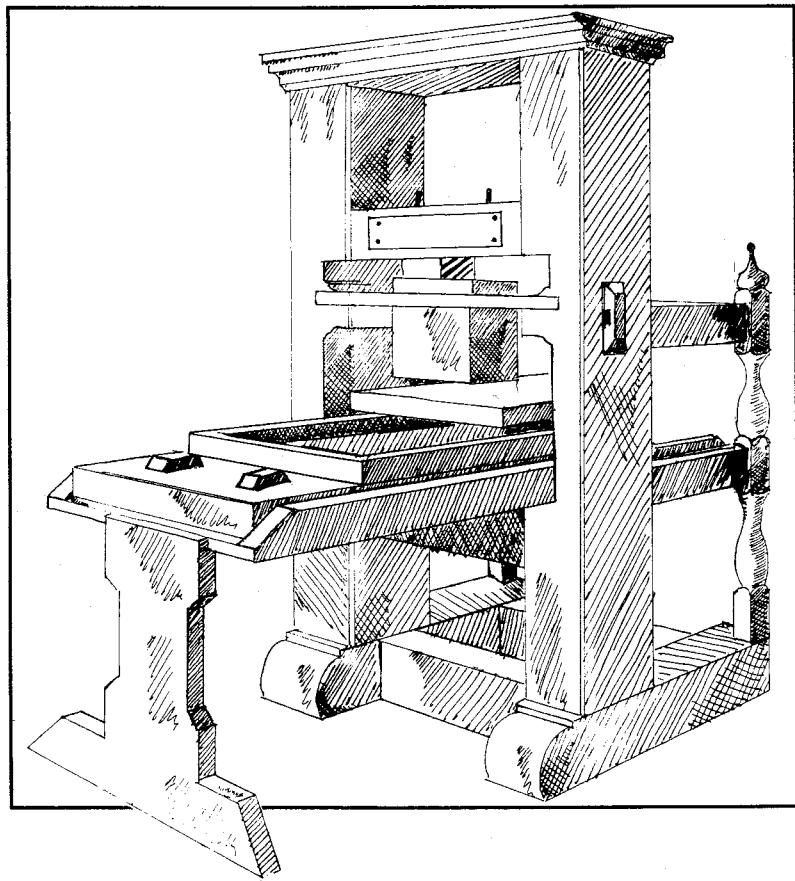


TUGBOAT

The Communications of the **TEX** Users Group



Volume 12, Number 2, June 1991

TeX Users Group

Memberships and Subscriptions

TUGboat (ISSN 0896-3207) is published four times a year plus one supplement by the TeX Users Group, 653 North Main Street, P.O. Box 9506, Providence, RI 02940, U.S.A.

1991 dues for individual members are as follows:

- Ordinary members: \$45
- Students: \$35

Membership in the TeX Users Group is for the calendar year, and includes all issues of *TUGboat* for the year in which membership begins or is renewed. A membership form is provided on page 331.

Subscription rates: United States \$45 a year; all other countries, \$45.

Second-class postage paid at Providence, RI, and additional mailing offices. Postmaster: Send address changes to the TeX Users Group, P.O. Box 9506, Providence, RI 02940, 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.

The privileges of institutional membership include:

- designating individuals to receive *TUGboat* subscriptions and be accorded the status of individual TUG members, the number of such designees depending on the category of membership chosen;
- reduced rates for the purchase of additional subscriptions;
- reduced rates for TUG meetings/courses for all staff members, and for rental/purchase of videotapes.
- acknowledgement in every issue of *TUGboat* published during the membership year.

For further information, contact the TUG office.

TeX is a trademark of the American Mathematical Society.

TUGboat © Copyright 1991, TeX Users Group

Board of Directors

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

Nelson Beebe, *President*

Malcolm Clark, *European Coordinator*,

Vice-President for the U.K.

Bernard Gaulle, *Vice-President for GUTenberg*

Roswitha Graham, *Vice-President for the Nordic countries*

Kees van der Laan, *Vice-President for NTG*

Joachim Lammarsch, *IBM VM/CMS*

Site Coordinator, Vice-President for DANTE

Barbara Beeton, *TUGboat Editor*

Lance Carnes, *Small Systems Site Coordinator*

Bart Childs, *DG MV Site Coordinator*

John Crawford, *Prime 50 Series Site Coordinator*

Allen Dyer

David Fuchs

Regina Girouard

Raymond Goucher, *Founding Executive Director*†

Dean Guenther

Hope Hamilton

Doug Henderson, *METAFONT Coordinator*

Alan Hoenig

Patrick Ion

David Kellerman, *VAX/VMS Site Coordinator*

David Kratzer

Pierre MacKay, *UNIX Site Coordinator*

Craig Platt, *IBM MVS Site Coordinator*

Christina Thiele

Hermann Zapf, *Wizard of Fonts*†

†*honorary*

See page 203 for addresses.

Addresses

General correspondence: 401-751-7760

TeX Users Group

P. O. Box 9506

Providence, RI 02940

Payments:

TeX Users Group

P. O. Box 594

Providence, RI 02901

Parcel post,

delivery services:

TeX Users Group

653 North Main Street

Providence, RI 02904

Telephone

401-751-7760

Fax

401-751-1071

Electronic Mail

(Internet)

General correspondence:

TUG@Math.AMS.com

Submissions to *TUGboat*:

TUGboat@Math.AMS.com

Publishers have, no doubt, influenced the course of history no less than kings or generals; the publishers' trouble being that they had to earn money during operations.

Ch. Enschedé,
quoted in S. L. Hartz
The Elseviers and their Contemporaries (1955)

TUGBOAT

COMMUNICATIONS OF THE *TEX* USERS GROUP
TUGBOAT EDITOR BARBARA BEETON

VOLUME 12, NUMBER 2 JUNE 1991
PROVIDENCE • RHODE ISLAND • U.S.A.

TUGboat

During 1991, the communications of the *TeX* Users Group will be published in four issues. Two issues will consist primarily of Proceedings, one of *TeX90*, Cork (Vol. 12, No. 1), and the other of the 1991 TUG Annual Meeting (Vol. 12, No. 3).

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

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

Submitting Items for Publication

The deadline for submitting technical items for Vol. 12, No. 4, is August 13, 1991, and for news items, September 10, 1991; the issue will be mailed in November. (Deadlines for future issues are listed in the Calendar, page 307.)

Manuscripts should be submitted to a member of the *TUGboat* Editorial Committee. Articles of general interest, those not covered by any of the editorial departments listed, and all items submitted on magnetic media or as camera-ready copy should be addressed to the Editor, in care of the TUG office.

Contributions in electronic form are encouraged, via electronic mail, on magnetic tape or diskette, or transferred directly to the American Mathematical Society's computer; contributions in the form of camera copy are also accepted. The *TUGboat* "style files", for use with either plain *TeX* or *LATEX*, will be sent on request; please specify which is preferred. For instructions, write or call Karen Butler at the TUG office.

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

***TUGboat* Advertising and Mailing Lists**

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

***TUGboat* Editorial Committee**

Barbara Beeton, *Editor*

Ron Whitney, *Production Assistant*

Helmut Jürgensen, *Associate Editor, Software*

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

Don Hosek, *Associate Editor, Output Devices*

Victor Eijkhout, *Associate Editor, Macros*

Jackie Damrau, *Associate Editor, LATEX*

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

See page 203 for addresses.

Other TUG Publications

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

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

Trademarks

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

APS μ5 is a trademark of Autologic, Inc.

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

LaserJet, *PCL*, and *DeskJet* are trademarks of Hewlett-Packard, Inc.

METAFONT is a trademark of Addison-Wesley Inc.

PC T_EX 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.

UNIX is a trademark of AT&T Bell Laboratories.

General Delivery

President's Introduction

Nelson H. F. Beebe

News Items

Some technical magazines have a regular column each issue reporting recent newsworthy events in their fields. Because typography has such a broad range of applications, it would be impossible for a single unaided individual to write a similar one for *TUGboat*. I've been collecting small items of news that should be of interest to readers of *TUGboat*, and present them here as a small sample of what could be a regular feature, were there sufficient contributions from others.

Music typesetting

Those who have previously followed the work of Eric Foxley [2], John Gourlay [3], Daniel Taupin [11], and Andrea Steinbach and Angelika Schofer (authors of *m_{tex}*, available in some *TeX* archives) on the typesetting of music should read the recent paper by Blostein and Haken [1]. It will also give you a chance to compare the "new typography", and extensive use of font scaling, introduced in the July 1990 issue of the Communications of the ACM with that of older issues. The change in the Communications of the ACM is so radical that it seems worthy of further comment. I shall refrain from doing so here, but I invite readers with sounder typographical backgrounds than mine to offer their scholarly views. I raise this point because some TUG members have called for a "new look" to *TUGboat*.

The internals of *TeX*

David Salomon has expanded upon his earlier presentations of *TeX* output routines in these pages [8, 9] in some fine work which we hope to see published as a *TeXniques* issue.

A grand challenge

At the *TeX90* meeting in Cork, Timothy Murphy [7] raised an important point that I had not heard discussed before. In compilation of computer languages, a number of automated techniques and programs were developed during the 1970s and 1980s for the generation of lexical analyzers and parsers directly from a concise specification of the language grammars. Much of this is based on pioneering work

of none other than *TeX*'s own author, Don Knuth, on what are now known as *LR(k)* grammars [5].

The advantages of such an approach are now widely understood in the computer science community. In particular, these methods provide for rapid experimentation in language design, since grammar changes can be immediately incorporated in a correct working parser. Timothy Murphy suggested that we really ought to have a rigorous grammatical description of *TeX*, and a reimplementation of the *TeX* program based on that grammar.

Such an approach is, I believe, critical to future development of *TeX*'s descendants. I do not, however, know whether it is even possible to define *TeX*'s parsing by a grammar that is amenable to automatic parser generation. This could easily be enough work for a graduate thesis.

METAFONT oddities

Sharp-eyed readers have sometimes observed peculiar things happening in the fine details of characters from METAFONT, particularly where lines join. The Grand Wizard explains it all in [6].

TUGlib archive

The *TeX* Users Group bibliography collection that I've been maintaining in the TUGlib archives at Utah continues to grow, thanks to the contributions of several colleagues who are acknowledged in the collections. Readers with network access can send an e-mail message with the text

send index from tex/bib

to tuglib@math.utah.edu [128.110.198.2]. Alternatively, those with Internet connections can more conveniently use anonymous ftp to [math.utah.edu](ftp://math.utah.edu) and look in the directory <pub/tex/bib>.

There are now 65 book entries about *TeX* and digital typography, 45 journal articles (these *exclude* articles from the various *TeX* user group publications), and a list of nearly 100 periodicals that are typeset with *TeX*, *exclusive* of the many AMS journals. Contributions to these bibliographies are invited from the *TUGboat* readership.

We expect to have current copies available for distribution at the TUG91 meeting in July. While it has been suggested that these bibliographies could be published as a *TeXniques* issue, I am disinclined to do so, for the reason that they are continually being updated, sometimes more than once a week, so any printed copy would rapidly be out of date.

TUGlib archive access via e-mail is still small, but the log records over 800 accesses in the last year from 173 users. The anonymous ftp access to the

archives records 735 logins during the last two weeks of April 1991. It seems that these services are being found useful.

Checksums

Robert M. Solovay at the University of California at Berkeley took up my challenge to write an idempotent file checksummer in WEB, and succeeded. I expect to incorporate it into regular use for files in the TUGlib archives, and of course, the code will be available as well.

Literate programming

The 40 entries in the literate programming bibliography in the TUG archive now need to be supplemented with the contents of a recent annotated bibliography on the subject [10].

Usenet newsgroups `sci.math.symbolic` and `comp.text.tex` carried a posting on 26 April 1991 about a new version (1.3) of TeX/Mathematica available for anonymous ftp from `chem.bu.edu` [128.197.30.18] in the directory `pub/tex-mathematica`. This incorporates support for literate programming in Mathematica.

TeX and computer architecture

The recent book by Patterson and Hennessy [4] on computer architecture has attracted rave reviews. A foreword by noted computer architect Gordon Bell says *To advance computing, I urge publishers to withdraw the scores of books on this topic so a new breed of architect/engineer can quickly emerge.* The authors of the book are the originators of the Berkeley and Stanford RISC chips which have revolutionized the computing industry in the last decade. Throughout the book, they use three widely-available programs to justify architectural decisions. One of these programs is TeX; the others are the Free Software Foundation's C compiler, `gcc`, and the circuit-design program, `spice`.

E-mail lists and archives

For a number of years, *TUGboat* editor Barbara Beeton has coordinated a small `tex-implementors` electronic mail list as a forum for discussing implementation issues, and sharing news about, and problems with, TeX and METAFONT software. Now there are some new lists.

`tex-archive` is intended to contain addresses of archive site coordinators, so that they too can be kept aware of recent happenings. We have endeavored to make this list reach major sites that we know of, but if your archive site is not included, and you

wish to be added, send a message to the human answering to `tex-archive-request@math.utah.edu`.

People interested specifically in the problems of font design and font naming can communicate on the `tex-fonts` list; a message to `tex-fonts-request@math.utah.edu` will get you on it. One problem that has received considerable discussion is how to map the names of thousands of existing fonts into the limited filenames of some operating systems, in such a way as to preserve document portability, not require forbidden changes to TeX itself, and still be understandable by users.

The three lists above are not intended for wide distribution; their goal is to reach a small core of experts.

Some new lists may be of wider interest.

The formation of a group of Irish TeX users was discussed at the TeX90 meeting in Cork, and on 22 February 1991, UKTeX issue 91.009 carried an announcement about the establishment of ITALIC (Irish TeX And LATEX Interest Community). Users in Ireland and abroad are invited to signal their interest in this group and join the mail discussion list `italic-1` by sending a one-line electronic mail message to `listserv@irlearn.bitnet` saying `subscribe italic-L <your real name>`.

An announcement on 29 April 1991 appeared in the `tex-euro` list about the formation of a list to discuss Greek TeX. To join, send a message to `listserv@dhdurz1.bitnet` with the text `subscribe ellhnikia` (ellhnikia is Greek for Greek, in Roman).

The TeXhax issue of 21 April 1991 (volume 91, issue 20) carried a posting about a new list for the discussion of Turkish TeX. To join, send a message to `listserv@trmetu.bitnet` with the text `subscribe yunus <firstname> <middle-name> <lastname>`. The list is named after Yunus Emre, a fifteenth century Turkish poet and philosopher.

A connection to the Usenet `comp.text.tex` list is now available for Bitnet sites; send a message to `listserv@shsu.bitnet` with the text `subscribe info-tex`. This list has been created by George Greenwade at Sam Houston State University in Huntsville, TX. A mirror image of the Aston archive is being established there, which will at long last make a North American copy of that archive readily accessible.

Finally, for those of you who have net access and have tried to answer the question "On what archive will I find this software?", you should know about the `archie` server at McGill University. This is a new project that is an archive of archives: the

archie server communicates about once a month with known archive sites around the world, fetches a directory listing of their holdings, and stores it in a database. A request like `prog dvialw` returns a list of all archives containing my PostScript DVI driver, including their file time stamps. I intend to use this to announce updates.

The **archie** server is available both via e-mail, and via telnet to the host machine, `quiche.cs.-mcgill.ca` [132.206.2.3], where you can login with username **archie** (no password) and search the data base. On-line help is available. Via e-mail, a request `help` to `archie@quiche.cs.mcgill.ca` will get you started. **archie** is already overworked, and the plan is to clone him at other sites.

REVTEX

In early April, 1991, the American Physical Society released Version 2.0 of the *REVTEX* macro package, which is a *LATEX*-based system for preparation of manuscripts for the *Physical Review A, B, C, and D* journals, and soon, for *Physical Review Letters* and *Reviews of Modern Physics*.

This new version is incompatible with earlier ones, and should replace them. Instructions are provided in a `README` file for the conversion of manuscripts that used earlier versions. The new *REVTEX* is now compatible with *BIBTEX*.

Most *TEx*ware archive sites should have the new version available by the time you read this.

An industry leader's view

During the last week of March, the Novell Developer's Conference was held across the street from my office, and I took the opportunity to listen to keynote speakers each morning. One of them, Steve Jobs, the co-founder of Apple Computer, and the founder of NeXT Computer, said that in his view, the two most significant events in computing during the last two decades have not been hardware, windows, graphical user interfaces, or the Macintosh. They have been (a) the spreadsheet, and (b) desktop publishing. These two areas have been responsible for the spread of personal computers so that several tens of millions have now been sold, and the manufacturing economies of scale have revolutionized the rest of the computing industry.

TEx Users Group Board changes

The past year has been a difficult time for the TUG Board. A severe division arose in the Board over future directions, and management. This division was approximately between the Executive and Finance Committee members on one side, and part of

the remainder of the Board on the other. Actions were taken at the Cork meeting to partially revise *TEx* Users Group bylaws, and to terminate the contract of the Executive Director effective at the end of 1990.

These divisions have been so strong that three of my fellow elected officials have resigned their offices. In the interests of maintaining at least some continuity, I will remain in office until the expiration of my term at the end of 1991. At that time, I will step down, and will not in the future be a candidate for *TEx* Users Group office. I have chosen not to appoint interim officers to fill the three Executive Committee vacancies. It is difficult for me to write this; I have suffered many sleepless nights over Board issues.

The Board has revised the election procedures, and an election will not be held at the TUG91 meeting. Instead, a mail ballot will be sent to the full membership. Details should appear in the first issue of a newsletter which is in preparation.

The new directions chosen by the Board will require significant financial resources to carry out. The fact that the TUG budget will suffer a significant loss this year, mostly due to falling course revenues (which in turn may be related to the state of the economy), is a very real problem.

Most TUG members are probably unaware that their membership/subscription dues contribute less than 25% of the budget (a figure that can be readily determined from the Treasurer's Reports published periodically in *TUGboat*). Although dues have been increased slightly, they do not produce sufficient additional income to cover the deficit, which may this year be more than 10% of the budget.

For comparison with another non-profit organization, the Association for Computing Machinery (ACM) 1990 budget shows membership providing 30% and subscriptions 12% of their annual budget, which is about 32 times larger than TUG's. The ACM has 18 times as many members as TUG, so their budget share per member is about double that of TUG.

The future

I regret the state of TUG finances, because it prevents us from supporting research into digital typography at a time when I believe strongly that such research is essential. We must not forget that *TEx* was designed *before* personal computers and workstations, *before* bitmapped graphics displays, *before* cheap memory, *before* fast RISC processors, *before* laser printers, *before* graphics standards, *before* PostScript, and *before* SGML.

All of these have resulted in limitations in TeX's design that *must* be removed in future systems. I believe that TeX has many unique features, the most important of which are high quality mathematical typography, support for multiple languages and character sets, programmability, and public access to the source code.

The desktop publishing industry is responding to all of the above issues, and there is the very real risk that the benefits of TeX will be lost to commercial systems which will once again produce captive markets and ultimately, throttle development.

References

- [1] Dorothea Blostein and Lippold Haken. Justification of printed music. *Communications of the ACM*, 34(3):88–99, March 1991.
- [2] Eric Foxley. Music—a language for typesetting music scores. *Software—Practice and Experience*, 17(8):485–502, August 1987.
- [3] John S. Gourlay. A language for music printing. *Communications of the Association for Computing Machinery*, 29(3):388–401, 1986.
- [4] John L. Hennessy and David A. Patterson. *Computer Architecture—A Quantitative Approach*. Morgan Kaufmann Publishers, 1990.
- [5] Donald E. Knuth. On the translation of languages from left to right. *Information and Control*, 8(6):607–639, 1965.
- [6] Donald E. Knuth. A note on digitized angles. *Electronic Publishing—Origination, Dissemination, and Design*, 3(2):99–104, May 1990.
- [7] Timothy Murphy. PostScript, QuickDraw, and TeX. *TUGboat*, 12(1):64–65, March 1991.
- [8] David Salomon. Output routines: Examples and techniques. Part I: Introduction and examples. *TUGboat*, 11(1):69–85, April 1990.
- [9] David Salomon. Output routines: Examples and techniques. Part II: OTR techniques. *TUGboat*, 11(2):212–236, June 1990.
- [10] Lisa M. C. Smith and Mansur H. Samadzadeh. An annotated bibliography of literate programming. *ACM SIGPLAN Notices*, 26(1):14–20, January 1991.
- [11] Daniel Taupin. Musique en TeX: écriture de musique polyphonique ou instrumentale en “plain TeX”. *Cahiers GUTenberg*, (5):62–73, May 1990. An English-language translation is available from the author.

◇ Nelson H. F. Beebe
 Center for Scientific Computing
 Department of Mathematics
 South Physics Building
 University of Utah
 Salt Lake City, UT 84112
 USA
 Tel: (801) 581-5254
 FAX: (801) 581-4148
 Internet: Beebe@math.utah.edu

Editorial Comments

Barbara Beeton

Newsletter news

The biggest piece of news to report here is the appearance of a new TUG publication—a newsletter. (You should probably have received your copy by now.) While the present incarnation is just a prototype, its usefulness will be examined in detail against the mechanics and cost of production.

A newsletter is presumed to be a timely publication, providing information about events, scheduled or spontaneous, and carrying questions and answers with a short turnaround. A newsletter is a suitable forum for reports on TUG activities and news from the Board of Directors. (A significant part of the prototype issue is devoted to such matters.) Therefore, it should appear more frequently than *TUGboat*, probably about six times a year.

If you have opinions on this subject that you wish to share, send them to the TUG Publications Committee, in care of the TUG office (the address is at the top of the list on page 203), or via e-mail to TUG-Pubcomm@Math.AMS.com.

TUGboat items in electronic archives

I have seen a number of inquiries about whether macros, style files, and similar items published in *TUGboat* are available from the various electronic archives. This is an old question, and the answer is also rather old, although it has unfortunately not yet been put into action. Several years ago, it was agreed in principle, and confirmed by the TUG board of directors, that such items should be made available electronically, one form of which is installation in the archives.

The fact that this hasn't actually happened to any great extent is due to the fact that the *TUGboat* production staff is very limited (it consists of Ron Whitney and myself) and overworked (I am employed full time by the American Mathematical Society, and *TUGboat* is my avocation; Ron is at present acting manager in charge of the TUG office).

The post-processing of items published in *TUGboat* is non-trivial. Use of such features as non-standard fonts or multiple files must be documented or changed. The author's address must be verified. The author should get questions on how to use the items, not the *TUGboat* staff. Other documentation and identifying material at the beginning of the file must be checked. Any special code added explicitly for *TUGboat* should be removed. Although we try very hard to avoid inserting explicit line and page breaks, that isn't always possible. And, unfortunately, we've never had time to write up guidelines for doing this, so it's hard to turn it over to someone else.

Nevertheless, we would like to try to do a better job of electronic distribution, and if you'd like to volunteer to help, please let us know. (Write to us at *TUGboat@Math.AMS.com* on the Internet.) Because of the limits on our available time, we ask that you have a stable e-mail connection and be familiar with the use of the *TUGboat* styles; the current versions of *tugboat.sty*, *ltugboat.sty*, and *tugboat.com*, as well as the documentation file, *tubguide.tex*, can be found at *labrea.Stanford.edu* in the directory */tex/tugboat* as well as at other archive sites.

TUG Resource Directory

Accompanying this issue is a supplement; take a close look at it. Last year, the corresponding supplement was the annual membership list. The membership list is still there, but it has been joined by other useful information.

Short descriptions are given of other *TeX* user associations, including the areas covered, officers, how get in touch, and sometimes more. For several countries of eastern Europe, where formal associations are still in the process of being formed, the names of central contacts are given.

Sources of *TeX*-related software are listed in another section. This includes electronic sources accessible by ftp or servers, other sources of public-domain implementations of *TeX*, commercial vendors, and Don Hosek's list of output device drivers (formerly included in the body of *TUGboat*).

Another section provides information on electronic discussion and help lists.

The final section is the current contents of the *TeX* bibliography from the TUGlib archive.

Updates to these resource lists will appear in the "Resources" column of *TUGboat* as previously.

(Mis)information about *TeX* in non-*TeX* publications

From time to time, I come across references to *TeX* in rather unexpected places, and also unexpected references in the more usual places. Many references are knowledgeable and informative, but sometimes, it's clear that whoever wrote the item didn't have the foggiest idea of what s/he was talking about. (It's been said that some publicity seekers welcome *any* reference, as long as their name is spelled correctly; well, sometimes in the case of *TeX* that isn't even true.)

TeX users can do a lot to improve this situation.

First, send a description (or, better still, a copy) of the reference to the *TeX* Users Group office. Be complete—include the name of the book or periodical, date of publication, the name and address of the publisher, and the numbers of the page or pages on which the (mis)information appeared. Include any comments or suggestions you think might be helpful in writing to the publisher. The e-mail address is *TUG@Math.AMS.com*, and the postal address can be found at the top of page 203. (If you happen to see a good, informative reference, send that too—if appropriate, it will be added to the growing *TeX* bibliography. To avoid duplication, you might want to check the latest version of the *TeX* bibliography first; see Nelson Beebe's instructions for accessing the TUGlib archive, page 205, or the latest TUG Resource Directory for a printed version.)

Next, if you ever have the opportunity, write an article for some magazine or journal whose readers are doing things where *TeX* could be a useful tool, to describe what *TeX* can do, and how to learn more about it. Or give a talk to a group of people at a meeting of a publishing, scientific or computer-related society. Point out the fact that, though *TeX* software is in the public domain, the many different implementations must all adhere to a single standard in order to be referred to as *TeX*, and one of the consequences is that input prepared in a standard way for one computer will yield exactly the same results on another. A number of articles have been written in this vein, and references can be found in the *TeX* bibliography.

Finally, put in a reference to TUG. There are many users of *TeX* who don't know that TUG exists, or how to get in touch. There are thousands of users of personal computers without access to electronic

networks in this position. One of the TUG Board's hot discussion topics is how to reach these people. If you can help, or have any good ideas, do get in touch.

DANTE meeting, Vienna

The tenth meeting of the German *T_EX* interest group took place in Vienna on 21–23 February 1991, at the Technical University of Vienna. It was my pleasure to attend this meeting and to meet a most enthusiastic, well-organized, and growing group of *T_EX* users.

The meeting was divided into a number of technical sessions, each with a distinct theme, among them *L^AT_EX*, auxiliary software, fonts, and news from other user groups. A business meeting occupied part of one afternoon, and workshops were offered the last morning on the subjects of *L^AT_EX*, WEB, and SGML.

Rainer Schöpf presented an update on the status of *L^AT_EX3.0*. Barbara Burr, the editor of DANTE's newsletter, *Die T_EXnische Komödie*, described features of GNU Emacs that could be used to make *L^AT_EX* input easier and more reliable, and also the GNU documentation system, *T_EXinfo*. A presentation that I (as a former linguistics student) found particularly fascinating was the description of how *T_EX* has been used in the preparation of a linguistic atlas, with phonetic transcriptions of selected words superimposed on maps to delineate the areas in which distinct pronunciations are found, assisting the study of patterns of linguistic change.

In the font area, Konrad Neuwirth surveyed what fonts can be used with *T_EX* besides Computer Modern. Yannis Haralambous introduced Scholar*T_EX*, intended not only to provide fonts for many different languages, but also to simplify the handing of multilingual documents in a scholarly setting. The 256-character font agreed on in Cork was presented by Norbert Schwarz and Peter Breitenlohner.

Users group news included a presentation by Malcolm Clark on the state of user groups in Europe and America. Jiří Veselý spoke on *T_EX* in Czechoslovakia. The final technical presentation was by Joachim Lammarsch, describing DANTE's software program.

In addition to the technical program and business meeting, attendees were welcomed to a buffet supper in the banner room of the Rathaus, with musical accompaniment appropriate to the waltz capital of the world. Some of us chose to enjoy more musical adventures, in this 200th anniversary year

of Mozart's death, attending a performance of "Die Zauberflöte" at the Staatsoper. Wonderful!

I would like to congratulate Hubert Partl and Irene Hyna on their excellent meeting arrangements—it was so well organized that it seemed to run itself. And I would like to thank everyone who had any part in the meeting for their kind hospitality.

Publications of other *T_EX* organizations

In previous issues, abstracts of *TUGboat* articles have appeared in other languages, particularly German. On thinking about this, however, it seemed obvious that what would be most useful to *TUGboat* readers was information on what appears in other *T_EX* publications.

This new approach starts with this issue, where we have abstracts of two issues of the *Cahiers GUTenberg*, the official publication of the Groupe francophone des Utilisateurs de *T_EX*. Coverage will be expanded to the publications of other *T_EX* organizations in future *TUGboat* issues.

TUG's annual meeting

The preliminary program for this year's meeting indicates that there will be a higher concentration of presentations than ever before on how *T_EX* is actually used by publishers. Those of us who really do production work know that there are times when it's necessary to resort to more "traditional" methods (like cut-and-paste for inserting figures or applying page numbers or even running heads to items submitted in camera-ready form; careful reading of the *TUGboat* "Production notes" column will uncover many such references). There are also times when other software is needed as well, working in consort with *T_EX*, to provide features not built into "canonical" *T_EX* (e.g. graphics) or to make certain phases of manuscript preparation more convenient (e.g. translators from the input to word processors or SGML). A realistic approach to such requirements is needed for success in using *T_EX* as a production tool, and I expect to learn much from this summer's speakers.

I'd like to put in a plug here for the annual question-and-answer session. Last year we tried a new approach. Instead of expecting the attendees to come up with questions on the spot, we solicited questions from readers of several of the electronic discussion lists. (We got started too late to put a notice in *TUGboat*.) And we promised to search out answers, report them at the meeting Q&A session, and publish the results in *TUGboat*. The response was not large, but the questions were good ones, and the ability to research answers beforehand

meant both that there were fewer cases of "I don't know" and there was a transcript to be published in the Proceedings. We were happy with the success of this approach, and are using it again this year. If you have any good general questions, or ones that you think would be instructive, or even specific problems that you've tried for a long time to solve without success, send them in, and we will do our best. For presentation at the meeting, questions that have relatively short answers will be favored, but all questions will be published, along with whatever answers we can dig up.

