces a newline character, no matter what the value of \newlinechar might be in the surrounding environment. The impending \endgroup will restore \newlinechar to its previous value. One reason for using ^^J (instead of, say, ^^M directly) is to allow e.g. \mesj{xxx^^Jxxxx} to be written inside a definition, as is sometimes convenient. This would be difficult with ^^M instead of ^^J because of catcodes.

```
\def\sendmesj{\newlinechar`\^^J%
  \actively\def\^^M{^^J}%
  \immediate\write\sixt@@n{\mesjtext}\endgroup}
```

Given the support functions defined above, the definition of \mesj is easy: Use \mesjsetup to clear all special catcodes, then set up \sendmesj to be triggered by the next assignment, then read the following balanced-braces group into \mesjtext. As soon as the definition is completed, TEX will execute \sendmesj, which will send the text and close the group that was started in \mesjsetup to localize the catcode changes.

```
\def\mesj{\mesjsetup \afterassignment\sendmesj
  \def\mesjtext}
```

The \sendprompt function is just like \sendmesj except that it uses \message instead of \write, as might be desired when prompting for user input, so that the on-screen cursor stays on the same line as the prompt instead of hopping down to the beginning of the next line. In order for newlines to work with \message we must use a visible character instead of ^^J. When everyone has TeX version 3.1415 or later this will no longer be true. The choice of ! might be construed (if you wish) as editorial comment that ! should not be shouted at the user in a prompt.

```
\def\sendprompt{%
  \newlinechar`\!\relax \actively\def\^^M{!}%
  \message{\mesjtext}\endgroup}
```

This function is like \mesj but uses \sendprompt instead of \sendmesj.

```
\def\promptmesj{\mesjsetup
  \afterassignment\sendprompt \def\mesjtext}
```

Arg #1 of \storemesj is the control sequence under which the message text is to be stored.

```
\def\storemesj#1{\mesjsetup
  \catcode`\#=6 % to allow arguments if needed
  \afterassignment\endgroup
  \gdef#1}
```

While \storemesj\foo{...} is more or less the same as \def\foo{...} with special catcode changes, \fmesj\foo{...} corresponds to \def\foo{\mesj{...}}, that is, after \fmesj\foo the function \foo can be executed directly to send the message. Thus \storemesj is typically used for storing pieces of messages, while \fmesj is used for storing entire messages.

To read the parameter text #2, we use the peculiar #{ feature of TeX to read everything up to the opening brace.

```
\def\fmesj#1#2#{\mesjsetup
  \catcode`\#=6 % restore to normal
```

The parameter text #2 must be stored in a token register rather than a macro to avoid problems with # characters. The \long prefix is just to admit the (unlikely) possibility of using \fmesj to define something such as an error message saying 'You can't use #1 here' where one of the possibilities for #1 is \string\par.

```
  \toks@{\long\gdef#1#2}%
```

Define \@tempa to put together the first two arguments and [pseudo]argument #3 and make the definition of #1.

```
  \def\@tempa{%
    \edef\@tempa{%
      \the\toks@{%
```

The abbreviation \nx@ = \noexpand is from grabhedr.sty.

```
      \begingroup\def\nx@\mesjtext{\the\toks2 }%
        \nx@\sendmesj}%
    }%
    \@tempa
    \endgroup % Turn off the \mesjsetup catcodes
  }%
  \afterassignment\@tempa
  \toks2=}
```

\xmesjsetup is like \mesjsetup except it prepares to allow control sequence tokens and normal comments in the message text. For TeXnicians' convenience certain other features are thrown in.

Here, unlike the setup for \xreadline, I don't bother to remove the outerness of \bye, \newif, etc., because I presume the arguments of \xmesj, \fxmesj, \storexmesj, \fxmenu, etc. are more likely to be written by a TeXnician than by an average end user, whereas \xreadline is designed to handle arbitrary input from arbitrary users.

```
\def\xmesjsetup{\mesjsetup
```

Throw in pseudo braces just in case we are inside an \halign with \\ let equal to \cr at the time when \xmesjsetup is called. (As might happen in AMS-TeX.)

```
  \iffalse{\fi
  \catcode`\\=0 \catcode`\%=14
```

Define \% \\ \{ \} \& to produce the corresponding single characters, category 12.

```
  \begingroup \lccode`\0=`\\\lccode`\1=`\{%
  \lccode`\2=`\}\lccode`\3=`\%%
  \lowercase{\endgroup \def\\{0}\def\{{1}%
    \def\}{2}\def\%{3}}%
  \iffalse}\fi
  \edef\&{\string &}%
```

Let & = \noexpand for expansion control inside the argument text; let active ^^M = \relax so that newlines will remain inert during the expansion.

```
  \actively\let\&=\noexpand
  \actively\let\^^M=\relax
```

Define \. to be a no-op, for terminating a control word when it is followed by letters and no space is wanted.

```
  \def\.{}%
```

Support for use of \xmesj inside a definition replacement text or macro argument: control-space \␣ = \space, tilde ~ prints as itself, \^^M (i.e., a lone backslash at the end of a line) will produce a newline, also \^^J, while finally \par = blank line translates to two newlines.

```
  \def\ { }\edef~{\string ~}%
```

Define `\^^M` to produce an active `^^M` character, which (we hope) will be suitably defined to produce a newline or whatever.

```
  \begingroup \lccode`\~=`\^^M%
    \lowercase{\endgroup \def\^^M{~}}%
  \let\^^J\^^M \def\par{\^^M\^^M}%
}
```

`\xmesj` uses `\xmesjsetup` and `\edef`.

```
\def\xmesj{\xmesjsetup \afterassignment\sendmesj
  \edef\mesjtext}
```

`\promptxmesj` is analogous to `\promptmesj`, but with expansion.

```
\def\promptxmesj{\xmesjsetup
  \afterassignment\sendprompt \edef\mesjtext}
```

And `\storexmesj` is like `\storemesj`, with expansion. Since we allow arguments for the function being defined, we also must define `\#` to produce a single category-12 `#` character so that there will be a way to print `#` in the message text.

```
\def\storexmesj#1#2#{\xmesjsetup
  \catcode`\#=6 % to allow arguments if needed
  \edef\#{\string##}%
  \afterassignment\endgroup
  \xdef#1#2}
```

And `\fxmesj` is the expansive analog of `\fmesj`.

```
\def\fxmesj#1#2#{\xmesjsetup
  \catcode`\#=6 % restore to normal
  \edef\#{\string##}%
  \toks@{\long\xdef#1#2}%
  \def\@tempa{%
    \edef\@tempa{%
      \the\toks@{\begingroup
      \def\nx@\nx@\nx@\mesjtext{\the\toks\tw@}%
      \nx@\nx@\nx@\sendmesj}}%
    \@tempa % execute the constructed xdef
    \endgroup % restore normal catcodes
  }%
  \afterassignment\@tempa
  \toks\tw@=}
```

1.7 Reading functions

The `\readline` function gets one line of input from the user. Arguments are: `#1` default to be used if the user response is empty (i.e., if the user just pressed the return/enter key), `#2` macro to receive the input.

```
\def\readline#1#2{%
  \begingroup \count@ 12 %
  \def\do##1{\catcode`##1\count@ \do}%
  \xp@\do\otherchars{all \@gobbletwo}%
  \xp@\do\controlchars{all \@gobbletwo}%
  \xp@\do\highchars{all \@gobbletwo}%
```

Make spaces and tabs normal instead of category 12.

```
  \catcode`\ =10 \catcode`\^^I=10 %
  \catcode`\^^M=9 % ignore
```

Reset endline char to normal, just in case.

```
  \endlinechar`\^^M
```

We go to a little trouble to avoid `\gdef`-ing `#2`, in order to prevent save stack buildup if the user of `\readline` unknowingly carries on doing local redefinitions of `#2` after the initial read.

```
  \read\m@ne to#2%
  \edef#2{\def\nx@#2{#2}}%
  \xp@\endgroup #2%
  \ifx\@empty#2\def#2{#1}\fi
}
```

`\xreadline` is like `\readline` except that it leaves almost all catcodes unchanged so that the return value is executable tokens instead of strictly character tokens of category 11 or 12.

```
\def\xreadline#1#2{%
  \begingroup
```

Render some outer control sequences innocuous.

```
    \xp@\let\csname bye\endcsname\relax
    \xp@\let\csname newif\endcsname\relax
    \xp@\let\csname newcount\endcsname\relax
    \xp@\let\csname newdimen\endcsname\relax
    \xp@\let\csname newskip\endcsname\relax
    \xp@\let\csname newmuskip\endcsname\relax
    \xp@\let\csname newtoks\endcsname\relax
    \xp@\let\csname newbox\endcsname\relax
    \xp@\let\csname newinsert\endcsname\relax
    \xp@\let\csname +\endcsname\relax
    \actively\let\^^L\relax
  \catcode`\^^M=9 % ignore
  \endlinechar`\^^M% reset to normal
  \read\m@ne to#2%
  \toks@\xp@{#2}%
  \edef\@tempa{\def\nx@#2{\the\toks@}}%
  \xp@\endgroup \@tempa
  \ifx\@empty#2\def#2{#1}\fi
}
```

`\readchar` reduces the user response to a single character.

```
\def\readchar#1#2{%
  \readline{#1}#2%
```

If the user's response and the default response are both empty, we need something after `#1` to keep `\@car` from running away, so we add an empty pair of braces.

```
  \xdef#2{\xp@\@car#2#1{}\@nil}%
}
```

\readChar reduces the user response to a single uppercase character. (This is useful to simplify testing the response later with \if.)

```
\def\readChar#1#2{%
  \readline{#1}#2%
  \changecase\uppercase#2%
```

Reduce #2 to its first character, or the first character of #1, if #2 is empty. The extra braces {} are to prevent a runaway argument error from \@car if #2 and #1 are both empty.

```
  \xdef#2{\xp@\@car #2#1{}\@nil}%
}
```

The function \changecase uppercases or lowercases the replacement text of its second argument, which must be a macro. The first argument should be \uppercase or \lowercase.

```
\def\changecase#1#2{\@casetoks\xp@{#2}%
  \edef#2{#1{\def\nx@#2{\the\@casetoks}}}#2}
```

We allocate a token register just for the use of \changecase because it might be used at a low level internally where we don't want to interfere with other uses of the scratch token registers 0–9.

```
\newtoks\@casetoks
```

A common task in reading user input is to verify, when an answer of a certain kind was requested, that the response has indeed the desired form — for example, if a nonnegative integer is required for subsequent processing, it behooves us to verify that we have a nonnegative integer in hand before doing anything that might lead to inconvenient error messages. However, it's not easy to decide how best to handle such verification. One possibility might be to have a function

```
 \readnonnegativeinteger\foo
```

to do all the work of going out and fetching a number from the user and leaving it in the macro \foo. Another possibility would be to read the response using \readline and then apply a separate function that can be used in combination with \if, for example

```
  \readline{}\reply
  \if\validnumber\reply ... \else ... \fi
```

For maximum flexibility, a slightly lower-level approach is chosen here. The target syntax is

```
  \readline{}\reply
  \checkinteger\reply\tempcount
```

where \tempcount will be set to -\maxdimen if \reply does *not* contain a valid integer. (Negative integers are allowed, as long as they are greater than -\maxdimen.) Then the function that calls \checkinteger is free to make additional checks on the range of the reply and give error messages tailored to the circumstances. And the handling of an empty \reply can be arbitrarily customized, something that would tend to be inconvenient for the first method mentioned.

The first and second approaches can be built on top of the third if desired, e.g. (for the second approach)

```
\def\validnumber#1{TT\fi
  \checkinteger\tempcount#1%
  \ifnum\tempcount>-\maxdimen }
```

The curious TT\fi...\ifnum construction is from TEXhax 1989, no. 20 and no. 38 (a suggestion of D. E. Knuth in reply to a query by S. von Bechtolsheim).

Argument #2 of \checkinteger must be a count register; #1 is expected to be a macro holding zero or more arbitrary characters of category 11 or 12.

```
\def\checkinteger#1#2{\let\scansign@\@empty
  \def\scanresult@{#2}%
  \xp@\scanint#1x\endscan}
```

To validate a number, the function \scaninit must first scan away leading + or - signs (keeping track in \scansign@), then look at the first token after that: if it's a digit, fine, scan that digit and any succeeding digits into the given count register (\scanresult@), ending with \endscan to get rid of any following garbage tokens that might just possibly show up.

Typical usage includes initializing \scansign@ to empty, as in the definition of \checkinteger.

```
\let\scansign@\@empty
\def\scanresult@{\tempcount}%
\xp@\scanint\reply x\endscan
```

Assumption: \reply is either empty or contains only category 11 or 12 characters (which it will if you used \readline!). If a separate check is done earlier to trap the case where \reply is empty, for example, by using a nonempty default for \readline, then the x before \endscan is superfluous.

Arg #1 = next character from the string being tested. The test whether #1 is a decimal digit is similar in spirit to the test \if!#1! to see if an argument is empty (*The TEXbook*, Appendix D, p. 376).

```
\def\scanint#1{%
  \ifodd 0#11 %% is #1 a decimal digit?
%%   If so read all digits into \scanresult@
%%   with sign prefix
    \def\@tempa{\afterassignment\endscan
```

```
   \scanresult@=\scansign@#1}%
 \else
  \if -#1\relax
```

Here we flipflop the sign; watch closely.

```
   \edef\scansign@{%
    \ifx\@empty\scansign@ -\fi}%
   \def\@tempa{\scanint}%
  \else
```

A plus sign can just be ignored.

```
   \if +#1\relax
    \def\@tempa{\scanint}%
   \else
    \def\@tempa{%
     \scanresult@=-\maxdimen\endscan}%
 \fi\fi\fi
 \@tempa
}
```

```
\def\endscan#1\endscan{}
```

```
\newcount\dimenfirstpart
```

```
\newtoks\dimentoks
```

\scandimen is similar to \scanint but has to call some auxiliary functions to scan the various subcomponents of a dimension (leading digits, decimal point, fractional part, and units, in addition to the sign). The minimum requirements of TEX's syntax for dimensions are a digit or decimal point plus + the units; all the other components are optional (*The TEXbook*, Exercise 10.3, p. 58).

When scanning for the digits of a fractional part, we can't throw away leading zeros; therefore we don't read the fractional part into a count register as we did for the digits before the decimal point; instead we read the digits one by one and store them in \dimentoks.

The function that calls \scandimen should initialize \scansign@ to \@empty, \dimenfirstpart to \z@, \dimentoks to empty {}, and \dimentrue@ to \@empty.

Test values: 0pt, 1.1in, -2cm, .3mm, 0.4dd, 5.cc, .10000000009pc, \hsize, em.

```
\def\scandimen#1{%
 \ifodd 0#11
  \def\@tempa{\def\@tempa{\scandimenb}%
   \afterassignment\@tempa
   \dimenfirstpart#1}%
 \else
```

The following test resolves to true if #1 is either a period or a comma (both recognized by TEX as decimal point characters).

```
  \if \if,#1.\else#1\fi.%
   \def\@tempa{\scandimenc}%
  \else
   \if -#1% then flipflop the sign
    \edef\scansign@{%
     \ifx\@empty\scansign@ -\fi}%
    \def\@tempa{\scandimen}%
   \else
    \if +#1% then ignore it
     \def\@tempa{\scandimen}%
    \else % not a valid dimen
     \def\@tempa{%
      \scanresult@=-\maxdimen\endscan}%
 \fi\fi\fi\fi
 \@tempa
}
```

Scan for an optional decimal point.

```
\def\scandimenb#1{%
 \if \if,#1.\else#1\fi.%
  \def\@tempa{\scandimenc}%
 \else
```

#1

If the decimal point is absent, we need to put back #2 and rescan it to see if it is the first letter of the units.

```
  \def\@tempa{\scanunitsa#1}%
 \fi
 \@tempa
}
```

Scan for the fractional part: digits after the decimal point.

```
\def\scandimenc#1{%
```

If #1 is a digit, add it to \dimentoks.

```
 \ifodd 0#11 \dimentoks\xp@{%
   \the\dimentoks#1}%
  \def\@tempa{\scandimenc}%
 \else
```

Otherwise rescan #1, presumably the first letter of the units.

```
 \def\@tempa{\scanunitsa#1}%
 \fi
 \@tempa
}
```

```
\def\scanunitsa#1\endscan{%
```

Check for true qualifier.

```
 \def\@tempa##1true##2##3\@tempa{##2}%
```

The peculiar nature of \lowercase is evident here as we are able to apply it to only the test part of the conditional without running into brace-matching problems. (Compare the braces in this example to something like \message{\iffalse A}\else B}\fi.)

```
 \lowercase{%
  \xp@\ifx\xp@\end
```

```
    \@tempa#1true\end\@tempa
  }%
```

No true was found:

```
    \let\dimentrue@\@empty
    \def\@tempa{\scanunitsb#1\endscan}%
  \else
    \def\dimentrue@{true}%
    \def\@tempa##1true##2\@tempa{%
      \def\@tempa{##1}%
      \ifx\@tempa\@empty
        \def\@tempa{\scanunitsb##2\endscan}%
      \else
        \def\@tempa{\scanunitsb xx\endscan}%
      \fi}%
    \@tempa#1\@tempa
  \fi
  \@tempa
}
```

Scan for the name of the units and complete the assignment of the scanned value to \scanresult@. Notice that, because of the way \scanunitsb picks up #1 and #2 as macro arguments, p t is allowed as a variation of pt. Eliminating this permissiveness doesn't seem worth the speed penalty that would be incurred in \scanunitsb.