If you have e-mail access, send your questions to `TUG-Q@TAMVM1.Bitnet` or to `TUG-Q@TAMVM1.TAMU.edu`.

If you have to use paper, write to "TUG91 Q&A Session" in care of the TUG office (the first address on page 203).

I expect to see many familiar TeX users in Dedham, and hope to meet many new ones as well.

of Stuttgart which stocks a fine collection of TeX programs, amongst it emTeX. Additionally DANTE is making efforts to distribute TeX to people not in the possession of a network connection.

Talks were scheduled for Oct. 11, beginning with a review by Joachim Schrod on the unholy marriage of non-standard font files and drivers not conforming to the TUG driver committee standards. The combination of two non-standard components may work, but often as a nonstandard font and a standard dvi-driver are used, problems arise. Character spacings will be incorrect and significantly degrade the readability of output. Unfortunately, even commercial implementations of drivers have these problems.

Peter Abbot presented the usage statistics of the Aston Archive-Server for the last months. Problems with the Rutherford gateway into BITNET still persist but are tackled with a new encoding scheme (`vvencode`) which is under development at Aston. Some participants surmised the problems were caused by left lane traffic in the UK part of the BITNET cable....

Applications of LATEX control statements, as in `ifthen.sty`, were discussed in a talk by H. Kopka. It was followed by a report by A. Lingnau on the development of an online documentation system for mathematical software using LATEX as the typesetting engine.

An approach to documenting LATEX style files was presented by W. Kaspar and received lively attention of the audience. Many of the available style files out there lack documentation. The solution is not as trivial as it seems. For example, copyright issues make documentation of certain styles *within* the style file itself a contentious issue. The goal of the discussion was to design a multilingual documentation format for style files, and a style file to print them. The documentation should include information on other required styles and style combination that cause problems.

Some 30 popular LATEX style files have been documented by volunteers in this manner and efforts are under way to compile a complete list of styles available. It was decided to pull different approaches together until the next German TeX meeting in Vienna (Feb. 20-22, 1991) and resume discussion.

Urs Widmer informed on the use of TeX in typesetting Chinese and Japanese texts, highlighting problems with the fonts and the many different character encoding schemes in use for Chinese, Japanese and Korean.

TeX in Germany

Walter A. Obermiller

Göttingen was the place to be for TeX users and implementors, as the 9. annual meeting of the German-language TeX users was held from October 10-12, 1990. Formerly in a geographically "marginal" position, the old university town Göttingen is now located in the center of Germany. Approximately 150 participants from Germany, Austria, Belgium and the UK attended the meeting. It was jointly sponsored and organized by the German-language TeX-users group DANTE and the German Society for Scientific Computing (GWDG), which is based in Göttingen.

In the wake of the meeting, the 3. member conference of DANTE was held in the afternoon on Oct 10. Chairman Joachim Lammarsch presented an encouraging outlook on DANTE's future. DANTE has had a *boom* year; membership increased from 150 at the beginning of 1990 to a staggering 858 with another 100 applications still pending. These figures and the age structure, with students prevailing, show that DANTE is doing its part to promote TeX in Germany.

The TeX server in Heidelberg has been complemented by the DANTE FTP-server at the University

Merits and some problems of producing texts with diacritical marks and multilingual editions using **TEX** were shown by G. Koch in examples from Hebrew texts and bilingual book projects.

Philosophical and technical insights into printing old German documents with appropriate old fonts were presented by Yannis Haralambous. He introduced old German Fraktur, Gotisch, Schwabacher and Initialen fonts he created with **METAFONT**, and showed old German works of print he has typeset in them. The fonts are *fantastic!* Yannis will make them available in Heidelberg soon.

G. Bienek announced the availability of a new **TEX** bulletin board. It is accessible under the number +49-8024-8416 and will carry the offerings of the normal DANTE distribution.

A new twist to including graphics in **TEX** documents was introduced by F. Sowa. A preprocessor converts Tag Image File Format (TIFF) graphic files into a .pk font file which is then simply printed by including a .tex file also produced in the process. The bitmap is distributed over a number of characters, so even old drivers should be able to handle them. His graphic inclusion mechanism hence does not require the use of a \special. The preprocessor is capable of dealing with simple bitmaps and has dithering capabilities to deal with grayscale and RGB pictures.

From the user interface "department", L.P. Kurdelski presented a **Smalltalk** based system to simplify the creation and processing of **TEX** documents on PCs and their output via printers connected to a heterogeneous network, so as to relieve the user from having to deal with DOS or network software.

The final day of the meeting had four tutorials on the agenda. H. Kopka introduced the use of **PICTEX** macros, while B. Burr gave an introductory **LATEX** tutorial. A tutorial on how to change **LATEX** style files (H. Partl) and one on **METAFONT** (F. Sowa) were followed by a final discussion.

Conclusion

Many different uses of **TEX** in Germany are reflected by the talks outlined above. As more and more people are using **TEX** here, many more will be thought of. As all **TEX** meetings that I have attended this meeting too was very short.

The challenges for the immediate future are clear: introduction of the new 256 character Latin fonts, for which the encoding scheme has been tentatively approved at the Cork meeting. Only new German hyphenation patterns and a revised **german.sty** will then be needed to make full use

of **TEX**'s hyphenation capabilities for German language texts.

◊ Walter A. Obermiller
Max-Planck Institut für Chemie
Geochemistry Division
Postfach 3060
6500 Mainz
FRG
walter@mpch-mainz.mpg.dbp.de

Philology

Russian **TEX**

Basil Malyshev, Alexander Samarin and Dimitri Vulis

For processing Russian texts [1] by **TEX** one should adjust **TEX** to use: Russian language hyphenation, coding of the Russian characters, and fonts with the cyrillic symbols.

TEX 3.0 can be adjusted without changes!

The hyphenation patterns described in [2] are used for Russian language. Actually **TEX** is bilingual—the Russian and English hyphenation patterns are loaded by following file:

```
\language=0          % English
\lefthyphenmin=2
\righthyphenmin=3
\input ehyphen.tex
\language=1          % Russian
\lefthyphenmin=2
\righthyphenmin=2
\input cyrdef.tex
\input rhyphen.tex
\language=0          % English as default
```

In the file **cyrdef.tex** proper **catcode**, **uccode**, **lccode** and **mathcode** are set for cyrillic characters.

Switching between Russian and English hyphenation is performing by \language: setting \language=0 means English, \language=1 means Russian. English words are not hyphenated if the Russian patterns are active and reversely. Another possibility is to merge the English and Russian hyphenation patterns as a single language.

*T**EX* can use any 8-bit coding scheme for Russian characters—"alternative"¹, KOI-8², ISO 8859-5³, etc. The hyphenation patterns and .tfm files should correspond to the coding scheme being used. Russian *T**EX* works with virtual fonts; each of them consists of an original Computer Modern font (below the 128th code) and a font with cyrillic characters (above the 128th code). For *T**EX* the cyrillic characters are completely equal in "rights" with Latin characters. One could define new commands as Russian words! The simultaneous usage of cyrillic and Latin characters does not require any additional commands for switching or separating them.

For creating the .tfm file of a virtual font the program *tfmerge* was designed. It merges the .tfm for a Computer Modern font and .tfm for a cyrillic font into a virtual font .tfm and .vf in accordance with the coding scheme of the Russian characters.

The managing of the program *tfmerge* is performed by a file which contains the table of correspondence between the position of a character in a virtual font and the position of the same character in a cyrillic font. E.g., the correspondence between alternative coding into virtual fonts and "phonetic"-like coding (with swapped lower/upper case) into cyrillic fonts [3] is expressed by the following pairs:

```
\200:a
\201:b
\202:w
...
\360:Q
```

The left part of a pair (up to delimiter character ":") is the position in virtual fonts and the right part is the position in cyrillic fonts. The position can be specified by a symbol or by an octal number. When sorted by left parts the file becomes simpler:

```
\200:abwgdevzijklmnoprstufhc^{} \177yx|`q
\240:ABWGDEVZIJKLMNOP
\340:RSTUFHC^{}[_YX]\QQ
```

The correspondence between ISO 8859-5 coding in virtual fonts and the "phonetic"-like coding in cyrillic fonts is expressed by:

```
\260:abwgdevzijklmnoprstufhc^{} \177yx|`q
\320:ABWGDEVZIJKLMNOPRSTUFHC^{}[_YX]\QQ
```

The calling sequence of *tfmerge* program is:

```
TFMERGE [-d] cmr10 cmyr10 xcmr10
```

¹ Used mainly on the IBM PC, alphabetically ordered, almost identical to Microsoft's codepage 866.

² Used on some UNIX-like systems, based on "phonetic" correspondence between Latin and cyrillic characters.

³ Used on VAX/VMS, alphabetically ordered.

where *cmr10* and *cmyr10* are source Latin and cyrillic fonts. *xcmr10* is the virtual font, which will consist of two files: *xcmr10.tfm* and *xcmr10.vf*. They are accepted by the standard program *vftopl*.⁴

Note that such merging is correct because the cyrillic fonts are created by METAFONT on the base of the same setup files, like *cmr10.mf*. The font parameters are identical for Latin and cyrillic fonts.

The files are merged by the following couples:

<i>cmbx*</i>	<i>cmcbyn*</i>	<i>xcmbx*</i>
<i>cmbxsl10</i>	<i>cmcbyn10</i>	<i>xcmbxsl10</i>
<i>cmbxti10</i>	<i>cmcbyn10</i>	<i>xcmbxti10</i>
<i>cmbxsl10</i>	<i>cmcbyn10</i>	<i>xcmbxsl10</i>
<i>cmbxti10</i>	<i>cmcbyn10</i>	<i>xcmbxti10</i>
<i>cmmi5</i>	<i>cmyr5</i>	<i>xcmmi5</i>
<i>cmmi6</i>	<i>cmyr6</i>	<i>xcmmi6</i>
<i>cmmi*</i>	<i>cmcti*</i>	<i>xcmmi*</i>
<i>cmmib10</i>	<i>cmcbyn10</i>	<i>xcmmb10</i>
<i>cmr*</i>	<i>cmyr*</i>	<i>xcmr*</i>
<i>cmsl*</i>	<i>cmcs1*</i>	<i>xcmsl*</i>
<i>cmsl10</i>	<i>cmcs110</i>	<i>xcmsl10</i>
<i>cmss*</i>	<i>cmcss*</i>	<i>xcmss*</i>
<i>cmssbx10</i>	<i>cmcssbx10</i>	<i>xcmssbx10</i>
<i>cmssdc10</i>	<i>cmcssdc10</i>	<i>xcmssdc10</i>
<i>cmssi*</i>	<i>cmcssi*</i>	<i>xcmssi*</i>
<i>cmti*</i>	<i>cmcti*</i>	<i>xcmti*</i>
<i>cmtt*</i>	<i>cmctt*</i>	<i>xcmtt*</i>

Mathematical Italic fonts *cmmi** are merged with Cyrillic Text Italic fonts and proper *mathcode*'s have been set. One can use Russian letters in math.

To use another realization of the cyrillic fonts one should create the table of correspondences for *tfmerge* and select other couples for merging.

Files *plain.tex* for *T**EX* and *1fonts.tex* for *L**A**T**E**X* should be changed to substitute the references to the Latin font being merged by references to the proper virtual font.

For a VAX/VMS realization, creating .fmt files requires only setting the parameter *trie_size* to 16000. For SB30TEX (on MS-DOS) some .tfm files are not preloaded, because the size of the .tfm files is increased. For *EmT**EX* [3a] the options -i -o -8 -mt:12700 have been set.

The main problem with virtual fonts is that not all .dvi drivers can handle the virtual fonts. To avoid that problem the program *PosTeX* has been designed. It reads a .dvi file, expands the virtual fonts and writes a new .dvi file which is accepted by any .dvi driver.

There are some .dvi drivers which already accept the virtual fonts, e.g. from the fine collection

⁴ Really more comprehensive utilities are needed to create virtual fonts.

by Eberhard Mattes. One can use such .dvi drivers without PosTeX. But extensive usage of virtual fonts with a previewer will require more memory and increase the time before showing the first page. For a .dvi file which consists of a single cyrillic character, three files `xcmr10.vf`, `cmr10.pk` and `cmcyr10.pk` are opened instead of one file `cmcyr10.pk`, and total required memory is increased by one-third. This example only shows that extensive usage of virtual fonts will require more economic realization.

Another reason to use the program PosTeX is the portability of .dvi files. Our virtual fonts refer to local coding of Russian characters and immediately after TeXing a .dvi file is not portable. After being transforming by PosTeX a .dvi file becomes portable—it refers only to ordinary fonts.

It's also possible to enter Russian text using pure ASCII, for people who don't do much Russian TeXing, but need to set an occasional citation. In this case, control sequences can be used, and it is necessary to specify the boundary between Russian and non-Russian text to switch the hyphenation patterns. E.g., for printing this article in TUGboat the WNcyr realization of the cyrillic characters by Thomas Ridgeway has been used.

An integration of Russian into “International L^AT_EX” by Joachim Schrod is done by the following Russian.sty file:

```
\def\contentsname{Содержание}
\def\listfigurename{Список рисунков}
\def\listtablename{Список таблиц}
\def\abstractname{Аннотация}
\def\partname{Часть}
\def\chaptername{Глава}
\def\appendixname{Приложение}
\def\refname{Литература}
\def\bibname{Библиография}
\def\indexname{Алфавитный указатель}
```

```
\def\figurename{Рис.}
\def\tablename{Табл.}
\def\enclname{Вложение}
\def\ccname{Копия}
\def\headtoname{К:}
\def\pagename{Страница}
\def\today{\number\day\space\ifcase\month
\or января\or февраля\or марта\or апреля
\or мая\or июня\or июля\or августа
\or сентября\or октября\or ноября
\or декабря\fi\space\number\year}
\language=1
```

The macros \Alph and \alph are redefined too.

References

- [1] Barbara Beeton, *Mathematical symbols and cyrillic fonts ready for distribution (revised)*. TUGboat **6** (1985), no. 3, pp. 124-128.
- [2] Dimitri Vulis. *Notes on Russian TeX*. TUGboat **10** (1989), no. 3, pp. 332-336.
- [3] Н.Л. Глонти и др. *Метапроект кирилловского алфавита для печатающих устройств с высоким разрешением*: Препринт ИФВЭ. 90-66, Протвино, 1990.

- ◊ Basil Malyshev
Institute for High Energy Physics
142284, Protvino, USSR
malyshev@m9.ihep.su
- ◊ Alexander Samarin
Institute for High Energy Physics
142284, Protvino, USSR
samarin@vxcern.cern.ch
- ◊ Dimitri Vulis
CUNY Graduate Center
NY, USA
dlv@cunyvms1.bitnet

Serbo-Croatian Hyphenation: a **TeX** Point of View

Cvetana Krstev

On Serbo-Croatian

Serbo-Croatian is one of the South-Slavic languages. It is characterized, as other Slavic languages, by a rich morphology. A particular feature of the language is its almost fully phonological orthography, i.e. on a word level, one letter corresponds to each phoneme and vice versa. As a result, the written text practically represents a phonemic transcription of speech. Still, the Serbo-Croatian literary language has two main pronunciations, ekavian and jekavian, which reflect the different development of the pronunciation of the old Slavic sound **ѣ**. Sound **ѣ** is usually replaced by vowel **e** in ekavian dialect (for instance, **dete**, **mleko**, **večan**, **čovek**) while in jekavian dialect it is usually replaced either by two-syllable group **ije** (**dijete**, **mlijeko**) or by one-syllable group **je** (**vječan**, **čovjek**). Those differences in pronunciation are recorded in the written text. Accent has a distinctive role in Serbo-Croatian and as it is not marked in written texts there is a number of homographs.

Two alphabets are in use: Latin and Cyrillic. The Serbo-Croatian Latin alphabet is different from the English alphabet. Both letters with diacritics — **ć**, **č**, **ž**, **š**, **đ** — and digraphs — **dž**, **lj**, **nj** — are in use and they all have a separate place in the alphabet. The order of the Serbo-Croatian Latin alphabet is therefore as follows: **a, b, c, ć, č, d, dž, đ, e** and so on. As the letters **q, w, x** and **y** don't exist in the Serbo-Croatian alphabet, the total number of letters is 30. Transcription of foreign words and names is compulsory in Serbo-Croatian of ekavian pronunciation while jekavian pronunciation allows the orthography of the source language.

While all the letters with diacritics are assigned separate keys on the standardized national keyboard as well as the positions in the national version of 7-bit code [1, 2, 3], neither keys nor codes are provided for digraphs so they are input by striking two keys, i.e. by entering two codes. Besides that, although the standard provides a separate key for the letter **đ**, the keyboards of old typewriters often did not have it. As a result, this letter was—and sometimes still is—recorded as the digraph **dj**, in spite of orthographic rules.

Serbo-Croatian Cyrillic has the equivalent 30 letters but with neither diacritics nor digraphs. The

order of the letters in the Serbo-Croatian Cyrillic alphabet is completely different from the order in the Latin alphabet. The Serbo-Croatian Cyrillic alphabet is also different from the Russian alphabet as there are letters which do not exist in Russian Cyrillic: **Ђ, ј, Ђ, њ, Ѓ, ћ**, and vice versa, which is important as the Russian Cyrillic was the basis for the development of appropriate international coding standards.

The digraphs of the Serbo-Croatian Latin alphabet can cause problems when using formatting and typesetting programs, particularly for hyphenation and automatic transcription from the Latin to the Cyrillic alphabet. These problems can be caused by each combination — **lj**, **nj**, **dž** and **dj** — which in the text may represent both digraphs and consonant clusters. A digraph is always transcribed into one Cyrillic letter and is never hyphenated. For instance, **nadžak-baba** is transcribed into **наџак-баба** and in both cases is hyphenated as **na-džak-ba-ba**. On the other hand, a consonant cluster is always transcribed into two Cyrillic letters and can, in principle, be hyphenated. For instance, **nadživeti** is transcribed into **наџивети** and is hyphenated as **nad-ži-ve-ti**.

Serbo-Croatian Hyphenation Rules

Several sets of hyphenation rules for Serbo-Croatian were proposed on different occasions [4, 5], but to none of them did linguists give unqualified support. Thus, the *Serbo-Croatian Orthography Book* [6] only gives the recommendations on how to hyphenate words, avoiding formulating precise rules. These recommendations can briefly be described as follows [7]:

- (a) Two adjacent vowels should be divided, but it is not wrong if they are not. For instance, **ža-oka** or **žao-ka**.
- (b) It is not allowed to carry over to the next line two or more final consonants without a vowel. For instance, not **mlado-st** but **mla-dost**.
- (c) If there is only one consonant between two vowels, the consonant belongs to the second vowel and it is carried over with it to the next line. For instance, not **bor-ac** but **bo-rac**.
- (d) If there are two or more consonants between two vowels, only those consonants that can be easily pronounced with the vowel that follows can be carried over to the next line (for instance, **zem-mlja** but also **zem-lja**). On the other hand, it is not recommended to carry over to the next line

a consonant cluster which is difficult to pronounce (for instance, not **bu-mbar** but **bum-bar**).

(e) If the constituent parts of a compound word can be distinguished, the break point is between those parts and each part is further hyphenated as if it were a separate word (for instance, **raz-vuči** and **raz-oružati**). If those parts can't be distinguished, the word is hyphenated as if it were not compound (for instance, not **raz-um** but **ra-zum**). In both the examples the words are formed of prefix (**raz-**) and stem, but in the first case this prefix can be distinguished while in the latter case it can't be recognized any more, and the word is hyphenated according to the previous rules ((a)-(d)).

Although it follows from this rule that the recognized prefix can be further hyphenated as a separate word, it is considered a good typographic practice not to hyphenate a polysyllabic prefix. For instance, the word **novootvoren** with prefix **novo-** should be hyphenated **novo-o-tvo-ren** rather than **no-vo-o-tvo-ren**.

The recognition of vowels is fundamental for hyphenation in Serbo-Croatian, as hyphen positions often coincide with syllable boundaries and syllables are formed around the central phonemes which are usually vowels. The Serbo-Croatian alphabet, both Latin and Cyrillic, has five vowels: **a, e, i, o, u**. However, the phoneme **r** can take on the role of a vowel and be a central phoneme of a syllable in the following circumstances:

- in interconsonantal position (**svr-stavanje**);
- when preceding a consonant, at the beginning of a word (**r-vanje**);
- when following a vowel, in compounds (**po-r-vati se**).

Besides that, the sonants **r, l, m** and **n** when preceded by a consonant, at the end of a word, also behave as vowels (**ma-sa-kr** and **bi-ci-kl**). Marginal phonemes are consonants. Two types of syllables can be distinguished: open syllables, with the structure **-V-** or **-CV-**, where **V** is any vowel in the sense described above and **C** is any consonant, and closed syllables, with any other structure.

The application of recommendations (a)-(c) is almost self-evident and beyond questioning. Recommendation (d) refers to the identification of closed syllables which means that the consonant cluster has to be divided. Still, the *Serbo-Croatian Orthography Book* gives no strict rules for the division of consonant clusters, but only introduces the intuitive notions of consonant clusters that are easy

or difficult to pronounce and illustrates them by few examples.

A semantic criterion, whose automatic implementation can be difficult to achieve, is introduced into hyphenation by recommendation (e). For example, **ob-** is a prefix in the word **ob-istiniti** but not in the word **o-bi-čaj**. The particular problem here is to decide whether the constituent parts of a compound word can be distinguished. The *Serbo-Croatian Orthography Book* gives no hints on how to make such a decision (for instance, whether the parts in the word **preduzeti** can be recognized). An additional problem to the application of this recommendation is posed by homographs. For instance, in the word **podići**, **podidem** the prefix is **pod-** while in the word **podići**, **podignem** the prefix is **po-**.

Definition of Hyphenation Rules

Research into some aspects of consonant clusters in Serbo-Croatian has been undertaken before [8], but their occurrences in contiguous text have not been investigated. It was thus necessary to establish which consonant clusters do occur in Serbo-Croatian in order to state precisely the notions of easy- and difficult-to-pronounce consonant clusters. For that purpose, the analysis of consonant cluster occurrences has been undertaken that was based on the corpus of modern Serbo-Croatian texts of ekavian dialect [9]. In addition to frequency dictionaries that take into account the position of a consonant cluster in a word — initial, final or medial — results were obtained pointing out the occurrences of consonant clusters **dj, dž, nj** and **lj** as well as of those occurring only at the junction of the prefix and the word stem.

This analysis has made it possible to formulate the following rules for consonant cluster division in Serbo-Croatian:

I Binary consonant clusters.

(a) Two consonants are carried over to the next line ($-C_1 C_2 V$) only if they are usual at the beginning of a word in Serbo-Croatian, that is, if $C_1 C_2$ belongs to one of the following sets:

1. $C_1 \in \{s, z, š, ž\}$ (C_1 is a fricative) and
 C_2 is any consonant; or
2. $C_1 = m$ and $C_2 \in \{n, l, r\}$; or
3. $C_1 = v$ and $C_2 \in \{l, r\}$; or
4. $C_1 \notin \{r, l, lj, v, j, m, n, nj\}$ and
 $C_2 \in \{r, l, lj, v, j, m, n, nj\}$ (C_2 is a sonant).

(b) In all other cases, one consonant must be left in the current line ($C_1-C_2 V$).

II Three-member consonant clusters.

(a) Three consonants are carried over to the next line ($-C_1C_2C_3V$) only if they are usual at the beginning of a word in Serbo-Croatian (for instance, **izdajni-štvo**), that is if

- $C_1 \in \{s, z, š, ž\}$ and
- C_2 is any consonant, and
- $C_3 \in \{r, l, lj, v, j\}$ (C_3 is a sonant);

(b) Two consonants are carried over to the next line ($C_1-C_2C_3V$) only if two consonants C_2C_3 are usual at the beginning of a word in Serbo-Croatian (for instance, **rot-kva**—see Ia).

(c) One consonant is carried over to the next line ($C_1C_2-C_3V$) only if the two consonants that remain in the current line (C_1C_2) are usual (or possible) at the end of a word in Serbo-Croatian (for instance, **funk-cija**), which means that C_1C_2 belongs to one of the following sets:

1. $C_1 \in \{r, l, lj, v, j, m, n, nj\}$ (C_1 is a sonant) and
 C_2 is any consonant; or
2. $C_1 \in \{s, z, š, ž, f, h\}$ (C_1 is a fricative) and
 $C_2 \notin \{r, l, lj, v, j, m, n, nj\}$ (C_2 is not a sonant); or
3. $C_1C_2 \in \{ps, bz, ks, gz, pt, bd, kt, gd\}$.

III Four-member consonant clusters.

(a) Three consonants are carried over to the next line ($C_1-C_2C_3C_4V$) only if these three consonants are usual at the beginning of a word in Serbo-Croatian (see IIa). For instance, **demon-stracije**.

(b) Two consonants are carried over to the next line ($C_1C_2-C_3C_4V$), only if the two consonants which are left in the current line are usual at the end of a word in Serbo-Croatian (see IIc) and if the two consonants which are carried over to the next line are usual at the beginning of a word in Serbo-Croatian (see Ia). For instance, **student-ski**.

Moreover, it should be stressed that no four-member consonant cluster was identified on the analyzed corpus that would not have the structure (a) or (b).

In order to enable the implementation of recommendation (e) given by the *Codex* it was necessary to analyze the use of prefixes in Serbo-Croatian. For that purpose a list of 79 common Serbo-Croatian prefixes of 2 or more letters in length was composed on the basis of traditional Serbo-Croatian grammar textbooks, consisting of 52 basic prefixes and 27 phonologically altered prefixes that occur as a result of phonological changes at the junction between the prefix and the word stem. For instance, the basic

prefix **iz-** has three altered prefixes: **is-** (in front of **p, t, k, f, c, h**), **iž-** (in front of **d** and **dž**) and **iš-** (in front of **č** and **č**).

The occurrences of all these prefixes were analyzed on the corpus of modern Serbo-Croatian texts of ekavian dialect [10]. For each prefix string, it was thus possible to establish its productivity and its ability to be combined with other prefixes. Also, such prefix strings were identified that are prefixes in some, but not all, cases (for instance, **ob-** is a prefix in **obuhvatiti** but not in **običaj**). Finally, instances were identified where the prefix is not the longer but the shorter prefix string (for instance, **naj-** is the prefix in **najelegantnija** while in **najednom** the prefix is **na-**).

The Serbo-Croatian prefixes were, exclusively for the purpose of hyphenation, classified on the basis of this analysis in such a way that the recognition of prefixes of the same group requires the same conditions. The prefix types are as follows:

1. **The break point is always after a prefix string of type 1.** A prefix was assigned type 1 either because all instances of the prefix string in the corpus were prefixes or because it never occurred. For all the prefix strings that never occurred in the corpus the additional check was done in the Serbo-Croatian dictionary that supported the decision to categorize them as type 1. For instance, the prefixes of type 1 are **anti-** (**antihrist**)—always a prefix—and **iž-** (**izdžikljati**)—never occurred.
2. **The break point is after a prefix string of type 2, except when a difficult-to-pronounce consonant cluster follows it.** A prefix falls into this group if it is monosyllabic and ends with a vowel. The prefix **za-** is of type 2 (**za-svoditi**, but **zač-koljica** because **čk** is a difficult-to-pronounce consonant cluster, that is, condition Ia is not satisfied).
3. **The break point is after a prefix string of type 3, except if it is followed by a vowel, when additional information is needed.** All prefixes of type 3 end with a consonant. If a prefix string of type 3 is followed by a consonant, it cannot be wrong to break the word after it, because the prefix string without that last consonant can't be a prefix for various reasons. For instance, **eks-** is a type 3 prefix because **ek-** is not a prefix at all. On the other hand, **naj-** is a type 3 prefix although **na-** can be a prefix, but as no initial consonant cluster beginning with consonant **j** exists, **na-** can't be a prefix when the prefix string **naj-** is followed by a consonant. On the other hand, **eks-** and **naj-**

are not prefixes in the words *ekser* and *najaviti*, respectively.

4. Additional information is always necessary to decide whether a prefix string of type 4 is a prefix. Prefix strings of type 4 are polysyllabic. If we consider the break point after the prefix to have a higher priority than other possible break points and if, on the other hand, we do not want to miss any break point, we can never tell beforehand if the first break point in a word is after a prefix string of type 4. For instance, *polu-* is a prefix in the word *polu-pismen* while in the word *polupati* the prefix is *po-*. In the former case the break point after *po-* is possible but should be avoided due to typographical conventions, while in the latter case the break points after *po-* and *polu-* have equal value.

5. Additional information is always necessary to decide whether a prefix string of type 5 is a prefix, except when its final consonant and consonants that follow form a difficult-to-pronounce consonant cluster. Prefixes of type 5 end with a consonant and for all of them the prefix string without the final consonant can also be a prefix. We can be sure that a word can be hyphenated after a longer prefix string only if the final consonant and consonants that follow form a difficult-to-pronounce consonant cluster, no matter whether that prefix string is really a prefix. For instance, the word *ob-zidati* must be hyphenated after *ob-* because *bz* doesn't satisfy the condition Ia (and *ob-* is actually a prefix). On the other hand, *ob-* is a prefix in *ob-lizati* while in *o-bližnji* the prefix is *o-*.

6. The additional information is always necessary to decide whether a prefix string of type 6 is a prefix, except when its final consonant and consonants that follow form the difficult-to-pronounce consonant cluster. Prefixes of type 6 end with a consonant and are followed by a consonant, as they are all phonologically altered prefixes. Also, for all of them a prefix string without the final consonant can also be a prefix. As for the prefixes of type 5, we can be sure that the longer prefix string is a prefix in a word only if the final consonant and consonants that follow form a difficult-to-pronounce consonant cluster. For instance, the word *is-psovati* must be hyphenated after *is-* as *sps* isn't an initial consonant cluster in Serbo-Croatian. On the other hand, *is-* is a prefix in *is-kititi* while in *i-skakati* the prefix is *i-*.