The method for detecting a valid units string is to define the scratch function \@tempa to apply TeX's parameter-matching abilities to a special string that will yield a boolean value of true if and only if the given string is a valid TeX unit. This extraordinary ploy should only be attempted by experienced TeX programmers possessed of the profoundest understanding of the language.

...Ha ha! Just kidding. Actually you simply have to realize that \lowercase and \uppercase are rather odd, then experiment to see what you can get away with.

```
\def\scanunitsb#1#2{%
  \def\@tempa##1#1#2##2##3\@nil{##2}%
  \def\@tempb##1{T\@tempa
    pcTptTcmTccTemTexTinTmmTddTspT##1F\@nil}%
```

Force lowercase just in case the units were entered with uppercase letters (accepted by TeX, so we had better accept uppercase also).

```
  \lowercase{%
  \if\@tempb{#1#2}%
  }%
   \scanresult@=\scansign@
     \number\dimenfirstpart.\the\dimentoks
     \dimentrue@#1#2\relax
  \else
    \scanresult@=-\maxdimen
  \fi
```

Call \endscan to gobble garbage tokens, if any.

```
  \endscan
}
```

Argument #2 must be a dimen register; #1 is expected to be a macro holding zero or more arbitrary characters of category 11 or 12.

```
\def\checkdimen#1#2{%
  \let\scansign@\@empty \def\scanresult@{#2}%
  \let\dimentrue@\@empty
  \dimenfirstpart\z@ \dimentoks{}%
  \xp@\scandimen#1xx\endscan
}
```

Finish up.

```
\restorecatcodes
\endinput
```

Part 2
Menu functions: menus.sty

This file requires grabhedr.sty and dialog.sty. If grabhedr.sty is not already loaded, load it now and call \fileversiondate, since it's too late to apply \inputfwh to *this* file. See the documentation of \trap.input in grabhedr.doc.

```
\csname trap.input\endcsname
\input grabhedr.sty \relax
\fileversiondate{menus.sty}{0.9q}{6-Jul-1994}
\inputfwh{dialog.sty}
```

2.1 Function descriptions

```
\fmenu\foobar{
  ⟨preliminary text⟩
}{
  ⟨menu lines⟩
}{
  ⟨following text⟩
}
```

Defines \foobar as a function that puts the preliminary text, the menu lines (list of choices), and the after text on screen. Normal usage:

```
\foobar            % print the menu on screen
\readline{}\reply  % read the answer
```

(See the description of \readline in dialog.doc.) In the various text parts all special characters have category 12 except for braces, as with \mesj. Note the recommended placement of the braces: no closing brace falls at the end of a line, except the very last one. Because of the special catcodes in effect when reading the final three arguments, a ^^M or % between arguments would be read as an active character or category-12 character respectively, instead of being ignored. But actually, after some rather arduous difficult programming, I managed to make it possible to write just about anything (except brace characters) between the arguments and have it be ignored, so the recommended style is not mandatory. The first and last newline of each argument are stripped off anyway in order to produce consistent clean connections with \menuprefix etc.; see below.

Menu functions created by \fmenu are allowed optionally to have arguments, like functions created with \fmesj (from dialog.sty), so that pieces of text can be inserted at the time of use. This makes it possible for several similar menus to share the same menu function if there are only minor variations between them.

```
\menuprefix, \menusuffix
\inmenuA, \inmenuB
```

The text \menuprefix will be added at the beginning of each menu; \menusuffix will be added at the end. The text \inmenuA and \inmenuB will be added between the first and second, respectively second and third parts of the menu; their default values produce a blank line on screen. (But \inmenuA will be omitted if the first part is empty, and \inmenuB will be omitted if the last part is empty.) To change any of these texts, use \storemesj or \storexmesj. For example:

```
\storemesj\menuprefix{********* MENU **********}
```

```
\menuprompt
```

Furthermore, the function \menuprompt is called at the very end of the menu, so that for example a standard prompt such as `Enter a number:` could be applied at the end of all menus, if desired. To change \menuprompt, use \fmesj or \fxmesj.

```
\menuline, \endmenuline
\menutopline, \menubotline
```

Each line in the middle argument of \fmenu (the list of choices) is embedded in a statement \menuline...\endmenuline. The default definition of \menuline is to add two spaces at the beginning and a newline at the end. Lines in the top or bottom part of the menu are embedded in \menutopline...\endmenuline or \menubotline...\endmenuline respectively. (Notice that all three share the same ending delimiter; if different actions are wanted at the end of a top or bottom line as opposed to a middle menu line, they must be obtained by defining \menutopline or \menubotline to read the entire line as an argument and perform the desired processing.)

An enclosing box for a menu can be obtained by defining \menuline and its relatives appropriately and using \fxmenu (see below).

```
\fxmenu\foobar{
  ⟨preliminary text⟩
}{
  ⟨menu lines⟩
}{
  ⟨following text⟩
}
```

Similar to \fmenu but with full expansion in each part of the text, as with \xmesj.

To get an enclosing box for a menu, write `\.` at the end of each menu line (to protect the preceding spaces from TEX's propensity to remove character 32 at the end of a line, regardless of its catcode), and then make sure that \menuline and \endmenuline put in the appropriate box-drawing characters on either side. I.e.:

```
\fxmenu\foobar{
First line                                        \.
Second line                                       \.
}{
Third line                                        \.
...
}{
Last line                                         \.
}
```

With the /o option of emTEX, you can use the box-drawing characters in the standard PC DOS character set. A more detailed example in menus.doc is omitted here for the sake of brevity.

```
\nmenu\Alph\foobar#1{
  ⟨preliminary text⟩
}{
  ⟨menu lines⟩
}{
  ⟨following text⟩
}
```

\nmenu and \nxmenu are like \fmenu, \fxmenu except that they automatically number each line of the middle part of the menu. (This allows menu choices to be added or deleted without tedious renumbering.) The first argument indicates the type of numbers to be used: \alph, \Alph, \arabic, \roman, \Roman (following LATEX). These are not yet implemented.

The function \menunumber (taking one argument) is applied to each automatically generated number. The default value is to add brackets and a space after:

```
\def\menunumber#1{[#1] }
```

but by redefining \menunumber you can add parentheses or extra spaces or what have you around each number. Internally a line of an autonumbered menu is stored as

```
\menuline\menunumber{5}Text text ...\endmenuline
```

```
\optionexec\answer
```

This is a companion function for \readChar and the menu functions: it checks to see if the answer is equal to any one of the characters ? Q X, and if so executes \moption? or \moptionQ or \moptionX respectively, otherwise executes

```
\csname moption\curmenu C\endcsname
```

where C means the character that was read and \curmenu is a string identifying the current location in the menu system. (\optionexec pushes and pops \curmenu when going between menus, to keep it up to date.)

Thus the major work involved in making a menu system is to define the menu screens using \fmenu, \fxmenu, and then define corresponding functions \moptionXXX that display one of the menu screens, read a menu choice, and call \optionexec to branch to the next action.

```
\optionfileexec\answer
```

Like \optionexec, but gets the next menu from a file instead of from main memory, if applicable. This is not yet implemented. The technical complications involved in managing the menu files are many — for example: How do you prevent the usual file name message of TeX from intruding on your carefully designed menu screens, if \input is used to read the next menu file? Alternatively if you try to use \read to read the next menu file, how do you deal with catcode changes?

```
\lettermenu{MN}
```

This is an abbreviation for

```
\menuMN \readChar{Q}\reply \optionexec\reply
```

It calls the menu function associated with the menu name MN, reads a single uppercase letter into \reply, and then calls \optionexec to branch to the case selected by the reply.

```
\if\xoptiontest\answer ... \else ... \fi
```

The function \xoptiontest returns a boolean value; it is designed for use with \readline or \xreadline, to trap the special responses ? Q q X x before executing some conditional code. It returns true if and only if the replacement text of \answer is a single character matching one of those listed. This is used when you are prompting for a response that can be an arbitrary string of characters, but you want to allow the user still to get help or quit with the same one-character responses that are recognized in other situations.

2.2 Definitions

We start by using the \localcatcodes function from grabhedr.sty to save current catcodes and set new catcodes for certain significant characters, as explained (at more length) in dialog.sty.

```
\localcatcodes{\@{11}%
  \~{13}\"{12}\#{6}\^{7}\`{12}\${3}\:{12}}
```

\menuprefix is a string added at the beginning of each menu to pretty it up a little (or uglify it a little, depending on your taste). The length of the default string is 70 characters, not counting the two newline characters. By using \storexmesj we get embedded newlines corresponding to the ones seen here. [That is, except for the extra line break (where the newline character is commented out), needed to make this fit in *TUGboat*'s column width.]

```
\storexmesj\menuprefix{
===================================%
===================================
}
```

The default value for \menusuffix is the same as for \menuprefix.

```
\let\menusuffix=\menuprefix
```

The default for \inmenuA and \inmenuB is a single newline, which will produce a blank line on screen because they will occur after an \endmenuline, which also contains a newline.

```
\storemesj\inmenuA{
}
\storemesj\inmenuB{
}
```

The default value for \menuline is two spaces. This means that each line in the middle section of a menu defined by \fmenu or \fxmenu will be indented two spaces.

```
\storemesj\menuline{  }
```

By default, no spaces are added at the beginning of a line in the top or bottom section of a menu:

```
\def\menutopline{}
\def\menubotline{}
```

\endmenuline is just a newline.

```
\storemesj\endmenuline{
}%
```

This definition of \menunumber adds square brackets and a following space around each item number.

```
\def\menunumber#1{[#1] }
```

This definition of \menuprompt is suitable for the purposes of listout.tex but will probably need to be no-op'd or changed for other applications.

```
\def\menuprompt{\promptmesj{Your choice? }}
```

Each of the three pieces of a menu gets its own token register.

```
\newtoks\menufirstpart
\newtoks\menuchoices
\newtoks\menulastpart
```

The 'arguments' of \fmenu are #1 menu name, #2 optional argument specifiers, #3 preliminary text, #4 list of menu choices, #5 following text. But at first we read only the first two because we want to change some catcodes before reading the others. The auxiliary function \fxmenub is shared with \fxmenu.

Because of the catcode changes done by \mesjsetup, newlines, spaces, or percent signs between the three final arguments will not be ignored. To get around this, we use the peculiar #{ feature of TEX, in intermediate scratch functions called \@tempa, to read and discard anything that may occur between one closing brace and the next opening brace. Token register assignments are used to read the arguments proper.

```
\def\fmenu#1#2#{\mesjsetup
  \catcode`\#=6 % for parameters
  \toks@{\fxmenub{\gdef}{\begingroup}{}#1{#2}}%
  \def\@tempa##1##{%
    \def\@tempa####1####{%
      \def\@tempa{\the\toks@}%
      \afterassignment\@tempa \menulastpart}%
    \afterassignment\@tempa \menuchoices}%
  \afterassignment\@tempa \menufirstpart
}
```

Before proceeding to define \fxmenub, we must deal with a subproblem. What we will have to work with is three pieces of text in the token registers \menufirstpart, \menuchoices, and \menulastpart, containing active ^^M characters to mark line breaks, including possibly but not necessarily ^^M at the beginning and at the end of each piece. What we would like to do, for each piece, is to remove the first ^^M, if there is one, and the last one, if there is one. The function \stripcontrolMs does this.

The technical details behind \stripcontrolMs found in menus.doc are skipped here for the sake of brevity, as they are unlikely to be interesting except to real TEX exegetes.

The argument of \stripcontrolMs is a token register. The text of the token register will be stripped of a leading and trailing ^^M if either or both are present, and the remainder text will be left in the token register.

```
\begingroup \lccode`\~=`\^^M
\lowercase{%
\gdef\stripcontrolMs#1{\expandafter\stripM
  \expandafter$\the#1$~$$\stripM#1}
}% end lowercase
```

```
\lowercase{%
\gdef\addmenulines#1#2#3{%
```

Add #2 at the beginning and #3 at the end of every line of token register #1.

```
  \def ~##1~##2{%
    #1\expandafter{\the#1#2##1#3}%
    \ifx\end##2\expandafter\@gobbletwo\fi~##2}%
  \edef\@tempa{\nx@~\the#1\nx@~}#1{}%
  \@tempa\end}
}% end lowercase
\endgroup % restore lccode of ~
```

The function \fxmenub is the one that does most of the hard work for \fmenu and \fxmenu. Argument #4 is the name of the menu, #5 is the argument specifiers (maybe empty). Arguments #1#2#3 are assignment type, extra setup, and expansion control; specifically, these arguments are \gdef \begingroup \empty for \fmenu or \xdef \xmesjsetup and an extra \noexpand for \fxmenu.

That this function actually works should probably be regarded as a miracle rather than a result of my programming efforts.[4]

```
\def\fxmenub#1#2#3#4#5{%
  \stripcontrolMs\menufirstpart
  \stripcontrolMs\menulastpart
  \stripcontrolMs\menuchoices
  \addmenulines\menuchoices\menuline\endmenuline
  \actively\let\^^M\relax % needed for \xdef
```

Define #4. Expansion control is rather tricky because of the possibility of parameter markers inside \menufirstpart, \menuchoices or \menulastpart.

```
  \toks@{\long#1#4#5}% e.g. \xdef\foo##1##2
```

If \menufirstpart is empty, we don't add the separator material \inmenuA.

```
  \edef\@tempa{\the\menufirstpart}%
```

[4] Let's see, three miracles is a prerequisite for sainthood in the Catholic church — only two more needed for Don Knuth to be a candidate ...

```
  \ifx\@tempa\@empty
    \let\nxa@\@gobble
  \else
    \addmenulines\menufirstpart
      \menutopline\endmenuline
    \let\nxa@\nx@
  \fi
```

If \menulastpart is empty, we don't add the separator material \inmenuB.

```
 \edef\@tempa{\the\menulastpart}%
  \ifx\@tempa\@empty
    \let\nxb@\@gobble
  \else
    \addmenulines\menulastpart
      \menubotline\endmenuline
    \let\nxb@\nx@
  \fi
```

Set up the definition statement that will create the new menu. #2 = begingroup or \xmesjsetup.

```
\edef\@tempa{{#3\nx@#3#2%
    \def#3\nx@#3\mesjtext{%
      #3\nx@#3\menuprefix
      \the\menufirstpart #3\nxa@#3\inmenuA
      \the\menuchoices #3\nxb@#3\inmenuB
      \the\menulastpart #3\nx@#3\menusuffix}%
    #3\nx@#3\sendmesj
    #3\nx@#3\menuprompt}}%
  \toks2 \expandafter{\@tempa}%
  \edef\@tempa{\the\toks@\the\toks2 }%
```

Temporarily \relaxify \menuline etc. in order to prevent their premature expansion if \xdef is applied.

```
  \let\menutopline\relax \let\menuline\relax
  \let\menubotline\relax \let\endmenuline\relax
  \let\menunumber\relax
  \@tempa % finally, execute the \gdef or \xdef
  \endgroup % matches \mesjsetup done by \fxmenu
}% end \fxmenub
```

Expanding analog of \fmenu.

```
\def\fxmenu#1#2#{\xmesjsetup
  \toks@{\fxmenub{\xdef}{\xmesjsetup}\nx@#1{#2}}%
  \def\@tempa##1##{%
    \def\@tempa####1####{%
      \def\@tempa{\the\toks@}%
      \afterassignment\@tempa \menulastpart}%
    \afterassignment\@tempa \menuchoices}%
  \afterassignment\@tempa \menufirstpart
}
```

```
\def\notyet#1{%
  \errmessage{Not yet implemented:  \string#1}}
```

These two functions aren't implemented yet.

```
\def\nmenu#1#2#3#4#5{\notyet\nmenu}
```

```
\def\nxmenu#1#2#3#4#5{\notyet\nxmenu}
```

2.3 Menu traversal functions

For reliable travel up and down the menu tree, we need to push and pop the value of \curmenu as we go along. Among other things, \curmenu is used to repeat the current menu after a help message.

```
\newtoks\optionstack
```

```
\let\curmenu\@empty
```

Start of a stack element.

```
\let\estart\relax
```

End of a stack element.

```
\let\eend\relax
```

name space constriction

```
\def\pushoptions#1{%
  \edef\pushtemp{\estart
    \def\nx@\curmenu{\curmenu}%
    \eend
    \the\optionstack}%
  \global\optionstack\expandafter{\pushtemp}%
  \edef\curmenu{\curmenu#1}%
}
```

```
\def\popoptions{%
  \edef\@tempa{\the\optionstack}%
  \ifx\@empty\@tempa
    \errmessage{Can't pop empty stack
      (\string\optionstack)}%
  \else
    \def\estart##1\eend##2\@nil{%
      \global\optionstack{##2}%
      \let\estart\relax##1}%
    \the\optionstack\@nil
  \fi
}
```

The X option is a total exit from the menu maze, as compared to \moptionQ, which returns you to the previous menu level.

```
\fmesj\moptionX{Exiting . . .}
```

```
\def\repeatoption{%
  \csname moption\curmenu\endcsname}
```

```
\def\moptionQ{\popoptions \repeatoption}
```

The sole reason for using \fxmesj rather than \fmesj here is to use % to comment out the initial newline, as the line break was needed only for convenient printing of this documentation within a narrow column width.

```
\fxmesj\badoptionmesj#1{%
?---I don't understand "#1".}
```

The function `\optionexec` takes one argument, which it uses together with `\curmenu` to determine the next action. The argument is expected to be a macro containing a single letter, the most recent menu choice received from the user.