Production of Hyphenation Patterns

For Serbo-Croatian, as for some other languages—French and Polish, for example—dictionaries that indicate the hyphenation break points for all the entries do not exist [12, 13]. Thus, in order to apply the PATGEN program [14] for the automatic generation of a pattern dictionary to be used with the typesetting system *TEX* [15], it would be necessary to include that information in some machine-readable dictionary of Serbo-Croatian. But that would not be enough as this dictionary would have to contain, in addition to the usual dictionary entries, all the derived forms. As Serbo-Croatian is a language with a very rich morphology, this extended dictionary would be at least ten times larger than the original one, and it would be time-consuming and erroneous to prepare it for PATGEN.

For this reason, the decision was made to produce the pattern dictionary “by hand”, founding the generation on the precise hyphenation rules described in previous sections and checking the obtained results on the sample words extracted from the existing Serbo-Croatian paper dictionaries. In this section will be described the procedure which generates the pattern dictionary to be used for Serbo-Croatian texts of the ekavian dialect that use digraphs *dj*, *lj*, *nj* and *dž* encoded as two separate codes [11].

(a) The break point is between two vowels (recommendation (a) of the *Orthography Book*). From this rule, 25 patterns were generated of the form

$$V1V,$$

where *V* belongs to the set {*a*, *e*, *i*, *o*, *u*} of “real” vowels.

(b) The break point is before a consonant surrounded by two vowels (recommendation (c) of the *Orthography Book*). From this rule, 105 patterns were generated of the form

$$1CV,$$

where *C* belongs to the set $\mathcal{A} \setminus \{a, e, i, o, u, dj, nj, lj, dž\}$, where \mathcal{A} denotes Serbo-Croatian alphabet.

(c) As the analysis of consonant cluster occurrences showed, the strings *dj*, *nj*, *lj*, *dž* are much more frequently digraphs than consonant clusters. Therefore, the following four patterns were produced which disable the division of digraphs:

$$1d2j \quad 1n2j \quad 1l2j \quad 1d2ž$$

The same analysis also showed that the consonant clusters *dj*, *nj*, *lj* and *dž* occur only at the junction of a prefix with a stem, so these cases will be covered in items (j)–(o). At this point, only 4 patterns were

added for the identification of prefixes *in-* and *kon-* which had not been included in the prefix analysis as they are not usual in Serbo-Croatian:

.in3jekc	.in3junkt
.ko2n3jug	.ko2n3junk

(d) The break point is not between two consonants of a binary consonant cluster if this consonant cluster is usual at the beginning of Serbo-Croatian words (rule Ia for consonant cluster division). From this rule, 205 patterns were generated of the form

$$1C_1C_2$$

where C_1 and C_2 are such consonants that C_1C_2 satisfies rule Ia.

Having in mind that consonant clusters *mnj* and *v1j*, whose existence is confirmed on the corpus, do not satisfy rule Ia, in contrast to *mn* and *v1*, two more patterns were introduced:

$$2m3nj \quad 2v3lj$$

The patterns of form (c) function as desired if a digraph is followed by a vowel. On the other hand, if the digraph is followed by a consonant, the break point before the digraph has to be disabled in all the cases where the resultant consonant cluster does not satisfy rule Ia. So, 11 more patterns were added:

2ljn	2ljk	2ljs	2ljš
2ljc	2ljd	2ljb	2njs
2njc	2djs	2dž3b	

The generation of these patterns was based on the results of the analysis of consonant cluster occurrences.

(e) The phoneme *r* between two consonants behaves as a vowel, which means that the break point can, in principle, be after *r*. Analysis of the occurrences of the phoneme *r* in interconsonantal position, as part of the analysis of consonant cluster occurrences, showed that only 106, of 576 possible strings of the form C_1rC_2 , were actualized. Besides that, the actualized strings had such a form that the pattern $1C_12r$ was generated in step (d) while pattern $1r2C_2$ was not generated at all. When pattern $1C_12r$ is applied to string C_1rC_2 , the obtained result is

$$1C_12rC_2$$

and that is precisely the pattern necessary to identify the phoneme *r* in interconsonantal position — no new patterns need to be generated.

(f) The break point is before a three-member consonant cluster that is usual at the beginning of Serbo-Croatian words (rule IIa). It may seem that the implementation of this rule requires the generation of 460 patterns of the form

$$1C_12C_22C_3$$

However, analysis of consonant cluster occurrences showed that C_2 is actualized only as a plosive consonant (*p*, *b*, *k*, *g*, *t*, *d*), the fricative *f*, the affricate *c* or *č*, or the sonant *m* or *v*. That means that patterns of the form $1C_12C_2$ and $1C_22C_3$ have already been generated in step (d) as $C_1 \in \{s, š, z, ž\}$, $C_2 \notin \{r, l, lj, j, n, nj\}$ and C_3 is a sonant. When these two patterns are applied, the necessary pattern $1C_12C_22C_3$ is obtained. This means that for implementation of this rule no new patterns need to be generated.

(g) The break point in a three-member consonant cluster that is not usual at the beginning of Serbo-Croatian words is between the first and the second consonant only if the two last consonants of the cluster are usual at the beginning of Serbo-Croatian words (rules IIb and IIc). The three-member consonant cluster $C_1C_2C_3$ that can match no pattern of form (f), while C_2C_3 cannot match any pattern of form (d) either, will be divided as $C_1C_2-C_3$, because C_3V matches the pattern of form (b). This means that for the implementation of these rules no new patterns need to be generated either.

However, the analysis of consonant cluster occurrences, as well as the interactive checking of the correctness of the generated patterns, showed that there are some cases when it is better to divide a three-member consonant cluster as $C_1C_2-C_3$, although C_2C_3 is usual at the beginning of Serbo-Croatian words, which means that pattern $1C_22C_3$ exists. Those cases occur mainly in the words of foreign origin and very often in derivative forms. For instance, it is better to hyphenate *konkursni* as *konkurs-ni* than as *konkur-sni*, or *tekstil* as *teks-til* than as *tek-stil*. In order to cover these cases too, 6 more patterns were generated:

ur2s3n	k2s3t	k2t3n
12t3n	n2t3n	or2f3n

(h) The majority of four-member consonant clusters in Serbo-Croatian appear at the junction of a word stem with suffixes *-sk(i)* and *-stv(o)*. Namely, the analysis of consonant cluster occurrences showed that only 3 of the 22 four-member consonant clusters that were identified on corpus did not have the form *CCsk* or *Cstv*. The breaks between the stem and the suffix will thus be provided by the patterns of form (d) and (f) respectively. The remaining identified consonant clusters are also correctly handled, as they match the patterns of type (f) or the appropriate patterns for prefix recognition: *nstr* in *demon-stracija* is controlled by the pattern of type (f) while *kspl*

in *eks-plozija* and *jstr* in *naj-strašniji* are controlled by the patterns of form (l) for prefixes *eks-* and *naj-* respectively.

(i) The break point cannot be before the final consonants (recommendation (b) of the *Orthography Book*). The patterns of form (b) provide that at least one vowel is carried over to the next line. However, the patterns of form (d) and (f) that match the consonant clusters satisfying the conditions Ia and IIa respectively would allow the break points before these consonant clusters at final positions as well. The analysis of consonant cluster occurrences showed that only 4 of the 33 final consonant clusters that were identified on corpus match the patterns of form (d). In order to overcome this problem it would thus be necessary to add 4 more patterns:

2st. 2sl. 2st. 2kl.

But as the hyphenation routine of *TEX* hyphenates the words in a way that enables at least three letters to be carried over to the next line, these patterns were not included in the pattern dictionary for Serbo-Croatian.

Aside from these final clusters, the special case of the phonemes *r*, *m*, *n* and *l* at the end of a word behind a consonant should be mentioned. An example is the word *masakr*, in which the final *r* behaves as a vowel. But the final consonant cluster *kr* matches the pattern of form (d), which means that this word would be correctly hyphenated: *ma-sa-kr*.

Before we continue to describe the generation of patterns that control the hyphenation at the junction of prefixes and word stems, it should be noted that the pattern dictionary produced thus far has 362 patterns.

(j) Prefix strings of type 1 are always prefixes. In other words, the break point is always after a prefix string of type 1. For the 27 prefixes of type 1, 36 patterns were generated which make break points after the prefix strings possible. For instance, for the prefix *anti-* only one pattern was generated, *.an2ti*, while for the prefix *beš-*, which is a variant of the prefix *bez-* that is realized in front of *č* and *ć*, two patterns were generated: *.be2š3č* and *.be2š3ć*. On the other hand, for the prefixes *nat-*, *op-*, *ot-*, *pot-* and *pret-*, which are also variants of the basic prefixes, no pattern was generated as the hyphenation after these prefixes is controlled by the patterns for consonant cluster division (steps (a)-(i)).

(k) The question whether a prefix string of type 2 is a prefix in a word or not, does not influence

word hyphenation. So, for the 19 prefixes of type 2 no pattern was generated.

(l) The break point is after a prefix string of type 3, except when it is followed by a vowel when additional information is needed. The generation of patterns for prefixes of this type will be illustrated on the example of the prefix *naj-*, which has a high frequency in Serbo-Croatian as it is used to express the superlative of adjectives. First of all, the prefix *.na2j3* is generated. Then, the cases when *naj-* in front of a vowel is not a prefix have to be covered, because *naj-* can, in principle, be a prefix for all Serbo-Croatian adjectives that begin with a vowel. So, for instance, the patterns *.na3j4av* (for words as *na-ja-vi-ti* or *na-jav-lji-va-či-ca*) and *.na3j4el* (for words as *na-je-la* or *na-je-lo*) are generated. Finally, as the adjectives *avetinjski* and *elementaran*, which have superlatives exist, the patterns *.na4j5avet* and *.na4j5elem* had to be generated. In that way, 19 patterns were generated for the prefix *naj-* and a total of 108 patterns for the 9 prefix strings of type 3.

(m) Additional information is always necessary in order to decide whether a prefix string of type 4 is a prefix in a word or not. It should be remembered that all prefixes of type 4 are polysyllabic and, despite the recommendation given by the *Orthography Book*, it is usually considered better not to hyphenate the prefix itself. Patterns should, therefore, prevent the hyphenation of a prefix string in the cases when it is a prefix. The generation of patterns for prefixes of type 4 will be illustrated on the example of the prefix *preko-*, that occurs, for example, in words *prekomerno* or *prekosutra*. However, the analysis of occurrences of the prefix string *preko-* in the corpus, as well as a lookup in Serbo-Croatian dictionaries, showed that the instances where *pre-*, rather than *preko-*, is a prefix, are more frequent. Some examples are *pre-koračiti*, *pre-kositi*, and *pre-komandovati*. Because of that, only the patterns that provide the correct hyphenation in the cases when *preko-* is prefix were entered in the pattern dictionary. For instance, for the examples given above two patterns were generated: *.pre2kome* and *.pre2kosu*. Similarly, 9 patterns were generated for the prefix *preko-*, and a total of 59 patterns for the 14 prefix strings of type 4.

(n) The break point is compulsory after a prefix string of type 5 only if the last consonant of the prefix string and the initial consonants of the word stem form a difficult-to-pronounce consonant cluster. In all other cases, additional information is needed. The generation of patterns for prefixes

of type 5 will be illustrated on the example of the prefix **od-**, that occurs, for instance, in words **odraditi** or **odsvirati**. In view of the fact that the analysis of occurrences of this prefix string in the corpus, as well as a lookup in Serbo-Croatian dictionaries, showed that it is a very frequent prefix, pattern **.od3** was entered in the pattern dictionary. However, in front of the vowel **i** the prefix string **od-** very often is not a prefix (for instance, in words **odignuti** or **odista**), so the pattern **.od4i** is generated. At the end, in the word **odigrati** the prefix string **od-** is a prefix, so the pattern **.od5igr** is added. In that way, 35 patterns were generated for the prefix **od-**, and a total of 151 patterns for the 6 prefixes of type 5.

(o) Prefix strings of type 6 are similar to the prefix strings of type 5, except that, being variants of the basic prefixes, they can be prefixes only if they are followed by a consonant of a certain kind. The generation of patterns for the prefixes of type 6 will be illustrated on the example of the prefix **ras-**, which occurs, for instance, in words **rastumaciti** or **rascvetati**. The prefix **ras-** is a variant of the prefix **raz-** that emerges from the substitution of the voiced consonant **z** by its unvoiced counterpart **s** in front of the unvoiced consonants **p, k, t, f, c** or **h**. Thus, for the prefix **ras-** we introduce the patterns

.ra2s3p	.ra2s3k	.ra2s3t
.ra2s3f	.ra2s3c	.ra2s3h

However, **ras-** is not always a prefix in front of these consonants. Moreover, in words **rastaviti** and **rastezati**, for instance, the prefix is **ra-** and therefore the patterns **.ra3s4ta** and **.ra3s4te** are added. At the end the pattern **.ra4s5tanj** is generated because the word **rastanjiti** has the prefix **ras-**. Similarly, for the prefix **ras-** 20 patterns were generated, and a total of 82 patterns for the 4 prefixes of type 6.

(p) Combinations of two, rarely three, prefixes can be identified in Serbo-Croatian. Examples of the latter case are, for instance, **is-po-raz-boljevati** or **po-iz-o-stavljati**. For the solution of the hyphenation problem, only those combinations are interesting for which additional information is needed to identify the second prefix. The analysis of occurrences of a combination of prefixes in the corpus, as well as a lookup in Serbo-Croatian dictionaries, showed that besides the very frequent prefixes **naj-**, which expresses the superlative of adjectives, and **ne-**, which expresses the negative form of nouns, adjectives and adverbs, prefixes that combine with other prefixes are **o-, po-, pro-, za-** and **novo-**.

The generation of patterns needed to identify the combination of prefixes will be illustrated on the example of the combination **o-bez-**, which occurs in the words **obezvrediti** and **obezglaviti**, while in the word **obezubiti** the combination **o-be-** appears. Bearing in mind that patterns **.ob3** and **.ob4e** were generated in step (n), two patterns, **.obe2z3v** and **.obe2z3g**, were added for the examples given above.

To the prefixes **naj-** and **ne-** a partial solution was applied. For prefix **naj-** only the patterns that correspond to the combinations of the prefix **naj-** with prefixes that participate in adjectival derivation were generated. Similarly, for the prefix **ne-** only patterns that correspond to the combinations of the prefix **ne-** with prefixes that participate in adjectival, adverbial and nominal derivation were generated. At the same time, for the second prefix in each combination, only the patterns that "reflect the rule" were taken into consideration. For example, for the combination **naj-bez-** (for instance, in words **najbezgrešniji** and **najbezvoljniji**) only one pattern was generated **.najbe2z3** while the other patterns generated for the recognition of the prefix **bez-** were not taken into account. In this manner, for the recognition of the combination of prefixes 90 patterns were generated.

(r) The pattern dictionary was amended with an exception dictionary which contained only seven words. The words were added to the exception dictionary for one of these two reasons:

- The pattern dictionary would have to be expanded with a pattern that matches only one word. Such is the case of the undeclined word **po-dne** which due to the pattern **.po2d3** generated in step (o) wouldn't otherwise have any break point.

- The word is a homograph and its break points depend on the meaning. For instance, the word form **uzori** can be the nominative plural of the noun **uzor**, in which case the break points are **u-zo-ri**, or the second person singular of the imperative of the verb **uzorati**, which has a prefix **uz-**, and in that case the word should be hyphenated as **uz-ori**. Such a word forms are added to the exception dictionary to suspend all the break points.

The complete pattern dictionary has 888 patterns. The highest coefficient that appears in some pattern is 5. The analysis of occurrences of prefixes of a particular type in the corpus showed that the 362 patterns generated in steps (a)-(i) and which reflect the hyphenation rules can be expected to

provide the breaks for approximately 97% of the words in some document.

Conclusion

TEX has been in steady use at the Faculty of Mathematics at the University of Belgrade for several years now. The Latin alphabet is predominant, but Cyrillic has been used too. However, the Serbo-Croatian version of *TEX* has not been produced yet, in the sense that there are no font tables, neither for the Latin nor for the Cyrillic alphabet, that would reflect the national standard 7-bit codes. This means that for the Latin alphabet the commands `\v` and `\'` are used to set the diacritics. It is well known that words containing such diacritics can't be hyphenated by *TEX*. For this reason, the generated pattern dictionary has still not been included in any *TEX* implementation.

The validity of the generated pattern dictionary was, thus, tested with the command `\showhyphens`. As a Serbo-Croatian dictionary in machine-readable form was not available, chosen words found in corpus and traditional dictionaries were used as the arguments of this command. All the words were coded using only the letters of the English alphabet. The words were chosen in a way to reflect most of the problems of digraphs, consonant clusters and prefix recognition. On the basis of this test it can be said that the generated pattern dictionary provides word hyphenation according to the formulated hyphenation rules. However, the undertaken test can, by no means, be considered exhaustive enough and the true validity of the produced pattern dictionary will be confirmed only through regular use.

There are some aspects of Serbo-Croatian hyphenation that have not been covered by the performed analysis. First of all, the problem of compound words is still unsolved. As in Serbo-Croatian many compound words are formed by inserting the vowel `o-/-e-` between the constituent parts (for example, `glavo-bolja` or `gluvo-nem`), all those words will be correctly hyphenated by the existing rules. As for the compound words that are formed simply by connecting the constituent parts, many of them will be correctly hyphenated, as `krompir-čorba` or `star-mali`.

In the end, it should be stressed that this pattern dictionary was generated for use with the Latin alphabet which uses the four digraphs `dj`, `lj`, `nj` and `dz`. If the letter `đ` were used instead of the digraph `dj`, the dictionary would have to be extended. First of all, patterns of form (b) and (d)

should be added: for example, patterns `1da`, etc. and `1dr`, etc. Also, in all patterns for prefix recognition every occurrence of the string `dj` in which `d` and `j` are not separated by a digit should be replaced by `đ`. For instance, the pattern `.na4d5redj` should be replaced by `.na4d5red`. It should be noted that during the generation of patterns for prefix recognition, the digraphs were always treated as separate letters. For instance, two patterns were generated — `.na4d5red` and `.na4d5redj` — although one pattern, `.na4d5red`, would have been enough. This strategy would facilitate the replacement of a digraph with a separate letter. Of course, all the patterns that refer to the digraph `dj` could be deleted, but that is not necessary at all. Namely, for the hyphenation routine of *TEX*, the letters `đ` and `dj` could be considered as two different letters that can occur in the same document, which sometimes even happens.

If the Cyrillic alphabet were used instead of the Latin alphabet, then the same procedure that was suggested for the replacement of the digraph `dj` by the letter `đ` should be applied to the letters `љ`, `њ` and `Џ`. Of course, the patterns that refer to digraphs should be deleted as they would have no meaning any more.

It should be stressed once again that the obtained results apply only to Serbo-Croatian of ekavian dialect as some crucial results are based on the analysis of corpus of texts of ekavian dialect. However, besides some small lexical differences, the differences between the two dialects are mostly the result of the different pronunciation of the sound `ž`. As a result, in jekavian dialect some new consonant clusters may be introduced: for instance, `cvjetova` (jekavian) *vs.* `cvetova` (ekavian) or `snjegova` (jekavian) *vs.* `snegova` (ekavian). According to hyphenation rule Ia, most of those consonant clusters wouldn't be hyphenated which is in this case appropriate. Nevertheless, the produced pattern dictionary should be applied to the texts of jekavian dialect only with due precaution.

Version 3.0 of *TEX* introduces some new possibilities. For example, with *TEX* 3.0 the user can specify the smallest number of letters that may be left in a current line and the smallest number of letters that may be carried over to the next line. For Serbo-Croatian this is an important improvement, as the old restrictions lead to situations where many 6-letter words with 2 legal break points would not be hyphenated (for instance, `u-sko-ro`) and even some 7-letter words, as `u-stre-li`. If some user would want to set the smallest number of letters that may be carried over to the next line to 2, the

only change that has to be made in a pattern dictionary is the addition of the 4 patterns generated in step (i). The further reduction of the limits would, however, require the detailed check of all patterns, especially those for the prefix recognition.

Acknowledgement

I want to express my gratitude to Dr. Janusz Bień, from the Institute of Informatics at Warsaw University, for all the support and information he gave me during my work as well as for the careful reading of the manuscript. I also owe a lot to Prof. Jacques Désarménien from "Laboratoire de typographie informatique" at the University Louis-Pasteur, Strasbourg, whose work in solving the hyphenation problem for French guided me during my research. In addition, I thank Prof. Ljubomir Popović, from the Faculty of Letters at Belgrade University, who helped me define the hyphenation rules for automatic application.

References

- [1] International Organization for Standardization, *Information processing—ISO 7-bit coded character set for information interchange*, ISO 646, (1983)
- [2] Savezni zavod za standardizaciju, *Skup znakova za razmenu podataka kodiranih sa 7 bitova za srpskohrvatsko latinično pismo*, [7-bit coded character set for Serbo-Croatian Latin alphabet for information interchange], JUS I.B1.002, (1986)
- [3] Savezni zavod za standardizaciju, *Jedinice za unos podataka—Tastatura sa 47 tipki za slovenačko i hrvatsko latinično pismo*, [Data entry units—Keyboards 47 keys for Slovenian and Croat Latin alphabet], JUS I.K1.002, (1986)
- [4] Belić, A., *Pravopis srpskohrvatskog književnog jezika*, [Standard Serbo-Croatian Orthography Book], Belgrade, pp. 15–18, (1934)
- [5] Stevanović, M., *Savremeni srpskohrvatski jezik*, [Contemporary Serbo-Croatian Language], Belgrade, pp. 152–156, (1964)
- [6] *Pravopis srpskohrvatskog književnog jezika*, [Standard Serbo-Croatian Orthography Book] Novi Sad – Zagreb, 130–131, (1960)
- [7] Vitas, D., *Podela na slogove srpskohrvatskih reči*, [Syllable Division of Serbo-Croatian Words], Informatika, Ljubljana, 3/1981
- [8] Tolstoja, S. M., *Načal'nye i konečnye sočetanija soglasnyh v serbsko-hrvatskom jazyke*, [Initial and Final Consonant Clusters in Serbo-Croatian], Issledovaniya po serbsko-hrvatskom jazyku, Moscow, pp. 3–38, (1972)
- [9] Krstev, C., *Frekvencijski rečnik konsonantskih grupa u srpskohrvatskom jeziku i problem rastavljanja na slogove*, [A Frequency Dictionary of Consonant Clusters in Serbo-Croatian and the Problem of Syllabification], Proceedings of the 2. Scientific Meeting "Computer Processing of Linguistic Data", Institute "Jožef Stefan", Bled, pp. 389–404, (1982)
- [10] Krstev, C., *Rastavljanje reči srpskohrvatskog jezika na kraju retka*, [Hyphenation of Serbo-Croatian words at the end of the line], Proceedings of the 3. Scientific Meeting "Computer Processing of Linguistic Data", Institute "Jožef Stefan", Bled, pp. 289–301, (1985)
- [11] Krstev, C., *Programski sistemi za obradu teksta*, [Text Processing Systems], Master Thesis, Faculty of Mathematics, Beograd, (1989)
- [12] Désarménien, J., *French hyphenation by computer: application to TEX*, Technology and Science of Informatics, Gauthier-Villars & John Wiley & Sons, Vol. 6, No. 1. (1987)
- [13] Kolodziejska, H., *Dzielenie wyrazów polskich w systemie TEX*, [Hyphenation of Polish words by TEX], Report No. 165, Institut of Informatics, Warsaw University, (1987)
- [14] Liang, F. M., *Word Hy-phen-a-tion by Computer*, Ph. D. Thesis, Department of Computer Science, Stanford University, Report No. STAN-CS-83-977, (1983)
- [15] Knuth, D. E.: *The TEXbook*, Addison-Wesley, Reading, Mass., (1984)

◊ Cvetana Krstev
 Computer Laboratory
 Faculty of Sciences
 Studentski trg 16
 11000 Belgrade
 Yugoslavia
 Bitnet: xpmf101@yubgss21

On TeX and Greek...

Yannis Haralambous

1 Some historical background

The first person who made greek characters for TeX was D. E. Knuth; each CM font contains the greek uppercase characters. These are excellent for greek text as well as for mathematics. Only two remarks could be made: first, the Υ is actually a calligraphic form of Y and the latter is more frequently used in Greece; second, the typewriter Γ, Δ, ..., Ω don't have much in common with greek typewriting.

Knuth also did the well known lowercase greek characters $\alpha, \beta \dots \omega$ for use in mathematics. These are not suitable for greek text.

The American Mathematical Society added a digamma (F), probably for the sake of completeness (the author would be interested to find out in which area of mathematics a digamma is or could be used).

Then came Silvio Levy [1], who created an entirely new family of greek fonts (roman, slanted, boldface, typewriter). These fonts are partly coded in 8-bit and Silvio uses ligatures and macros to obtain the characters in the ASCII range 129–256.

As a matter of fact, Silvio's idea was to have as many combinations of accents, breathings and subscript iota as possible and at the same time integrate a new feature concerning sigmas, namely an automatic choice between the medial σ and the final ς . For this, he included every possible combination $\sigma + \langle character \rangle$ as characters of the font (obtained by ligatures) and defined \mathbf{s} to be ς in all other cases. This is a brilliant idea, but has a major disadvantage: instead of 2 font positions σ, ς it needs 58. This forces him, because of lack of space, to use the \accent primitive in the case of the combination $\langle breathing \rangle + \langle grave accent \rangle$ („, „). Of course, as he already points out, this doesn't harm hyphenation since these accent combinations appear only on monosyllables. Also, Silvio announced an hyphenation table, without specifying for which kind of Greek.

A year later, Klaus Thull and the author [2] made some changes to Silvio's fonts:

- reducing them to 128 characters, because of problems with some drivers,
- making a new family of one-accent fonts (the official system in Greece),
- adding a SMALL CAPITALS font, a few ancient greek characters (Ϝ, Φ, Λ, Ψ, Α) and hyphenation patterns for modern "post-1974" greek (δημοτική).

2 What comes next

Since the release of TeX 3.0, many efforts have been undertaken to establish standards for 256-character CM fonts. These fonts – or at least tentative versions of them – have been presented in Cork 1990 and in Vienna 1991 and contain a maximum of accented characters to suit most of the occidental languages.

Besides supporting 8-bit input, the new TeX also allows hyphenation of different languages in the same document. In this way, soon we will have different pattern tables included in the plain format of the standard TeX-distribution. As a matter of fact, Mike Ferguson has proposed to coordinate such a collection of hyphenation patterns.

And what about greek? Before considering hyphenation problems, a standard font table should be fixed! I think it is time to take this step: choosing a standard greek font table and starting to work with it. In the appendix of this note, I propose a font table. All combinations of characters, accents, breathings and subscript iota are present as individual characters, reached by ligatures. Special symbols by Oxford in epigraphical texts [3] are included. Note that positions '013 and '040 are blank on purpose to maintain compatibility with PostScript fonts¹.

Of course this is just a proposal and I would like to collect your opinions and ideas. Other possible choices are:

1. Silvio's font table (see [1, p. 24]),
2. the greek standard 8-bit ASCII code,
3. greek PC (for example [4, p. 212]) or Macintosh (greek family script) 8-bit ASCII code, and finally
4. an entirely new one.

Choices 2 and 3 are more "open to the outside world" but have a major disadvantage: they don't contain all characters needed for classical greek.

Concerning multi-accent and one-accent systems, I think it is only natural to share the same font table (in the second case all breathings and the subscript iota will be missing and all accents replaced by the "universal accent", which also needs to be standardized!). Of course many greek TeX-users will say that 'multi-accented characters are only old junk'; I would answer that the first principle of TeX is universality, therefore the fonts used for classical or modern greek should be compatible.

¹ Have you ever tried to use cmr10 in Illustrator 3.0?

3 Problems.

Besides a choice of the font table, there are some problems which should be discussed, concerning the transliteration of Greek in 7-bit ASCII. The latter can still be useful for electronic mail and other media or devices which don't allow 8-bit. I think that Silvio's transliteration

α	β	γ	δ	ϵ	ζ	η	θ	ι	χ	λ	μ
a	b	g	d	e	z	h	j	i	k	l	m
v	ξ	o	π	p	σ, ς	τ	u	ϕ	χ	ψ	ω
n	x	o	p	r	s	t	u	f	q	y	w

has become a standard (I still get messages from people using χ for χ , θ for θ , υ for υ , which can cause quite a confusion...). In [2] a slight modification to Silvio's transliteration is proposed: c for ζ . I have another modification to propose: v, V for the letter digamma \digamma , F. This is some kind of rehabilitation of this letter, which was rather common in some ancient greek idioms (who would deny the beauty of Sapfo's

ἀλλὰ κὰμ μὲν γλῶσσα φέαγε, λέπτον
δ' αὐτικα χρῶι πῦρ ὑπαδεδρόμακεν,
δηπάτεσι δ' οὐδ' ἐν δρρημ' ἐπιρρόμβεισι
[δ' ἀκοναι]

But there is a problem: the circumflex accent. Silvio encodes it as ~ which – at least visually – is the most natural choice. But ~ is active in TeX and plays an important rôle in line breaking. Silvio has to enclose greek text inside a group where ~ is non-active; this brings several inconveniences, which could be avoided. I propose to change the transliteration of the circumflex accent. The next most natural choice would be ^ (in french "accent circonflexe"). But unfortunately, the same problem would appear. If you look at the standard ASCII table (see [5, p. 367]) only the four characters *, +, =, | are possible choices. The characters * and | should be eliminated because of their uses in various TeX-constructions. Only + and = remain unused. Therefore I propose = as a 7-bit transliteration of the circumflex accent. A sequence like

Ποῦ πῆγε ὁ ζῆλος τῶν παλιῶν ἡμερῶν;
would be typed as

```
Po=u p=hge <o z=hloc t=wn pali=wn  
<hmer=wn?
```

For the minor inconvenience of typing \$\$ when you really want a =² you will be using greek fonts in a straightforward manner, like CM fonts.

² according to Knuth ([5, p. 51]): You can also type + and =, to get the corresponding symbols + and =; but it's much better to use such characters only in math mode, i.e., enclosed between two \$

Similar problems arise with diaeresis ($\delta\alpha\lambda\upsilon\tau\iota\chi$) and subscript iota ($\bar{\iota}\pi\sigma\gamma\rho\mu\mu\eta\eta$). All this has to be discussed.