Common options such as `?`, `Q`, or `X` that may occur at any level of the menu system are handled specially, to cut down on the number of control sequence names needed for a csname implementation of the menus.

```
\def\optionexec#1{%
  \if ?#1\relax \let\@tempa\moptionhelp
  \else \if Q#1\relax
    \ifx\curmenu\@empty \let\@tempa\moptionX
    \else \let\@tempa\moptionQ \fi
  \else \if X#1\relax \let\@tempa\moptionX
  \else
```

Because special characters, including backslash, are deactivated by `\readChar`, we can apply `\csname` without fearing problems from responses such as `\relax`.

```
    \expandafter\let\expandafter\@tempa
      \csname moption\curmenu#1\endcsname
    \ifx\@tempa\relax
      \badoptionmesj{#1}\let\@tempa\repeatoption
    \else
      \pushoptions{#1}%
    \fi
  \fi\fi\fi
```

We save up the next action in `\@tempa` and execute it last, to get tail recursion.

```
  \@tempa
}
```

Really big menu systems could get around TeX memory limits by storing individual menus or groups of menus in separate files and using `\optionfileexec` in place of `\optionexec` to retrieve the menu text from disk storage instead of from main memory. However there are a number of technical complications and I probably won't get around to working on them in the near future.

```
\def\optionfileexec#1{\notyet\optionfileexec}
```

The function `\xoptiontest` must return true if and only if the macro `#1` consists entirely of one of the one-letter responses `? Q q X x` that correspond to special menu actions. The rather cautious implementation with `\aftergroup` avoids rescanning the contents of `#1`, just in case it contains anything that's `\outer`.

```
\def\xoptiontest#1{TT\fi
  \begingroup \def\0{?}\def\1{Q}%
  \def\2{q}\def\3{x}\def\4{X}%
    \aftergroup\if\aftergroup T%
    \ifx\0#1\aftergroup T%
    \else\ifx\1#1\aftergroup T%
    \else\ifx\2#1\aftergroup T%
    \else\ifx\3#1\aftergroup T%
    \else\ifx\4#1\aftergroup T%
    \else \aftergroup F%
    \fi\fi\fi\fi\fi
  \endgroup
}
```

Default help message, can be redefined if necessary. The extra newlines commented out with `%` are here only for convenient printing within a narrow column width.

```
\fxmesj\menuhelpmesj{&\menuprefix%
A response of Q will usually send you back to %
the previous menu.
A response of X will get you entirely out of %
the menu system.
&\menusuffix%
Press the <Return> key ( Enter ) to continue:
}
```

```
\def\moptionhelp{%
  \menuhelpmesj \readline{}\reply \repeatoption}
```

`\moptionhelp` is the branch that will be taken if the user enters a question mark in response to a menu.

```
\def\moptionhelp{%
  \menuhelpmesj \readline{}\reply \repeatoption}
```

```
\expandafter\def\csname moption?\endcsname{%
  \moptionhelp}
```

The function `\specialhelp` can be used to provide a one-time alternate help message tailored to a specifc response given by the user. It defines the first argument (the macro containing the response) to contain `?`, then redefines `\menuhelpmesj` to use the message text given in arg `#2`.

```
\def\specialhelp#1#2{%
  \let\specialhelpreply=#1\def#1{?}\begingroup
  \def\menuhelpmesj{\let#1\specialhelpreply
    \promptxmesj{#2\
Press <return> to continue:}\endgroup}%
}
```

Init.

```
\def\specialhelpreply{}
```

```
\def\lettermenu#1{%
  \csname menu#1\endcsname
  \readChar{Q}\reply \optionexec\reply
}
```

Restore any catcodes changed locally, and depart.

```
\restorecatcodes
```

```
\endinput
```

Appendix
Miscellaneous support functions: grabhedr.sty

This file defines a function \inputfwh to be used instead of \input, to allow TeX to grab information from standardized file headers in the form proposed by Nelson Beebe during his term as president of the TeX Users Group. Usage:

\inputfwh{⟨*filename*⟩}

Functions \localcatcodes and \restorecatcodes for managing catcode changes are also defined herein, as well as a handful of utility functions, mostly from latex.tex: \@empty, \@gobble, \@gobbletwo, \afterfi, \fileversiondate, \trap.input.

The use of \inputfwh, \fileversiondate, and \trap.input as illustrated in \dialog.sty is cumbersome kludgery that in fact should be handled instead by appropriate functionality built into the format file. But alas, none of the major formats yet have anything along these lines. (It would also help if TeX made the current input file name accessible, like \inputlineno.)

By enclosing this entire file in a group, saving and restoring catcodes 'by hand' is rendered unnecessary. This is perhaps the best way to locally change catcodes, better than the \localcatcodes function defined below. But it tends to be inconvenient for the TeX programmer: every time you add something you have to remember to make it global; if you're like me, you end up making every change twice, with an abortive test run of TeX in between, in which you discover that a certain control sequence is undefined because you didn't assign it globally.

```
\begingroup
```

Inside this group, enforce normal catcodes. All definitions must be global in order to persist beyond the \endgroup.

```
\catcode96 12 % left quote
\catcode`\= 12
\catcode`\{=1 \catcode`\}=2 \catcode`\#=6
\catcode`\$=3 \catcode`\~=13 \catcode`\^=7
\catcode`\_=8 \catcode`\^^M=5 \catcode`\"=12
```

Make @ a letter for use in 'private' control sequences.

```
\catcode`\@=11
```

A.1 Preliminaries

For \@empty, \@gobble, ... we use the LaTeX names so that if grabhedr.sty is used with LaTeX we won't waste hash table and string pool space.

Empty macro, for \ifx tests or initialization of variables.

```
\gdef\@empty{}
```

Functions for gobbling unwanted tokens.

```
\long\gdef\@gobble#1{}
\long\gdef\@gobbletwo#1#2{}
\long\gdef\@gobblethree#1#2#3{}
```

The function \@car, though not really needed by grabhedr.sty, is needed by the principal customers of grabhedr.sty (e.g., dialog.sty).

```
\long\gdef\@car#1#2\@nil{#1}
```

To define \@@input as in LaTeX we want to let it equal to the primitive \input. But if a LaTeX format is being used we don't want to execute that assignment because by now \input has changed its meaning. And if some other format is being used it behooves us to check, before defining \@@input, whether \input still has its primitive meaning. Otherwise there's a good chance \inputfwh will fail to work properly.

```
\ifx\UndEFiNed\@@input % LaTeX not loaded.
```

This code shows a fairly easy way to check whether the meaning of a primitive control sequence is still the original meaning.

```
  \edef\0{\meaning\input}\edef\1{\string\input}%
  \ifx\0\1%
    \global\let\@@input\input
  \else
    \errhelp{%
Grabhedr.sty needs to know the name of the
\input primitive in order to define \inputfwh
properly. You might want to try to patch up the
problem by letting \input = \primitiveinput
before inputting grabhedr.sty.}
    \errmessage{%
      Non-primitive \noexpand\input detected}%
  \fi
\fi
```

Scratch token register.

```
\global\toksdef\toks@=0
```

Sonja Maus's function for throwing code over the \fi ("An Expansion Power Lemma", *TUGboat* vol. 12, no. 2, June 1991). (Except that she called this function \beforefi.)

```
\long\gdef\afterfi#1\fi{\fi#1}
```

We will be using `\noexpand` a lot; this abbreviation improves the readability of the code.

```
\global\let\nx@\noexpand
```

Another convenient abbreviation.

```
\global\let\xp@\expandafter
```

A.2 Reading standard file headers

The function `\inputfwh` ('input file with header') inputs the given file, checking first to see if it starts with a standardized file header; if so, the filename, version and date are scanned for and stored in a control sequence.

For maximum robustness, we strive to rely on the fewest possible assumptions about what the file that is about to be input might contain.

Assumption 1: Percent character `%` has category 14. I.e., if the first line of the file to be input starts with `%`, it is OK to throw away that line.

```
\begingroup \lccode`\.=`\%%
\lowercase{\gdef\@percentchar{.}}%
\endgroup
```

The function `\fileversiondate` is not only a useful support function for `\inputfwh`, it can also be used by itself at the beginning of a file to set file name, version, and date correctly even if the file is input by some means other than `\inputfwh`—assuming that the arguments of the `\fileversiondate` command are kept properly up to date.

```
\gdef\fileversiondate#1#2#3{%
  \xp@\xdef\csname#1\endcsname{#2 (#3)}%
  \def\filename{#1}\def\fileversion{#2}%
  \def\filedate{#3}%
  \message{#1 \csname#1\endcsname}%
}
```

And now apply `\fileversiondate` to this file.

```
\fileversiondate{grabhedr.sty}{0.9g}{6-Jul-1994}
```

Currently (July 1994) `filehdr.el` by default adds a string of equal signs (with an initial comment prefix) at the very top of a file header. This string must be scanned away first before we can start looking for the real information of the file header.

```
\xdef\@headerstart{%
  \@percentchar\@percentchar\@percentchar\space
  ==================================%
  ==================================}
```

The purpose of this function is just to scan up to the opening brace that marks the beginning of the file header body. Everything before that is ignored, not needed for our present purposes.

```
\gdef\@scanfileheader#1@#2#{\@xscanfileheader}
```

Throw in some dummy values of version and date at the end so that all we require from a file header is that the filename field must be present. The version and date fields can be present or absent, in any order, but the corresponding TEX variables `\fileversion` and `\filedate` will not get set properly unless the order is: filename, [. . . ,] version, [. . . ,] date.

```
\long\gdef\@xscanfileheader#1{%
  \@yscanfileheader#1{} version = "??",
  date = "??",\@yscanfileheader}
```

This function assumes that filename, version, and date of a file are listed in that order (but not necessarily adjacent). It's possible for the version and date to be missing, or out of order, but in the latter case wrong values may be passed on to the `\fileversiondate` call. Trying to handle different orderings would be desirable but I haven't yet been struck by a suitable flash of insight on how to do it without grubby, time-consuming picking apart of the entire file header.

```
\long\gdef\@yscanfileheader
  #1 filename = "#2",#3 version = "#4",%
  #5 date = "#6",#7\@yscanfileheader{%
  \endgroup
  \csname fileversiondate\endcsname{#2}{#4}{#6}%
}
```

This function has to look at the first line of the file to see if it has the expected form for the first line of a file header.

```
\begingroup
\lccode`\$=`\^^M
\lowercase{\gdef\@readfirstheaderline#1$}{%
  \toks@{#1}%
  \edef\@tempa{\@percentchar\the\toks@}%
  \ifx\@tempa\@headerstart
    \endgroup \begingroup
    \catcode`\%=9 \catcode`\^^M=5 \catcode`\@=11
```

Double quote and equals sign need to be category 12 in order for the parameter matching of `\@xscanfileheader` to work, and space needs its normal catcode of 10.

```
    \catcode`\ =10 \catcode`\==12 \catcode`\"=12
  \xp@\@scanfileheader
  \else
    \message{(* Missing file header? *)}%
    \afterfi\endgroup
  \fi}
\endgroup
```

An auxiliary function.

```
\gdef\@xinputfwh{%
  \ifx\next\@readfirstheaderline
```

Sanitize a few characters. Otherwise an unmatched brace or other special character might cause a problem in the process of reading the first line as a macro argument.

```
    \catcode`\%=12 \catcode`\{=12 \catcode`\}=12
    \catcode`\\=12 \catcode`\^^L=12
    \catcode`\^=12
%    Unique terminator token for the first line.
    \catcode`\^^M=3\relax
  \else \endgroup\fi
}
```

Auxiliary function, carries out the necessary `\futurelet`.

```
\gdef\@inputfwh{\futurelet\next\@xinputfwh}
```

Strategy for (almost) bulletproof reading of the first line of the input file is like this: Give the percent sign a special catcode, then use `\futurelet` to freeze the catcode of the first token in the input file. If the first token is *not* a percent character, then fine, just close the group wherein the percent character had its special catcode, and proceed with normal input; the first token will have its proper catcode because we did not change anything except the percent character. Otherwise, we still proceed with 'normal' input execution, but by making % active and defining it suitably, we can carry out further tests to see if the first file line has the expected form (three percent signs plus lots of equal signs).

```
\gdef\inputfwh#1{%
  \begingroup\catcode`\%=\active
  \endlinechar`\^^M\relax
  \lccode`\~=`\%\relax
  \lowercase{\let~}\@readfirstheaderline
  \xp@\@inputfwh\@@input #1\relax
}
```

A.3 Managing catcode changes

A survey of other methods for saving and restoring catcodes would be more work than I have time for at the moment. The method given here is the best one I know (other methods use up one extra control sequence name per file, or don't robustly handle multiple levels of file nesting).

The `\localcatcodes` function changes catcodes according to the character/catcode pairs given in its argument, saving the previous catcode values of those characters on a stack so that they can be retrieved later with `\restorecatcodes`. Example:

```
\localcatcodes{\@{11}\"\active}
```

to change the catcode of `\@` to 11 (letter) and the catcode of " to 13 (active). In PLAINTEX you'd better be careful to use + instead of `\+` in the argument of `\localcatcodes` because of the outerness of `\+`.

The way this function works is by using token registers 0 and 4 to accumulate catcode assignment statements: in `\toks0` we put the statements necessary to restore catcodes to their previous values, while in `\toks 4` we put the statements necessary to set catcodes to their new values.

```
\gdef\localcatcodes#1{%
  \begingroup \toks@{}\toks4 {}%
  \def\do##1##2{\ifx \end##1%
```

Finished processing the list; Take the accumulated contents of `\toks@` and add them as a new element at the top of the catcode stack. Adding the { } makes the new element easily poppable.

```
      \ifx\@empty\@catcodestack
        \gdef\@catcodestack{{}}%
      \fi
      \toks2 \xp@{\@catcodestack}%
      \xdef\@catcodestack{%
        {\the\toks@}\the\toks2 }%
      \xp@\endgroup\the\toks4 \relax
    \else
```

Add a catcode-restore statement at the beginning of `\toks@`.

```
      \edef\0{\catcode`\nx@##1=
        \the\catcode`##1\relax \the\toks@}%
      \toks@\xp@{\0}%
```

Add a catcode-setting statement at the end of `\toks4`.

```
      \toks4 \xp@{\the\toks4\relax
        \catcode`##1=##2\relax}%
      \afterfi\do
    \fi
  }%
  \do#1\end\relax
}
```

Initialize the stack with an empty element; otherwise popping the next-to-last element would wrongly remove braces from the last element. But as a matter of fact this init is just for show since `\localcatcodes` is careful to add an empty element whenever necessary.

```
\gdef\@catcodestack{{}}
```

The function `\restorecatcodes` has to pop the stack and execute the popped code.

```
\gdef\restorecatcodes{%
  \begingroup
  \ifx\@empty\@catcodestack
    \errmessage{Can't pop catcodes;
      \nx@\@catcodestack = empty}%
    \endgroup
  \else
    \def\do##1##2\do{%
      \gdef\@catcodestack{##2}%
```

Notice the placement of `#1` after the `\endgroup`, so that the catcode assignments are local assignments.

```
      \endgroup##1}%
    \xp@\do\@catcodestack\do
  \fi
}
```

A.4 Trapping redundant input statements

The utility `listout.tex` calls `menus.sty`, which calls `dialog.sty`, and all three of these files start by loading `grabhedr.sty` in order to take advantage of its functions `\fileversiondate`, `\localcatcodes`, and `\inputfwh`. But consequently, when `listout.tex` is used there will be two redundant attempts to load `grabhedr.sty`. The straightforward way to avoid the redundant input attempts would be to surround them with an `\ifx` test:

```
\ifx\undefined\fileversiondate
  \input grabhedr.sty \relax
  \fileversiondate{foo.bar}{0.9e}{10-Jun-1993}
\fi
```

This method has a few drawbacks, however: (1) the conditional remains open throughout the processing of everything in `grabhedr.sty` and the `\fileversiondate` statement, which makes any `\else` or `\fi` mismatch problems harder to debug; (2) if `\undefined` becomes accidentally defined the `\ifx` test will fail; (3) choosing the right control sequence to test against `\undefined` requires a little care.

In a situation where we know that the file to be input has had `\fileversiondate` applied to it, if it was already input, then we have a failsafe control sequence that we can test to find out whether the file has already been input — the name of the file. Assuming a standard form for the input statement (one that will work with either plain TeX or LaTeX, and makes as few assumptions as possible), we can write a function that will trap input statements and execute them only if the given file has not yet been loaded:

```
\csname trap.input\endcsname
\input grabhedr.sty \relax
\fileversiondate{foo.bar}{1.2}{1993-Jun-07}
```

The function `\trap.input` scans for an input statement in canonical form and executes it if and only if the file has not yet been input (more precisely, if the control sequence consisting of the file name is undefined, which means that it has not had `\fileversiondate` applied to it). The canonical form that I consider to be the best is `\input` ⟨*full file name*⟩␣`\relax`. Having the `\relax` means that the input statement will not try to expand beyond the end of the line if `\endlinechar` is catcoded to 9 (ignore), as is done rather frequently now by progressive TeX programmers. The `\relax` would ordinarily render the space after the file name unnecessary, but I prefer leaving the space in to avoid interfering with redefinitions of `\input` to take a space-delimited argument that are occasionally done to achieve other special effects (see, for example, "Organizing a large collection of stylefiles", by Angelika Binding, *Cahiers GUTenberg*, numéro 10–11, septembre 1991, p. 175.) LaTeX's argument form `\input{...}` cannot, unfortunately, be part of the canonical form if PLAINTeX compatibility is required.

```
\expandafter\gdef\csname trap.input\endcsname
  \input#1 \relax{%
    \expandafter\ifx\csname#1\endcsname\relax
      \afterfi\inputfwh{#1}\relax
    \fi}
```

End the group that encloses this entire file, and then call `\endinput`.

```
\endgroup
\endinput
```

◇ Michael Downes
49 Weeks Street
North Smithfield, RI 02895
U.S.A
mjd@math.ams.org

LaTeX

LaTeX V3: Philology & Typography Reports to Read, Reports to Do

Bernard GAULLE

Some time ago Frank Mittelbach proposed several *Research tasks* that might be done by volunteers in order to bring some proposals to the LaTeX V3 design team. The list of tasks (today 24) is still alive and is nearly about the following topics: Syntax for tables, chemistry, commutative diagrams, index or bibliographical commands; Experimenting with `\emergencystretch`; Conventions for footnotes and endnotes, for `^^` usages and about typography; Multilingual requirements; Most commonly used LaTeX styles; Math font handling; Converting numbers to textual form; Rewrite of MakeIndex in WEB; etc. (For the latest information about these tasks and the volunteers involved, get the file `vol-task.tex` from the archive on the server located at `Niord.SHSU.edu` in `[FILESERV.LTX3PUB]`).

A few tasks still have no coordinator or even no volunteers; don't hesitate to join us.

Some groups have already started to report (like VTs 2, 4, 5, 11, 15 and 16) and continue to work. So don't hesitate to comment on their reports if you think it useful that your opinion be known before any decision is definitely taken.

As coordinator of the VT15 group about multilingual documents, I would like to give an account of the work done and suggest that people send me reports, especially about the parts of the task which are still void because we simply don't know what are the typographic habits in your country.

The first part of our work has been reported; it was most recently updated in March, and is available on the server at `Niord.SHSU.edu` (`vt15d02.tex`, V1.02). In this report we discuss the standard and non-standard TeX mechanisms for language processing, character sets, filters and font encodings. We suggest what could be a language definition in LaTeX V3. Where does a language apply? How does the user switch from one language to another? What could be a *default* language, a *main* language? ... All these questions are discussed in the document that ends with an important choice: *With LaTeX V3, every document is multilingual.*

Now we have to collect as many typographic country habits as possible, either national or local but used by numerous people for years. After that we will establish a list of common usages that require specific mechanisms. And then report again.

If you are aware of specific things, for example, bibliographic habits, please summarize them and let us know. If you are involved in any "Typographer's Inn" and know how print correctly, let's say, a caption title in an Ethiopian text, please send us few words, preferably in English. References as well as samples are welcome too. It will greatly help for the design of LaTeX V3, the LaTeX *for* the next century.

◇ Bernard GAULLE
IDRIS-*CNRS*, BP 167, F–91403
ORSAY Cedex, FRANCE
email: gaulle@idris.fr

Tough table becomes easy with PiCTeX, but it's even easier with LaTeX

Claudio Beccari

In *TUGboat* 14.4 (1993) p. 420, Kevin Carmody presents an interesting way of using PiCTeX for producing a table with "gnomons" that Carmody says "defeated my best efforts to typeset it in plain TeX".

No doubt PiCTeX can be used to draw almost any simple shape with text in it, but I'd rather have liked to see something tougher than the simple table with "gnomons", which, maybe, is not so simple to set in plain TeX (although careful reading of *The TeXbook* where ruled tables are dealt with and an intelligent use of `\multispan` should be sufficient to overcome the little intricacies of that table), but is almost trivial with LaTeX (see Figure).

Due to the repetition of the same construct in almost half the entries, with the help of the LaTeX command `\newcommand`, `\m` was defined in order to reduce typing, exactly as Leslie Lamport suggests to do; the vertical spacing of the array was adjusted with the redefinition of `\arraystretch`; in order to have the last column as wide as the others, the last entry was set within a zero-width box as is suggested in the LaTeX book; in other words, just plain LaTeX:

```
\begin{displaymath}
\newcommand{\m}[1]{\multicolumn{1}{r|}{#1}}
\renewcommand{\arraystretch}{1.6}
\begin{array}{*{10}{r}}
\m{1}   &\m{3}  &\m{5}  & ...   &\m{19} \\
```

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

Figure 1: Infinite Rectangular Array

```
\cline{1-1}
1       &\m{4}  &\m{7}  & ...    &\m{28} \\
\cline{1-2}
1       &5      &\m{9}  & ...    &\m{37} \\
...
\cline{1-9}
1       &12     &23     & ...    &
               \m{\makebox[0pt][r]{100}}\\
\cline{1-10}
\end{array}
\end{displaymath}
```

An experienced LaTeX user could further reduce typing by means of smarter definitions, but this is not the point. I suppose that also TeX users could do the same, since after all both `\multicolumn` and `\cline` are made up essentially with `\multispan`.

In spite of this let us thank Kevin Carmody for drawing our attention to the facilities offered by PiCTeX for typesetting something unusual.

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

Letters

On the review of *TeX in Practice*

I was interested to read Frank Pappas's review of TeX in Practice by Stephan von Bechtolsheim. I recently purchased a copy of the series and could not help but react to the quality problems myself. However, Pappas was a bit too quick to remove responsibility from the author and lay it all on Springer-Verlag. As a compositor/typesetter for over 30 technical books for various publishers, I would like to share some thoughts with the TUG community.

1. Technical authors, particularly those of us who are TeXnicians, tend to equate knowledge of TeX with knowledge of bookmaking. TeX in Practice is only one of many TeX books that suffer from poor writing, ugly design, lack of attention to composition details, or typesetting errors.

2. Technical authors believe, and publishers are too quick to accept, that a book authored with TeX is "just going to typeset itself." There is nothing about electronic manuscripts or TeX that eliminates the traditional needs for copyediting, proofreading, manual page make-up, hand-crafted tables, and so forth.

3. Publishers desire to save money by deferring many aspects of book production to the author. This has come to include complete coding of source files, creation or alteration of style files, art preparation, proofreading, page make-up, and indexing. In the old days, publishers did not believe that authors could do these things, possibly because the authors did not know how to handle hot lead. Why, in the age of desktop publishing, do authors suddenly know any more about such crafts? We may be "empowered," but we are not necessarily enlightened.

At the risk of appearing self-serving, I admonish authors and publishers alike: nothing has changed from traditional bookmaking, except that manuscripts are now provided in electronic files. Authors, you don't know all that much about bookmaking. Publishers, do not trust that authors know all that much about bookmaking. Just as an author expects her publisher to listen carefully regarding the content of the book, so also should she expect advice from the publisher on book production, and then heed that advice.

Paul C. Anagnostopoulos
Windfall Software
433 Rutland Street
Carlisle, MA 01741
508 371-2316

Abstracts

Baskerville
Contents of Recent Issue

Volume 4, Number 1 February 1994

SEBASTIAN RAHTZ, Editorial; Colophon; p. 1

The editor requests material for *Baskerville*, lists the schedule of deadlines for each issue of 1994 (each issue will have a special theme), and announces the first release of LaTeX2ε which was followed by the launch of *LaTeX Companion*. The TUG '94 meeting is announced.

This issue deals especially with PostScript issues, and was created entirely with the test distribution of LaTeX2ε.

ANGUS DUGGAN, Mixing and matching PostScript fonts; pp. 2–5

Author's words excerpted "The Apple LaserWriter was the product that introduced PostScript to the world, bringing in its wake a major change in the publishing and printing industry. PostScript is now used everywhere from the home to high quality printing presses..."

Unfortunately, the LaserWriter 13 typefaces look terrible when used together. "Fortunately the outline descriptions of fonts in PostScript allow us to do something about the variations in width, height and weight of the fonts" and the author goes on to describe methods used to scale fonts *anamorphically*.

"The techniques described in this note make minor distortions to a set of typefaces in order to make their use together more pleasing. These techniques are only useful in limited circumstances *i.e.* when the only fonts you can rely on using are the base LaserWriter fonts. As noted in the introduction, there are circumstances where this is the case, and in these cases almost anything is better than the ugly sight of Times, Helvetica and Courier mixed together in their natural state."

ALAN JEFFREY, Building virtual fonts with 'fontinst'; pp. 6–7

Author's Introduction: "This document gives a brief overview of the `fontinst` package. The `fontinst` package is used to build *virtual fonts* (VFs) which allow PostScript fonts to be used as drop-in replacements for the Computer Modern fonts in TeX."

BERTHOLD HORN, Do you really need virtual fonts?; pp. 8–10

The author acknowledges that many people feel strongly about these issues and tries to sway opinion, going into some detail about why virtual fonts are *not* needed for use of non-CM fonts—*or* for re-encoding. He explains what virtual fonts *may* actually be useful for (and why even for those purposes there are better ways of going about things). The author is affiliated with Y&Y, and states that, "...I hope what I say will not sound *too* much like advertising."

YANNIS HARALAMBOUS, Further thoughts on virtual fonts; pp. 10–12

The author responds to some of the issues mentioned in the above article, with the statement that "METAFONT can do things PostScript cannot even dream of." He mentions several tools used to convert PostScript fonts to METAFONT, opening new horizons to digital typography. There is a utility called DVICopy (written by Peter Breitenlohner) which can *de-virtualize* a document's DVI file, replacing characters from a virtual font by the real character(s) they represent. Yannis states that in the past few years there has been much more development in public domain TeXware than in commercial software around TeX. He suggests virtual fonts may be one of the innovations that all commercial products have not yet adopted, but that we should, "think twice before giving away virtual fonts in exchange for something poorer (PostScript font re-encoding), when we can equally well use both at the same time and produce even better results."

M. GOOSSENS and S. RAHTZ, Colour slides with LaTeX and **seminar**; pp. 12–16

This article provides an introduction to the use of the LaTeX package **seminar**, in conjunction with a PostScript printer, and the set of macros called PSTricks. **seminar** can be used with almost all other LaTeX packages but its main job is to produce transparencies, though it can also make accompanying notes from the same file. It is compatible with AMS-LaTeX and LaTeX2ε.

JONATHAN FINE, Back(s)lash; pp. 16–17

This the first in a series of columns devoted to the subtleties of programming TeX. The focus will be on the primitive commands and low-level features of TeX. The column is devoted to `\csname` and to avoiding its side effects.

R.A. Bailey, Topical tip: Numbering theorems and corollaries in LaTeX; pp. 17–18

Good tips on how to use the `thm` environment to choose how to label your theorems and corollaries.

Malcolm Clark, Malcolm's Gleanings; pp. 18–21

Computers and Typography, 'compiled by' Rosemary Sassoon, is reviewed by Malcolm. The review is based on one which appeared in the *Information design journal*, vol. 7, no. 2, 1993, pp.161–6.

Information design journal: Malcolm describes this journal as interesting, appropriate and local. The range of papers is broad, both in content and in style, and he recommends it to the reader.

Nonsense: Malcolm has described the Christmas "birth" of the test release of LaTeX2ε with great wit, going on to discuss the pitfalls of the LaTeX logo(s), to a query about NTS "taking over the world". He winds up questioning the trend of *TUGboat* getting thinner and less timely.

Letters to the editor; pp. 21–22

Includes letters from Alan Hoenig and Phil Taylor.

Obtaining TeX; pp. 22–23

This notice lists the UK CTAN archive addresses and lists the main directories. There are instructions on obtaining Unix tapes and PC and Mac disks.

UKTUG Business Reports; p. 23

Volume 4, Number 2 April 1994

Sebastian Rahtz, Editorial; Colophon; p. 1

The schedule of deadlines is listed. An *apology* is given to Alan Jeffrey about his article in *Baskerville* 4.1, in which one of the font tables suffered from font-encoding problems. The corrected table is included here.

The issue was created entirely with the test distribution of LaTeX2ε, using Y&Y's implementation of TeX, and printed on a Hewlett Packard LaserJet 4. The editor used a slightly extended Windows ANSI encoding, using Y&Y's comprehensive font tools to re-encode the fonts, and then using Alan Jeffrey's *fontinst* to build a full set of EC-encoded virtual fonts.

Letters to the editor; p. 2

John Bowsher posts a letter with questions regarding AtariTeX, and the dvi_lj2 printer driver. There is another letter from Bowsher about the use of TeX logo when typeset using ITC New Baskerville Roman. *Editor's reply was that he is quite right ... but we need clarification from Knuth ....*

Michel Goossens, LaTeX2ε, an overview; pp. 2–11

Summary: This article gives an overview of the new or extended user commands available with LaTeX2ε, the new LaTeX release, compared to the previous version, LaTeX 2.09. After introducing the new preamble commands, the extensions for defining new commands and environments, and handling length and boxes are discussed. The new font selection commands are explained, both for text and math, and it is shown how to easily use different font families. A list of supported class and package files is given and new possibilities for controlling page contents and floats are discussed. Most of this material is described in much greater detail in *The LaTeX Companion* and in the second edition of the *LaTeX Reference Manual.*

Laurent Siebenmann, Using virtual fonts with Textures; pp. 12–17

Excerpted from author's Introduction: "For *Textures* users, virtual fonts offer the first opportunity to exploit the high performance of the scalable CM/S fonts, which *Textures* provides, in a way that permits perfect linebreaking and kerning for accented west European languages.

"...This article is very much user-oriented in the sense that the main point is to deliver performance we have been dreaming of since Knuth's virtual font standard was unveiled in 1989, ...its sections are called 'tutorials'."

Tutorial 1: *Textures* virtual font resources
Tutorial 2: *Textures* 'Type 1' virtual fonts
Tutorial 3: *Textures* DM virtual fonts
Tutorial 4: Trying out extended CM fonts for *Textures*
Tutorial 5: Tutorial on setting up DM fonts for *Textures* 1.6X
Tutorial 6: Maintaining preview speed with *Textures*' virtual fonts

The software described in this article, with the exception of *Textures*, has its current master posting on `matups.matups.fr` and most of it is mirrored on the CTAN archives.

Jonathan Fine, TeX and SGML — Friend or Foe?; pp. 18–19

The author presents personal observations and opinions on the subject of promoting TeX as the typesetting system for SGML documents. SGML is a specification language for compuscripts while TeX can produce high quality typesetting of SGML compuscripts, such as the contents of a database, and the tagging process adds information to the compuscript, thereby making it more valuable. The author states a belief that a TeX format can be written that will parse and typeset suitable SGML compuscripts.

PETER ABBOTT, Tips for installing LaTeX2ε for OzTeX; p. 19

The author provides some tips for the installation of LaTeX2ε files, obtained from the CTAN archives, on a Mac IIfx, for use with OzTeX version 1.6. (It took over 3 hours!)

MALCOLM CLARK, Malcolm's Gleanings; pp. 20–21

Malcolm announced the demise of *TeXline*; the last issue appeared in February 1992. He then goes on to list other means of dissemination of information about TeX and family by publications of LUGs. Electronic distribution of information is an obvious alternative, but who would bother subscribing to TUG or the UK group if one of the key benefits of membership, the publications, were available electronically? There are those who still prefer paper — the author being one. The author is hopeful that gadgets like the Sony DiscMan, and emerging technologies like Adobe's Acrobat may make portable electronic books a realistic alternative to information available via the Internet.

Carousel: Adobe Systems introduced the 'Multiple Masters' technology, which allows one to retain the 'look and feel' of a printed document when the correct fonts are not available. The solution was to develop a scheme where two font families were introduced which would attempt to emulate the characteristics of the 'missing' font(s). Multiple Masters were a prelude to Acrobat, which was originally code-named Carousel. An Acrobat document can be transmitted electronically without difficulty and read by anyone with an Acrobat reader. It no longer matters what software was used to prepare the original — you will still be able to see what the author intended, without having that software yourself.

Knuth, the video: University Video Communications, based at Stanford, now have a tape of Knuth's 'Computer musing' in which he discusses the One-Way Associative Law.

Sheep stealing in Barnet: Frederic Goudy's quotation is (mis)quoted in reference to an article in *Baskerville* 3(2), in which Eric Gill suggests letterspacing lower case as a substitute for italics. The misquote is, "Anyone who would letterspace lower case would steal sheep," — whilst Goudy is quoted correctly as saying, "Anyone who would letterspace black letter would steal sheep."

Public domain does not mean user unfriendly: Malcolm mentions several public domain software products in refute of a claim that 'important innovations (in TeX etc.) have always appeared first in public domain software'. Listed are TeX on the pc, done by Lance Carnes and David Fuchs; virtual fonts were released first by ArborText, Blue Sky's innovative Lightning Textures, and Michael Vulis' VTeX. Malcolm goes on to say that, "They ... supported us all for many years until public domain versions of the program finally became robust and reliable. I believe they still have a place in the development of TeX and TeX tools."