4 A Greek TeX Users (sub-) Group

There is a lot of work which remains to be done – and more and more people wanting to use TeX for greek, and potentially being able to contribute – :

- hyphenation tables should be made according to TeX 3.0 and the new standard font (once it's established). The recent release of a greek dictionary on electronic media should be exploited;
- LATEX and AMSTEX styles should be created;
- new fonts, like a typewriter font which would look like greek typewriting, and sans-serifs should be made;
- greek PostScript fonts should be gathered and the corresponding .tfm files created. They then could be used as virtual fonts;
- introductory and TeX-educational material should be translated to greek; and finally the most important:
- all this should be gathered, ordered, documented and made available by the maximum number of servers. Some part of it could also be included in the standard TeX distribution.

And before even starting, there should be a solid discussion on the standardization problems I mentioned in previous sections.

Therefore I solemnly propose the creation of some structure (or sub-structure of TUG) which will take the responsibility for distributing and coordinating the tasks. Please send me your opinions, ideas and proposals, either to my bitnet address or to my physical address. I propose that the deadline (minus a week) for submission of technical papers for the first regular issue of *TUGboat*, volume 13 (November 19), be a (tentative) deadline for your answers, after which I'll collect the informations and statistics, and announce them through the *TUGboat* (that is, after agreement of the editorial board).

I especially invite TeX users in Greece to contact me, because of my lacunary knowledge on TeX-activities in Greece.

5 Say it in Greek...

Και στὰ Ἑλληνικὰ τώρα: πρῶτα ἀπ' ὅλα θὰ γίθελα νὰ εύχαριστήσω τοὺς ἔκδότες τοῦ *TUGboat* ποὺ μοῦ ἐπέτρεψαν νὰ ἐκφραστῶ στὰ Ἑλληνικὰ μέσα ἀπ' τις στῆλες τοῦ περιοδικοῦ. Κι αὐτὸ γιὰ νὰ σᾶς προτείνω, μὲ πιὸ πολλὴ

signs, since that tells TeX to insert the proper spacing for mathematics.

έμφαση νὰ συνεργασθοῦμε δὲροι μάζι γιὰ νὰ φτιάξουμε ἔνα Ἑλληνικὸ *TeX* ποὺ νὰ ξεπερνᾷ τὶς νόρμες τῶν διαφόρων χωρῶν, νὰ εἶναι κοινὸ γιὰ τὰ ἀρχαῖα καὶ τὰ νέα, καὶ ποὺ νὰ ἐκμετολλεύεται δὲρες τὶς δυνατότητες τῶν ἐπεκτάσεων τοῦ *TeX*: *IATEX*, *AMSTEX* κ.δ.κ.

Προηγουμένως ἀνέφερα μιὰ σειρὰ ἀπὸ ἑργασίες ποὺ πρέπει ἀκόμα νὰ γίνουν. Νομίζω ὅτι μόνο μὲ μιὰ δργανωμένη προσπάθεια μποροῦμε νὰ ἀνταπεξέλθουμε σ' αὐτὸ τὸ ἔργο, σὲ πεπερασμένο χρόνο. Προτείνω λοιπὸν πρῶτα νὰ κάνω μιὰ καταγραφὴ

- τῶν ἀπόφεων σας πάνω στὰ προβλήματα ποὺ προανέφερα (κάθισκα τῶν γραμματοσειρῶν, μεταγλώττιση τῆς περισπωμένης, τῶν διαλυτικῶν καὶ τῆς ὑπογεγραμμένης),
- τοῦ τι ἔχει γίνει κιόλας, ποὺ νὰ μποροῦσε νὰ χρησιμοποιηθεῖ,
- τοῦ κατὰ πόσον θέλετε, ἀλλὰ καὶ μπορεῖτε νὰ βοηθήσετε.

Ἐπίσης προτείνω τὴν ἵδρυση μιᾶς ἑλληνικῆς ὁμάδας φίλων³ τοῦ *TeX*, εἴτε σὰν ὑποομάδας τοῦ *TeX* Users Group, εἴτε ὀνεξάρτητης (τὸ ὄνομα θὰ μποροῦσε νὰ εἶναι *EPTEX*: "Ελληνες Φίλοι τοῦ *TeX* ή κάτι άλλο). Αὐτὸ θὰ προωθοῦσε τὴν συνεργασία⁴ καὶ θὰ διευκόλυνε τὴν πληροφόρηση τῶν δὲροένα καὶ πιὸ πολλῶν –Ελλήνων καὶ ξένων– ποὺ θέλουν νὰ γράφουν σὰν Ἐλληνικὰ μὲ τὸ *TeX*.

Σᾶς προσκαλῶ λοιπὸν νὰ δώσετε δυναμικὰ τὸ παρὸν καὶ νὰ μοῦ μεταφέρετε τὶς ἀπόψεις, ἰδέες, καὶ προποντὸς τὶς προτάσεις σας. Ἐπίσης προβλέπονται συζητήσεις πάνω σ' αὐτὸ τὸ θέμα στὰ Συνέδρια τοῦ *TeX* ποὺ θὰ γίνουν στὸ Dedham (ΗΠΑ) στὶς 15 μὲ 18 Ιουλίου, στὸ Παρίσι στὶς 23 μὲ 26 Σεπτεμβρίου καὶ στὸ Ἀμβούργο (κατὰ πᾶσα πιθανότητα) κάποτε τὸ φθινόπωρο τοῦ 1991.

6 And a motto

Following the tradition of the *The TeXbook* of placing the motto at the end, here is mine:⁵

Ποῦ πᾶς γυμνὸς καλὲ μου Μιγκέλ;
γέμισε δὲ τόπος μὲ κηλίδες ἀπουσίας.

Τὶς τεφροδόχους κλεῖσαν σὲ κρυφὴ σπηλιὰ
καὶ μᾶς πουλᾶνε ἐνέσεις εὐθυμίας.
(Ἀλκαῖος)

◦ Yannis Haralambous
101/11, rue Breughel
59650 Villeneuve d'Ascq
France
FAX: +33 20 91 05 64
Bitnet: yannis@fricit181

³ εἰσαι φίλοις; γίνε μέλοις!...

⁴ ή θὰ τὴν χάλαγε τελείως...

⁵ sorry for the lack of greek sans-serif quotation fonts, it's one of the things which have to be done.

7 Appendix: The font table

	'0	'1	'2	'3	'4	'5	'6	'7	
'02x			[]	{}	<	>			"1x
'03x	~	~	~	~	~	~	~	~	
'04x	!	"	#	\$	%	&	'		"2x
'05x	()	*	.	,	-	.	/	
'06x	0	1	2	3	4	5	6	7	"3x
'07x	8	9	:	.	:	=	:	;	
'10x	@	A	B	Σ	Δ	Ε	Φ	Γ	"4x
'11x	H	I	Θ	K	Λ	M	N	O	
'12x	Π	X	P	Σ	T	Υ	F	Ω	"5x
'13x	Ξ	Ψ	Z	[§]	Δ	¶	
'14x	`	α	β	ς	δ	ε	φ	γ	"6x
'15x	η	ι	θ	χ	λ	μ	ν	ο	
'16x	π	χ	ρ	σ	τ	υ	φ	ω	"7x
'17x	ξ	ψ	ζ	“	”	”	—	—	
'20x	ά	ά	ά	ά	ά	ά	ά	ά	"8x
'21x	ά	ά	ά	ά	ά	ά	ά	ά	
'22x	ά	ά	ά	ή	ά	ά	ά	ά	"9x
'23x	ή	ή	ή	ή	ή	ή	ή	ή	
'24x	ή	ή	ή	ή	ή	ή	ή	ή	"Ax
'25x	ή	ή	ή	”	ή	ή	ή	ή	
'26x	ώ	ώ	ώ	ώ	ώ	ώ	ώ	ώ	"Bx
'27x	ώ	ώ	ώ	ώ	ώ	ώ	ώ	ώ	
'30x	ώ	ώ	ώ		ώ	ώ	ώ	ώ	"Cx
'31x	ι	ι	ι	ι	ι	ι	ι	ι	
'32x	ι	ι	ι	ι	ι	ι	ι	ι	"Dx
'33x	ι	ι	ι		ι	ι	ι	ι	
'34x	è	é	é	é	ò	ó	ó	ó	"Ex
'35x	é	é	é	é	ó	ó	ó	ó	
'36x	í	í	í	í	ü	ú	ú	ú	"Fx
'37x	ø	η	φ	ø	ø	?	~	-	
	"8	"9	"A	"B	"C	"D	"E	"F	

References

- [1] Silvio Levy: Using Greek Fonts with *TeX*, *TUGboat*, 9 (1988) 20-24.
- [2] Klaus Thull and Yannis Haralambous: Typesetting Modern Greek with 128 Character Codes, *TUGboat*, 10 (1989) 354-358.
- [3] Marcus N. Tod (ed), A Selection of Greek Historical Inscriptions, Oxford 1948.
- [4] Χρῆστος Κοῖλιας, 'Οδηγὸς Χρήστη τοῦ MS-DOS v. 3.3, Αθήνα 1988.
- [5] Donald E. Knuth, The *TeXbook* (9th printing), Reading 1989.

JemTEX 2.00 available for Japanese

François Jalbert

I have just released Version 2.00 of my Japanese [I]A[T]EX system.

JemTEX is a freeware package containing everything needed to typeset beautiful Japanese text. You should, of course, already have a Japanese text editor, *TeX*, and *METAFONT*. Turbo-Pascal sources and executables are included for DOS computers. UNIX users are now supplied with a C program (*gcc*) so they too can enjoy *JemTEX*.

If you are interested in *METAFONT* code for japanese and chinese, the program *jis2mf* will interest you. This much improved program generates *METAFONT* code automatically out of 24×24 bitmap files. Smoother and better positioned japanese characters are the main improvements. *METAFONT* code for 61 japanese fonts of 128 characters covering punctuation, english, hiraganas, katakanas, and kanjis (level 1 and 2) is included indirectly in *JemTEX*.

My program *jem2tex* will turn the output of your favorite DOS or UNIX japanese text editor into a standard *TeX*, *LATEX*, or *MUTEX* document. Thanks to several users from Japan, it handles fine points of japanese punctuation, spacing, and hyphenation much better than before. Switching to C for *jem2tex.exe* has also improved the speed substantially since Turbo-C has buffered I/O for non-text files, unlike Turbo-Pascal.

The file *JEMTEX2.ZIP* includes a 40-page-long user's guide *jguide.tex* where you can find all the details. It is (or soon will be) available from:

- SIMTEL (USA) (26.2.0.74)
(tenex FTP or e-mail server)
- utsun (Japan) (133.11.11.11) (binary FTP)

Please feel free to contact me if you wish more information, or to be added to the mailing list which is only now officially being started. I plan on using it to keep everybody informed of new versions and bug fixes.

◊ François Jalbert
220 Forest
Châteauguay, QC
Canada J6J 1R1
jalbert@IRO.UMontreal.CA

Fonts

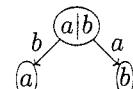
Labelled diagrams in *METAFONT*

Alan Jeffrey

1 Diagrams in *METAFONT*

In *TUGboat* 11(5), Alan Hoenig described a method of producing diagrams in *METAFONT* with labels provided by *TeX*. His method relied on passing information around via font dimensions. This is a standard method of passing information from *METAFONT* to *TeX*, but it has some drawbacks:

- There are only a limited number of font dimensions available, and each label uses up two of them.
- As *METAFONT* can only communicate with *TeX* via font dimensions, each label has to be assigned a font dimension, and it is difficult for the correspondence between font dimensions and labels to be kept automatically.
- Since *TeX* is providing the labels, and *METAFONT* is providing the diagrams, the diagrams have to be kept in a different file from the labels.
- There is no communication between *TeX* and *METAFONT*, so *METAFONT* cannot change the diagram depending on the size and shape of the labels. This is rather inconvenient for diagrams such as



where the shape of the ovals depends on the size of the contents.

Fired with enthusiasm by Alan's talk at the European *TeX* Users Group meeting, I stole the best of his ideas, and slightly modified them to produce a simple *METAFONT-TeX* interface. This allows *TeX* code to be embedded within a *METAFONT* program, for example

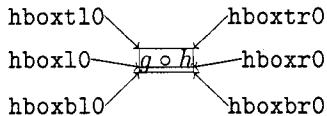
```
begindiagram(2,30pt#,7pt#,2pt#);
hboxes(0);
pickup pencircle scaled 0.4pt;
.5[hboxl0,hboxr0] = (.5w,0);
draw hboxb10..hboxt10
    ---hboxtr0..hboxbr0
    ---cycle;
setbox0 "$g \circ h$";
enddiagram;
```

produces the diagram *(g o h)*. The new facilities used are:

`begendiagram(2,30pt#,7pt#,2pt#)` starts off diagram 2, which is 30pt wide, 7pt tall and 2pt deep.

`hboxes(0)` says that the only label we'll be using is number 0. This has a similar syntax to `labels`, so you can say `hboxes(1,2,7)` or `hboxes(3 upto 9)`.

`hboxl0` is the left point of label number 0, at the baseline. Similarly, `hboxb10` is the bottom left, `hboxtr0` is top right, and so on. In this example, these points are



You can also use the numeric variables `hboxwd0`, `hboxht0` and `hboxdp0` which are the width, height and depth of label 0, and `hboxwd#0`, `hboxht#0` and `hboxdp#0` which are their sharp equivalents.

`setbox0 "$g \circ h"` sets label number 0 to be `g o h`.

`enddiagram` finishes it all off.

The rest of the diagram is standard METAFONT. Within a TeX document you can use

`\diagramfile{example}` to load in the diagrams kept in `example.mf`,
`\diagramf{2}` to get the second diagram, and
`\everylabel` which is a token register added to every label, in the same fashion as `\everymath`. It should be set *before* saying `\diagramfile`.

These commands behave well inside groups, so if you say

```
\diagramfile{foo}
{\diagramfile{baz}\diagramf{1}}
\diagramf{2}
```

you get the first diagram from `baz` and the second diagram from `foo`.

2 How it all works

In the `diagramf` package, TeX and METAFONT communicate by auxiliary files, in a similar fashion to the MG TeX-PostScript interface ('Problems on the TeX/PostScript/graphics interface', *TUGboat* 11(3)).

When you run METAFONT on `example.mf` it reads in `example.dim`, which specifies the dimensions of all the boxes. In our example, part of `example.dim` is

```
wd#[2][0] := 20.3344pt#
ht#[2][0] := 6.94444pt#
dp#[2][0] := 1.94444pt#;
```

So, in diagram 2, label 0 has width 20.3344pt, height 6.94444pt and depth 1.94444pt. From this, METAFONT calculates where to put each label, and outputs a `.dia` file, containing TeX code. For example `example.dia` contains¹:

```
\newdiagram{2}
\diagramlabel{0}{4.88908pt}{0pt}
$g \circ h
\enddiagramlabel
\diagramchar{2}
\endnewdiagram
```

This tells TeX that diagram number 2 contains label 0 at coordinates (4.88908pt, 0pt) consisting of `$g \circ h`. The diagram is character number 2 in the `example` font.

Similarly, when TeX encounters the instruction `\diagramfile{example}` it loads in `example.dia` and produces `example.dim`. And so we can have our METAFONT cake and eat it in TeX.

Well, almost. Unfortunately for all these grand ideas, METAFONT has *no* file-handling capabilities at all! The only files METAFONT generates are the `.tfm`, `.gf` and `.log` files.

This is rather annoying, but fortunately we can steal an idea from Section 7 of the Dirty Tricks appendix in *The METAFONTbook*. There, Knuth uses the `.log` file as a means of communicating between METAFONT jobs. Similarly, we use the `.log` file as a way of sending messages to TeX. Our `texoutput` macro is defined

```
def texoutput text t =
  for s = t:
    message s & "% diagramf";
  endfor
  message ""
enddef;
```

So `texoutput "Fred", "Ethel"` produces the output

```
Fred% diagramf
Ethel% diagramf
```

You can then use your favourite file-handling utility to filter the `.log` file, keeping only the lines containing `% diagramf`. On my UNIX set-up, for example, I have an alias `diagramf` `example` which expands out to

```
touch example.dim
mf example
grep "% diagramf" example.log > example.dia
echo Labels written on example.dia.
```

¹ Actually, each line ends with `% diagramf`.

The crucial line in this is the `grep`, which takes all the lines from `example.log` containing % `diagramf` and puts them in `example.dia`.

And so we've achieved labelled diagrams in METAFONT. The `diagramf` package is free software, and is available from the Aston archive.

3 Acknowledgements

The inspiration, and many of the original ideas, for this article came from Alan Hoenig's talk on the same subject at Cork. I'd also like to thank Jeremy Gibbons and Damian Cugley for comments, advice and allowing me to bounce ideas off them.

- Alan Jeffrey
Programming Research Group
Oxford University
11 Keble Road
Oxford OX1 3QD
`Alan.Jeffrey@prg.ox.ac.uk`

© 1990 Alan Jeffrey

Graphics

X Bitmaps in \TeX

Reinhard Fößmeier

Abstract

A new \LaTeX style, `bitmap.sty`, allows the direct inclusion of bitmaps from the X Window System in \LaTeX documents. With a tiny modification, the macros can be used with plain \TeX , too.

Resumo

Nova ordonaro bitmap.sty por \LaTeX permesas rektan enkludon de bit-matricoj el la fenestro-sistemo X en \LaTeX -aj dokumentoj. Post eta modifo, la makrooj estas uzeblaj ankaŭ por simpla \TeX .

1 Introduction

The X Window System uses a special C language syntax to describe bitmaps, images made up from black and white pixels. The syntax consists of several C definitions that specify the width and height of the bitmap and possibly the position of a "hot spot", and the declaration of a character array for the bitmap information, with initializers in hexadec-

imal notation (see figure 1). Each pixel line starts with a new byte; the last byte in a line is normally padded with zero bits.

```
#define bildo_width 50
#define bildo_height 30
static char bildo_bits[] = {
    0x00, 0x60, 0xff, ...
    ...
    ... 0xe0, 0x01};
```

Figure 1: Example of the C bitmap format in X

Apart from being suited for inclusion in C programs where it can be processed by the Xlib routines `XCreateImage` or `XCreatePixmapFromBitmapData`, this format can be read by the Xlib routine `XReadBitmapFile` or written by `XWriteBitmapFile`. It is also supported by several programs, including a graphic editor (`bitmap`), conversion programs from/to ASCII character maps (`atobm`, `bmtoda`), and a screen dump utility (Bruce Schuchardt's `xgrabsc`). So it has become a true standard for the representation of bitmaps.



Figure 2: An example from the X Window System bitmaps (`xlogo64`)

2 The `bitmap.sty` style

A new \LaTeX style "bitmap" provides a macro to include and print such bitmaps in \LaTeX documents. The macro is called `\Bitmap` and has two arguments: the name of the file containing the bitmap, and the pixel size desired. The latter is saved in `\bmppsiz` and used to set the value of `\baselineskip`:

```
1 \newcount\bmhpocz
2 \newcount\bmwid
3 \newif \ifbmbblack
4 \newdimen\bmrlen
5 \newdimen\bmppsiz
6 \catcode',=\active
```

```

7 \def\Bitmap#1#2{\bgroup
8 \bmposz=#2
9 \baselineskip=\bmposz
10 \parindent=0pt
11 \bmrlen=0pt

```

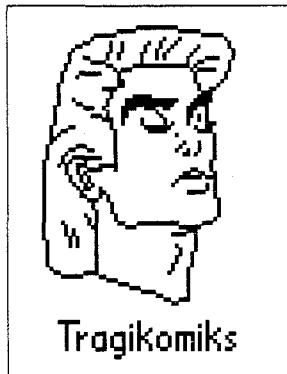


Figure 3: Example of an included bitmap, taken from an Asterix cartoon

The comma, separating the hex-numbers in the file, is made an active character; its function is to analyse the hexadecimal information and translate it into *TeX* \vrules. It also throws away the leading *0x* of the hex numbers; the second and third argument (the hex figures) are swapped and each given to \HexFig for further analysis. Then, \bmhpz is incremented by 8 columns and checked against the width of the bitmap; if the former exceeds the latter, the line is terminated and \bmhpz is reset to 1.

```

12 \catcode',=\active
13 \def,\#10x##2##3{%
14 \HexFig{##3}\HexFig{##2}%
15 \advance\bmhpz 8
16 \ifnum\bmhpz>\bmwid
17 \0\hfil \vskip 0pt
18 \bmhpz=1 \bmrlen=0pt
19 \fi}

```

The \HexFig macro analyses a hexadecimal figure and translates it to four calls of the macros \0 or \1, for white and black pixels, respectively. As the hex figures A to F normally are not capitalized in bitmap files, \uppercase is used to convert them if necessary.

```

20 \def\HexFig##1{%
21 \uppercase{\ifcase "##1}
22 \0\0\0\0\or
23 \1\0\0\0\or
24 \0\1\0\0\or
25 \1\1\0\0\or

```

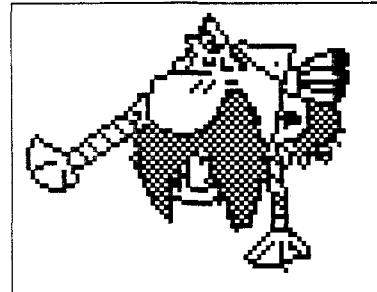


Figure 4: Another example (Abraracourcix, Gaulish tribe chief)

```

26 \0\0\1\0\or
27 \1\0\1\0\or
28 \0\1\1\0\or
29 \1\1\1\0\or
30 \0\0\0\1\or
31 \1\0\0\1\or
32 \0\1\0\1\or
33 \1\1\0\1\or
34 \0\0\1\1\or
35 \1\0\1\1\or
36 \0\1\1\1\or
37 \1\1\1\1\fi
38 }

```



Figure 5: The bitmap logo of the *Akademio Internacia de la Sciencoj (AIS) San Marino*

Black pixels are printed as rectangular spots, produced by \vrule. To save space in *TeX*'s memory, \0 and \1 collect sequences of equal bits and put them together to longer \vrules. Invisible \vrules of height 0 are used for the white space rather than \hskips because they, too, save a little memory space. The length of the current sequence of 0 or 1 bits is kept in the dimension \bmrlen, the color of the current pixel is remembered in the value of \ifbmbblack.

```

39 \def\0{\ifbmbblack
40 \vrule width \bmrlen height \bmposz
41 \bmrlen=\bmposz \bmbblackfalse
42 \else
43 \advance \bmrlen\bmposz
44 \fi}

```

#1:	<code>#define bildo_</code>	(discarded)
#2:	50	width
#3:	<code>#define ... _bits</code>	(discarded)
#4:	=	(discarded)
#5:	0x00, 0xe0, ... 0x01	picture

Table 1: Arguments of `\BmContent` in the example of figure 1

```

45 \def\1{\ifbmbblack
46 \advance\bmrlen\bpsiz
47 \else
48 \vrule width \bmrlen height Opt
49 \bmrlen=\bpsiz \bmblacktrue
50 \fi}

```

The only pieces of information used from the bitmap file are the width of the bitmap and its picture information; height and hot spot coordinates are discarded. The contents of the file are read with `\@@input`; a macro `\BmContent` gathers the width as its second and the picture information as its fifth argument. The other arguments are ignored; their only purpose is to get rid of the rest of the bitmap header. `\BmContent` is prefixed with `\expandafter`, to capture the result of `\@@input` in its arguments.

```

51 \def\BmContent
52 ##1_width ##2 ##3[] ##4 ##5;{%
53 \bmwid##2
54 \bmhpz=1
55 ,##5
56 }% end of \BmContent
57 \expandafter\BmContent\@@input #1
58 \egroup}% end of \Bitmap

```

Finally, the comma's `\catcode` is reset to "other":

```
59 \catcode',=12 % other
```

In the example of figure 1, the arguments of `\BmContent` would be like shown in table 1. Obviously, the 3rd and 4th argument could be combined into one; yet this way provides some more security against erroneous use.

The macros can be used with plain `TEX`, too, if `\@@input` in the last line of `\BmContent` is replaced with `\input`. The `LATEX` macro `\input` doesn't work with the `\expandafter` technique.

Figure 2 shows the X logo bitmap, included in this document from the X bitmap file *xlogo64*. Figures 3 to 7 show some more examples, which may serve to show the versatility of even small bitmaps. All bitmaps presented here are far under 100x100 pixels.



Figure 6: Yet another example ...

3 Possible extensions

To draw boxes around bitmaps, they can be put into a `\vbox`. In its present form, the macro doesn't compute the necessary width of such a box, although this information is available from the bitmap header. This functionality can be achieved by adding to `\BmContent`, after the definition of `\bmwid` in line 53, the statement

```
\hsize\bmwid\bpsiz
```

Sometimes it is interesting to have statistics about the complexity of a bitmap. To this end, a `\newcount\Complty` may be introduced, initialized to zero and incremented by

```
\advance\Complty1
```

either in `\1` before the `\ifbmbblack` at line 45 (to count black pixels) or in `\0` within the `\ifbmbblack` at line 39 (to count `\vrules`). The result may be issued by a `\message` statement.

4 Problems, conclusion

Unfortunately, typesetting bitmaps is slow. A 60x85 bitmap (like the one shown in figure 6) can easily consume more of `TEX`'s time than an ordinary text page. Moreover, the many `\vrules` consume a lot of `TEX`'s space, even with runlength encoding. Tests show that drawing bitmaps with two printing characters (e. g., `-` and `+`) for `\0` and `\1` saves some time but needs even more space. For really big bitmaps, `Big TEX` may have to be used; it was, however, not necessary for this article.



Figure 7: ... and another.

This problem could be alleviated by creating 16 special drawing characters for the bitmap patterns corresponding to the 16 hexadecimal figures. The complexity of a bitmap, however, will always be reflected in the cost of its typesetting. Remember that the aforementioned bitmap contains about as many pixels as there are characters on an average page.

Serious problems may arise if the bitmap file contains C language comments. They are discouraged when using `bitmap.sty`. The *bitmap* editor discards them anyway, so they aren't normally used; however, the *terminal* bitmap used in the X System contains a comment.

If the bitmap width is not a multiple of 8, the algorithm in “\,” depends on the last byte in each line being padded with 0's. This could be changed by putting the check of `\bmhp0z` against `\bmwid` into `\1`. It turned out to be hardly ever necessary in practice.

Pictures with small pixel sizes come out better if a multiple of the printer resolution is chosen for the pixel size. (Oh well, I know *TeX* input should be device independent...) Here sometimes the “big point” ($1\text{bp} = 1/72 \text{ in}$) unit is useful if the resolution is related to inches; e.g. the resolution of a 300dpi printer is 0.24bp ; with 400dpi, 0.18bp .

Although the use of bitmaps in documents is limited by *TeX*'s resources, they provide a comfortable way to put small images into documents and a useful interface to the X Window System, e.g. for documentation.

- ◊ Reinhard Fößmeier
iXOS Software GmbH
Bretonischer Ring 12
D-W-8011 Grasbrunn
Germany
`refo@ixos.de`

Output devices

Report on the DVI Driver Standard

Joachim Schrod
Secretary
TUG DVI Driver Standards Committee

The DVI Driver Standard will be available in several stages. The basic stage is now called level 0. It covers only those driver capabilities which are really necessary to output a DVI document on an output device. All other driver capabilities will be called features (and may even be realized outside a driver). In the future we will publish several additional standard documents which will cover ranges of features; those documents will represent “tiers” built upon level 0 or on previous tiers. In this way they will be available as future stages of a complete standard. (One may doubt whether the standard will ever be complete as there may be always new features to standardize.)

The basic stage, level 0, consists of three parts:

- (1) The pure standard document telling what a driver must be able to do.
- (2) Definitions of all file formats spoken of in part 1.
- (3) A rationale describing why the committee has chosen the given definition in part 1, recalling discussions that led to particular decisions.

A draft of the level 0 document is about to be published for public review. Part 1 of the draft is (almost) ready; a few spelling errors and such have to be removed. Part 2 was ready, but D.E. Knuth has changed the GF documentation, and this change must be incorporated. Part 3 exists only in part.

The committee will publish the draft as soon as possible. It may be that the draft of the rationale will not be finished in time; in that event we will publish part 1 by itself. This is considered to be useful (although not desirable) so that we will get responses very soon—and especially to change the status from “draft” to “released” as soon as possible. The file formats will not be published in *TUGboat*; they are available on several file servers. For people who do not have access to file servers I've prepared a brochure covering all file formats.

When complete, the standard will be published in the *TeXniques* series. The style will be modified slightly to follow formal standards conventions. The body of the standard will form the main text; this will be followed by a number of “annexes”. The

file formats will come immediately after the main text, as "normative" annexes; that is, these format specifications are an integral part of the standard, but the presentation of each is self-contained and too large to be appropriate in the main text. Finally, the rationale will appear as an "informative" annex, to present information that is not an integral part of the standard, but is intended to help a user in understanding it.

Future work

What tier will come next, i.e., what driver feature will be looked at next, is still unclear. There is public pressure to tackle the area of graphics inclusion at an early date; others want to touch areas such as page selection, etc., first. So this remains an open problem. We invite all parties to bring proposals to the committee. My personal opinion is that a proposal for a new tier received early will be handled early. So if someone is eager to see a specific topic addressed, he or she should do work on this topic and send us the result of the work. (We will be glad to acknowledge contributors.)

◊ Joachim Schrod
 Technical University of Darmstadt
 Institut für Theoretische
 Informatik
 Alexanderstraße 10
 W-6100 Darmstadt
 Federal Republic of Germany
 Bitnet: xitijsch@ddathd21

Resources

Review of *3 : 16 Bible Texts Illuminated*

Karl Berry and Kathryn A. Hargreaves

3 : 16 Bible Texts Illuminated, by Donald E. Knuth. A-R Editions (801 Deming Way, Madison, WI 53717-1903; (608) 836-9000), 1991. ISBN 0-89579-252-4. 268 pages, paperbound.

In about three-fourths of *3 : 16*, his first book since *Computers & Typesetting*, Knuth studies — indeed gives an exegesis of — chapter 3, verse 16 of

every† book in the Bible. 59 calligraphers (from 26 countries) fill the remaining quarter of the book with their renderings of the verses. Hermann Zapf, one of the world's leading typographers, contributed the illustration for John 3 : 16 and the cover design.

The main text consists of four pages per Biblical book: a left-hand page with a summary of the book as a whole, the page of calligraphy, and two pages of discussion about the 3 : 16 of that book. In a foreword, Knuth discusses how he came to study the Bible using the statistical procedure of stratified sampling, and describes his reactions to the experiment (including a few quantitative conclusions) in an afterword. Mathematically-minded people will appreciate this novel application of statistics to the Bible. We wished we could have attended the Sunday morning Bible classes led by Knuth upon which he based the text.

The book design is attributed to both Knuth and Zapf. The typeface is Computer Modern (Knuth didn't do any of the calligraphy, so it seems only fair that all the typeset letters should be his own design). The book designers and calligraphers use up to four colors: the text is black, the name of the Biblical book is printed in rust red on the summary page, and the verse is printed in a blue-green on the left-hand discussion page, inset in the text. (It must have been a lot of fun to figure out those 59 *parshapes*.) The calligraphy also uses a light ochre. The book designers reduced the calligraphic works (or perhaps just had the calligraphers make them) so they would all fit in approximately the same rectangle, and then set the calligrapher's name and Biblical reference well underneath in sans — giving credit without detracting from the calligraphy.

We can read the title's word "Illuminated" in two different ways. First, through linguistic and historical analysis, Knuth "sheds new light on" the meanings of the Bible verses — finding humanistic interpretations that reinforce our contemporary experience. In fact, he gives his own fresh translation of the verses. (He wholeheartedly recommends this to anyone interested in Bible study.) He also gives a generous sprinkling of those personal insights for which he is well-known. Knuth's writing has the elegance to which we've grown accustomed; the book can be enjoyed even by those with no background

† Well . . . , almost every. Some books don't have that many verses. Knuth "decided to omit all such books, because they turn[ed] out to be similar to other books that are long enough to be included."

in Bible study, and even no particular enthusiasm for it.

The other way we can read the title's "Illuminated" is in the sense of "illuminated manuscript". Manuscript books from the Middle Ages were handwritten and illuminated by monks (although they sometimes used outside illuminators). As in 3:16, the monks used black ink for the text and a red ink (made from clay) for titles. Gold, red, and blue were the illuminators' favorite colors. Instead of gold leaf, the book designers use an ochre ink. Knuth chooses to "illuminate" his text not with illustrations and decorations subservient to it, but with calligraphic renderings of the verses. That calligraphy is commonly used today for producing religious artifacts is not lost on us, but Knuth is more likely incorporating in 3:16 two of his areas of interest—theology and typography.

Many of 3:16's calligraphers "illuminate" their verses in a straightforward way using literal illustrations: Ismar David (p. 19) uses a pyre for a verse about burning offerings, John Prestianni (p. 27) a map for a verse about to whom God assigns what territories, Satyanarayan Mallayya Wadish-erla (p. 47) an eye (elegantly drawn by extending the letterforms) with a teardrop for a verse about a man sobbing, Andrzej Kot (p. 55) a fish for a verse about the promise of a dry riverbed being filled with water, Allen Q. Wong (p. 63) a pomegranate-decorated pillar for a verse about such, Lili Cassel Wronker (p. 131) a broad-leaved branch for a verse about a plant springing up to shade Jonah's head.

We think the real power of calligraphy, however, is that calligraphers can use graphic manipulation to "illuminate", if you will, a text's meaning(s); for instance, they can (1) modulate letters in regards to weight, color, shape, drawing quality (i.e., rough, smooth, quickly executed or carefully rendered) or rendering method (traditional broad-edged pen, brushes, drawing, or cutting), or (2) choose particular letterforms that evoke either historical (e.g., ecclesiastical or avante-garde) or formal (e.g., delicate or bloated) meaning. They can also arrange things in an evocative composition or manipulate the background. When they do this, we can see the words in addition to reading them—and calligraphy becomes a *visual language*. Some of 3:16's calligraphers do this:

R. Williams (p. 11) gives the word 'troubles' in his verse an unstable baseline, and 'desire' a decorative style. Fritz Eberhardt (p. 75) evokes shredded clothes by rendering words about such using a dry pen. By putting hands atop the outer stems of the verse's first letter, 'W', Luigi Cesare

Maletto (p. 79) evokes a supplicant with arms outstretched skyward for Job's wish he'd never been born. Kris Holmes (p. 83) configures her night prayer verse (the familiar "Now I lay me down to sleep . . .") in a circle, evoking daily repetition of such. Sheila Waters (p. 91) uses a rough chiaroscuro to darken the background for the two instances of 'wickedness' in a verse about the abundance of evil in the world. Georgia Deaver (p. 107) uses violent brushstrokes to slash out the 'V' starting the lamentation "Viciously he ground my teeth on gravel . . .". Steven Skaggs (p. 119) renders a block of intertwining repetitions of the word "orgies" for a verse about such. Peter Fraterdeus (p. 127) uses all caps for the command "LISTEN TO THIS", round, fat brush-painted letters for the words "you fat cows of Bashan", and gold background for words of warning directed at some affluent women. Karlgeorg Hoefer (p. 143) puts words about terror on heaving baselines and words about calm waiting on smooth, straight ones. Timothy R. Botts (p. 167) renders in elongated, thin red strokes, words about baptism with fire. Alfred Linz (p. 179) uses as background what looks like either an aerial photograph or a close-up photograph of a rock and rough, broken letters (graffiti left behind?) for a verse about a "trail of wreckage and misery". Neenie Billawala (p. 203) incorporates "staff" lines (which can double as typographic rules) and handwritten "musical notes" into a verse about singing. Margo Snape (p. 207) puts a verse discouraging sexual immorality on a gold background marbled to evoke licking flames. Leonid Pronenko (p. 215) puts (almost) each word of a verse about the mystery of true religion on a piece of torn paper reminiscent of ransom notes pieces. He emphasizes the word 'mystery' both by making it heavier and by splitting it across two pieces of paper. Jean Evans (p. 235) reduces letterforms to geometric black shapes, making the words hard to read—a modernist play on the verse's "making his words hard to understand" and "people distort his words". Rick Cusick (p. 243) splatters with ink a verse about metaphoric spitting and lashes out the word "spit" with quick, rough penstrokes.

Knuth's illuminations of the verses reveal to us what a complex, multilayered book the Bible is, whose understanding requires much work on the part of the reader. Historically, it has produced a range of interpretations. Calligraphy can also layer meaning and form, requiring the reader/viewer to sort things out; for instance, Robert Borja (p. 35) cuts up the background of his rectangle with color to get a sword-shape for a verse about such. He also plays on the word 'double-edged' by mirroring on

the left side the darker text (which is about a sword strapped onto a right thigh) on the right. Both the Bible and calligraphy can require us to participate in the interpretation, and the reward for doing so is a satisfaction worth seeking.

- ◊ Karl Berry
135 Center Hill Rd.
Plymouth, MA 02360
`karl@cs.umb.edu`
- ◊ Kathryn A. Hargreaves
135 Center Hill Rd.
Plymouth, MA 02360
`letters@cs.umb.edu`

LATEX for engineers and scientists

Book review by Nico Popelier

LATEX for Engineers and Scientists, by David J. Buerger. McGraw-Hill, 1990. ISBN 0-07-008845-4. 198 pages, paperbound.

Although the *LATEX* manual is a useful book, it is not suitable as an introduction, as a book for beginning users. *LATEX for engineers and scientists* by David J. Buerger, published last year, at first sight appears to be a good introduction. In the preface the author writes: '[this book] was written to provide a fast and easy way to learn how to produce technical documents with *LATEX*.' And indeed, *LATEX for engineers and scientists* is a book that doesn't frighten readers by its length and is easy to read. It describes *BIBTEX* and *Makelindex*, it gives exercises – with answers – that are really not bad, and it contains an index – although I find that a bit short – and a glossary.

Unfortunately my general opinion about this book is not positive: both the contents of the book and the quality of the book as a printed product leave a lot to be desired.

My overall impression of the contents of the book can be summarised in a few points.

- The author has not quite grasped the concept of a document style and the separation between logical and visual structure, two fundamental concepts of *LATEX*.
- The author does not distinguish between *LATEX* proper and *LATEX plus* the standard document

styles. There are many document styles beside the standard ones, so this distinction is essential.

- In several examples *LATEX* and *TEX* commands are mixed. My opinion is that in examples only *LATEX* commands should be used. If the author insists on mentioning the *TEX* equivalents, he should explain what sort of functionality *LATEX* adds.
- Some functions of *LATEX*, among which at least one important function, are not explained in the book.
- Explanations in the book are sometimes confusing or sloppy. In a few cases they are even incorrect.

I will give some examples:

- In chapter 4, Formatting environments, the author starts with the `center`, `flushleft` and `flushright` environments, and then goes on to treat the `list` and `quotation` environments. The main purpose of *LATEX*'s markup instructions is describing the logical structure of the text. In a book on *LATEX*, descriptions of logical design should come before descriptions of visual design.
- The custom description list on p. 28 refers to layout parameters of the `list` environment that cannot be found in the index or the glossary. Although the author includes in his book instructive page-layout diagrams¹ that are unfortunately absent in the *LATEX* manual, he forgets to include the equally useful list-layout diagram that is printed on p. 113 of the *LATEX* manual.
- The custom description list given as an example on p. 28 is a variation on the `description` environment described in the *LATEX* manual. In this example the items are typed as `\item[{\bf Fox}]`; as a result there is no clear separation between form and contents. A better way would be to define the layout of the items in the definition of the customised list. That way, one only has to type `\item[Fox]`.
- On p. 39 the author gives a table of the typeface sizes that correspond to *LATEX* commands such as `\small`, `\normalsize` and `\large`. The correspondence given in the table is valid only for the standard document styles and not for *every* document style. By failing to make this distinction, the author suggests that the table is universally valid, which it isn't.
- In chapter 6 the author treats only the `$... $` and not the `\(... \)` construction for in-line mathematical formulae. `$... $` and `$$... $$` give formulae in a more or less fixed layout. If one uses *LATEX*'s `\(... \)` and `equation` environment

¹ Similar diagrams have appeared in *TUGboat*.

instead, the user lets the document style control the formula layout. Furthermore, the L^AT_EX notation for formulae has opening and closing tags that are not identical, which results in fewer errors.

- In chapter 7, on p. 52, the author introduces the \lefteqn command without any explanation. This is a command that a lot of users find confusing: they often think that \lefteqn puts an equation flush with the left margin of the text.
- In chapter 8 the author gives a confusing description of the two environments **table** and **tabular**. The **tabular** environment produces a table, i.e. an arrangement of cells in rows and columns, possibly with horizontal and vertical rules². The **table** environment creates a floating object, i.e. a part of the document for which L^AT_EX tries to find a good place to print it. In most cases, the **table** environment contains a caption that starts with the word ‘Table’³ and a **tabular** environment for the actual table contents.

However, Buerger writes (italics mine):

Tables *created* with the **tabbing** or **tabular** environments— ...

The \begin{table}[] or \begin{figure}[] command will *create* a table or figure.

- On p. 64 the author explains the use of \label and \ref. He instructs the reader to put the \label command after sectional-unit commands and after the \caption command of a **figure** or **table** environment. However, there is no information on where to put the label in **equation** and **eqnarray** environments.
- In chapter 10, Organizing a document, the author uses in an example

```
\topmargin 0mm
\def\BibTeX{ ... }
instead of the LATEX equivalents
\setlength{\topmargin}{0mm}
\newcommand{\BibTeX}{ ... }
```

- In chapter 10 the author fails to distinguish between L^AT_EX proper and the standard document styles. On p. 68 the author writes:

Title information is automatically centered.

and (italics by the author):

You can produce an abstract placed below the title information ... by typing the following command *before* the \maketitle command.

² An imprecise definition of a table, I know!

³ To be precise: this is specified by the document style, but it should be ‘Table’ or something equivalent.

In both cases the behaviour the author describes is that of the standard document styles: in other document styles a title could be left-justified and emphasised phrases could be printed in a boldface font. In the second case, the author is also definitely wrong since the \maketitle command defined in the standard document styles does not print the abstract, but only the title, author and date.

The author is also inconsistent with notation: for example, in pages vii-xiii, the table of contents, list of figures and list of tables, I found ‘L^AT_EX’, ‘LaTeX’ and ‘LaTEX’! I sometimes got the feeling that the book was written or at least finished in some haste.

Some examples of features of L^AT_EX that are missing in L^AT_EX for engineers and scientists:

- The author writes that the \include command is similar to the \input command, except that it starts on a clean page. He doesn’t mention one of the nicest mechanisms in L^AT_EX: cross-referencing between sub-documents if some of the sub-documents are excluded from the current formatting run by means of \includeonly.
- The only information on T_EX’s units was the sentence ‘There are 72.27 points to an inch’, and I found it in the chapter on error messages!
- One of the sample input files contains the \; command, without explanation and without treating other, similar commands.

So far, I have only criticised the author. However, I think the publisher of this book, McGraw-Hill, can be blamed for a few things as well. Concerning the quality of the book as a printed product: the book was produced from camera-ready pages prepared by the author on a laser printer. Computer Modern is a good typeface, if only you use it on a printing device of sufficiently high quality. Laser printer quality is, I’m afraid, not good enough and I hope this book is one of the last books on T_EX-related matters produced in such a way. As for the contents of the book: it seems likely that McGraw-Hill did not ask an expert to review the book, otherwise they would have asked the author to rewrite parts of it.

L^AT_EX for engineers and scientists is not a bad book, but it is not a good book either. It can be used, but I can’t really commend it.

Nico Poppelier
Elsevier Science Publishers
Academic Publishing Division
R&D Department
Sara Burgerhartstraat 25
Amsterdam, The Netherlands
n.poppelier@elsevier.nl

Q & A

JUST PLAIN Q&A: The Russians are Coming!

Alan Hoenig

We've received three good questions, all of them from Andrei Khodulev of Mir Publishers in Moscow (Soviet Union). I hope more people will respond. Contact the TUG office or me directly (ajhjj@cunyvm on Bitnet, or by telephone, (516) 385-0736) with your TeX questions.

Cutouts Spanning Paragraphs

Mr. Khodulev needs to know how to use `\parshape` to cross several paragraphs. Suppose, for example, he needs to leave space for a 3cm high cutout. Suppose the paragraph at which the cutout starts is only 24pt high. Can we ask TeX to somehow continue the effects of `\parshape` across paragraph boundaries? Furthermore, we need to specify the cutout in absolute measure, not using the numbers of lines, as `\parshape` expects. Is this possible?

The answer to this question formed the subject of an earlier article by Tom Reid in *TUGboat* 8, no. 3, 315-320 (November 1987). This article contains many useful TeX hack tricks and deserves to be better known than it is. Tom presents a set of macros which do what the Russians require. We note that his macros don't use numbers of lines. Simply set a box to the required height and width:

```
\setbox\figbox=
  \vbox to 4 cm {\hbox to2cm{\hss}\vss}
```

for example. Tom's macros do the rest.

An alternative solution to the same problem appears in [1].

By the way, TeX can easily be asked to convert from measure to lines. We can determine the lines by "dividing" the given measure by the value of `\baselineskip`. Here's one way to define a macro `\tolines` to do that. In practice, the command `\tolines 5pt` or `\tolines 3cm` places the number of lines equivalent to the measure in the register `\n`. Note that we don't need any grouping symbols in the argument.

```
\newcount\n
\def\tolines{%
  \afterassignment\dotolines
  \dimen0= }
\def\dotolines{\n=\dimen0
  \ifdim\dimen0<0pt \n=-\n \fi
```

```
\converttolines
\ifdim\dimen0<0pt \n=-\n \fi }
\def\converttolines{%
  \advance\n by\baselineskip
  \advance\n by-1
  \divide\n by\baselineskip }
```

(A brief explanation. The register `\n` stores the results of the calculation. The `\afterassignment` trickery is necessary so we may use the macro as `\tolines 1in` or `\tolines 4.5\baselineskip` without using braces around the argument to the macro. We add one scaled point less than an extra value of `\baselineskip` to make sure we always round up after the divide operation; normally, TeX's `\divide` operation truncates the fractional portion.) Positive values of `\n` refer to lines down the page, while negative values refer to lines up the page.

Nested Loops

Mr. Khodulev both poses and answers his own question. The problem: something seems wrong if you use nested loops in what would appear to be an obvious way, such as the following.

```
\newcount\iii \newcount\jjj
\iii=0
\loop
\jjj=0
\loop
\message{(\the\iii,\the\jjj) }
\advance\jjj by 1
\ifnum\jjj<3\repeat
\advance\iii by 1
\ifnum\iii<3\repeat
```

The naïve user would expect nine pairs of numbers, but you see only these: (0,0), (0,1), (0,2), (1,3), and (2,4). The difficulty vanishes when one surrounds the inner loop with braces:

```
\newcount\iii \newcount\jjj
\iii=0
\loop
{\jjj=0
\loop
\message{(\the\iii,\the\jjj) }
\advance\jjj by 1
\ifnum\jjj<3\repeat}
\advance\iii by 1
\ifnum\iii<3\repeat
```

(The output is nine pairs of numbers as you expect.) The `\loop` macro of plain TeX is not primitive, and its definition demands that inner loops be grouped as shown.

The moral: Always surround nested loops with curly braces.

Fontdimens and Physical Fonts

TeX associates a series of parameters with each font, and looks for these values in the *tfm* file. These font dimensions are accessible to a TeX user via the `\fontdimen` command. (Their significance is summarized in tables on pages 433 and 447 of *The TeXbook*.) Mr. Khodulev has uncovered some puzzling behavior when he tries to alter the `\fontdimens` for his own uses.

For the sake of concreteness, we will use `\fontdimen2`, which specifies the normal interword space for a font. Suppose you wanted to increase the interword space for a certain font in special places in the document. You might try (as he apparently did) something like the following.

```
\font\rm=cmr10
\font\specrm=cmr10
\fontdimen2\specrm=9.99pt
```

You might expect that when you typeset using `\rm`, you get the normal interword spacing (3.33 pt), while you would extra large spaces only when using `\specrm`. In fact, after the `\fontdimen` declaration above, any `\fontname` tied to the physical font `cmr10` has its `\fontdimen` changed. As if to add insult to injury, you cannot attempt to surround changes to `\fontdimen` within a group, since `\fontdimen` assignments are *always* global.

The following lines of code present one way of resolving the problem. The font definitions are encumbered with longer names than usual, but the actual mechanics of changing fonts are relegated to macros with names that closely resemble normal font calls. These macros have been designed to be used so the user thinks they are font calls, and the rare appearance of `\aftergroup` helps make this syntax possible.

```
%% First, fonts.
\font\roman=cmr10
\font\specroman=cmr10
%% Next, the special registers
\newdimen\savedvalue
\savedvalue=\fontdimen2\roman
\newdimen\specialvalue
\specialvalue=9.99pt
%% Finally, definitions.
\def\rm{%
\fontdimen2\roman=\savedvalue }
\def\specrm{%
\aftergroup\restoredimen}
```

```
\fontdimen2\specroman=\specialvalue
\specroman }
\def\restoredimen{%
\fontdimen2\roman=\savedvalue }
```

Mr. Khodulev did not specify his need in any more detail, so these macros should be revised as necessary. With these macros and definitions in force, the source text

```
\rm Here is some text.
{\specrm Here is some spaced out text.}
Here is more text, hopefully
back to normal.
```

```
\rm Here is more text.
\specrm Here is some spaced out text.
\rm Text is back to normal.
```

produces

Here is some text. Here is some spaced out text. Here is more text, hopefully back to normal.

Here is more text. Here is some spaced out text. Text is back to normal.

Bibliography

[1] Hoenig, Alan, "Line-Oriented Layout with TeX," in *TeX: Applications, Uses, Methods*, ed. Malcolm Clark. London: Ellis Horwood (1990).

◇ Alan Hoenig
17 Bay Ave.
Huntington, NY 11743 USA
(516) 385-0736
ajhjj@cunyvm.bitnet

Tutorials

The `\if`, `\ifx` and `\ifcat` Comparisons

David Salomon

Large, small, long, short, high, low, wide, narrow, light, dark, bright, gloomy, and everything of the kind which philosophers term accidental, because they may or may not be present in things,—all these are such as to be known only by comparison.

— Leon Battista Alberti

A general note: Square brackets are used throughout this article to refer to the TeXbook. Thus [209] refers to page 209, and [Ex. 7.7] to exercise 7.7, in the book. Advanced readers are referred to the actual WEB code by the notation [§495].

The ability to make decisions is a mandatory feature of any programming language, and TeX, being a programming language (with emphasis on typesetting), is no exception. There are 17 control sequences [209, 210] that compare various quantities, and they are used to make decisions and to implement loops. Most are easy to use even for beginners, but the three commands `\ifx`, `\if` and `\ifcat` are different. They are harder to learn, are executed in different ways, are intended for different applications, and are confusing. Hence this tutorial.

All three have the same syntax, and must follow one of the forms below

```
<command><comparands><then part>\else
    <else part>\fi

<command><comparands><then part>\fi

<command><comparands>\else<else part>\fi
```

They start with one of the commands `\if`, `\ifx` or `\ifcat`, and test their comparands, in a way that will be described later, to see if they agree or match. If the test is successful, the *then* part is executed; otherwise, the *then* part is skipped and the *else* part is executed. The two parts may consist of any tokens (control sequences, text, and even other *ifs*).

The two parts are optional. Any of them, or even both, may be omitted (in practice, of course, one never omits both). If the *else* part is omitted, the `\else`, of course, should be omitted as well. It may also happen that the process of evaluating the comparands creates extra tokens, which become included in the *then* part. The `\fi` is important. It serves to indicate the end of the *else* part or, in the absence of that part, the end of the *then* part. Even more important, in the case of nested *ifs*, there should be `\fis` to indicate the end of any of the inner `\ifs`, as well as that of the outer one.

`\ifx`

Of the three comparisons, `\ifx` is the simplest and most useful one. It compares its two comparands without looking too deep into their values or meaning. The test

```
\def\qwe{trip} \def\rty{trip}
\ifx\qwe\rty
  \message{yes}\else\message{no}
\fi
```

will compare the two macros and, since their definitions are the same, the *then* part will be executed, displaying ‘yes’ in the log file and on the terminal. Similarly, the test `\ifx\qwe\rty True\else False\fi` results in the tokens ‘True’.

A comparison always results in the expansion of either the *then* or the *else* parts. As mentioned before, each part may contain any tokens. Thus we may have, e.g.:

```
\baselineskip=\ifx\ab 24pt\else 36pt\fi
\pageno=\count\ifx\ab 0 \else 1 \fi
\message{\ifx\ab success\else failure\fi}
\ifx\ab true\else false\fi
\def\M{\ifx\ab yes\else no\fi}
```

Or even something more sophisticated, such as:

```
\newif\ifSome
\csname
  Some\ifx\ab true\else false\fi
\endcsname
\ifSome ...
```

The `\csname`, `\endcsname` pair creates one of the control sequences `\Sometrue`, `\Somefalse`, after which the test `\ifSome` is meaningful. Refs. 2 and 3 discuss `\csname`.

The following example is interesting. It shows the meaning of the words “...the *then* or *else* parts are expanded.”

```
\ifx\ab \x<argument> \else \y<argument> \fi
Depending on how \a, \b are defined, either \x or \y is expanded, and its argument used in the expansion. However, if the argument is left outside the \ifx, it is not used in expanding either macro:
```

```
\ifx\ab \x \else \y \fi <argument>
```

If `\x` is expanded, its argument will be the `\else`; if `\y` is expanded, its argument will be the `\fi`. This is easy to verify with `\tracingmacros=1`.

The macros compared may have parameters.

Thus

```
\def\qwe#1{samething}
\def\rty#1{samething}
\ifx\qwe\rty
```

evaluates to ‘yes’. This suggests one use for `\ifx` namely, comparison of strings. To compare two strings, place them in macros, and compare the macros. `\ifx` is, in fact, heavily used in ref. 1 for this purpose. However

```
\def\qwe#1{samething}
\def\rty{samething}
\ifx\qwe\rty
```

will result in ‘no’. Macros must have the same number of parameters to be considered equal by `\ifx`.

Can `\ifx` be used to compare a macro and a character? After defining `\def\aa{a}`, both tests `\ifx\aa*, \ifx*\aa` are, surprisingly, a failure. However, after defining `\def\aa{a}`, both comparisons `\ifx\aa\aa`, `\ifx\aa\aa` are successful.

A similar example is a test for a null macro parameter. A straight comparison `\ifx#1\empty...` does not work. We first have to define a macro `\inner` whose value is `#1`, and then compare `\ifx\inner\empty` (See definition of `\empty` on [351]).

```
\def\testnull#1\\{\def\inner{#1}
\ifx\inner\empty
  \message{yes}\else\message{no}
\fi}
```

The test ‘`\testnull \\`’ displays ‘yes’ on the terminal, while ‘`\testnull *\\`’ displays ‘no’.

To understand these results, we obviously need to know the rules for evaluating `\ifx`. They are numbered

- if both quantities being compared are macros, they should have the same number of parameters, the same top level definition, and the same status with respect to `\long` and `\outer`;
- in any other case, the quantities compared should have the same category code and the same character code.

A macro does not normally have either a character or a category code. However, for the purpose of rule 2, a macro is considered to have character code 256 and category code 16. So when a macro is compared to a character, they will not match.

Rule 1 implies that `\ifx` can be used to compare macros, but it does not expand them and does not look too deep into their meanings. They are considered equal if they look the same on the surface (see examples later). Rule 2 implies that `\ifx` can compare two characters, but not, e.g., a character and a string, or two strings. Thus

- `\ifx AA` is a match by rule 2.
- `\ifx A#1` can be used, inside a macro, to see whether the first parameter is the letter ‘A’.
- `\ifx Aa` is a failure since the comparands have different character codes.
- `\ifx {abc}{abc}` fails since it compares a ‘{’ to an ‘a’.
- `\ifx A{B}` fails since it compares the ‘A’ to the ‘{’.
- The test `\ifx*\aa` above fails because of rule 1.

With these rules in mind, the following discussions and examples are easy to understand.

To compare a macro `\aa` to a string `{abc}`, we first define `\def\b{abc}`, and then compare `\ifx\aa\b`. If the string is a parameter of a macro, we can say `\def\mac#1{\def\inner{#1} \ifx\aa\inner...}`. However, to compare `#1` to a single character, we can simply say

```
\def\mac#1{\ifx*#1...}
```

after which the expansion `\mac*` will be successful. There are some complex examples using this construct on [375–377].

To understand the meaning of ‘top level definition’, consider the following. Defining `\def\aa{*}` `\def\b{b}`, the test `\ifx\aa\b` is a success. However, the test

```
\def\aa{b} \def\c{d}
\def\b{tests} \def\d{tests}
\ifx\aa\c
```

is a failure, since `\ifx` compares only the top level definitions, and does not bother to expand `\b` and `\d` to find out that they are equal. Both tests

```
\def\qwe{\par} \def\rty{\par} \ifx\qwe\rty
and
```

```
\let\qwe=\par \let\rty=\par \ifx\qwe\rty
are successful, but
```

```
\let\xxx=\par \def\qwe{\par}
\def\rty{\xxx} \ifx\qwe\rty
```

fails, since `\ifx` does not expand its comparands to find their deep meaning.

This is an important feature of `\ifx` that has several consequences. One consequence is that `\ifx` is not bothered by undefined control sequences. It simply considers them all to be equal, so an `\ifx` comparison of two undefined control sequences always results in a match. Another consequence of the same feature is that two defined control sequences—such as e.g., `\if` and `\ifcat`—can be compared by `\ifx` without worrying about side effects resulting from their expansions. The comparison `\ifx\if\ifcat` is a failure, whereas `\ifx\if\if` is a success. It is also possible to compare the control sequence `\ifx` to itself, by means of `\ifx`. Thus `\ifx\ifx\ifx yes\else no\fi` results in ‘yes’. Note that the `\fi` matches the first `\ifx`, and the other `ifs` shouldn’t have any matching `\fis` since they are being compared, not executed. Consequently, the test `\ifx\fi\fi yes\else no\fi` also produces ‘yes’, as does `\ifx\message\message\message{yes} \else\message{no}\fi`

An interesting effect occurs when we try (perhaps as a serendipitous error)

```
\ifx\message \message{yes}\else
    \message{no}\fi
```

The `\ifx` compares the two tokens following, which are `\message` and `\message`. They are equal, so the word ‘yes’ is typeset. It is not displayed in the log file or on the terminal because the control sequence `\message`, which would normally have displayed it, has been used in the comparison. Continuing along the same lines, the test `\ifx \message{yes} \else \message{no}\fi` compares the control sequence `\message` to the ‘{’. They are not equal, because of rule 2 and, as a result, the `else` part is expanded, displaying ‘no’ in the log file, etc. Note that the string ‘yes’ becomes part of the `then` part, and is skipped.

The reader should be able, at this point, to easily figure out the results of the following tests:

- ```
\def\qwe#1{trip} \def\rty#1{trip}
1. \ifx\qwe\message{yes}\else\message{no}\fi
2. \ifx\qw\message{yes}\else\message{no}\fi
3. \ifx\q\message{yes}\else\message{no}\fi
4. \ifx\message{yes}\else\message{no}\fi
5. \ifx\message{yes}\else\message{no}\fi
6. \ifx message{yes}\else\message{no}\fi
7. \ifx mmessage{yes}\else\message{no}\fi;
```

Tests 1–3 are similar, the control sequences `\qwe`, `\qw`, `\q` are compared to `\message`, which results, of course, in a ‘no’. The undefined `\qw` and `\q` do not produce any errors. In 4, the control sequence `\\"` (which is normally undefined) is compared to the first ‘m’ of ‘message’ and, in 5, `\message` is compared to the ‘{’. Test 6 compares the first two letters ‘me’ of ‘message’ to each other. The 6 tests result in a ‘no’ being displayed in the log file.