Hints and Tips: One of the 'events' at an RHBNC meeting was a LaTeX tables workshop, in which Geeti Granger of John Wiley & Sons provided a set of values (listed below) for parameters to make floating tables appear just where you want them.

parameter	default	suggested
topnumber	2	2
bottomnumber	1	2
totalnumber	3	4 or 2
dbltopnumber	2	2
`\topfraction`	0.7	0.9
`\bottomfraction`	0.3	0.5
`\textfraction`	0.2	0.1
`\floatpagefraction`	0.5	0.8
`\dbltopfraction`	0.7	0.9
`\dblfloatpagefraction`	0.5	0.8

JONATHAN FINE, Backslash — About `\def` and `\let`; pp. 22–23

This article is another in the series of 'topical tips', and will be added to a library of common TeX questions, to answer queries from those who do not participate in electronic network discussions.

The author explains the difference between `\let` and `\def`, introducing beginners to *tokens*. The execution of a `\def` will always create a macro, which in execution will assign a macro meaning to a control sequence. The `\let` command is used to transfer an existing meaning to a control sequence.

Answers to the two exercises which appeared in the previous issue were given, and two exercises on the current article are provided.

The UKTeX Users Group; pp. 23–24

Summary of group aims and activities, and a programme of meetings for 1994 – 5.

11th July, 1994 — *LaTeX fonts and graphics: a hands-on tutorial*

19th October, 1994 — *Annual General Meeting*
Winter, 1994 — *TEX SGML & electronic publishing*
Easter, 1994 — *Maths is what TEX does best of all*

(Summaries by Mimi Burbank)

Les Cahiers GUTenberg Contents of Recent Issues

Numéro 16 — février 1994 PSTricks et Seminar

[Editor's note: This thematic issue of the *Cahiers* is dedicated to color! Well, that's the side effect of the extended presentation of two well-known style files by Timothy van Zandt: PSTricks and Seminar. The three main articles were compiled by Denis Girou, with a preface by Timothy himself –Ch.]

TIMOTHY VAN ZANDT, Préface ; pp. 1–4

"It isn't easy to explain why someone would spend part of their free time doing TEX development work, but two reasons are the fun that programming with TEX gives you and the satisfaction you feel when others use and appreciate what you've created." Timothy's preface then goes on to talk about plans for enhancements, including easy use with LATEX2ε, and concludes with information on where the files can be found on CTAN (including additional style files used in the articles).

MICHEL GOOSSENS AND SEBASTIAN RAHTZ,
Composition en couleur avec LATEX [Colour composition using LATEX]; pp. 5–20

Author's summary: "In this article we show some basic principles underlying the use of colour. After explaining how colour is reproduced in printers, we show one implementation with LATEX of using it in your work and give a few examples of colour use. We also explain how to make simple colour separations using PostScript and dvips."

DENIS GIROU, Présentation de PSTricks
[Presenting PSTricks]; pp. 21–70

Author's summary: "The macro commands in the PSTricks package, developed by Timothy van Zandt, offer impressive new capabilities to (LA)TEX users, by giving them direct access to much of the power of PostScript, including complete support of color. In addition, Seminar, a development associated with PSTricks, allows users to easily make transparencies of great quality. This article [presents], through a large number of examples, from the simplest to the more complex, the diverse features of PSTricks, from the point of [view] of an end user."

MICHEL GOOSSENS AND SEBASTIAN RAHTZ,
Préparer les transparents avec Seminar [Preparing transparencies with Seminar]; pp. 71–82

Author's summary: "This article reviews the Seminar document class, which offers a user-friendly approach to the generation of transparencies with LATEX. It allows you to manage in an easy way your transparencies by letting you combine all LATEX commands with more specific commands for defining their layout. Moreover, in combination with the PSTricks macros, Seminar becomes an even more [capably] performing tool for creating your transparencies."

Compiler's note

This issue of the *Cahiers* is an exuberant *display* of colour in order to *discuss* colour. While one may easily find fault with the finer details of reproduction — colour bleeding, for example — the purpose of the entire issue is to demonstrate that colour and TEX can be made to work together. Refinements are always going to happen, but the ball has to start rolling somewhere. The articles are in French, but the information is not carried in the text; rather, it emerges via the verbatim examples, along with dozens and dozens of images, which convey the meaning and the necessary code needed to achieve those results. All the code elements are, of course, in English. And the source files on CTAN include documentation in English, so there's no lack of instruction available. Well, of course, translations always happen when a work is extremely popular and in demand, so who knows — we may yet see translations of PSTricks and Seminar documentation come on line at some point!

But these are such brief comments. Should these abstracts whet your appetite for more, there is indeed more to come — the 1994 TUG Annual Meeting in mid-summer, with its theme of "Innovation", has a whole day (Monday, the 1st of August) dedicated to "Colour and LATEX": papers by Leslie Lamport and Tom Rokicki; papers by the authors included in issue no. 16 of the *Cahiers*; papers by Michael Sofka and James Hafner, Friedhelm Sowa and Angus Duggan, Jon Stenerson and Johannes Braams. And that's only one day. The proceedings issue of *TUGboat* will definitely be one to wait for by your mailbox!

Numéro 17 — mai 1994

CHRISTIAN ROLLAND, Éditorial: Movement in the air ; pp. 1–2

Rolland talks about the new LaTeX2e release (the 'movement in the air'), and that readers will see more and more references to it in the pages of the *Cahiers* — the current issue contains articles ranging from theory to practice, innovation to initiation. The editorial concludes with notes on upcoming issues: one on ligatures, and another on TeX and Arabic.

EMMANUEL SAINT-JAMES, Esquisse d'une typographie applicative [Outline for a functional typography]; pp. 3–19

The purpose of this article is to develop some ideas for a functional approach to computerised text processing for the future. The author begins with a short history of types of computer programming languages, and brings that analysis to the world of text processing, demonstrating where underlying differences in approach and emphasis have caused confusion and overlap. He concludes with a proposal to re-evaluate the various steps currently used in text processing, to achieve a more functional theoretical framework.

[Not an easy piece to read; a background in computer programming, especially its theoretical history, would help the reader follow the arguments provided. Nevertheless, a thought-provoking piece for anyone who has an interest in the larger picture.]

DANIEL TAUPIN, L'avenue de TeX ... c'est Word[Perfect]! Une ébauche: RTFLaTeX [The Future of TeX ... is Word[Perfect]! A rough sketch of RTFLaTeX]; pp. 20–31

Whatever its qualities, TeX is used far less than Word (WordPerfect across the Atlantic). So we think that the future of TeX — as well as its varieties of LaTeX or AMS-TeX — will be via powerful conversion software, operating in both directions.

Unless one is willing to accept poor conversions, such software presents fundamental, sometimes insurmountable, difficulties; nevertheless, we will present a first reasonable attempt, namely RTFLaTeX. [translation of the **Résumé**]

A pragmatic analysis aimed at reconciling TeX with the wide-spread presence of Word[Perfect], particularly in a shared environment.

MICHEL GOOSSENS, Personnaliser les listes LaTeX [Customising LaTeX lists]; pp. 32–47

Author's summary: "In this article, which is a slightly reworked version of part of the third chapter of the book *The LaTeX Companion* (Addison-Wesley, January 1994) we take a look at LaTeX's list environments. First the various parameters and commands controlling the standard LaTeX lists, `enumerate`, `itemize`, and `description` are discussed. Then the general `list` environment is introduced and we tell you how to build custom layouts by varying the values of the parameters controlling this environment."

A welcome tutorial on that most useful and yet often most frustrating of LaTeX environments to customise.

JACQUES ANDRÉ, Multi-colonnage : comment en faire en LaTeX? [Multicolumns: how to do it in LaTeX]; pp. 49–62

Author's summary: "This paper is a tutorial on `multicol`, a LaTeX style to format multi-column pages."

Moving beyond the default two-column option in LaTeX, the tutorial gives samples of input and output, providing a handy and quick overview of some of the options available with this style, developed by Frank Mittelbach.

BERNARD GAULLE, Nos outils de bureau [Office tools]; pp. 63–66

Author's summary: "In this article the author relates his adventures in installing TeX, LaTeX and METAFONT for the operating system UNIX which was nearly new to him."

This is the second installment in a series; the first, in *Cahiers* 15 (43–45) and summarised in *TUGboat* 14 **#2** (146), described the process of finally selecting TeX to produce the in-house documentation. The third episode will be entitled "Good Impressions".

(Compiled by Christina Thiele)

Die TEXnische Komödie Contents of Recent Issues

1/1993, 5. Jahrgang (Mai 1993)

Luzia DIETSCHE, [Editorial]; p. 3

A short statement commenting on the delay of the current issue and articles scheduled for the next issue.

○ *Hinter der Bühne : Vereinsinternes* [Behind the scenes : Club matters]; pp. 4–26:

Joachim LAMMARSCH, Grußwort [Welcome message]; pp. 4–5

A short comment on club matters by the reelected president of DANTE.

[Luzia DIETSCHE], Protokoll der 8. Mitgliederversammlung von DANTE, Deutschsprachige Anwendervereinigung TEX e.V. [Protocol of the 8th assembly of members of DANTE]; pp. 5–23

This is the official report on the members' meeting held in Chemnitz (March '93). It starts with short accounts of the various hardware platforms and on german.sty (as presented by the appointed coordinators), followed by a report on the situation of DANTE (mainly organizational matters), the modalities of newly instituted funds, relations with TUG (Joachim Lammarsch complains that he had not been involved by the budget committee when it compiled the treasurer's report), and the status of the NTS project (now directed by Philip Taylor). The results of the election of board members are given.

Luzia DIETSCHE, Aktive Mitglieder der letzten Jahre [Members active during past years]; pp. 23–25

A list of those DANTE members who have been active in some way.

Lutz-Peter KURDELSKI, Stammtisch in Bremerhaven [Regular meetings in Bremerhaven]; p. 26

The creation of a local group in Bremerhaven is announced.

○ *TEX-Theatertage* [TEX theatre festival]; pp. 27–31:

Stephan ENGELKE, DANTE '93 in Chemnitz; pp. 27–31

This is a detailed report on the proceedings of the DANTE '93 conference in Chemnitz.

○ *Von fremden Bühnen* [On other stages]; pp. 32–34:

[LATEX3 PROJECT TEAM], News from LATEX3 (in English); pp. 32–34

A call for volunteers to help in the development of LATEX3.

○ *Bretter, die die Welt bedeuten* [The stage is the world]; pp. 35–49:

Ulrich WISSER, Amiga und TEX [Amiga and TEX]; pp. 35–41

A comparative survey of TEX implementations for the Amiga (including some benchmarks).

Peter KREBS, METAFONT-Reparaturen — für Laien [METAFONT repairs — for laymen]; pp. 41–42

In answer to a question (3/92) the author gives an example how METAFONT code can be modified.

Martin WALLMEIER, Probleme mit langen Überschriften in LATEX und ein Lösungsvorschlag [Problems with long headings in LATEX and a suggested solution]; pp. 43–46

This article deals with the problems posed by long section headings under LATEX.

Steffen STEINHÄUSER, Serienbriefe und vieles mehr ... [Mailmerge and much more]; pp. 46–49

The author describes his finder.sty for processing data bases like lists of addresses.

○ *Was Sie schon immmer über TEX wissen wollten ...* [What you always wanted to know about TEX ...]; pp. 50–51:

Luzia DIETSCHE, Fußnotenvariante [Footnote variants]; pp. 50–51

A hint how to change the appearance of LATEX footnotes.

○ *TEX-Beiprogramm* [TEX co-features]; pp. 52–57:

Lothar MEYER-LERBS, So kommen Sie an Ihr TEX auf dem Mac [How to get one's TEX for the Mac]; pp. 52–53

A list of TEX implementations for Macintosh.

Markus ERLMEIER, Mailbox für AmigaTEX-Software [A mailbox for AmigaTEX software]; pp. 53–56

This article describes a mailbox providing AmigaTEX.

Ekkehard HUNDT, Wettbewerb [Contest]; pp. 56–57

The purpose of the contest is to produce the logo of Addison-Wesley.

○ *Rezensionen* [Reviews]; pp. 58–59:

Ekkehard HUNDT, Der Vers 3:16 — Ein Informatiker treibt fröhliche Theologie [Verse 3:16 — A computer scientist does happy theology]; pp. 58–59

A review of *3:16*, the bible commentary by Donald E. Knuth.

○ *Leserbrief(e)* [Letter(s)]; pp. 60–63:

Heinz KUSZNIER and Luzia DIETSCHE, Artikel und die darin beschriebene Software [Articles, and the software described in them]; pp. 50–61

A letter addressing the problem that many readers of the 'Komödie' do not have easy access to the software mentioned, and an answer by Luzia Dietsche.

Steffen STEINHÄUSER, Anmerkungen zu `script.1` [Remarks on `script.1`]; pp. 62–63

The author shows how `script` style can be used with NFSS.

○ *Spielplan* [Repertory]; pp. 64–67:

The international and national calendar, and announcements of conferences (World Wide Window on TEX).

○ *Adressen* [Addresses]; pp. 68–71:

Various addresses related to DANTE and TUG, the addresses of all persons who have contributed to this issue, and the addresses of the coordinators in charge of the various hardware platforms.

2/1993, 5. Jahrgang (September 1993)

Luzia DIETSCHE, [Editorial]; p. 3

The editor explains that — due to circumstances — this issue does not contain (as promised) a large variety of articles but only three long ones.

○ *Hinter der Bühne : Vereinsinternes* [Behind the scenes : Club matters]; pp. 4–5:

Joachim LAMMARSCH, Grußwort [Welcome message]; pp. 4–5

A short report dealing with organizational matters.

Joachim LAMMARSCH, Fonds zur Unterstützung von Mitgliedern [Funds tp support members]; p. 5

A fund intended for the purpose of supporting members who cannot afford the reduced membership dues could be instituted.

○ *TEX-Theatertage* [TEX theatre festival]; pp. 6–18:

Ulrik VIETH, Bericht von der 14. Tagung der TUG [Report on the 14th TUG conference]; pp. 6–18

This is a detailed report on the TUG '93 conference in Birmingham dealing with both the proceedings and the social events.

○ *Von fremden Bühnen* [On other stages]; pp. 19–37:

Phil TAYLOR and Walter OBERMILLER, Die Zukunft von TEX [The future of TEX]; pp. 19–37

This article discusses in some detail those features of TEX which some consider as limitations or defects, various proposed extensions, and the strategies available to implement them. It closes with a short report on the status of the NTS project.

○ *Bretter, die die Welt bedeuten* [The stage is the world]; pp. 38–53:

Heinz WERNTGES, Grafik-Import in LATEX [Importing graphics under LATEX]; pp. 38–53

The article describes and compares some important tools for including graphics in TEX documents. It was first published in *c't* 12/92 (a computer journal).

○ *Was Sie schon immer über TEX wissen wollten ...* [What you always wanted to know about TEX ...]; pp. 54–55:

Luzia DIETSCHE, Kopfzeilen mal anders [Headline variations]; pp. 54–55

Some hints how to customize headlines under LATEX.

○ *Leserbrief(e)* [Letter(s)]; pp. 56–61:

Lutz BIRKHAHN, Gegendarstellung zum Artikel "Amiga und TEX", 1/1993 [Response to the article "Amiga and TEX", 1/1993]; pp. 56–58

The author disagrees with the article mentioned; in particular he criticizes the benchmark results given there.

Franz STROBL, Fraktur und eqnarray
[Fraktur fonts and eqnarray]; pp. 58–59

Albrecht MEHRL, Anfängerfrage?
[Beginners' questions?]; pp. 59–61

Luzia DIETSCHE, Anmerkung der Redaktion
[Remark of the editor]; p. 61

Two letters asking several questions, and a call for articles answering these and similar questions.

○ *Spielplan* [Repertory]; pp. 62–63:

The international and national calendar.

○ *Adressen* [Addresses]; pp. 64–67:

Various addresses related to DANTE and TUG, the addresses of all persons who have contributed to this issue, and the addresses of the coordinators in charge of the various hardware platforms.

3/1993, 5. Jahrgang (Dezember 1993)

Luzia DIETSCHE, [Editorial]; p. 3

In her opening statement the managing editor of the journal apologizes for the delayed publication, and invites readers to contribute articles.

○ *Hinter der Bühne : Vereinsinternes*
[Behind the scenes : Club matters]; pp. 4–27:

Joachim LAMMARSCH, Grußwort
[Welcome message]; pp. 4–6

In his column the president of DANTE addresses various organizational issues, such as problems with the change of Germany's postal codes, moving the office, a new firm called *DANTE Ltd.* (no connection to DANTE), problems with organizing a reduced rate for 'active' members, and a complaint in connection with the elections held in Chemnitz. He further announces *Offizin*, the first book published by DANTE, and Stephan von Bechtolsheim's *TEX in Practice*. He further reports progress of the NTS project, and deplores that some persons (from outside of DANTE) would hinder the project, often for personal reasons.