Test 7 compares the first two letters ‘mm’ of ‘mmessage’ to each other. They are equal, so everything up to the `\else` is expanded. This results in the tokens ‘essageyes’.

#### **Exercise 1.** What are the results of:

- ```
\def\\{message}
(a) \ifx\\{message{yes}\else\message{no}\fi
(b) \ifx\\\\ yes\else no\fi
```

and why?

More about undefined control sequences. The test `\ifx\aa\b`, where `\aa`, `\bb` are undefined, results in a match. This means that all undefined control sequences have the same meaning. On the other hand, the test `\ifx\aa\relax`, where `\aa` is undefined, is a failure. This means that an undefined control sequence is not equal to `\relax` (at least not its upper level meaning).

However, when the name of an undefined control sequence is synthesized by a `\csname-\endcsname` pair, that control sequence is made equal to `\relax`. Thus if `\aa` is undefined, the construct `\csname aa\endcsname` (which creates the name `\aa`) is set equal to `\relax`, and the test `\expandafter\ifx\csname aa\endcsname\relax` is a success. This is the basis of [Ex. 7.7]. It describes a macro `\ifundefined` that determines if any given string is the name of a defined macro.

```
\def\ifundefined#1{%
    \expandafter\ifx\csname#1\endcsname\relax}
```

The test

```
\ifundefined a
    \message{yes}\else\message{no}\fi
```

displays ‘yes’ in the log file if `\aa` is undefined.

Exercise 2. Define a macro `\ifdefined#1` that will be the opposite of `\ifundefined` and be used in the same way.

What if `\aa` has been defined as `\relax`? Predictably, the test ‘`\let\aa=\relax \ifundefined a`’ is successful. It is (somewhat) more surprising that the test ‘`\def\aa{\relax} \ifundefined a`’ is a failure. This difference is a direct consequence of the difference between `\let` and `\def`. Following is a short discussion of that difference, which is important in advanced applications, where macros are defined and compared.

The general form of `\let` is

```
\let<control sequence>=<token>
```

It defines the control sequence as being identical to the token. This is similar, but not identical to, `\def<control sequence>{<token>}`, and the following illustrates that difference. After `\def\aa{X}` `\let\gg=\aa \def\hh{\aa}`, the sequence `\gg\hh` produces ‘XX’. If we now redefine `\aa`, the meaning of `\hh` will change (since it was defined by `\def`) but `\gg` will not change. Thus `\def\aa{*} \gg\hh` produces ‘X*’.

As a result, we can say that `\let\aa=\bb` assigns `\aa` that value of `\bb` which is current at the time the `\let` is executed, and this assignment is permanent. In contrast, `\def\aa{\bb}` assigns `\aa` the name `\bb`.

When `\a` is expanded, its expansion causes an expansion of `\b`, so the result is the value of `\b`. Each expansion of `\a` may, therefore, be different since `\b` may be redefined.

A more formal way of saying the same thing is: A `\let` makes a copy of the definition of `\b`, and that copy becomes the definition of `\a`; in contrast `\def` sets a pointer to point to the definition of `\b`, and that pointer becomes the definition of `\a`.

Back to `\ifx`. The comparands of an `\ifx` are not limited to just macros, primitives, or characters. They can also be:

- font names. `\font\abc=cmr10 \font\xyz=cmr10 \relax \ifx\xyz\abc` produces ‘yes’.
- Active characters (see [Ex. 7.3]). The result of `\let\aa=\ifx\aa` is a match.

Exercise 3. Why does `\def\aa{\~} \ifx\aa{\~}` fail?

- Names of the same TeX register. A test such as `\countdef\me=3 \countdef\you=3 \ifx\you\me` is a success.
- Macros defined at run time, such as in:

```
\def\tone{\count0=9 A }%
\message{Enter a definition}%
\read16 to\note
\ifx\tone\note
  \message{yes}\else\message{no}%
\fi
```

If the user enters ‘`\count0=9 A`’ from the keyboard, in response to the message, there will be a match. Entering anything else, such as ‘`\count0=9 a`’, will result in a failure. In either case the value of `\count0` will not be changed (by the way, what is it?), nor will the letters ‘A’ or ‘a’ be typeset. Notice that a message entered from the keyboard must terminate with a carriage return which, in turn, is converted by TeX into a space. This is why the definition of `\tone` must end with a space (to avoid that, change the value of `\endlinechar` as explained on [48]).

`\if`

The second comparison, `\if`, is executed in a completely different way. TeX expands the token following the `\if` (if it is expandable), then expands its expansion (if possible), and so on until only unexpandable tokens (characters or unexpandable control sequences) are left. If less than two unexpandable tokens are left, the process is repeated with the next input token. The process ends when there are two or more unexpandable tokens to be

compared, or when an `\else` or a `\fi` are encountered. The final result is a string of unexpandable tokens, the first two of which are compared by *character code* but not by category code. The rest of the tokens, if any, are added to the *then* part.

If a comparand is an unexpandable control sequence, rather than a character, it is assigned a character code 256 and a catcode of 16. Thus the tests `\if\hbox\vbox`, `\if\hskip\vskip`, `\if\hbox\kern`, succeed. (See [209] for exceptions regarding the use of `\let`.) This also implies that comparing a primitive to a character always fails.

There is also the case where evaluating the comparands results in just one unexpandable token. Such a comparison should not be used since its result is undefined. Unfortunately, no error message is given by TeX. The advanced reader is referred to [§495] for the details of such a case.

The first example is simple `\def\aa{*}`. Both tests `\if\aa*`, `\if*aa` are successful (compare with the similar `\ifx` test above).

After `\def\aa{\b}`, `\def\cc{\d}`, `\def\bb{**}`, `\def\dd{***}`, the test `\if\aa\cc` is a ‘yes’. However, `\def\aa{\b}`, `\def\cc{\d}`, `\def\bb{testing}`, `\def\dd{testing}`, `\if\aa\cc` will fail, since the two tokens compared are the first two characters resulting from the expansion of `\aa`, which are ‘te’. As mentioned above, the rest of `\aa` (the string ‘sting’) and the whole of `\cc` (the string ‘testing’) do not participate in the comparison, are added to the *then* part, and are therefore skipped. More insight into the working of `\if` is provided by the test

```
\def\aa{\b} \def\cc{\d}
\def\bb{testing} \def\dd{testing}
\if\aa\cc \message{yes}\else\message{no}\fi
```

It compares the first two t’s of `\aa`. They are equal, so TeX expands everything up to the `\else`. It displays ‘yes’, and also typesets the rest of `\aa` (‘sting’) and the whole of `\cc` (‘ttsting’). Note that, again, `\cc` is not used in the test.

Similar results are obtained in the experiment

```
\def\tone{*}
\message{Enter a}\read16 to\note%
\if\tone\note
```

Assuming that the user enters ‘*`\count90=89`’, the result will be a match, and `\count90` will also be set to 89. However, if the user enters ‘?`\count90=89`’, the comparison will fail, and `\count90` will not be affected. Similarly, if the user enters ‘*`abc`’, the comparison will be successful, and the string ‘`abc`’ will be typeset. Entering, ‘?`abc`’ however, will result in ‘no’, and the string ‘`abc`’ will be skipped.

The test `\if\s`, where `\s` is undefined, results in the message `! Undefined control sequence`, since `\if` always tries to expand its comparands.

Defining `\def\w{xyz}`, the test

```
\if x\w yes\else no\fi
```

is a success, since the first token of `\w` is an ‘x’. However, The other two tokens are added to the *then* part, and the result of the test is the string ‘zyzes’. Sometimes it is desirable to discard that part of `\w` that does not participate in the comparison. This is a special case of the general problem of how to extract the first token of a macro `\w` and discard the rest.

One way of doing it is:

```
\def\tmp#1#2\{#1}
\ifx\w\empty
\def\W{}
\else
\def\W{\expandafter\tmp\w\}
\fi
```

When `\W` is expanded, the first step is to expand `\w`, and the second, to expand `\tmp`. The first argument of `\tmp` is thus the first token of `\w`, and the second argument, the rest of `\w`. The result of expanding `\tmp` is thus the single token ‘x’, and that token becomes the definition of `\W`. The test `\if x\W yes\else no\fi` now results in the string ‘yes’. This method works even if `\w` is `\empty`.

Exercise 4. Perform the test:

```
\if\the\count90 \the\count90
\message{yes}\else\message{no}\fi
for \count90 set to 1, 11 and 12.
```

The next example is the two tests `\let\aa=\if\aa`, `\def\b{\aa} \if\b`. In the first test, the `\let` makes `\aa` equivalent to the active character ‘~’. In the second one, the `\def` makes `\b` a macro whose definition is the same active character ‘~’. The `\if` expands its comparands, so it ends up comparing ‘~’ to ‘~’. Both tests thus result in a match.

Having mentioned active characters, let’s use them to further illustrate the behaviour of `\if`. The following:

```
\def\aa{*~}
\hbox{Mr. Drofnats}
\hbox{Mr.\if*\aa\fi Drofnats}
\hbox{Mr.\if+\aa\fi Drofnats}
results in:
Mr. Drofnats
Mr. Drofnats
Mr.Drofnats
```

which is easy to explain. The test `\if*\aa\fi` expands `\aa` and only uses its first character (the ‘*’). The second character (the tilde) remains and affects the space between ‘Mr.’ and ‘Drofnats’ (it has the effect of `\frenchspacing`). In contrast, the test `\if+\aa\fi` expands `\aa` and, since there is no match, *skips* the second character. As a result, there is no space between ‘Mr.’ and ‘Drofnats’.

The `\noexpand` command can be used to suppress expansion during an `\if`. Assuming the definitions `\def\q{A}`, `\def\p{9}`, the test `\if\p\q` fails since it compares the characters ‘A’, ‘9’; however, the test `\if\noexpand\p\noexpand\q` is a success (even if the macros involved are undefined).

\ifcat

The third comparison, `\ifcat`, is less useful. It works like `\if`, expanding its comparands, and resulting in a string of characters, of which the first two are compared by category codes [37], but not by character codes. For example, the catcode of ‘&’ is 4 (alignment tab) and the catcode of ‘8’ is 12 (other). If we change the catcode of ‘8’ to 4 and compare `\catcode`8=4 \ifcat 8&`, we get a ‘yes’. It is hard, however, to find simple, practical examples for `\ifcat` (the examples on the notorious [377] are hardly simple or practical).

Similar to an `\if`, there is also the case where expanding the comparands results in a non-expandable control sequence, rather than a character. In such a case, `TEX` assigns it a character code 256 and a catcode of 16. Thus all the following comparisons `\ifcat\hbox\vbox`, `\ifcat\hskip\vskip`, `\ifcat\hbox\kern`, succeed. (Again, see [209] for exceptions concerning the use of `\let`.)

The category code of a character can be typeset by the command `\the\catcode`A`. It can be displayed in the log file by `\showthe\catcode`A`. This does not work for control sequences since they have no catcode. When comparing control sequences with an `\ifcat`, they are first expanded, and the first two tokens are compared. For example, after defining `\def\aa{\&} \def\bb{+} \def\cc{true}` the comparison `\ifcat\aa\bb` fails, since the catcodes of ‘&’ and ‘+’ are different. However, the comparison `\ifcat\bb\cc` is a ‘yes’ since the comparands are ‘+’ and ‘=’. The string ‘true’ is typeset.

It is possible to compare macros without expanding them. Assuming the definitions of `\aa`, `\bb` above, the test `\ifcat\noexpand\aa\noexpand\bb` results in a match since it does not expand the macros, and they are treated as undefined (category code 16).

Exercise 5. With `\c` defined as above, what is the result of `\ifcat\c`?

Perhaps the simplest practical example of `\ifcat` is a test for a letter. Assuming that the parameter of macro `\suppose` is supposed to be a letter or a string starting with a letter. The macro can be defined as: `\def\suppose#1{\ifcat A#1...\fi...}`.

Exercise 6. If the parameter of `\suppose` is a string, only the first character will be used by the `\ifcat`, and the rest will be added to the `then` part, perhaps interfering with the rest of the macro. Generalize the definition of `\suppose` to suppress the rest of the parameter during the `\ifcat`.

The examples

```
\def\af~} \ifcat\af~  
\let\b=~ \ifcat\b~
```

are identical to the ones shown earlier, in connection with `\if`. They behave the same as in that case, resulting both in a ‘yes’.

Active characters may also be compared with an `\ifcat`, since they all have the same catcode (13). After defining `\catcode`?=13 \def?{::}, \catcode`!=13 \def!{;}`, the test `\ifcat?!` is a success, seemingly confirming the above statement. A deeper look, however, shows that the test expands the two active characters, and compares the catcodes of their values! The values just happen to have the same catcode. To actually compare the catcodes of the active characters, a `\noexpand` should be used to prevent their expansions. Thus the test `\ifcat\noexpand?!\noexpand!` compares the catcodes of the active characters without expanding them (and is also a success).

Nested ifs

In principle, it is possible to nest `ifs` one inside another. An `if` may be a comparand of another `if`, or it may be nested in either the `then` or the `else` part of another `if`. However, because our three `ifs` work in different ways, not every combination of nested `ifs` is valid. In general, a nested `if` is written as

```
\if..\if<inner>\fi..\else..\if<inner>\fi..\fi
```

where any of the inner `ifs` may have an `else` part, and may itself be nested by other `ifs`. However, as the examples below show, such an `if` should be carefully analyzed before it is used, since it tends to produce unexpected results.

Since `\if` evaluates its comparands, they can be other `ifs`. Defining `\def\af{}, \def\b{**}`, the test

```
\if\ifx\af\b\else\if\af\b23\fi\fi\else4\fi
```

(see [Ex. 20.13g]) is an `\if` with an `\ifx` as a comparand. The `\ifx`, in turn, has another `\if` nested in its `else` part.

The process starts when the outer `\if` evaluates its comparands in order to come up with two tokens for comparison. It activates the `\ifx` which, in turn, compares `\af` and `\b`. They are not equal, so the ‘1’ is skipped, and TeX starts executing the `else` part of the `\ifx`. This part contains the inner `\if`, which evaluates `\af` and `\b`, compares the two asterisks, and results in the ‘23’. The outer `\if` is now equivalent to `\if23\else4\fi`, which typesets the ‘4’.

The test

```
\if\ifx\af\b\else\if\af\b22+\fi\fi\else3\fi
```

is similar, it has the ‘+’ left over after the comparison, so it gets typeset.

Exercise 7. What gets typeset by the following?

```
\if\ifx\b\b\else\if\af\b2\fi\fi+\else3\fi
```

The `\ifcat` comparison is similar to `\if` in that it first evaluates its comparands. As a result, other comparisons may be used as comparands, and may also be nested inside an `\ifcat`. The following tests can be analyzed similarly to the ones above:

```
\ifcat\ifx\af\b\else\if\af\b234\fi\fi\else5\fi  
\ifcat\ifx\af\b\else\if\af\b22+\fi\fi\else3\fi  
\ifcat\ifx\b\b\else\if\af\b2\fi\fi\else3\fi
```

Since `\ifx` does not evaluate its comparands, they cannot be other `ifs`. Trying, e.g., `\ifx\if\af\b...`, the `\ifx` would simply compare the `\if` to the `\af`. We cannot even use braces to separate the inner and outer `ifs` `\ifx{\if\af\b...}`..., since the `\ifx` will compare the ‘`{`’ with the `\if`, and TeX will eventually complain of an ‘Extra `{`’.

We can, however, nest an `if` (of any type) in the `then` or `else` parts of an `\ifx`. The test

```
\ifx\af\b\else\if\af\b ok\fi\fi
```

(with `\af`, `\b` defined as above) typesets ‘ok’. The `\ifx` compares `\af` and `\b` and finds them different. It skips to the `else` part and expands it. The inner `\if` is thus executed in the usual way; it finds two identical tokens (the two asterisks of `\b`), and typesets ‘ok’.

Examples

1. A practical example is macro `\flexins` below. It lets the user decide, *at run time*, whether any

floating insertion should be a `\midinsert` or a `\topinsert`. The user is prompted to enter either ‘mid’ or ‘top’ from the keyboard. In response, the macro uses a nested `\ifx` to create either a `\midinsert` or a `\topinsert`.

```
\def\flexins{%
  \def\b{mid } \def\d{top }
  \message{mid or top? }\read-1 to\ans
  \csname
    \ifx\ans\b mid%
    \else
      \ifx\ans\d top\fi
    \fi insert%
  \endcsname
}

\flexins
<Insertion material>
\endinsert
```

What if the user enters none of these inputs. Clearly `\flexins` should be extended so it can recover from a bad input. It is a good idea to expand `\flexins` recursively, in such a case, to give the user another chance to enter a valid input. The first try is:

```
\def\flexins{%
  \def\b{mid } \def\d{top }
  \message{mid or top? }\read-1 to\ans
  \csname
    \ifx\ans\b mid\else
      \ifx\ans\d top\else \flexins\fi
    \fi insert\endcsname
}
```

It does not work! When TeX expands `\flexins` recursively, it is still inside the `\csname`. During the recursive expansion it finds `\def\b`, but `\def` is not expandable, and thus not supposed to be inside a `\csname` [40]. The result is an error message (which one?).

We now realize that we have to delay the recursive expansion of `\flexins` until we get out of the `\csname`-`\endcsname` pair. The final version is:

```
\def\flexins{%
  \def\b{mid } \def\d{top }
  \def\badinsert{\flexins}
  \message{mid or top? }\read-1 to\ans
  \csname
    \ifx\ans\b mid\else
      \ifx\ans\d top\else bad\fi
    \fi insert\endcsname
}
```

In the case of bad input, the `\csname`-`\endcsname` pair creates the control sequence name `\badinsert`. We redefine it to simply expand `\flexins`, which then asks the user for another input.

2. (Proposed by R. Whitney.) This is a generalization of the previous example. Macro `\yesno` below prompts the user to respond with a ‘Y’ or a ‘N’, but also accepts the responses ‘y’, ‘n’. It does the following:

- Prompts the user with a question where the response can be ‘Y’, ‘N’, ‘y’, or ‘n’.
- Reads the response into `\ans`.
- Uses `\ifx` to compare `\ans` to macros containing one of the valid responses.
- If a match is found, uses `\csname` to create the name of, and expand, one of the macros `\yesresult`, `\noresult`. These macros should be predefined to do anything desirable.
- If no match is found, expands `\badresult`, which, in turn, should expand `\yesno` recursively.

```
\def\y{y } \def\n{N } \def\Y{Y }
\def\N{N } \def\badresult{\yesno}
\def\yesresult{(whatever)}
\def\noresult{(whatever)}
\def\yesno{%
  \message{Respond with a Y or N! }
  \read-1 to\ans
  \csname
    \ifx\ans\y yes\else
      \ifx\ans\Y yes\else
        \ifx\ans\n no\else
          \ifx\ans\N no\else bad%
        \fi
      \fi
    \fi
  \fi result\endcsname
}
\yesno
```

A different version of `\yesno` uses `\if` instead of `\ifx`. We start with:

```
\def\badresult{\yesno}
\def\yesresult{(whatever)}
\def\noresult{(whatever)}
\def\yesno{%
  \message{Respond with a Y or N! }
  \read-1 to\ans
  \csname
    \if\ans\y yes\else
      \if\ans\Y yes\else
        \if\ans\n no\else
          \if\ans\N no\else bad%
        \fi
      \fi
    \fi
  \fi result\endcsname
}
\yesno
```

```

    \fi
    \fi
    \fi
\fi result\endcsname
}

```

It compares `\ans` to the token 'y' instead of the macro `\y`, but it does not work! Macro `\ans` contains a 'y' (or 'Y' or whatever), followed by a space. The space gets added to the *then* part, which then becomes '`\yes`', creating the control sequence `\yesresult`. To get this to work, the first token of `\ans` has to be extracted, and all the other ones discarded. This can be done, as shown elsewhere, by

```

\def\tmp#1#2{\{#1}
\def\sna{\expandafter\tmp\ans\\}

```

Macro `\sna` now contains just one character, and the next version is:

```

\def\badresult{\yesno}
\def\yesresult{\(whatever)}
\def\noresult{\(whatever)}
\def\yesno{%
  \message{Respond with a Y or N! }
  \read-1 to\ans
  \def\tmp##1##2{\{##1}
  \def\sna{\expandafter\tmp\ans\\}%
  \csname
    \if y\sna yes\else
      \if Y\sna yes\else
        \if n\sna no\else
          \if N\sna no\else bad%
        \fi
      \fi
    \fi
  \fi result\endcsname
}

```

Note that it works for any response that's a string starting with one of the four valid characters.

Exercise 8. Extend this example. Define a macro `\triresponse` that accepts the responses 'left', 'right', 'center', or any strings that start with 'l', 'r', or 'c'. The macro then expands one of the (predefined) macros `\doleft`, `\doright`, `\docenter` or `\dobad`.

3. A practical example of the use of `\ifcat` arises when style files are used. If such a file has internal macros, they can be made private by declaring `\catcode`@=11`, and giving the macros names that include the '@'. At the end of the file, a matching `\catcode`@=12` should be placed. The problem occurs when such a style file, say `b.sty`, is `\input` by another file, `a.sty`, that also contains the

pair `\catcode`@=11, \catcode`@=12`. A simple test should reveal the problem to readers who still don't see it. The solution is to place the test

```
\ifcat @\chardef\catcount=12
```

```
\else
```

```
  \chardef\catcount=\catcode`@
```

```
\fi
```

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

at the beginning of `b.sty`, and reset at the end to `\catcode`@=\catcount`.

Exercise 9. Use `\ifcat` to solve the following problem: Given `\def\foo#1{...}`, devise a test to see if, in the expansion `\foo...`, the argument is delimited by a space. Normally, such a space is automatically absorbed by TeX and cannot be recognized.

4. A compound macro argument.

Macro `\compndArg` accepts a compound argument and breaks it down into its components. The argument should be of the form `xxx,xxx,...,xxx`; (the ',' separates the individual components and the ';' delimits the entire argument). The macro accepts the argument (without the ';', of course), it appends ';;' to the argument, and makes the whole thing the argument of `\pickup`, which is then expanded.

```

\def\compndArg#1;{\pickup#1;;}
\def\pickup#1,{%. Note that #1 may be \null
\if:#1\let\next=\relax
\else\let\next=\pickup
  \message{'#1'}% use #1 in any way
\fi\next}

```

Macro `\pickup` expects its arguments to be delimited by a comma, so it ends up getting the first component of the original argument. It uses it in any desired way and then expands itself recursively. The process ends when the current argument becomes the semicolon. Note the following:

- This is also an example of a macro with a variable number of parameters. The compound argument may have any number of components (even zero, see below).
- The method works even for an empty argument. The expansion '`\compndArg ;`' will cause `\pickup` to be expanded with a null argument.
- The macros do not create spurious spaces. In many macro lines, the end-of-line character gets converted to a space, which is eventually typeset if the macro is invoked in horizontal mode. Such lines should be identified, with a test such as `C\compndArg g;D`, and should be terminated by a

'%'. Try the above test with and without the '%' in the first line of \pickup.

Conclusion

The main source of the confusion surrounding the various \if comparisons is the inability to find out exactly what TeX is comparing. In future extensions of TeX it would be useful to have a control sequence \tracingcomparands such that setting \tracingcomparands=1 would show, on the terminal, the actual quantities compared.

Answers to exercises

1. (a) displays 'no' on the terminal since it compares the macro \\ to the letter 'm'; (b) typesets 'yes' since it compares two identical macros.

2. Macro \ifundefined supplies the \ifx, and the matching \else and \fi are provided outside. We want \ifdefined also to supply an if that can be completed outside. We start with the test for an undefined macro

```
\expandafter\ifx\csname#1\endcsname\relax
and create either an \iffalse (if the macro is undefined) or an \iftrue (in case it is defined), to be matched outside. The first version is:
```

```
\def\ifdefined#1{%
\expandafter\ifx\csname#1\endcsname\relax
\let\next=\iffalse\else\let\next=\iftrue\fi
\next}
```

But it fails! The reason is explained at the bottom of [211]. The next, working, version is:

```
\def\macaf{\let\next=\iffalse}
\def\macbf{\let\next=\iftrue}
\def\ifdefined#1{%
\expandafter\ifx\csname#1\endcsname\relax
\maca\else\macb\fi \next}
```

After which, we can say:

```
\ifdefined a \message{yes}\else
\message{no}\fi
```

3. Because \let\= defines \a as an active character, whereas \def\={\~} defines \a as a macro (whose value is the active character '\~'). The \ifx does not look too deep into the meaning of its comparands, so it decides that a macro is not equal to an active character. In contrast, the \if comparison, discussed later, which looks deeper into the meaning of its comparands, returns a 'yes' for both tests.

4. Just do it. It's worth it. Then do the similar test

```
\if\the\count90\the\count90
\message{yes}\else\message{no}\fi
```

5. A success, since it compares the catcodes of the two letters 't', 'r'. It also typesets 'ue'.

6. Macro \tmp expands to the first character of the parameter.

```
\def\suppose#1{\def\tmp##1##2{\{##1\}%
\ifcat A\tmp#1\...\else...\fi...}}
```

7. The \ifx compares \b and \b, and they, of course, match. The '1' is thus the first token left for the outer \if to compare. The rest of the \ifx (\else\if\b2\fi\fi) is skipped. Next comes the '+', followed by the else part of the outer \if, with the '3'. The outer \if can now be written as \if1+\else3\fi which, of course, typesets the '3'.

8. Answer not provided.

9. We place an expansion of \isnextspace at the end of \foo. This sets \next to the token following the parameter of \foo. The \ifcat can then be used to compare the category of \next to that of a space. The following test

```
\def\spacecheck{%
\ifcat\next\space
\message{yes}\else\message{no}\fi}
\def\isnextspace{\futurelet\next\spacecheck}
\def\foo#1{#1\isnextspace}
\foo{A} \foo{B}. \foo{C} D
```

produces 'yes no yes' on the terminal.

References

1. Greene, A. M., *BASiX—An Interpreter Written in TeX*, TUGboat 11 (1990), no. 3, pp. 385–392.
2. Bechtolsheim, S., \csname and \string, TUGboat 10 (1989), no. 3, pp. 203–206.
3. Hendrickson, A., Getting *TEXnical*, TUGboat 11 (1990), no. 3, pp. 359–370.

◊ David Salomon
 California State University,
 Northridge
 Computer Science Department
 Northridge, CA 91330
 dxs@cms.secs.csun.edu

Some tools for making indexes: Part I

Lincoln Durst

Three previous episodes in this series of tutorials appear in earlier issues of *TUGboat* [10#3 (November 1989, pages 390–394), 11#1 (April 1990, pages 62–68), and 11#4 (November 1990, pages 580–588)]. The present installment is more or less independent of its predecessors; the approach, however, is similar in spirit. In fact, some of the ideas here inspired development of tools described in the earlier pieces.

There are four or so main steps in making an index using *TeX* as, indeed, there are when one makes one by hand, as in the good old days. These steps are (1) selecting the items that seem to be reasonable candidates for index entries, (2) sorting them into the proper order (alphabetical order for words, numerical order for page numbers) or into something near to the proper order, (3) combining or eliminating duplicates and resolving inconsistencies between entries that may have come from widely separated parts of the book but that turn up near each other after the sort, and (4) arranging to have the index typeset the way an index is supposed to look. Times have not really changed; these steps, listed in the order they must be performed are still, in spite of technical advances, arranged in order of increasing difficulty.

It is an easy matter to mark the terms in the text that are reasonable candidates for the index and cause them to be written into a file in the order in which they appear in the text. Sorting that file is not as simple as sorting the file of macro names for entries in the bibliography since the terms are paired with page numbers and, unless you're careful or cunning, a simple ASCII sort of numerals, may give you things like

1 < 11 < 111 < 2 < 21...

(cognoscenti call this “lexicographic order”). After an automatic sort, there will be other problems as well. Any inconsistencies resulting from capitalization vs. noncapitalization, plural forms vs. singular forms, etc., will have to be resolved. And, finally, there is the problem of distinguishing between major entries and subsidiary ones.

It is a matter of taste whether the last question mentioned here should be considered first of all or last of all: The problem is philosophical since it comes down to whether such logical distinctions must be settled before one starts writing (this makes the problem of what is to be arranged or rearranged quite an abstract question) or whether it only makes sense to worry about the problem after the list of

items to be organized is at hand. Long-time readers of *TUGboat* may notice here an echo of the variant of the chicken-and-egg dilemma posed by Richard Southall over 6 years ago [5#2, November 1984, page 80] about whether it is desirable (or even possible) to create a manuscript before its printed version has been designed.

This installment is devoted to the first two steps discussed above: The creation of the file of reminders and some of the problems encountered in sorting it.

Creation of the file of reminders. Here we consider a simplified version of Knuth's indexing macros for *The TeXbook*, described there in Appendix E.*

Knuth makes provisions for four kinds of items in his index, of which

Arabic, by, \char, <dimen>

are examples. Ordinary mortals may be able to survive with one category of items for an index. It is surely easier to grasp what's happening if we restrict our attention to the first of Knuth's four cases. Once you understand the elementary case, please feel free to proceed on your own if you decide to write a book about *TeX*.

Knuth's scheme is to mark terms in the text file that he may want to include in the index (*The TeXbook*, pages 415–416, 423–424). One point here is that this marking can be done at the time of writing, or at any time during revision. Knuth refers to these selections as “index reminders”, since they will not automatically create the final entry; many will be edited during later stages of index construction. Suppose Gauss is mentioned in the text, then the word “Gauss” should be marked as a reminder, even though the entry in the index corresponding to it may in the end be expanded to

Gauss, Karl Friedrich (1777–1855)...

Knuth's way of marking a term in the text is to write `^{\{Gauss\}}` or `^{^{\{Gauss\}}}`: in the former case, if the word “Gauss” is to be printed in the text at that point and, in the latter case, if it is not to be printed in the text although it is being considered for the index. For example, the source file for FIGURE 1 contains the following passage:

* Other ways to make indexes have been discussed in earlier issues of *TUGboat*, beginning with the first issue, October 1980 (1#1, Winograd & Paxton, Appendix A, pages [1]–12) and, more recently, in the November 1989 issue (10#3, David Salomon, pages 394–400). Readers interested in alternatives to what we consider here may profitably consult those sources.

```
By means of ^{stops} the performer has
within his power a number of combinations
for varying the ^{tone} and
^{power, dynamic}
dynamic power.
```

In *The T_EXbook*, items of the second category, those with two "hats", ^{...}, are said to be "silent". (There is a reason, as we shall see below, for putting the silent entry just before the word(s) to which it applies; putting it on a separate line may make reading the text on the screen a little easier because the whole line can then be skipped.)

Knuth gives code that will write the marked items into a file (in the order of their appearance in the text), as well as code that provides the option of listing them in the margin of the page, such as shown in FIGURE 1. Here is a stripped-down version [cf. *The T_EXbook*, pages 415, 423]:

```
%%% index.rem, first fragment %%%
\newinsert\margin \dimen\margin=\maxdimen
\count\margin=0 \skip\margin=0pt
\newif\ifsilent \newwrite\inx
\immediate\openout\inx=\jobname.inx
\def\specialhat{%
  \ifmmode\def\next{\;}%
  \else\let\next=\beginxref
  \fi
  \next}
\catcode`^=\active
\let ^=\specialhat
```

First we have the allocation of an insert to hold the list in the margin. The dimension of the insert is allowed to be as long as the whole page (\maxdimen), but it occupies a vertical space of zero length (\count\margin=0) so that it takes no text area away from the current page: As we have seen before, T_EX can swallow, with no noticeable qualms, some assertions that might seem to be outrageously contradictory. [For more about inserts, which are covered with characteristic brevity in *The T_EXbook*, see David Salomon's tutorial in the November 1990 issue of *TUGboat* (11#4, pages 588–605).] Next, an \if-switch is allocated to distinguish between the one-hat and the two-hat items and then the file to hold these things is allocated and opened. ^ is ultimately made active and \specialhat is defined to replace it. It is, of course, necessary to account for the fact that ^ may perfectly well be used in math mode in its usual way, thus \specialhat is defined so that for \ifmmode true, \specialhat is merely ^, but for \ifmmode false, it becomes \beginxref, the next item on the agenda [*ibid.*, page 423]:

```
%%% index.rem, second piece %%%
\def\beginxref{\futurelet\next
\beginxrefswitch}
```

```
\def\beginxrefswitch{%
  \ifx\next\specialhat
    \let\next=\silentref
  \else
    \silentfalse\let\next=\xref
  \fi
  \next}
\def\silentref{\silenttrue\xref}
```

We step through this fast shuffle gingerly. At this point T_EX has just seen one ^ not in math mode and \next is now \beginxref. The immediate question that must be settled is whether there is another ^ just after the first one. Consider the two cases ^{Gauss} and ^{Gauss}: In the first case, the next token T_EX will find is {" and, in the second case ^", i.e., \specialhat. \futurelet (cf. *The T_EXbook*, page 207) permits T_EX to sneak a peek at that token and set \next equal to it; then \beginxrefswitch is expanded. If a second hat is seen, the macro \silentref removes the ^, sets the switch \ifsilent to true, and calls the macro \xref; in the other case, \xref is called but \ifsilent remains false. [The discussion of \futurelet on page 207 is certainly terse; readers interested in other examples of its use may wish to consult Stephan v. Bechtolsheim's tutorial in *TUGboat*, 9#3, December 1988, pages 276ff.] Anyway, what happens here is that the macro \beginxrefswitch is expanded first, after which the \ifsilent switch is properly set and \xref is called.

Now it is time for \xref:

```
%%% index.rem, third part %%%
\def\xref#1{\def\text{#1}%
  \let\next=\text\makexref}
\def\makexref{%
  \ifproofmode\insert\margin{%
    \hbox{\marginfont\text}}%
  \xdef\writeit{\write\inx{%
    \text\space!0\space
    \noexpand\number\pageno.}}%
  \writeit
  \else
    \ifhmode\kern0pt\fi
  \fi
  \ifsilent\ignorespaces\else\next\fi}
```

The first two lines here replace fifteen (much longer) lines of code in the middle of page 424, where the four-way branch is made to accomodate the four types of index entries Knuth has to juggle. Notice that two copies of the text in the brackets are made by \xref, one of them, \text, is for the index file and the marginal note, and the other, \next, is the item to appear in the text for the case \silentfalse. The steps just mentioned are carried out by \makexref. Note \space!0\space in the definition; this is used to separate the term

for the index from the number of the page on which it appears. The only relevant criterion for selecting a separator for this purpose is that it should be something that will never appear in a candidate for the index. Thus any sufficiently unlikely juxtaposition of characters will do. Knuth uses !0, !1, !2, and !3, for his four cases and our notation is consistent with his. The period after \pageno is there to mark the end of the reminder. If your page numbers include periods (see below), you should terminate the reminder with some other character here and later (e.g., :). Two subtle points here: (1) The \kern0pt suppresses hyphenation when not in proof mode, for consistency with the result in the insert case (cf. Appendix H of *The T_EXbook*, pages 454, 455). (2) \ignorespaces suppresses any space, explicit or implied, at the end of "``{Gauss}"", which means that with, say,

\$1+i\$ is a ``{Gauss} gaussian prime.

or

\$1+i\$ is a
``{Gauss}
gaussian prime.

in the text file, a page break cannot occur between ``{Gauss} and gaussian: These two items are firmly attached to one another (as if by superglue which, of course, has no stretch).

To complete the construction of FIGURE 1, we have to monkey with the output routine, in order to get the insert into the margin when we are in proof mode. Knuth's version is on page 416; ours is a bit simpler because we're not trying to do all that he must:

```
%%% index.rem, output routine %%%
\newdimen\pageheight \pageheight=\vsize
\def\onepageout{\shipout
  \vbox{\offinterlineskip
    \makeheadline\pagebody}\advancepageno}
\def\makeheadline{\vbox to 2pc{%
  \ifodd\pageno\rightheadline
  \else\leftheadline\fi\vfill}}
\def\pagebody{\vbox to\pageheight{%
  \ifproofmode\InsertNotes\fi
  \unvbox255}}
\output={\onepageout}
\newdimen\remkern \remkern=-8pc
\def\InsertNotes{\ifvoid\margin\else
  \rlap{\kern\remkern\vbox to 0pt
    {\box\margin\vs}}\fi}
```

This writes over the \output defined in plain.tex, replacing it by \onepageout; here \offinterlineskip pushes the two boxes (\makeheadline and \pagebody) together by eliminating the space between them, and \vbox255 is the box that holds a full page of text. \InsertNotes puts in the contents

of the insert \margin when there are any. Finally (more or less) we need fonts for the reminders in the margin:

```
%%% index.rem, reminders in margin %%%
\font\remfont=cmtt8 \font\strutfont=cmr9
\newbox\rembox
\setbox\rembox=\hbox{\strutfont }
\def\remstrut{\vrule height 1\ht\rembox
  depth 1\dp\rembox width 0pt}
\def\marginfont{\remstrut\remfont}
```

Knuth puts his index reminders in the right margin; in our case we choose the left one because the right side is already overworked. Conventional wisdom among professional designers of books and other printed material, based on readability studies, dictates that printed text should contain an average of about ten words per line or fewer. This works out to a maximum line measure (\hsize) for ten point type of about 24 picas, with as many as 30 picas only for twelve point type. It follows that one column of 10pt text on an 8.5" x 11" sheet should always have left and right margins whose combined width is close to 4", i.e., nearly half the width of the sheet; this leaves plenty of room for marginal notes on both sides of the text. See, for example, Southall, *ibid.*, page 86. See also *Designing with type*, by James Craig (New York, Watson-Guptill; London, Pitman; revised edition 1980), page 128. *Typography: How to make it most legible*, by Rolf F. Rehe (Carmel, Indiana, Design Research International; fifth edition, 1984) has an extensive bibliography of the research reports of the relevant legibility studies.*

As usual, we have something for prepare.tex:

```
\newif\ifIndRemMark \IndRemMarkfalse
\def\IndexRemindersMarked{\IndRemMarktrue}
\newif\ifproofmode \proofmodefalse
\def\WriteIndexReminders{\proofmodetrue}
```

and something for compose.tex:

```
\ifIndRemMark\input index.rem\fi
```

There are two more options here for the driver file:

```
\IndexRemindersMarked
\WriteIndexReminders
```

The first option inputs index.rem in the composition run, in order to cope with all the \specialhats, and the second writes the reminders into the file and displays them in the margin at the time the index is to be prepared just prior to the final composition run. (prepare.tex, compose.tex, and driver files

* Can it be possible that there are graduate schools in the U. S. whose policies require that theses be presented in a form intended to make them difficult, tedious, or impossible to read? Anti-dissemination, surely!

HARPSICHORD**15**

Pianoforte
 strings, 2, 3, or 4
 strings
 case, harp shape
 case, wing shape
 pianoforte, grand
 harpsichord
 strings
 stops
 tone
 power, dynamic
 strings
 struck, by tangents
 tangents, See struck
 struck, by hammers
 hammers, See struck
 strings, plucked
 quill
 strings, twanging of

HARPSICHORD, **HARPSICON**, **DOUBLE VIRGINALS** (Fr. *clavecin*; Ger. *Clavicymbel*, *Kiel-Flügel*; Ital. *arpicordo*, *cembalo*, *clavicembalo*, *gravecembalo*; Dutch, *clavisinbal*), a large keyboard instrument (see **PIANOFORTE**), belonging to the same family as the virginal and spinet, but having 2, 3, or even 4 strings to each note, and a case of the harp or wing shape, afterwards adopted for the grand pianoforte. J. S. Bach's harpsichord, preserved in the museum of the Hochschule für Musik at Charlottenburg, has two manuals and 4 strings to each note, one 16 ft., two 8 ft. and one 4 ft. By means of stops the performer has within his power a number of combinations for varying the tone and dynamic power. In all instruments of the harpsichord family the strings, instead of being struck by tangents as in the clavichord, or by hammers as in the pianoforte, are plucked by means of a quill firmly embedded in the centred tongue of a jack or upright placed on the back end of the key-lever. When the finger depresses a key, the jack is thrown up, and in passing the crow-quill catches the string and twangs it. It is this twanging of the string which pro-

FIGURE 1 A

16**HARPY**

tone, brilliant
 tone, incisive
 power of expression
 accent
 pianoforte
 harpsichord
 orchestra
 harpsichord makers
 Ruckers
 Antwerp, See Ruckers
 Antwerp

duces the brilliant incisive tone peculiar to the harpsichord family. What these instruments gain in brilliancy of tone, however, they lose in power of expression and accent. The impossibility of commanding any emphasis necessarily created for the harpsichord an individual technique which influenced music composed for it to so great an extent that it cannot be adequately rendered upon the pianoforte.

The harpsichord assumed a position of great importance during the 16th and 17th centuries, more especially in the orchestra, which was under the leadership of the harpsichord player. The most famous of all harpsichord makers, whose names form a guarantee for excellence were the Ruckers, established at Antwerp from the last quarter of the 16th century. (K. S.)

[The writer is Kathleen Schlesinger, editor of *The Portfolio of Musical Archaeology* and author of *The Instruments of the Orchestra*. This text appears in *The Encyclopædia Britannica*, Eleventh Edition, Cambridge, England, 1910, Volume XIII, pages 15, 16.]

FIGURE 1 B

are discussed in the second of the tutorials in this series, cited above.)

A word of warning: In order to keep things simple, avoid all non-letters in reminders, except for ordinary punctuation, or you will have a terrible time when you try to sort the file. Do not use any diacritical marks, ties (~), or control sequences of any kind, and avoid math mode at all costs. These omissions may be corrected when the file for the index is brought to its final form. If necessary, put

```
... described in detail in his
~~{Heiligenstaat Testament}
Hei\li\gen\staat Test\-\-ment. ...
```

(See also the Postscript, below.)

Sorting the file of reminders. The first thing that must be done to the file of reminders is to sort it in order to get things into alphabetical order. If brevity has blessed you with no page numbers having more than one digit (or if, for some reason, they all happen to have the same number of digits), the problem is simple; use the utility sort that came with your operating system. Otherwise, heavier artillery may be indicated.

Kernighan & Ritchie, in *The C programming language* (second edition), Prentice-Hall 1988, show (section 5.6, pages 108–110) how to sort a file using Hoare's quick sort. If you prefer Shell's sort to Hoare's, see Harbison & Steele, *C: A reference manual* (second edition), Prentice-Hall 1987, pages 210, 211, for the souped-up Knuth-Sedgewick-Harbison-Steele version and incorporate that into the K&R code. It is not hard in either case to modify the code so that the parts of the lines preceding “!0” are sorted alphabetically and the parts following “!0” are sorted numerically. The trick is to change the criterion for swapping to require that two lines are to be interchanged if the alphabetic strings (a) are in the wrong order or (b) they are the same, but the page numbers are in the wrong order. If you are pig-headed enough to insist on page numbers having the form 1-12 [IBM, here 1-2 < 1-12], 2.17 [Zenith, here 2.2<2.17], K-9 [PCTEX], etc., you deserve the consequences.*

As hinted above, there's another way to skin the cat, if your page numbers are plain old natural numbers. Fortunately for us, ASCII codes (unlike

telephone dials and typewriter keyboards) are set up so that 0 precedes all the other digits. Hence by padding \pageno with enough initial zeros to give all page numbers the same number of digits, we can force lexicographic order to coincide with numerical order. You can use the following code, unless you have a thousand pages or more (if you do, you're the only one to be blamed),

```
\def\Pageno{%
  \ifnum\pageno>99
    \noexpand\number\pageno
  \else
    \ifnum\pageno>9
      \noexpand\number\pageno
    \else
      \noexpand\number\pageno
    \fi\fi}
```

and put “\Pageno.” in the definition of \makexref instead of “\noexpand\number\pageno.” With this change, you can use the utility sort that came with your operating system, and after you've done the sorting you can arrange to drop the, by then, superfluous zeros.

Postscript. On another occasion, we propose to consider steps one may take to convert the sorted file of reminders into a respectable index. One must expect that some “hand-work” will be necessary, if only to insert middle names, dates, or other details the author considers relevant or important. But there is quite a bit of routine work that may be automated. In addition, with some cunning, it is even possible to overcome the restrictions imposed so far that prohibit diacritical marks and other control sequences within reminders.

Note. The current version of the disk referred to in previous tutorials in this series contains the files **index.rem** and the source for FIGURE 1, in addition to the things mentioned earlier.

It is a pleasure, as always, to acknowledge the help, guidance, and encouragement, of those who have provided any of the above. In addition to all the usual suspects (cited here several times before), I must acknowledge my gratitude to Costa Mylonas, of Brown University and Athens, Greece, who challenged me to produce a collection of flexible tools, preferably simple ones, that might prove useful for authors who want to sit down at their computers and write books. These tutorials are a result of that challenge. I hope that others find them useful or, at the very least, instructive.

- ◊ Lincoln Durst
46 Walnut Road
Barrington, RI 02806
lkd@math.ams.com

* A lesser challenge is generated by typographical errors in the passages cited at the beginning of this paragraph. In K&R's code, replace **maxlines** by **MAXLINES**; in H&S's explanation, replace “smallest number” by “largest number”. (The errors appear in at least the first printings of these two second editions.)

The structure of the **T_EX** processor

Victor Eijkhout

The inner workings of **T_EX** are explained by its author [1] in terms of an analogy with the digestive tract. Apart from the fact that this gives rise to a whole genre of jokes¹, the analogy becomes definitely strained when regurgitation takes place in the mouth, or when the eyes take part in the process.

In this article² I will describe the **T_EX** processor as a multi-layered engine that successively transforms characters into tokens, tokens into lists, and from these lists builds a typeset page.

Four **T_EX** processors

The way **T_EX** processes its input can be viewed as happening on four levels. One might say that the **T_EX** processor is split into four separate units, each accepting the output of the previous stage, and delivering the input for the next stage. The input of the first stage is then the **tex** input file; the output of the last stage is a **dvi** file.

For many purposes it is most convenient and insightful to consider these four levels of processing as happening after one another, each one accepting the *completed* output of the previous level. In reality this is not true: **T_EX** is not something like a four-pass compiler. All levels are simultaneously active, and there is interaction between them.

The four levels are

1. The input processor. This is the piece of **T_EX** that accepts input lines from the file system of whatever computer **T_EX** runs on, and turns them into tokens. These are typically character tokens that comprise the typeset text, and control sequence tokens that are commands to be processed by the next two levels.
2. The expansion processor. A number of tokens generated in the first level – macros, conditionals, and a number of primitive **T_EX** commands – are subject to expansion. Expansion is the process that replaces some (sequences of) tokens by another (possibly empty) sequence.
3. The execution processor. Control sequences that are not expandable are executable, and

¹ Tokens being ‘sicked up again’ [2], output being ‘**T_EXrement**’ [3], or the particularly deplorable title of [4] ...

² This is basically the first chapter from my book, called (tentatively) ‘A **T_EXnician’s Reference Guide**’, and which is to appear with Addison-Wesley late this year.

this execution takes place on the third level of the **T_EX** processor.

One part of the activity here concerns changes to **T_EX**’s internal state: assignments and macro definitions are typical activities in this category. The other thing happening on this level is the construction of horizontal, vertical, and mathematical lists.

4. The visual processor. In the final level of processing the visual part of **T_EX** processing is performed. Here horizontal lists are broken into paragraphs, vertical lists are broken into pages, and formulas are built out of math lists. Also the output to the **dvi** file takes place on this level. The algorithms working here are not accessible to the user, but they can be influenced by a number of parameters.

1 The input processor

The input processor is that part of **T_EX** that translates whatever characters it gets from the input file into tokens. The output of this processor is a stream of tokens: a token list. Most tokens fall into one of two categories: character tokens and control sequence tokens. The remaining category is that of the parameter tokens; these will not be treated here.

1.1 Character input

For simple input text, characters are made into character tokens. However, **T_EX** can ignore some input characters: a row of spaces in the input is usually equivalent to just one space. Also, **T_EX** itself can insert tokens that do not correspond to any character in the input, for instance the space token at the end of an input line, or the **\par** token after an empty line.

Not all character tokens represent characters that are to be typeset. Characters fall into sixteen categories – each one specifying a certain function that a character can have – of which only two contain the characters that will be typeset. The other categories contain such characters as **{**, **}**, **&**, and **#**. A character token can be considered as a pair of numbers: the character code – usually the ASCII code – and the category code. It is possible to change the category code that is associated with a particular character code.

When the escape character **** appears in the input, **T_EX**’s behaviour in forming tokens is more complicated. Basically, **T_EX** builds a control sequence by taking a number of characters from the input and lumping them together into a single token.

The behaviour with which **T_EX**’s input processor reacts to category codes can be described as

a finite-state automaton with three internal states: N , new line, M , middle of line, and S , skipping spaces. These states and the transitions between them are treated in chapter 8 of *The T_EXbook*.

1.2 Two-level input processing

T_EX's input processor is in fact even a two-level processor. Due to limitations of the terminal, the editor, or the operating system, the user may not be able to input certain desired characters. Therefore, T_EX provides a mechanism to access with two superscript characters all of the available character positions. This may be considered a separate stage of T_EX processing, taking place prior to the three-state finite automaton mentioned above.

For instance, the sequence $\wedge +$ is replaced by k because the ASCII codes of k and $+$ differ by 64. Since this replacement takes place before tokens are formed, writing $\backslash vs \wedge + ip 5cm$ has the same effect as $\backslash vskip 5cm$. Examples more useful than this exist.

Note that this first stage is a transformation from characters to characters, without considering category codes. These come into play only in the second phase of input processing, where *characters* are converted to *character tokens* by coupling the category code to the character code.

2 The expansion processor

T_EX's expansion processor accepts a stream of tokens and, if possible, expands the tokens in this stream one by one until only unexpandable tokens remain. Macro expansion is the clearest example of this: if a control sequence is a macro name, it is replaced (together possibly with parameter tokens) by the definition text of the macro.

Input for the expansion processor is provided mainly by the input processor. The stream of tokens coming from the first stage of T_EX processing is subject to the expansion process, and the result is a stream of unexpandable tokens which is fed to the execution processor.

However, the expansion processor comes into play also when an `\edef` or `\write` is processed. The parameter token list of these commands is expanded as if the lists would have been on top level, instead of the argument to a command.

There is a special fascination to macros that work completely by the expansion processor. See the recent articles [4], [5], and [6] for some good examples.

2.1 The process of expansion

Expanding a token comprises the following steps:

- See if the token is expandable.
- If the token is unexpandable, pass it to the token list currently being built, and take on the next token.
- If the token is expandable, replace it by its expansion. For macros without parameters, and a few primitive commands such as `\jobname`, this is indeed a simple replacement. Usually, however, T_EX needs to absorb some argument tokens from the stream in order to be able to form the replacement of the current token. For instance, if the token was a macro with parameters, sufficiently many tokens need to be absorbed to form the arguments corresponding to these parameters.
- Go on expanding, starting with the first token of the expansion.

Deciding whether a token is expandable is usually a simple decision. Macros and active characters, conditionals, and a number of primitive T_EX commands (see the list on page 215 of *The T_EXbook*) are expandable, other tokens are not. Thus the expansion processor replaces macros by their expansion, it evaluates conditionals and eliminates any irrelevant parts of these, but tokens such as `\vskip` and character tokens, including characters such as dollar signs and braces, are passed untouched.

2.2 Special cases: `\expandafter`, `\noexpand`, and `\the`

As stated above, after a token has been expanded T_EX will start expanding the resulting tokens. At first sight the `\expandafter` command would seem to be an exception to this rule, because it expands only one step. What actually happens is that the sequence

`\expandafter<token1><token2>`

is replaced by

`(token1)expansion of token2`

and this replacement is in fact reexamined by the expansion processor.

Real exceptions do exist, however. If the current token is the `\noexpand` command, the next token is considered for the moment to be unexpandable: it is handled as if it were `\relax` (more about this control sequence follows below), and it is passed to the token list being built.

Example: in the macro definition

`\edef\aa{\noexpand\bb}`

the replacement text `\noexpand\bb` is expanded at definition time. The expansion of `\noexpand` is the next token, with a temporary meaning of `\relax`. Thus, when the expansion processor tackles the next

token, the `\b`, it will consider that to be unexpandable, and just pass it to the token list being built, which is the replacement text of the macro.

Another exception is that the tokens resulting from `\the<token variable>` are not expanded further if this statement occurs inside an `\edef` macro definition.

2.3 Braces in the expansion processor

Above, it was said that braces are passed as unexpandable character tokens. In general this is true. For instance, the `\roman numeral` command is handled by the expansion processor; when confronted with

```
\roman{number}\count2 3{4 ...}
```

TEX will expand until the brace is encountered: if `\count2` has the value of zero, the result will be the roman numeral representation of 103.

As another example,

```
\iftrue {\else }\fi
```

is handled by the expansion processor as if it were
`\iftrue a\else b\fi`

The result is a character token, be this a brace or a letter.

However, in the context of macro expansion the expansion processor will recognize braces. First of all, a balanced pair of braces marks off a group of tokens to be passed as one argument. If a macro has an argument

```
\def\macro#1{ ... }
```

one can call it with a single token

```
\macro 1 \macro \$
```

or with a group of tokens, surrounded by braces

```
\macro {abc} \macro {d{ef}g}
```

Secondly, when the arguments for a macro with parameters are read, no expressions with unbalanced braces are accepted. In

```
\def\{a#1\stop{ ... }
```

```
\{a bc{d\stop}e\stop
```

the argument is `bc{d\stop}e`. Only balanced expressions are accepted here.

3 The execution processor

The execution processor builds lists: horizontal, vertical, and math lists. Corresponding to these lists, it works in horizontal, vertical, or math mode. Of these three modes ‘internal’ and ‘external’ variants exist. In addition to building lists, this part of the *TEX* processor also performs mode-independent processing, such as assignments.

Coming out of the expansion processor is a stream of unexpandable tokens to be processed by

the execution processor. From the point of view of the execution processor, this stream contains two types of tokens:

- Tokens that signal an assignment (this includes macro definitions), and other tokens that are independent of the mode, such as `\show` and `\aftergroup`.
- Tokens that build lists: characters, boxes, and glue. Handling of these tokens depends on the surrounding mode.

Some objects can be used in any mode; for instance boxes can appear in horizontal, vertical, and math lists. The effect of such an object will of course still depend on the mode. Other objects are specific to one mode. For instance, characters (to be more precise: character tokens of categories 11 and 12) are intimately connected to horizontal mode: if the execution processor is in vertical mode when it encounters a character, it will switch to horizontal mode.

For the expansion processor a character token is just an unexpandable object. On the level of the execution processor, however, something is actually done with it. Some characters are typeset, but the execution processor can also encounter, for instance, math shift characters (usually \$), or braces. When a math shift character is found in the stream of tokens, math mode is entered (or exited if the current mode was math mode); when a left brace is found, a new level of grouping is entered.

One control sequence handled by the execution processor deserves special mention: `\relax`. This control sequence is not expandable, but the execution is ‘empty’. Compare the effect of `\relax` in

```
\count0=1\relax 2
```

with that of `\empty` defined by

```
\def\empty{}
```

in

```
\count0=1\empty 2
```

In the first case the expansion process that is forming the number stops at `\relax` because it is unexpandable, and the number 1 is assigned. In the second case `\empty` expands to nothing, so 12 is assigned.

4 The visual processor

TEX’s visual processor encompasses those algorithms that are outside direct user control: paragraph breaking, alignment, page breaking, math typesetting, and *dvi* file generation. Various parameters control the operation of these parts of *TEX*.

Some of these algorithms return their results in a form that can be handled by the execution proces-

sor. For instance, a paragraph that has been broken into lines is added to the main vertical list as a sequence of horizontal boxes with intermediate glue and penalties. Also, the page breaking algorithm stores its result in `\box255`, so output routines can dissect it. On the other hand, a math formula can not be broken into pieces, and, of course, shipping a box to the `dvi` file is irreversible.

5 Further examples

5.1 Skipped spaces

Skipped spaces provide an illustration of the view that `TEX`'s levels of processing accept the completed input of the previous level. Consider the commands

```
\def\af{\penalty200}
\af 0
```

Faulty reasoning

"The `\af` is encountered, expanded, the space then delimits the number"

would lead to the conclusion that this is equivalent to `\penalty200 0`. It is not. Instead, what results is

```
\penalty2000
```

because the space after `\af` is skipped in the input processor.

5.2 Internal quantities and their representations

`TEX` uses various sorts of internal quantities, such as integers and dimensions. These internal quantities have an external representation, which is a string of characters, such as `4711` or `91.44cm`.

Conversions between the internal value and the external representation take place on two different levels, depending on the direction the conversion goes. A string of characters is converted to an internal value in assignments such as

```
\pageno=12 \baselineskip=13pt
```

or statements like

```
\vskip 5.71pt
```

and all of these statements are handled by the execution processor.

On the other hand, the conversion of the internal values into a representation as a string of characters is handled by the expansion processor. For instance,

```
\number\pageno \romannumeral\year
\the\baselineskip
```

are all processed by expansion.

Note that in the `\baselineskip` example above the conversion from string of characters to internal

value was 'automatic'. The conversion the other way has to be forced by a command such as `\number`. Thus there is no danger that the sequence

```
\pageno=3 \count\MyCount=\pageno 5
```

will result in assigning either 15 or 35 to `\MyCount`.

As a final example, suppose `\count2=45`, and consider the statement

```
\count0=1\number\count2 3
```

The expansion processor tackles `\number\count2` to give the characters 45, and the space after the 2 is absorbed because it only serves as a delimiter of the number of the `\count` register. In the next stage of processing, the execution processor will then see the statement

```
\count0=1453
```

and execute this.

6 Conclusion

`TEX` is harder to understand than most programming languages. One reason for this is that the '`TEX` processor' consists of more than one level. In this article I have identified four levels of processing in `TEX`, and described what goes on on what level. Often the key to understanding `TEX`'s behaviour is to consider the four levels as working not simultaneously, but one after the other.

References

- [1] Donald Knuth, *The T_EXbook*, Addison-Wesley Publishing Company, 1984.
- [2] Angela Barden, Some `TEX` manuals, *TUGboat* 12(1991), no. 1, 166–170.
- [3] Ron Whitney, private communication.
- [4] Victor Eijkhout, Oral `TEX`, *TUGboat* 12(1991), no. 2, 272.
- [5] Alan Jeffrey, Lists in `TEX`'s mouth, *TUGboat* 11(1990), no. 2, 237–245.
- [6] Sonja Maus, An expansion power lemma, *TUGboat* 12(1991), no. 2, 277.

◇ Victor Eijkhout
Center for Supercomputing
Research and Development
University of Illinois
305 Talbot Laboratory
104 South Wright Street
Urbana, Illinois 61801-2932, USA
eijkhout@csrd.uiuc.edu

Warnings

Initiation rites

Barbara Beeton

Initial conditions always have to be looked at carefully if one wants to avoid nasty surprises, a fact known all too well to students of fluid dynamics, programmers, and other assorted technoids. Boundaries between TeX mode changes and other TeX structures are no exceptions to this rule.

Paragraph beginnings

Automatic hanging indent. There are times when one wants several successive paragraphs to be hanging indented. It's a great nuisance to have to type the instructions for a hanging indent into every paragraph, and though it's easy enough to make a simple substitution macro

```
\def\X{\par\noindent
      \hangindent\parindent\hangafter1 }
using it at the beginning of every paragraph is also a nuisance.
```

But wait—there's `\everypar`. Why can't we use that to make the hanging indented style automatic? Most of the time we can, but there are occasional problems, as the following example shows.

```
% for a test, make it small and loose

\parindent=0sp \hsize=2.5in
\hfuzz=2pc \parskip=.5pc

\everypar{\hangindent.5in \hangafter=1 }

{\it H}owdy boys and girls.
How are you doing today?

% But, wow! A strut fixes it!
\strut {\it H}owdy boys and girls.
How are you doing today?

% Also, not nesting it works
\it H\rm owdy boys and girls.
How are you doing today?

% Ordinary paragraphs work fine too.
Well, now is the time for all good men
to come to the aid of their party.
```

Here's what the output looks like.