[Luzia DIETSCHE and Joachim LAMMARSCH], Protokoll der 9. Mitgliederversammlung von DANTE, Deutschsprachige Anwendervereinigung TEX e.V. [Protocol of the 9th assembly of members of DANTE]; pp. 7–25

This is the official report on the members' meeting held in Kaiserslautern. It starts with short accounts on the various hardware platforms, software services, and german.sty (presented by the appointed coordinators), followed by a report on the situation of DANTE (2314 members by August 1993, organization, services, problems with the payment of TUG membership dues by DANTE on behalf of its members). Joachim Lammarsch gives reports on the meetings of various committees during the TUG conference in Birmingham, in particular, on the *Board of Directors* and the *Technical Council*. Joachim Schrod (representing Philip Taylor) reports the results of a meeting of the 'inner circle' of the NTS group, and announces plans to develop an eTEX (an extended TEX), and long term plans to develop NTS (a *New Typesetting System*, or *Next Generation Typesetting System*). The next DANTE conference will take place in Münster (February 94). DANTE e.V. will try to prevent the founding of a group under the name of *DANTE Ltd.*

Friedhelm SOWA, Kassenbericht im Zeitraum 1.1.92–31.12.92 [The treasurer's report for 1992]; pp. 26–27

○ *Bretter, die die Welt bedeuten*
[The stage is the world]; pp. 28–45:

Arne W. STEUER, Cassettenhüllen mit LATEX
[Labels for compact cassettes using LATEX]; pp. 28–30

The author describes his cassette.sty for printing labels for compact cassettes.

Jan SUCHANEK, Mathematikaufgaben mit Pascal und LATEX [Mathematics exercises with Pascal und LATEX]; pp. 29–32

The author uses Pascal to produce output in LATEX syntax.

Thomas K. GÖTZ, ASC2TEX: Ein Konverter für Blockgrafik [ASC2TEX: A converter for block graphics]; pp. 32–34

The author describes a preprocessor which converts screen prints (extended ASCII) to LATEX code (using the picture environment).

Ralph SCHLEICHER, Kompatible TEX-Makros für erweiterte Zeichensätze [Compatible TEX macros for extended character sets]; pp. 34–39

The author describes macros to handle 8-bit input in connection with both cm- and dc-fonts.

Dirk NITSCHKE, Markieren von Formeln
[The labelling of formulas]; pp. 40–42

The author describes how to produce 'non-standard' labels with LATEX.

Andreas SCHERER, Das Promillezeichen als TEX-Makro [The per mill sign as a TEX macro]; pp. 42–45

Macros and METAFONT code to produce the per mill sign.

○ *Was Sie schon immer über TEX wissen wollten ...* [What you always wanted to know about TEX ...]; pp. 46–47:

Harald GEWEHR, Vorsicht Datenverlust! [Attention: loss of data!]; pp. 46–47

The author warns that some TEX implementations for DOS may overwrite file `foo.tex` when TEX tries to write to `foo.tex.aux` (or similar).

○ *TEX-Beiprogramm* [TEX co-features]; pp. 48–56:

Norbert HESSE, Das TEX RunTime System für SCO Unix [The TEX RunTime System for SCO Unix]; pp. 48–52

The author has built a RunTime System avoiding the need for the (expensive) Development System.

Markus ERLMEIER, Amiga-News [News on the Amiga]; pp. 53–54

News on TEX implementations for the Amiga.

Lothar MEYER-LERBS, Viel Neues auf dem Mac [Much new on the Mac]; pp. 55–56

A short survey of TEX implementations for Macintosh computers.

○ *Leserbrief(e)* [Letter(s)]; pp. 57–58:

Jens GETREU, Anregungen zu NTS [Suggestions for NTS]; pp. 57–58

The author stresses the importance of compatibility and expresses the wish that eTEX will provide standards for the exchange of bitmap and vector graphics as well as for the use of the dc-fonts. A used phototypesetter is wanted.

○ *Spielplan* [Repertory]; pp. 59–62:

The international and national calendar, and the announcement of a conference (TEX meeting DANTE '94 in Münster).

○ *Adressen* [Addresses]; pp. 63–66:

Various addresses related to DANTE and TUG, the addresses of all persons who have contributed to this issue, and the addresses of the technical advisors in charge of the various hardware platforms.

4/1993, 5. Jahrgang (April 1994)

Luzia DIETSCHE, [Editorial]; p. 3

In her opening statement the managing editor of the journal reports a curious observation, namely, that more and more frequently authors tend to insert spaces in front of punctuation marks — which is contrary to German usage.

○ *Hinter der Bühne : Vereinsinternes* [Behind the scenes : Club matters]; pp. 4–7:

Joachim LAMMARSCH, Grußwort [Welcome message]; pp. 4–5

The president of DANTE addresses various organizational questions — DANTE now has installed its own software server `ftp.dante.de` — and announces the release of an ϵ-TEX by the NTS-project.

Luzia DIETSCHE, Verlorengegangene TUG-Mitglieder [Lost TUG members]; pp. 6–7

This note addresses organizational problems which occurred in the process of paying TUG membership dues via DANTE.

○ *Von fremden Bühnen* [On other stages]; pp. 8–10:

LATEX2$_\varepsilon$ – A New Version of LATEX (in English) ; pp. 8–10

This is the announcement of LATEX2$_\varepsilon$ from December 21, 1993.

○ *Bretter, die die Welt bedeuten* [The stage is the world]; pp. 11–35:

Frank HASSEL, Schachfigurensatz mit TEX und LATEX [Typesetting chess pieces with TEX and LATEX]; pp. 11–25

The author describes the chess fonts and `CHESS.STY` by Piet Tutelaers, and his own `BdFCHESS.STY` which adds tools for playing chess by mail. Several examples show input and output.

Andreas SCHERER, Graphiken mit `GNUPLOT` und METAFONT [Graphics with `GNUPLOT` and METAFONT]; pp. 26–30

The author briefly describes how GnuPlot can be used to produce METAFONT code for line graphics which then can be included in a TEX file as a font.

Oliver BRUSBERG, *MakeIdx* — Ein (anderer) Indexprozessor für LATEX [*MakeIdx* — An(other) index processor for LATEX]; pp. 30–33

The author describes his index processor which is essentially an extension of *MakeIdx* by Pehong Chen. The new features include explicit sort sequences, automatic replacement of strings, and correct sorting of numbers.

Andreas LAUSCHKE, Diese leidigen Mengenzeichen [These distressing set signs]; pp. 33–35

The author describes macros (using the LaTeX picture environment) which can be used to replace blackboard bold R, C, N, Z, and 1.

○ *Was Sie schon immer über TeX wissen wollten ...* [What you always wanted to know about TeX ...]; pp. 36–37:

Luzia DIETSCHE, Hokuspokus! [Hocus pocus!]; pp. 36–37

This note discusses a vanishing `\special` command in a LaTeX file, a bug which will be corrected in LaTeX2ε.

○ *TeX-Beiprogramm* [TeX co-features]; pp. 38–43:

Michael KAMLAGE, TeXSTAR — Eine Oberfläche für TeX und LaTeX [TeXSTAR — A shell for TeX and LaTeX]; pp. 38–39

The author presents his TeX shell (based on the Turbo Vision library of Turbo Pascal 7.0).

Michael SCHANK, TeX und OS/2 2.x [TeX and OS/2 2.x]; pp. 39–40

A brief discussion of TeX (in particular, emTeX) under OS/2.

Markus ERLMEIER, Rund um den Amiga [Around the Amiga]; pp. 40–43

News on TeX and TeX tools for the Amiga.

○ *Rezensionen* [Reviews]; pp. 43–48:

Stefan BREUER, WORD, TeX und Alcuin [WORD, TeX and Alcuin]; pp. 43–45

This article discusses the production of a scientific proceedings volume (*Science in Western and Eastern Civilization in Carolingian Times*, edited by P.L. Butzer and D. Lohrmann. Basel 1993). This volume was produced from camera ready copy. Work was first done using WORD on a Macintosh. Since some articles could not be realized using WORD, part of the book was done in LaTeX, which was adapted to produce a style corresponding to that of WORD.

Reinhard ZIERKE, The LaTeX Companion ['The LaTeX Companion']; pp. 46–48

A short summary of the contents of this new book.

○ *Leserbrief(e)* [Letter(s)]; p. 49:

Frank HREBABETZKY, Farbiger Druck [Colour printing]; p. 49

A wish for a `\special` command which passes its argument to the printer. Such a command could be used (e.g.) to start colour printing.

○ *Spielplan* [Repertory]; pp. 50–55:

The international and national calendar, and announcements of conferences (EuroTeX'94; TUG 15th Annual Meeting)

○ *Adressen* [Addresses]; pp. 56–59:

Various addresses related to DANTE and TUG, the addresses of all persons who have contributed to this issue, and the addresses of the technical advisors in charge of the various hardware platforms.

(compiled by Peter Schmitt)

Calendar

1994

Jul 7 DANTE TeX-Stammtisch at the Universität Bremen, Germany. For information, contact Martin Schröder (`115d@zfn.uni-bremen.de`; telephone 0421/628813).
First Thursday, 18:30, Universität Bremen MZH, 4th floor, across from the elevator.

Jul 6-8 C.N.E.D. 94: 3ième Colloque National sur l'Ecrit et le Document, Rouen, France. For information, contact Jacques Labiche (`labiche@la3i.univ-rouen.fr`).

Jul 11 UK TeX Users' Group, Cambridge University. LaTeX fonts and graphics: a hands-on tutorial. For information, e-mail `uktug-enquiries@ftp.tex.ac.uk`

Jul 14 DANTE TeX-Stammtisch, Wuppertal, Germany. For information, contact Andreas Schrell (`Andreas.Schrell@FernUni-Hagen.DE`, telephone (0202) 666889). Second Thursday, 19:30, Gaststätte Yol, Ernststraße 43.

Jul 18 DANTE TeX-Stammtisch in Bonn, Germany. For information, contact Herbert Framke (`Herbert_Framke@SU2.MAUS.DE`; telephone 02241 400018, Mailbox 02241 390082). Third Monday, Anno, Kölnstraße 47.

Jul 19 DANTE TeX-Stammtisch in Duisburg, Germany. For information, contact Friedhelm Sowa (`tex@ze8.rz.uni-duesseldorf.de`; telephone 0211/311 3913).
Third Tuesday, 19:30, at Gatz an der Kö, Königstraße 67.

Jul 24-29 SIGGRAPH'94: 21st International ACM Conference on Computer Graphics and Interactive Techniques. Orlando, Florida. (For information, contact `siggraph-94@siggraph.org`, telephone 312-321-6830.)

Jul 27 DANTE TeX-Stammtisch, Hamburg, Germany. For information, contact Reinhard Zierke (`zierke@informatik.uni-hamburg.de`; telephone (040) 54715-295).
Last Wednesday, 18:00, at TEX's Bar-B-Q, Grindelallee 31.

Jul 31 - Aug 4 **TUG 15th Annual Meeting**: Innovation, Santa Barbara, California. For information, contact Debbie Ceder (`tug94@tug.org`). (For the preliminary program, see *TUGboat* 15, no. 1, p. 68.)

Aug 4 DANTE TeX-Stammtisch at the Universität Bremen, Germany. (For contact information, see Jul 7.)

Aug 11 DANTE TeX-Stammtisch, Wuppertal, Germany. (For contact information, see Jul 14.)

Aug 15 DANTE TeX-Stammtisch in Bonn, Germany. (For contact information, see Jul 18.)

Aug 16 DANTE TeX-Stammtisch in Duisburg, Germany. (For contact information, see Jul 19.)

Aug 17 ***TUGboat* Volume 15, 3rd regular issue:**
Deadline for receipt of *technical* manuscripts (tentative).

Aug 31 DANTE TeX-Stammtisch, Hamburg, Germany. (For contact information, see Jul 27.)

Sep 1 DANTE TeX-Stammtisch at the Universität Bremen, Germany. (For contact information, see Jul 7.)

Sep 8 DANTE TeX-Stammtisch, Wuppertal, Germany. (For contact information, see Jul 14.)

Sep 7-10 CyrTUG94, Dubna, Russia. For information, contact Irina Makhovaya (`irina@mir.msk.su`).

Sep 14 ***TUGboat* Volume 15, 3rd regular issue:**
Deadline for receipt of news items, reports (tentative).

Status as of 28 February 1994

Sep 19 DANTE TeX–Stammtisch in Bonn, Germany. (For contact information, see Jul 18.)

Sep 20 DANTE TeX–Stammtisch in Duisburg, Germany. (For contact information, see Jul 19.)

Sep 26 – 30 EuroTeX '94, Sobieszewo, Poland. For information, contact Włodek Bzyl (`EuroTeX@Halina.Univ.Gda.Pl`). (See announcement, *TUGboat* 15, no. 1 p. 69.)

Sep 28 DANTE TeX–Stammtisch, Hamburg, Germany. (For contact information, see Jul 27.)

Oct 6 DANTE TeX–Stammtisch at the Universität Bremen, Germany. (For contact information, see Jul 7.)

Oct 13 DANTE TeX–Stammtisch, Wuppertal, Germany. (For contact information, see Jul 14.)

Oct 17 DANTE TeX–Stammtisch in Bonn, Germany. (For contact information, see Jul 18.)

Oct 18 DANTE TeX–Stammtisch in Duisburg, Germany. (For contact information, see Jul 19.)

Oct 19 UK TeX Users' Group,
University of Warwick.
Annual General Meeting.
For information, e-mail
`uktug-enquiries@ftp.tex.ac.uk`

Oct 26 DANTE TeX–Stammtisch, Hamburg, Germany. (For contact information, see Jul 27.)

Nov–Dec UK TeX Users' Group, location to be announced. Topic: TeX, SGML and electronic publishing.
For information, e-mail
`uktug-enquiries@ftp.tex.ac.uk`

Nov 3 DANTE TeX–Stammtisch at the Universität Bremen, Germany. (For contact information, see Jul 7.)

Nov 10 DANTE TeX–Stammtisch, Wuppertal, Germany. (For contact information, see Jul 14.)

Nov 15 DANTE TeX–Stammtisch in Duisburg, Germany. (For contact information, see Jul 19.)

Nov 21 DANTE TeX–Stammtisch in Bonn, Germany. (For contact information, see Jul 18.)

Nov 23 ***TUGboat* Volume 15,**
3rd regular issue:
Mailing date (tentative).

Nov 30 DANTE TeX–Stammtisch, Hamburg, Germany. (For contact information, see Jul 27.)

Dec 1 DANTE TeX–Stammtisch at the Universität Bremen, Germany. (For contact information, see Jul 7.)

Dec 8 DANTE TeX–Stammtisch, Wuppertal, Germany. (For contact information, see Jul 14.)

Dec 19 DANTE TeX–Stammtisch in Bonn, Germany. (For contact information, see Jul 18.)

Dec 20 DANTE TeX–Stammtisch in Duisburg, Germany. (For contact information, see Jul 19.)

Dec 28 DANTE TeX–Stammtisch, Hamburg, Germany. (For contact information, see Jul 27.)

1995

Jan 5 – 8 Linguistic Society of America,
69th Annual Meeting,
Fairmont Hotel, New Orleans.
For information, contact the
LSA office, Washington,
DC (202-834-1714,
`zzlsa@gallua.gallaudet.edu`).

Apr UK TeX Users' Group, location to be announced. Topic: Maths is what TeX does best of all.
For information, e-mail
`uktug-enquiries@ftp.tex.ac.uk`

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

TUG Business

Meet the Board, Part II

In the last issue, the biographies and personal statements of seven members of the Board appeared; these were originally intended for presentation with the ballot, but the shortfall of candidates rendered that inoperative. Below is the corresponding information for the rest of the Board.

The statements appear in alphabetical order by name of the Board member, with the ending year of that person's term of office in parentheses after the name. Terms of office end with the annual meeting of the year shown.

Barbara Beeton
For the Elections Committee

Michael J. Ferguson (1996)

INRS – Télécommunications
Université du Québec
16 Place du Commerce
Verdun H3E 1H6, Québec Canada
Phone: 514-765-7834
Internet: `mike@inrs-telecom.uquebec.ca`

Biography:

For over ten years now, I have been using, adapting, and promoting TEX for use in multilingual environments. I introduced TEX-78 at INRS-Télécommunications, a graduate research institute of the University of Québec, in 1980, with an early version of the INRSTEX document preparation package. The greatly enhanced features of TEX-82 allowed me to enhance INRSTEX to have full document preparation facilities, eventually including graphics. However there was much student dismay at the fact that this new wonderful system kept leaving the ends of words, punctuated by little black rectangles, in the margin of their theses. It became apparent that something had to be done to make TEX viable in our bilingual environment. With this incentive, I produced Multilingual TEX, a modification to TEX, the program, that allowed beautiful, and semantically consistent, documents in any language. Virtually all of the features of MLTEX, except for the ability to work multilingually with the cm fonts, have been incorporated in TEX-3. Most recently, I coordinated the activity that led to the new TEX Extended font standard.

Personal statement:

TEX is one of the few systems that has the capability of document preparation and production in any alphabet-based language. I feel strongly that its premier position should be protected and enhanced in this area. TEX, although powerful, is not without its deficiencies. It is vitally important that we search and find a path that allows us, and TEX to exploit the new and powerful computing facilities that are becoming available, while maintaining enough stability for confident use. I hope to work towards such a goal.

Peter Flynn (1996)

Computer Center
University College
Cork, Ireland
Phone: +353 21 276871 x2609
Internet: `cbts8001@iruccvax.ucc.ie`

Biography:

Peter Flynn was born in 1953 and educated in England. He built his first computer (analog!) at high school in 1969, and also resurrected and ran the school press for four years. He spent a year studying languages and working as a printer in Europe before moving to the London College of Printing, where he took his first degree in 1976. In 1978 he took his MA in management and computerised planning systems at the Central London Polytechnic. He worked in technology training and research for the Printing and Publishing Industry Training Board until 1982, when he joined the London office of United Telecom (Kansas) as a technical consultant, advising on database, communications and text systems in the UK and USA. In 1984 he moved to his present position as Manager of the Academic Computing Service at University College Cork, Ireland, providing student, graduate and faculty network and computing facilities on VM, VMS, MS-DOS, Mac and UNIX platforms.

Peter started using TEX in the early 1980s. He has written several public domain add-ons, including PC-WriTEX and SGML2TEX, and ran the 5th European TEX conference in Cork in 1990, where he also helped start the Irish TEX and LATEX Interest Community (ITALIC). He is involved as a consultant in several EU and academic research projects (AIM, COSINE/RARE, FRAMEWORK, CURIA) and has published articles on networking, TEX, statistics and programming. He is deputy director of EARN Ireland, a member of the Advisors Group of the Irish academic network, and secretary of the local branch of the Irish Computer Society. Among his other interests are typography, surfing, reading, cyberspace and early music.

Personal statement:

I would like to help TUG build on the changes which have taken place in the last few years by developing its resources in training, publicity and advice/information about TEX and its family of products. I want to increase the membership, and the activity rate of members, by making the use of TEXware more attractive, and by helping to develop the additions to the system to keep pace with changes in DTP elsewhere. I would like to continue to improve communications with users (members and non-members alike) especially by electronic means, and to build a community where TEX's advantages can be seen more clearly.

George D. Greenwade (1997)

Associate Professor of Economics and
Business Analysis
College of Business Administration
Sam Houston State University
Huntsville, Texas 77341-2118 USA
Phone: (409) 294-1265 (department);
(409) 204-1266 (office)
FAX: (409) 294-3612
Internet: `bed_gdg@SHSU.edu`

Personal statement:

My interest in serving of the TUG Board of Directors lies in my representing the "naive" end user of TEX. My personal usage focuses on LATEX and is primarily aimed at documents associated with my teaching and research. While I have authored a limited number of style options to achieve certain layout designs, the bottom line is that I believe that it is incumbent upon TUG to recognize the end-level users and to provide as much assistance as possible to them in getting, installing, learning, and using TEX. This in no way implies that the naive end user has failed to have representation on the TUG Board previously; instead, my explicit goal, if elected, is to serve as a representative of this group, focusing on effectively bridging the gap between high-level TEX users and developers and general end users with less expertise.

My involvement with TUG includes serving as Chair of the TUG Technical Working Group on TEX Archive Guidelines (WG 92-04) which has led to the successful Comprehensive TEX Archive Network (CTAN), where I also serve as Coordinator of the CTAN host `ftp.SHSU.edu`. In addition to my official involvement with TUG, I am also manager of three network-based mail lists dealing with TEX-related topics — `INFO-TeX`, `TeX-Pubs`, and `ctt-Digest`.

Yannis Haralambous (1997)

187, rue Nationale
59000 Lille
France
Fax: (33) 20.40.28.64
Internet: `yannis@gat.univ-lille1.fr`

Biography:

Born in Athens (Greece) in 1962, I came to Lille (France) in 1979. I received all my higher education there, including the Ph.D. in Pure Mathematics which I obtained in 1990. Since 1987 I've been teaching mathematics at the University of Lille. I speak Greek, French, German, English fluently and can get by in Spanish and Russian.

I became involved with TEX and METAFONT in 1987, when I met Klaus Thull in Berlin. Since volume 10(3) of the *TUGboat*, my submissions have appeared in almost every issue, with articles on Greek, Latin alphabet diacritical marks, Old German alphabets, Ottoman-Arabic-Persian, ScholarTEX, etc.

My TEX activity includes giving lectures and/or tutorials in the TUG meetings in Cork and Dedham, the DANTE meetings in Göttingen, Vienna and Dresden, and the European TEX Conference in Paris.

My working fields are primarily the creation of fonts and macro packages for typesetting in various languages, starting from Latin alphabet languages (West or ex-East) and going all the way to Greek, Middle East and Slavic languages — extant or historical, for scholarly use.

I believe that POSTSCRIPT should become a standard part of TEX, and that METAFONT could be used to obtain "meta"-families of POSTSCRIPT fonts — whether pre-existing or new; I'm planning to work on these issues.

Finally I have to say that what I like the most about TEX are the people which I am privileged to have met through it and because of it.

Personal statement:

TEX is one of the big ideas of the century. It means "intrusion of Art into Science and vice-versa". The software is free and accessible to everyone. In few words: TEX is an important cultural vehicle. TUG has to make TEX widespread while keeping its ideology and standards; to encourage and coordinate new developments; and, finally, to support newborn and pre-existing national User Groups — *no TEX user should ever be isolated.*

My personal modest contribution to TUG would be primarily on the level of European TEX-related activities: my geographical position allows easier travel and information exchange within the Old Continent.

Nico Poppelier (1996)

Elsevier Science Publishers BV
Academic Publishing Division
Information Technology Development
Molenwerf 1
1014 AG Amsterdam, The Netherlands
Internet: `n.poppelier@elsevier.nl`

Biography:

While doing research for my Ph.D. in theoretical nuclear physics, between 1984 and 1989, I became familiar with TEX and LATEX. During that time I also collaborated with Elsevier in one of their R&D projects; my job was to create document styles for two physics journals. In early 1990, after I had received my Ph.D. degree, I joined Elsevier in a permanent capacity as R&D Assistant in the R&D Department of the Academic Publishing Division.

One part of my work there is the implementation of procedures for the handling of author-prepared TEX files (compuscripts) for scientific journals and book series in the fields of physics, materials science, mathematics, computer science, engineering and technology.

Another important part of my work is research into the applications of SGML in scientific publishing.

Being a member of the TEX Users Group has helped me form the contacts needed for my work. I have attended the TUG conferences for two years and have found that these conferences are a unique opportunity to meet people, to exchange ideas and to discuss various developments. As a member of TUG's Board of Directors I bring knowledge of TEX and LATEX, ideas about the future of TEX and LATEX, ideas about the relation between TEX and SGML, and knowledge of four foreign languages, English, French, German and Japanese, beside my native language Dutch. Knowledge of more than one language could be useful, since TUG has become a truly international users group.

Personal statement:

I think that the best platform for the exchange of ideas, questions, answers and discoveries concerning the TEX system should be TUG. In my opinion

(1) TUG should inform users of TEX and related software all over the world, in particular the members of TUG, of changes and directions, and allow the opinion of the users to be heard.
(2) TUG should take the role of standardizing body, and see to it that TEX and related software remain standard, despite the growing number of applications based on or derived from the TEX system.
(3) TUG should sponsor and supervise the on-going development of the TEX system.
(4) TUG should sponsor and supervise research into computer programs for high-quality typesetting and font design.
(5) TUG should encourage new memberships and the active relation between TEX and participation of new members.

Jon Radel (1997)

P. O. Box 2276
Reston, VA 22090-0276 U.S.A.
Internet:`jon@radel.com`

Biography:

I was born in Minnesota in 1961, where I was also introduced to computers. While my love affair with computers is not matched in intensity by my interest in traditional printing, I have long been interested in the technology of the printed word. This, not surprisingly, led to an interest in using computers for printing, which first saw expression when I was an undergraduate and had access to an IBM mainframe running SCRIPT. The limitations of SCRIPT soon became apparent.

By the time Princeton University granted me my degree in Computer Science in 1985, I had heard of a new formatter that the Mathematics Department had — on a machine I couldn't touch. Shortly thereafter I noticed ads in *Byte* and realized that, if I purchased a copy of pcTEX, I could do what the Mathematics Department had been doing. I was fascinated by TEX because it used the terminology of traditional printing, and I quickly began to collect material about it.

By 1986 I had joined TUG, and I have attended all the North American meetings since 1988. *TUGboat*, and discussions at meetings, made me realize that there was a wealth of TEX material available only via the academic computer networks. I still had access to these networks, but the accounts I had were insecure enough that I could understand the problems of using TEX on isolated machines. For these isolated users I started my service of distributing TEX material on floppies in 1988. Starting slowly, I eventually developed enough of a reputation that I was sending disks all over the world, and the TUG staff started referring people to me. Unfortunately, a lack of time necessitated my stopping the service, and I never found the time to properly transfer the service to the TUG office.

I currently serve as the Internet Systems Administrator for the U.S. National Archives and maintain a keen interest in electronic publishing, both where the final product is paper and where it is images on a screen.

Personal statement:

Given my professional involvement with microcomputers, and my former floppy distribution, I view microcomputer users, particularly those not in academia, as my constituency. I would like to ensure continued commitment for TUG serving you as a source of information and support. I also think it important that TUG take a greater role in influencing and supporting the future of TeX and related software, particularly now that Donald Knuth is no longer developing TeX.

Sebastian Rahtz (1997)

ArchaeoInformatica
12 Cygnet Street
York Y02 1AG
UK
Phone: (44) 904 633904
Internet: `spqr@ftp.tex.ac.uk`

Biography:

I have been a TeX enthusiast from minutes after seeing the output from a copy of MicroTeX in 1986. I am a classicist and archaeologist by training, but have worked in humanities computing since 1984. After five years as a computer science lecturer at the University of Southampton, I left to become an independent consultant — supposedly to specialize in archaeological computing, but never failing to be sidetracked by TeXery. I now live in York, but work regularly on text-processing support for CERN in Geneva. I have a beautiful partner and a beautiful baby daughter.

I have been a PostScript fan for almost as long as a TeX fan, and have spent much time and effort on helping with the integration of PS fonts in LaTeX, graphics inclusion, and colour. As well as being a founding, and current, committee member of the UK TeX Users Group (and editor of its newsletter *Baskerville*), I have been closely associated with the UK TeX Archive since its inception, and am one of the three principal architects of the Comprehensive TeX Archive Network. Archives have caused me more work and grief than any reader can imagine...

My TUG mission is to promote archives and the cause of LaTeX2ε.

Production Notes

Barbara Beeton

Input and input processing

Electronic input for articles in this issue was received by e-mail, and was also retrieved from remote sites by anonymous ftp.

In addition to text and various coded files processable directly by TeX, the input to this issue includes a `.tfm` file created from the `.afm` file for a PostScript font. For this PostScript font it was necessary also to create a number of `.pk` files in various resolutions, using `ps2pk` after converting the `.pfa` form to `.pfb` with `t1binary`. More than 45 files were required to generate the final copy; over 70 more contain earlier versions of articles, auxiliary information, and records of correspondence with authors and referees. These numbers represent input files only; `.dvi` files, device-specific translations, and fonts (`.tfm` files and rasters) are excluded from the total.

Most articles as received were fully tagged for *TUGboat*, using either the `plain`-based or LaTeX conventions described in the Authors' Guide (see *TUGboat* 10, no. 3, pages 378–385). The macros are available from CTAN (the Comprehensive TeX Archive Network); see *TUGboat* 14, no. 2, p. 100. The TUG office will provide copies of the macros on diskette to authors who have no electronic access.

By number, 62% of the articles, and 70% of the pages in this issue are in LaTeX. Owing to technical problems in running multiple articles together under LaTeX2ε, production was accomplished using version 2.09, except for the article by Downes, which required features of LaTeX2ε that were not available in earlier versions. Philip Taylor's report on the meeting of the NTS group, tagged for the `plain`-based `tugboat.sty`, had to be processed by itself because it redefined some essential elements of the *TUGboat* style that affected anything run after it, even escaping a `\begingroup`... `\endgroup` quarantine.

Font work was required for the article by Ramasubramanian, Nickalls and Reed, "ASCII.sty". Unlike most font work done for *TUGboat*, this was not METAFONT; instead, following the instructions in the article, the program `ps2pk` was used to create bitmap fonts at typesetter resolution. My compliments to the authors for making the instructions so clear and easy to follow.

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

In addition to the report by Taylor, the following material was prepared using the `plain`-based `tugboat.sty`:

- the Abstracts from *Baskerville*, *Cahiers GUTenberg*, and *Die TEXnische Komödie*, starting on page 133.
- "Meet the Board", page 145.
- the TUG calendar, page 143.
- these Production notes.
- "Coming next issue".

Two articles dealing with fonts, one METAFONT and one PostScript, were unable to be processed to give the same results shown in PostScript files provided by the authors for verification. These articles were pulled from the issue and the problems are being investigated with the help of the authors and other technical adepts; they are expected to appear in the next regular issue. Unfortunately, the time spent in attempting to overcome the problems (more than 40 hours), has contributed significantly to the delay of this issue.

Output

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

Coming Next Issue

New techniques in METAFONT

Certain geometrical problems that arise very often in glyph design are not directly solvable by METAFONT's `plain` macros. Yannis Haralambous presents two such problems and solutions for them, along with a discussion of an approach that, although geometrically correct, does *not* work in real-world METAFONT practice and should be avoided. [Delayed by technical difficulties]

More new books

Reviews of the following are expected:

- Eitan Gurari, *Writing with TEX*
- Eitan Gurari, *TEX and LATEX: Drawing and Literate Programming*
- and possibly others ...

Macro Packages for Typesetting Commutative Diagrams

Gabriel Valiente Feruglio has examined the various packages for setting commutative diagrams, and prepared a comparative report on what each has to offer.

NOTICE

Change in *TUGboat* policy

Barbara Beeton

Owing to production difficulties with articles not prepared using the *TUGboat* styles (`plain` or LaTeX), and which do not conform to *TUGboat* guidelines in other respects, it has become necessary to make the policy for acceptance of articles more stringent.

Articles for which authors include their own formatting macros, and for which those macros are not relevant to the intellectual content of the article, may be returned to the author for modification if substitution of the *TUGboat* styles is more than a trivial exercise, or if macros provided by the author disrupt articles by other authors when multiple articles are run together.

If an article requires any style option or macro package in addition to the *TUGboat* styles, the author must include a full accounting of what is needed in the initial transmittal to the *TUGboat* editor. This includes macros, fonts, PostScript files, etc., — anything that a run of (LA)TeX will try to input, or that will be required to produce output on a 1200dpi typesetter.

If a requested style file or package is one that in available on CTAN, the version used must be the one on CTAN, unless the variance is the subject of the article, in which case special arrangements will be made. All other files required for processing the article must be delivered to the editor in usable form; this can be as a uuencoded file, a node and directory reference from which the editor can retrieve the files by ftp, on diskette, ...; if in doubt, contact the editor to make suitable arrangements.

I regret that it is necessary to make the procedures for submitting *TUGboat* articles more stringent. However, this has become necessary in light of the problems associated with processing some of the articles initially accepted for this issue in a reasonable length of time.

For the information of authors, here is a description of the system used for *TUGboat* production. The main processor is a UNIX box with TeX 3.141 implemented via `WEB2C`; local modifications consist mainly of increased memory block sizes, with 256k words of main memory, 72k bytes of string space, hash size of 9500 (multiletter control sequence names), and space for 255 fonts. Tom Rokicki's `dvips` is the output device driver. Proof is printed on a 300 dpi laser printer (several are available, and usage depends on what is least heavily loaded, or physically most convenient, when needed), and camera copy is processed through a 1200 dpi PostScript typesetter, either a Compugraphic 9600, or an ECRM ScriptSetter IV with a software RIP from Harlequin.

As always, constructive comments and suggestions are welcomed.

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

Brown University,
Providence, Rhode Island

California Institute of Technology,
Pasadena, California

Carleton University,
Ottawa, Ontario, Canada

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

CERN, *Geneva, Switzerland*

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

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

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

Cornell University,
Mathematics Department,
Ithaca, New York

CSTUG, *Praha, Czech Republic*

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

Escuela Superior de
Ingenieres Industriales,
Sevilla, Spain

European Southern Observatory,
Garching bei München, Germany

Fermi National Accelerator
Laboratory, *Batavia, Illinois*

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

GKSS, Forschungszentrum
Geesthacht GmbH,
Geesthacht, Germany

Grinnell College,
Computer Services,
Grinnell, Iowa

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

MacroSoft, *Warsaw, Poland*

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

Rogaland University,
Stavanger, Norway

Rutgers University,
Computing Services,
Piscataway, New Jersey

Space Telescope Science Institute,
Baltimore, Maryland

Springer-Verlag,
Heidelberg, Germany

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

Stanford Linear Accelerator
Center (SLAC),
Stanford, California

Stanford University,
Computer Science Department,
Stanford, California

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

United States Naval
Postgraduate School,
Monterey, California

Universität Augsburg,
Augsburg, Germany

University of California, Berkeley,
Space Astrophysics Group,
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

University of Heidelberg,
Computing Center,
Heidelberg, Germany

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

Universität Koblenz-Landau,
Koblenz, Germany

University of Manitoba,
Winnipeg, Manitoba

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

University of Salford,
Salford, England

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

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

University of Stockholm,
Department of Mathematics,
Stockholm, Sweden

University of Texas at Austin,
Austin, Texas

Università degli Studi di Trento,
Trento, Italy

Uppsala University,
Uppsala, Sweden

Villanova University,
Villanova, Pennsylvania

Vrije Universiteit,
Amsterdam, The Netherlands

Wolters Kluwer,
Dordrecht, The Netherlands

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

TeX Consulting & Production Services

North America

Abrahams, Paul
214 River Road, Deerfield, MA
01342; (413) 774-5500
Composition and typesetting of high-quality books and technical documents. Complete production services using any PostScript fonts. Assistance with book design and copy. I am a computer consultant with a computer science education.

American Mathematical Society
P. O. Box 6248, Providence, RI
02940; (401) 455-4060
Typesetting from DVI files on an Autologic APS Micro-5 or an Agfa Compugraphic 9600 (PostScript). Times Roman and Computer Modern fonts. Composition services for mathematical and technical books and journal production.

Anagnostopoulos, Paul C.
433 Rutland Street, Carlisle, MA
01741; (508) 371-2316
Composition and typesetting of high-quality books and technical documents. Production using Computer Modern or any available PostScript fonts. Assistance with book design. I am a computer consultant with a Computer Science education.

ArborText, Inc.
1000 Victors Way, Suite 400,
Ann Arbor, MI 48108;
(313) 996-3566
TeX installation and applications support. TeX-related software products.

Archetype Publishing, Inc.,
Lori McWilliam Pickert
P. O. Box 6567, Champaign, IL
61821; (217) 359-8178
Experienced in producing and editing technical journals with TeX; complete book production from manuscript to camera-ready copy; TeX macro writing including complete macro packages; consulting.

The Bartlett Press, Inc.,
Frederick H. Bartlett
Harrison Towers, 6F, 575 Easton
Avenue, Somerset, NJ 08873;
(201) 745-9412
Vast experience: 100+ macro packages, over 30,000 pages published with our macros; over a decade's experience in all facets of publishing, both TeX and non-TeX; all services from copyediting and design to final mechanicals.

Cowan, Dr. Ray F.
141 Del Medio Ave. #134,
Mountain View, CA 94040;
(415) 949-4911
Ten Years of TeX and Related Software Consulting: Books, Documentation, Journals, and Newsletters
TeX & LaTeX macropackages, graphics; PostScript language applications; device drivers; fonts; systems.

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.

Magus, Kevin W. Thompson
P. O. Box 390965, Mountain View
CA 94039-0965;
(800) 848-8037; (415) 940-1109;
`magus@cup.portal.com`
LaTeX consulting from start to finish. Layout design and implementation, macro writing, training, phone support, and publishing. Can take LaTeX files and return camera ready copy. Knowledgeable about long document preparation and mathematical formatting.

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
920 Addison, Palo Alto, CA 94301;
(415) 323-9624
Experienced in book production, macro packages, programming, and consultation. Complete book production from computer-readable copy to camera-ready copy.

Pronk&Associates Inc.
1129 Leslie Street, Don Mills,
Ontario, Canada M3C 2K5;
(416) 441-3760; Fax: (416) 441-9991
Complete design and production service. One, two and four-color

books. Combine text, art and photography, then output directly to imposed film. Servicing the publishing community for ten years.

Quixote Digital Typography, Don Hosek
555 Guilford, Claremont,
CA 91711; (909) 621-1291;
Fax: (909) 625-1342
Complete line of TEX, LATEX, and METAFONT services including custom LATEX style files, complete book production from manuscript to camera-ready copy; custom font and logo design; installation of customized TEX environments; phone consulting service; database applications and more. Call for a free estimate.

Richert, Norman
1614 Loch Lake Drive, El Lago, TX
77586; (713) 326-2583
TEX macro consulting.

TEXnology, Inc., Amy Hendrickson
57 Longwood Ave., Brookline, MA
02146; (617) 738-8029
TEX macro writing (author of MacroTEX); custom macros to meet publisher's or designer's specifications; instruction.

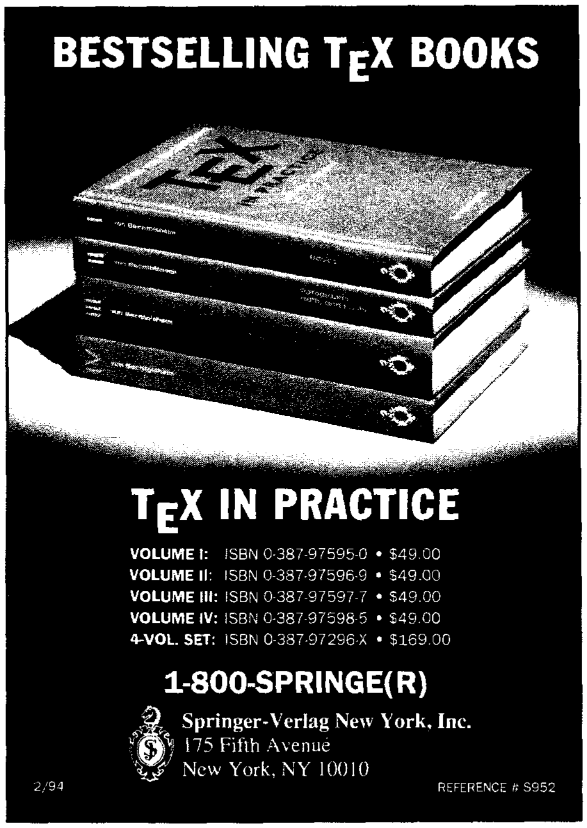
Type 2000
16 Madrona Avenue, Mill Valley,
CA 94941; (415) 388-8873;
Fax: (415) 388-8865
$2.50 per page for 2000 DPI TEX camera ready output! We have a three year history of providing high quality and fast turnaround to dozens of publishers, journals, authors and consultants who use TEX. Computer Modern, Bitstream and METAFONT fonts available. We accept DVI files only and output on RC paper. $2.25 per page for 100+ pages, $2.00 per page for 500+ pages.

Outside North America

TypoTEX Ltd.
Electronical Publishing, Battyány
u. 14. Budapest, Hungary H-1015;
(036) 11152 337
Editing and typesetting technical journals and books with TEX from manuscript to camera ready copy. Macro writing, font designing, TEX consulting and teaching.

Information about these services can be obtained from:

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



Index of Advertisers

Individual Membership Application

TeX
USERS
GROUP

Complete and return this form with payment to:

TeX Users Group
Membership Department
P. O. Box 869
Santa Barbara, CA 93102 USA

Telephone: (805) 963-1338
FAX: (805) 963-8358
Email: `tug@tug.org`

Membership is effective from January 1 to December 31 and includes subscriptions to *TUGboat, The Communications of the TeX Users Group* and the TUG newsletter, *TeX and TUG NEWS*. Members who join after January 1 will receive all issues published that calendar year.

For more information ...

Whether or not you join TUG now, feel free to return this form to request more information. Be sure to include your name and address in the spaces provided to the right.

Check all items you wish to receive below:

- □ Institutional membership information
- □ Course and meeting information
- □ Advertising rates
- □ Products/publications catalogue
- □ Public domain software catalogue

Name ______________________________

Institutional affiliation, if any ______________________________

Position ______________________________

Address (business or home (circle one)) ______________________________

____

____

City ______________________ Province/State ______________

Country ______________________ Postal Code ______________

Telephone ______________________ FAX ______________

Email address ______________________________

I am also a member of the following other TeX organizations:

Specific applications or reasons for interest in TeX:

There are two types of TUG members: regular members, who pay annual dues of $60; and full-time student members, whose annual dues are $30. Students must include verification of student status with their applications.

Please indicate the type of membership for which you are applying:

□ Regular at $60 □ Full-time student at $30

Amount enclosed for 1994 membership: $ __________

□ Check/money order payable to TeX Users Group enclosed
(checks in US dollars must be drawn on a US bank; checks in other currencies are acceptable, drawn on an appropriate bank)

□ Bank transfer:

TeX Users Group, Account #1558-816,
Santa Barbara Bank and Trust, 20 East Carrillo Street,
Santa Barbara, CA 93101 USA

your bank ______________________________

ref # ______________________________

□ Charge to MasterCard/VISA

Card # ______________________ Exp. date __________

Signature ______________________________

TEX
USERS
GROUP

Institutional Membership Application

Institution or Organization ______________________________

____

Principal contact ______________________________

Address ______________________________

____

City ____________________ Province/State __________

Country ____________________ Postal Code __________

Daytime telephone ______________ FAX ______________

Email address ______________________________

Each Institutional Membership entitles the institution to:

- designate a number of individuals to have full status as TUG individual members;
- take advantage of reduced rates for TUG meetings and courses for *all* staff members;
- be acknowledged in every issue of *TUGboat* published during the membership year.

Educational institutions receive a $100 discount in the membership fee. The three basic categories of Institutional Membership each include a certain number of individual memberships. Additional individual memberships may be obtained at the rates indicated. Fees are as follows:

Category	Rate (educ. / non-educ.)	Add'l mem.
A (includes 7 memberships)	$ 540 / $ 640	$50 ea.
B (includes 12 memberships)	$ 815 / $ 915	$50 ea.
C (includes 30 memberships)	$1710 / $1810	$40 ea.

Please indicate the type of membership for which you are applying:

Category ____ + ____ additional individual memberships

Amount enclosed for 1994 membership: $ ________

□ Check/money order payable to TEX Users Group enclosed (*payment in US dollars must be drawn on a US bank; payment in other currencies is acceptable, drawn on an appropriate bank*)

□ Bank transfer: your bank ______________________________

ref # ______________________________

TEX Users Group, Account #1558-816,
Santa Barbara Bank and Trust, 20 East Carrillo Street,
Santa Barbara, CA 93101 USA

□ Charge to MasterCard/VISA

Card # ______________________________ Exp. date ________

Signature ______________________________

Please attach a list of individuals whom you wish to designate as TUG individual members. Minimally, we require names and addresses so that TUG publications may be sent directly to these individuals, but we would also appreciate receiving the supplemental information regarding phone numbers, email addresses, and TEX interests as requested on the TUG Individual Membership Application form. For this purpose, the latter application form may be photocopied and mailed with this form.

Complete and return this form with payment to:

TEX Users Group
Membership Department
P. O. Box 869
Santa Barbara, CA 93102
USA

Membership is effective from January 1 to December 31. Members who join after January 1 will receive all issues of *TUGboat* and *TEX and TUG NEWS* published that calendar year.

For more information ...

Correspondence
TEX Users Group
P. O. Box 869
Santa Barbara, CA 93102
USA

Telephone: (805) 963-1338
FAX: (805) 963-8358
Email: tug@tug.org

Whether or not you join TUG now, feel free to return this form to request more information.

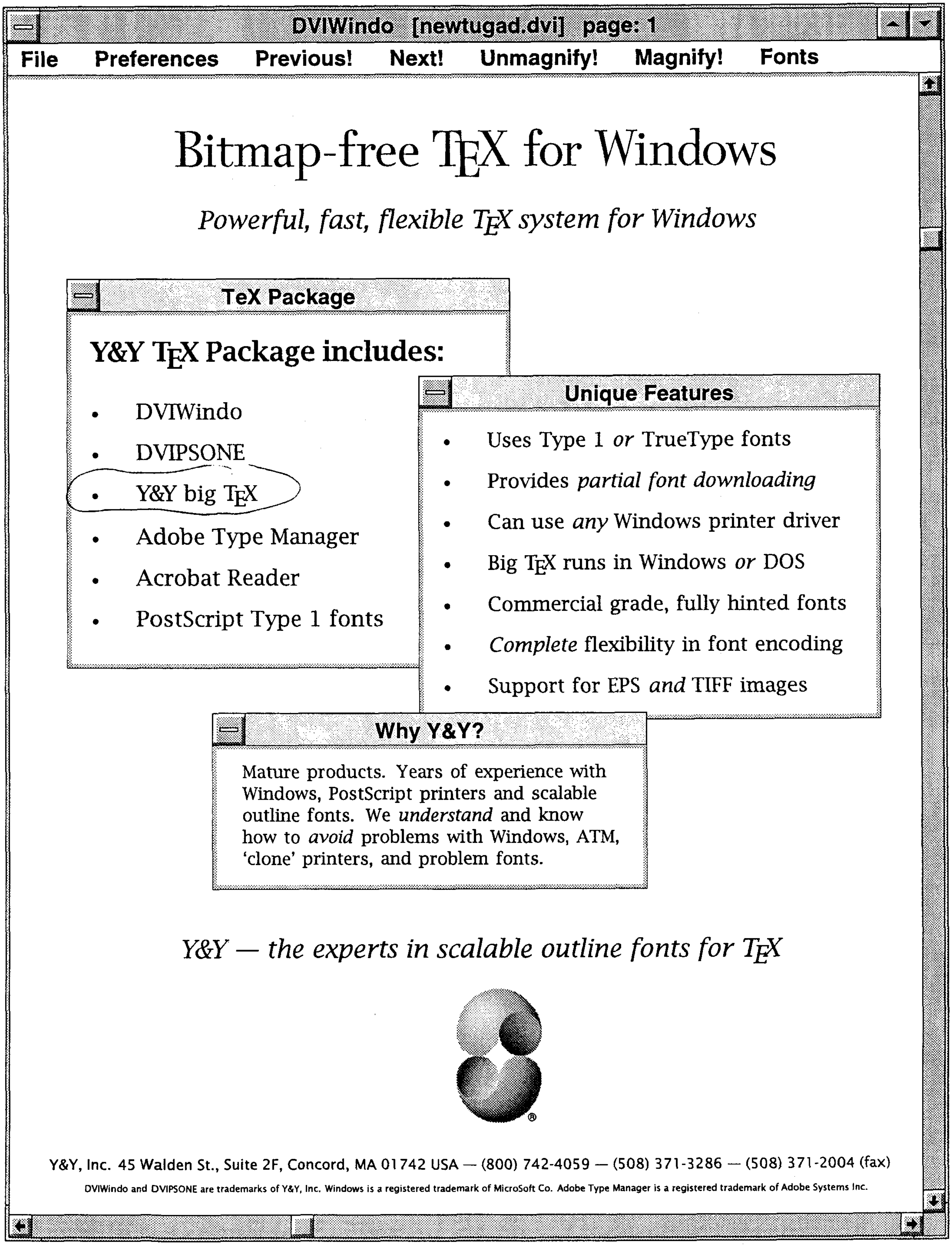
Check all items you wish to receive below:

□ Course and meeting information

□ Advertising rates

□ Products/publications catalogue

□ Public domain software catalogue



TUG '95

— St. Petersburg Beach, Florida —

July 24–28, 1995

The TeX Users Group is proud to announce that the **sixteenth** annual meeting will be held at the TradeWinds Hotel, in St. Petersburg Beach, Florida, July 24–28, 1995. We would like to extend a warm invitation to TeX users around the world—come join us at one of the largest and most beautiful resort beaches in Florida, as we explore where TeX is to be found and how its users are going far beyond—or are diverging from—its initial mathematical context.

The theme of the meeting will be "Real World TeX" and we plan to have demonstrations of pre- and post-processors, and the active participation of developers and vendors, in hopes that **you**, the user, may discover "hands-on" just what can be done with TeX, METAFONT, PostScript, and other utilities!

Commercial users of TeX are particularly encouraged to attend. The meeting will feature papers of interest to publishers and TeX vendors, a panel discussion addressing commercial users' needs and wants, and a gallery for displaying samples of TeX work.

There will be the usual courses associated with the meeting: *Intensive Courses* in LaTeX2$_\varepsilon$ and TeX, PostScript, Graphics, and perhaps other topics. The meeting itself will have excellent speakers, panel discussions, workshops, poster displays, Birds-of-a-Feather sessions (BoFs) and technical demonstrations.

Getting Information

A preliminary schedule will be available in February of 1995, so be sure to look for updates in *TeX and TUG NEWS* and *TUGboat*, on the World Wide Web, at `http://www.ucc.ie/info/TeX/tug/tug95sched.html`, as well as on the CTAN archives in `tex-archive/usergrps/tug/`.

Nearer the time of the conference, there will be an on-line form for registration (`tug95form.html`) located on the WWW cited above.

Send suggestions and requests to `tug95c@scri.fsu.edu`.

The TUG95 committee will be working with individuals who wish to share accommodations, to help defray expenses. The Bursary Fund is also available to assist TeX users who demonstrate need. All members are encouraged to consider contributing to the fund. To obtain more information about contributing to or applying for the Bursary Fund, please contact the TUG office by email to `tug@tug.org` or by post to the address

TeX Users Group
P.O. Box 869
Santa Barbara, CA 93102-0869 USA.

Deadlines

Submission of Abstracts	◇	January 13, 1995
Preliminary Papers Due	◇	March 17, 1995
Preprint Deadline	◇	June 23, 1995
Meeting Date	◇	July 24–July 28, 1995
Camera Ready Deadline	◇	August 25, 1995

TUGboat

Volume 15, Number 2 / June 1994

General Delivery

Dreamboat

Fonts

Book Reviews

Typesetting on Personal Computers

Macros

LaTeX

Letters

Abstracts

News & Announcements

Late-Breaking News

TUG Business

Forms

Advertisements