Howdy boys and girls. How are you
doing today?

Howdy boys and girls. How are you
doing today?

Howdy boys and girls. How are you
doing today?

Well, now is the time for all good
men to come to the aid of
their party.

Not exactly what we had in mind.

Why has the hanging indent disappeared from the first paragraph? It turns out that TeX applies `\everypar` immediately after it encounters a token in vertical mode that will cause it to shift into horizontal mode (*The TeXbook*, p. 105). In this example, the initial "H" is such a token. Unfortunately, this is inside a group, and all the default conditions are restored upon exiting the group; the transcript of a trace (slightly edited) shows this very clearly:

```
{begin-group character {}}

\it ->\fam \itfam \tenit
{\fam}
{select font cmti10}
{the letter H}
\everypar->\hangindent .5in \hangafter =1
{horizontal mode: \hangindent}
{\hangafter}
{the letter H}
{end-group character }}           ***
{restoring \hangafter=1}
{restoring \hangindent=0.0pt}
{restoring current font=\tenrm}
{restoring \fam=0}
{the letter o}
```

Here are some more things that won't work, because they don't put TeX into horizontal mode:

- `\relax`;
- an empty `\hbox` or `\vbox`; this has an additional nasty side effect—it leaves a “blank line” above the paragraph that is occupied by nothing but the empty box;
- an `\hrule` with `width0pt`, with an effect similar to that of an empty box; in this case, extra blank space depends on the height of the `\hrule`.

And here are several more things that *do* work:

- `\leavevmode` (the most neutral option);
- `\noindent`;
- `\indent` (because `\parindent=0sp`);

- a `\vrule` with `width0pt`;
- math mode with no contents: `$ $`; just be sure not to leave any space between the closing dollar sign and the first word of the sentence, or it will be there in the output.

Although this is a warning about beginnings, it seems prudent to put in just a few words about endings, at least of hanging indented paragraphs.

A paragraph doesn't actually end until *TEX* comes to `\par`, `\endgraf`, or some implicit shift to vertical mode, e.g. `\vskip`. If a paragraph that's supposed to be hanging indented is in a group, and the group ends before the shift into vertical mode, the hanging indentation is lost. *TEX* only applies certain "shape" adjustments (among them `\leftskip`, `\rightskip`, `\baselineskip`, and the like) upon the transition to vertical mode, so whatever those settings are at that transition, those are the ones that take effect.

Paragraphs that begin with boxes. This has already been touched on above. Neither an `\hbox` nor a `\vbox` will cause *TEX* to enter horizontal mode. (If *TEX* is already in horizontal mode, it will stay that way.) Instead, the box is set on a line by itself, at the absolute left margin. If you are using `\llap` to tack something on to the left of the first word in a paragraph, you should remember to precede it by `\leavevmode`; the answer to exercise 14.28 in *The T_EXbook* works because the first command activated by `\strut` is `\unhcopy`, which causes *TEX* to go into horizontal mode, and the answer to exercise 14.29 works because `\everypar` is not invoked until *TEX* is already in horizontal mode.

Table cells

The start of data in a cell of a table can also be an invitation for something to go wrong. The reason is simple: *TEX* will always expand the first token to see if it is `\noalign` or `\omit`, to make sure that items that must be processed before the template, or that ignore it, are handled properly. *The T_EXbook* (p. 240) tells us that macros are expanded until the next non-space token is found; if it is not `\noalign` or `\omit`, it will be put back to be read again after the first part of the template has been read. However, the order of expansion will now be something other than intended, and for some macros whose proper application depends on their being expanded in a particular order, the results may be unexpected.

An example from the *AMSFONTs User's Guide* will illustrate the problem. I wanted to present the cyrillic transliteration conventions in the form of a table. Since two columns (upper- and lowercase) would always be cyrillic, it seemed obvious that the cyrillic command should be put into the template. Here's what I tried first.

```
\halign{\cyr#\hfil\quad\# \hfil... \cr
      \u\i & \u\i ... \cr
      ... }
```

What I wanted was

и ѿ

What I got was

и ѿ

This only makes sense if you happen to know the layout of the cyrillic font, and even then not much sense at first.

Here's what happened. `\u` is expanded and determined to be an accent. Unfortunately, `\cyr` in the template hasn't been expanded yet, and one of the functions of `\cyr` is to invoke some special macros that treat certain combinations of accents and letters as single letters in the cyrillic alphabet. So the accent is assigned as a separate character (to be found in the position `\char'25` as in the default text font), and it is superimposed according to the usual accenting rules over whatever is found as `\char'20`.

Actually, this is fixed relatively easily; all that is needed is to begin each cyrillic cell with `\relax` (or some other "harmless" control sequence, as Don Knuth expressed it when I asked if he'd please explain to me what was happening). So:

```
\relax \u\i & \u\i \cr
```

As it turned out, I finally used another technique entirely to create the table. But I learned the dangers of macros being expanded out of order, and was also prepared for the questions that arose when users of the cyrillic font hadn't included the special accent macros in their definition of `\cyr` — the output looks just the same! Dreadful!

Moral: If the contents of some cells in your table don't look quite like what you expected, it may be worth putting `\relax` at the beginning and trying again before you start digging into the macros.

Solutions to the riddle from *TUGboat* 11#4

Frank Mittelbach

Puzzle:

Given a simple *TeX* document containing only straight text, is it possible for the editor, after deleting one sentence, to end up with a document producing an extra page?

We assume that the deleted text contains no *TeX* macros and that the document was prepared with a standard macro package like the one used for *TUGboat* production.

When I wrote down the riddle, I had a real life experience with the *TUGboat* layout in mind. In *ltugboat.sty* we have

```
\widowpenalty=10000
\clubpenalty =10000
```

Suppose that, a four line paragraph is broken across two pages. If we now reduce this paragraph to three lines, then *TeX* will no longer find any breakpoint in this paragraph.¹ So it will move the whole paragraph to the next page, thereby enlarging the document by at least one line. Similar problems might arise before displayed formulas if the default value for *\predisplaypenalty* is used.

Three other solutions were communicated to me by Bogosław Jackowski and Marek Ryćko. They all might occur in situations where only one word is removed from the input.

- *\lineskip* might get inserted when a paragraph is reformatted after deletions. If this parameter is positive² it will enlarge the paragraph height.
- If, after deletions, a footnote marker would have to be placed on the last line of a page, *TeX* will move the whole line to the next page.
- If the value of *\beforedisplayskip* is smaller than *\beforedisplayshortskip* removal of words preceding a formula might enlarge the document.

While the last solution might be classified under “obscure layouts”, all solutions show that parameter setting in *TeX* is a difficult art and might result in surprising results. There are many parameter settings buried inside plain *TeX* that have never been

¹ Before, there was a permissible breakpoint between the second and the third line.

² The default in *LATeX* is 1pt.

questioned. It would be nice if this short example arouses enough curiosity in you to play with them.³

One learns by experience — most of *TeX*’s world is still unexplored.⁴

³ You probably won’t believe this, but when I entered a few corrections to this note (as suggested by Chris Rowley), the current paragraph became a bit shorter, so that the word “them” moved up one line. Now, *ltugboat.sty* sets *\widowpenalty* to 10000, so that the column break could not be taken at the same place as before. Therefore one line from the preceding column was moved to this one.

⁴ An interesting and wide open field is the paragraph breaking mechanism, especially with the new *\emergencystretch* feature. Don Knuth has given us some hints about its potential but this has been never been systematically researched (or at least such research has never been published), which is a pity.

Macros

The bag of tricks

Victor Eijkhout

Hello everyone.

Using TeX, you inevitably gather a collection of useful little macros. Most of them are probably of use only to you, but maybe you have something that you're willing to share with the rest of the TUG audience. In that case, here's an outlet.

Starting this issue, I will present in this corner one or more handy little macros, without too much explanation, but otherwise ready for use. If you have something that you think is appropriate, send it to me or to *TUGboat*, and you may see it printed here, with due acknowledgements of course.

Here's this issue's gadget: saving and restoring category codes.

The plain format, the L^AT_EX format, and the L^AT_EX document styles contain lots of commands with the 'at' character (@) in them, in order to hide such commands from the user. Now, if you write some macros, you may want to do the same for the internal macros that should not be accessible to the user. You write at the start of the file containing these macros

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

(or maybe you use another character than @), and at the end you reset

```
\catcode`\@=12
```

But wait a second! Who told you that in the environment from which your file is \input the character @ has category 12?

The following macros allow you to write

```
\savecat @
```

```
...
```

```
\restorecat @
```

or, if you suspect that the character may have a strange category code such as 9, 14, or 15,

```
\savecat\@
```

```
...
```

```
\restorecat\@
```

The two forms can also be mixed.

Here are the macros.

```
\newcount\tempcount
\def\csarg#1#2{\expandafter#1%
  \csname#2\endcsname}
\def\storecat#1{%
  \tempcount=\escapechar
```

```
\escapechar=-1\relax
\csarg\edef\storecat{\string#1}%
{\catcode`\string#1=
  \the\catcode\expandafter`\string#1}%
\catcode\expandafter`\string#1=12\relax
\escapechar=\tempcount}
\def\restorecat#1{%
  \tempcount=\escapechar
\escapechar=-1\relax
\csname restorecat\string#1\endcsname
\escapechar=\tempcount}
```

Note that the \storecat macro sets the category of its argument initially to 12.

And here is some input to test the macros on:

```
\nonstopmode
\showthe\catcode`\%
\storecat\%
\catcode`\%=13
\showthe\catcode`\%
\restorecat%
\showthe\catcode`\%
\showthe\catcode`\~
\storecat`~%
\catcode`~=2
\showthe\catcode`\~
\restorecat`~%
\showthe\catcode`\~
Share and enjoy!
```

Note. Macros similar to the above appear in the macros *tugboat.com*. The editors of *TUGboat* have found such macros to be vital for maintaining their sanity in dealing with the incredible inventiveness of *TUGboat* authors.

TeX Macros for Producing Multiple-Choice Tests

Don De Smet

Since many members of the TeX community work in an academic environment, at least some of us teach. Sometimes we teach large classes. One facet of teaching is, unavoidably, testing. With a large class and no assistance in grading tests, the only reasonable way to give a test is with multiple-choice "computer-graded" examinations.

As a practical matter, when administering a multiple-choice test, steps must be taken to prevent students from copying answers from each other. Most students are inherently honest; however, in a crowded room where students are sitting close to each other it is unavoidable that one student can see his or her neighbor's choices of answers. Under these circumstances we have an obligation to keep this from becoming an advantage since, in a high-pressure situation, almost any normal person might be tempted to peek. We must prevent such temptations from occurring. The obvious solution is, of course, to use a number of different versions of the test in which questions and answers are scrambled. This insures that no useful information can be transferred by simply peeking while at the same time each student is given the same set of questions and the same choices for answers so that all students are treated equally.

TeX provides an ideal solution to this problem since TeX will let us easily produce different versions of a test from the same source. With TeX we can scramble the answers to an individual question; e.g., if the question is "How much is two minus two?" the answer "zero" can be choice A on one test and choice C on another. In addition, TeX allows us to re-arrange the questions; e.g., question number 1 on one test might be question number 7 on another. Further, TeX can put the correct answers to the test into a data file for later use provided we tell TeX what these answers are. (TeX is smart, but not quite smart enough to get the answers on his own.) TeX allows us to separate the mechanics of preparing a test from the effort of making good test questions. We can go as far as creating a different test for each student; however, there are good reasons to not get carried away. If it proves necessary to go back and re-grade a particular question (it happens—fortunately not very often) it is easier to correct a small number of different versions of a test. This, combined with the fact that four versions of a test are enough

so that in a normal rectangular room no student will have the same version as any nearest neighbor in any direction (and nine versions are enough to insure that no student will have the same version as any nearest or next nearest neighbor) means that four (or nine) versions of the test are adequate for most situations. Since a typical multiple-choice test has five possible answers, and these answers can be arranged 120 different ways, we have chosen to scramble only the answers and have a maximum of nine different versions of a test.

The set of macros given here has been used for some time to prepare tests for elementary Physics courses and it has worked well. Each student is assigned a particular seat (in our case the seats are labeled A-2, ..., A-9, B-2, ..., M-9). The tests are printed in this order with a version number and a seat designation on each test so there is a minimum of fuss when distributing them. Each student is asked to put his name, student identification number, and the test version number on the answer sheet for the test. Invariably a few students forget one of these, but this is not a serious problem, provided that the student at least sat in the correct seat, since there is some redundancy here.

At the University of Alabama the office of Test Services will take the individual answer sheets and provide a file on magnetic tape, disk, or by electronic mail that contains a record of each student's responses, including name, student identification number, test version number and answers to individual questions, known locally as a raw data file. After editing this file to get it into the correct format, we use a FORTRAN program that reads this file and the data file containing the test keys created by TeX. The program computes a grade for each student and also produces a table for statistical purposes that gives the number of students that chose a particular literal (unscrambled) answer for each question. We also produce a crude histogram, a set of grades by student identification number that can be posted, a file containing averages, etc.

We have written these macros so that they require two files for input. The first file, TESTOP.TEX, is the heading which includes instructions for all students taking the test. The other file, called ATEST.TEX contains the actual test questions and answers. Each of these ends with an \endinput statement. A sample of each file is included. The answer key (or keys) is put into a file TESTKEY.DAT. The names TESTOP.TEX, ATEST.TEX and TESTKEY.DAT can, of course, be changed.

In ATEST.TEX the macro \qn marks the beginning of each question. This is just an adaptation

of \item that automatically numbers the question. After this is a statement of the test question. This is followed by a macro \picks that takes five arguments, each enclosed between a \{} \} pair, representing the five choices for the answer to the question. The correct answer should be marked by **. This will be replaced by a * on the printed test if we are making a copy for proofreading purposes, but will be suppressed when we are actually printing tests. The macro \endq marks the end of the question and answers. Thus a typical question and its answers would read:

```
\qn How often have you been
late for class?
\picks{always}{usually}{sometimes\**}
{rarely}{never}
\endq
```

T_EX will choose one of four formats for printing the answers to a question. If all five of the answers fit comfortably onto a single line, then T_EX will do it this way. If not, T_EX determines the widths of the individual answers. If the three longest answers fit on a line, then we let T_EX put the first three answers on one line and the last two on a second line in tabular format. If not, but if the two longest answers do fit on a single line, the first two answers are placed on one line, the second two on a second line, and the last one on a third line. If the two longest answers are too long to fit together on a single line, T_EX puts one answer per line. If an answer is more than one line long, any additional lines are suitably placed under the first one.

A question and its answers should be a unit; there should not be a page break between the start of a question and the end of the answers. I initially used \nobreak, \goodbreak, and \medskip to achieve this, but this did not always work. \filbreak, on the other hand, seems to solve this problem.

The macros given here have several modes. When \makingprooftrue, a single copy of the test is generated. This single copy, version 0, does not have the answers scrambled. All other versions will have scrambled answers that can be unscrambled to compare to version 0. In this mode, the printed copy does not have a seat number printed on the test and the correct answer to each question is marked with a *.

When \makingprooffalse, the correct answers are not marked and the number of copies generated depends on \ifsmallset. For \smallsettrue only one copy of each version of the test is generated with the version number printed on it (except when there

is only one version), but without a seat number. This is useful if we have limited disk space or a slow printer, since we can still have many versions of the test. In this case we can make additional copies with a copy machine and put seat numbers on the tests by hand. It is not as pretty (or as automatic) as T_EX, but it is sometimes the most efficient way to do things. When \smallsetfalse, a test is printed for each seat number.

There is also a parameter \alternationnumber that should be set to either one, two, or three. This lets us select the number of different versions of the test. If \alternationnumber=1 we get a single version of the test, which does not require a seat number or a version number, so these are suppressed. If \alternationnumber=2 we get tests that alternate every second row and seat for a total of 4 versions of the test. If \alternationnumber=3 we get 9 versions of the test. Thus \alternationnumber=1 \smallsettrue gives one copy of the test without seat or version numbers, \alternationnumber=1 \smallsetfalse gives a set of tests without seat or version numbers, \alternationnumber=2 or 3 \smallsettrue gives one copy of each version of the test that includes a version number but no seat number, while \alternationnumber=2 or 3 \smallsetfalse gives a complete set of tests with seat and version numbers. These parameters are set near the beginning of the macros. Also, several other parameters (e.g., the number of seats in a row) are given values here. Assuming that we call this set of macros TESTMAKE.TEX, T_EX is run on TESTMAKE, and TESTMAKE reads the test heading and the test.

For one class in which we used these macros, we printed 90 copies of a test that was a little less than two pages long, thus the amount of time it took T_EX to process this set of tests was about the same as it would be for a 180-page document. The size of the .DVI file and the time required to print the tests were also comparable to those for a 180-page document. With a large computer and a fast printer this is acceptable; with a small computer or a slow printer it might not be.

A few closing words: Answers such as "none of the above" and "both A and C are true" should be avoided, since the answer scrambler is going to move the answers around. An answer such as "none of these choices", however, seems quite reasonable. It is not a good idea to put additional \newcount, \newdimen, etc. statements in TESTTOP or ATTEST; it is better to put these at the beginning of TESTMAKE along with your own favorite macros. As given here, TESTMAKE can put a maximum of 10 answers

into TESTKEY, but the extension to longer tests is straightforward. Simply look in the macros for `\ansx`; wherever it appears add `\ansxi`, `\ansxii`, etc. in the same context.

Although this set of macros has been used in its present form for more than a year, it has evolved from a few simple definitions and it will continue to evolve as new situations arise. There are probably better ways to do many of the things described here (a few of them were found by trial and error), but these macros have worked reasonably well for me. After using them a few times you get accustomed to the error messages generated when you make the usual typographical errors. Further, we can accumulate questions in TeX format, so when one last question is needed to finish writing a test, we will have a selection to choose from.

Finally, students prefer TeX made tests to typed tests. (Remember purple ditto masters?) In fact, on one occasion a student asked me how I prepared my tests, and when I told her I used TeX, I received a knowing nod of approval and after that she began to sit in the front row during class. I think there is some psychology involved in this.

A spoon full of sugar helps the medicine go down.

—Mary Poppins

An ounce of prevention is worth a pound of cure.

—Old proverb

A sample of TESTOP.TEX

```
\rightline{NAME $ \underline{\phantom{a}\hspace{3in} a}}$}
\phantom{a\hspace{3in} a}
\vskip2mm
\line{\bf PHYSICS 102 \hfil
FIRST EXAMINATION \hfil SPRING 1990}
\vskip1.5mm
\ifprintcode\line{\bf \hfil TEST
CODE NUMBER \number\version \hfil}
\fi
\vskip 1.0ex
{\multiply\baselineskip by 5
\divide\baselineskip by 4
This test consists of 4 multiple-choice
questions. Answer each question in the
space provided on the answer sheet. There
is one 'best' answer to each question.
Also {\bf PLEASE FILL IN YOUR NAME}
(last name first) on this sheet.
Remember to darken the appropriate
circles for your name.
\ifprintcode
At the top of this test is a code number.
{\bf Fill in this number on your answer
```

sheet. It must be put in the first column of {\sl SPECIAL CODES}.

\fi

Do not consult with anyone
during the test.}

```
\def\ldrfl{\leaders\hrule\hfill\ }
\vskip1mm
\line{\ldrfl {Good luck!} \ldrfl}
\vskip 1.2ex
\endinput
```

A sample of ATEST.TEX

```
\qn If a man and a half can dig a hole
and a half in a day and a half, how many
holes can three men dig in three days?
\picks{3}{6}{**}{2.25}{4.5}
\endq
\qn \TeX\ is
\picks{a very small mouse.}{a great big dog.}{a lonesome cowboy.}{a state of the union.}{a state of mind.}{**} \endq
\qn
A good way to recall the spectral classes
of main sequence stars is to remember the
phrase \picks{All Cows Eat Grass.}{Spectroscopy Prevents Daddy From Going
Home.}
\picks{Old MacDonald Had A Farm EIEIO.}{Oh Be A Fine Girl, Kiss Me.}{**}{Twinkle Twinkle Little Star.}
\endq
\qn We need a question and some answers
that each fill up more than one line.
In order to accomplish this we will ask
the following question: "What are the
basic food groups?"'
\picks{Sugar, salt, caffeine and fat.}{Burger, fries, coke and a candy bar.}{Fruit, cereal, meat}{or fish and dairy products.}{Nuts (eeech!), berries (eeech!), honey (mmmmm!) and anything we can beg, borrow or steal from the picnic basket of an unsuspecting camper.}{A book of verses underneath the Bough, a Jug of Wine, a Loaf of Bread---and Thou Beside me, singing in the Wilderness---Oh, Wilderness were Paradise enow!}
\endq
\endinput
```

And Finally, TESTMAKE.TEX

```
% no guarantees---use at your own risk.
\font\twentypt=cmr10 scaled\magstep4 % use a large bold font
\newcount\firstseatnum \newcount\lastseatnum
\newcount\firstseatletter \newcount\lastseatletter
\newcount\alternationnumber \newwrite\keywrite
\newif\ifmakingproof \newif\ifsmallset

%%%%%%%%% PUT CHANGES BETWEEN HERE %%%%%%
\hsize=6.00in\hoffset=0.250in
\vsiz=7.50in\voffset=0.250in
% size and offset adjusted for margins of both printer and driver
% page number should appear at the bottom of the test
\firstseatnum=2 % there is no first row in our room
\lastseatnum=2 %11 % row goes from 2 to 11, but 2 for testing
\firstseatletter=1 % starts with A
\lastseatletter=1 %12 % ends with M (but no I), but A for testing
% here is the name of the test heading
\def\getthetop{\input testop \relax}
% here is the name of the file containing the test
\def\getthetest{\input atest \relax}
%%% define a file for the test keys %%%%%%
\immediate\openout\keywrite=testkey.dat
\makingprooffalse % set either true or false
\alternationnumber=2 % use either 1, 2 or 3 here
% \smallsettrue % set either true or false
\smallsetfalse
%%%%% AND HERE %%%%%%
\newif\ifprintversion \newif\ifprintseatnumber
\newif\ifprintstar \newif\ifprintcode \newcount\numberofcopies
\numberofcopies=1 \printcodetrue

\ifsmallset \printseatnumberfalse \else \printseatnumbertrue \fi
\ifmakingproof \printstartrue \printseatnumberfalse
\else \printstarfalse \fi
\ifnum\alternationnumber=1 \printversionfalse \else \printversiontrue \fi

\ifnum\alternationnumber=1
\ifsmallset \else \printseatnumberfalse \fi \fi

% numberofcopies is set to 1 for a single copy, 2 for a single copy of
% each version and 3 for a fullblown set of tests

\ifsmallset \else \ifmakingproof \else \numberofcopies=3 \fi \fi

\ifsmallset \ifnum\alternationnumber=1 \else
\ifmakingproof \else \numberofcopies=2 \fi \fi \fi
\ifprintversion \printcodetrue\else\printcodefalse\fi % redundant

\newdimen\itemsize \itemsize=1dd
\def\qnf{\parindent=1dd\item{\number\questionno.}}
\def\endqf{\vskip2.0ex\filbreak\medskip}
```

```

\newcount\qsizecounter\newcount\anct\newcount\anso\newcount\ansi
\newcount\ansii\newcount\ansiii\newcount\ansiv\newcount\ansv
\newcount\ansvi\newcount\ansvii\newcount\ansviii\newcount\ansix
\newcount\ansx \newcount\version \newdimen\widp \widp=\hsize
\newcount\anscor\newcount\seatcounter\newcount\testitone
\newcount\testittwo\newcount\versionend\newcount\letternum
\newcount\questionno\newcount\testpageno\newcount\oldpagenumber
\newdimen\aaazz\newdimen\bbbzz\newdimen\ccczz\newdimen\dddzz
\newdimen\eeezz\newdimen\sizeline\newdimen\tmzz\newcount\bubbllbb
\newtoks\aayy\newtoks\bbyy\newtoks\ccyy\newtoks\ddyy\newtoks\eeyy
\letternum=\firstseatletter \version=0 \oldpagenumber=0

\newbox\testbox\newbox\aaabox\newbox\bbbbox
\newbox\cccbox \newbox\dddbox\newbox\eeebox

% next necessary to get alignment right: 01...10 in output file
\def\qwy{0\number\seatcounter}\def\qwz{\number\seatcounter}

% this writes a line to the test key file
\def\writeitout{\def\qwa{\ifnum\seatcounter<10 \qwy\else\qwz\fi}
\immediate\write\keywrite{%
VERSION-\number\version..SEAT-\LETTERPUT-\qwa...
\number\ansi \number\ansii \number\ansiii \number\ansiv \number\ansv
\number\ansvi \number\ansvii \number\ansviii \number\ansix \number\ansx}}}

\def\pickansform#1#2#3#4#5{%
formats answer depending on \qsizecounter
\aaayy={#1}\bbyy={#2}\ccyy={#3}\ddyy={#4}\eeyy={#5}
\global\anct=0 \global\anscor=0
% find the right answer for this version
\ctest\aaayy\ifresult\global\anscor=1 \advance\anct by 1 \fi
\ctest\bbyy\ifresult\global\anscor=2 \advance\anct by 1 \fi
\ctest\ccyy\ifresult\global\anscor=3 \advance\anct by 1 \fi
\ctest\ddyy\ifresult\global\anscor=4 \advance\anct by 1 \fi
\ctest\eeyy\ifresult\global\anscor=5 \advance\anct by 1 \fi
\ifnum\anct<1 \message{no right answer for question \the\questionno} \fi
\ifnum\anct>1 \message{more than one correct answer given
for question number \the\questionno} \fi
\ifcase\qsizecounter % \qsizecounter=0: one line for all answers
{\line{\hskip1pt a) #1 \quad
\hfil b) #2 \quad\hfil c) #3 \quad\hfil d) #4 \quad\hfil e) #5}}
\or % \qsizecounter=1: one or more lines per answer
{\parindent=0.8em % 1.0em is too much
\vbox{\parindent=0.8em % 0.9 is too much
\itemitem{a)} #1\vskip0.5ex \vbox{\itemitem{b)} #2} \vskip0.5ex
\itemitem{c)} #3\vskip0.5ex \vbox{\itemitem{d)} #4} \vskip0.5ex
\itemitem{e)} #5}}
\or% \qsizecounter=2: need 3 lines; 2 on first, 2 on second, one on third
{\vbox{\halign to \hsize{\hskip1pt ##\hfil
\tabskip=5em plus5em minus5em##\hfil\cr
a) #1 &b) #2 \cr
c) #3 &d) #4 \cr
e) #5 &\cr}}}
\else% \qsizecounter=3: three on first, two on second line

```

```

{\vbox{\halign to \hsize{\hskip1pt ##\hfil
\tabskip=5em plus5em minus5em&##\hfil& ##\cr
a) #1 &b) #2 &c) #3 \cr
d) #4 &e) #5 & \cr} } }
\fi}

\newif\ifresult % for the result of a test (taken from the TeXbook)
\def\ctest#1{\resultfalse\expandafter\c\the#1\ctest}
\def\c{\afterassignment\cc\let\next= }
\def\cc{\ifx\next\ctest \let\next\relax
\else\ifx\next*\resulttrue\fi\let\next\c\fi \next}
\c

% here is the answer scrambler
% computes \qsizecounter depending on lengths of answers and
% scrambles answers depending on value of \version
\def\picks#1#2#3#4#5{\nobreak\vskip1mm\nobreak
\ifnum \questionno > 10 \message{OOPS...this test has more than
10 questions. Time to fix things!!!} \fi

\setbox\testbox=\hbox{\hskip1pt a) #1 \hfil\quad b) #2 \hfil\quad
c) #3 \hfil\quad d) #4 \hfil\quad e) #5 \quad}
\sizeline=\wd\testbox
\ifdim\sizeline<\widp \global\qsizecounter=0 % answers fit on one line
\else% compute widths of individual answers--with a little extra space
\setbox\aaabox=\hbox{ia) #1\quad}\setbox\bbbbbox=\hbox{ia) #2\quad}
\setbox\cccbox=\hbox{ia) #3\quad}\setbox\dddbox=\hbox{ia) #4\quad}
\setbox\eeebox=\hbox{ia) #5\quad}
\%p. 121 TeXbook
\aaazz=\wd\aaabox \bbbzz=\wd\bbbbbox \ccczz=\wd\cccbox
\dddzz=\wd\dddbox \eeezz=\wd\eeebox
\global\bubblebb=1
\loop \ifnum\bubblebb>0 % bubble sort answers according to length
\global\bubblebb=0
\ifdim\aaazz<\bbbzz\temzz=\aaazz\aaazz=\bbbzz\bbbzz=\temzz
\global\advance\bubblebb by 1 \fi
\ifdim\bbbzz<\ccczz\temzz=\bbbzz\bbbzz=\ccczz\ccczz=\temzz
\global\advance\bubblebb by 1 \fi
\ifdim\ccczz<\dddzz\temzz=\ccczz\ccczz=\dddzz\dddzz=\temzz
\global\advance\bubblebb by 1 \fi
\ifdim\dddzz<\eeezz\temzz=\dddzz\dddzz=\eeezz\eeezz=\temzz
\global\advance\bubblebb by 1 \fi
\repeat % loop if not completely sorted
\global\qsizecounter=1 \temzz=\aaazz \advance\temzz by \bbbzz
% \temzz becomes width of widest and next widest answers
\ifdim\temzz<\widp \global\advance\qsizecounter by 1 \fi
\advance\temzz by \ccczz % \temzz is width of three widest answers
\ifdim\temzz<\widp \global\advance\qsizecounter by 1 \fi
\advance\temzz by \dddzz % \temzz is width of four widest answers
\ifdim\temzz<\widp \global\advance\qsizecounter by 1 \fi
\fi % close ifdim\sizeline
% scramble answers--move these around from time to time.
\ifcase\version \pickansform{#1}{#2}{#3}{#4}{#5}\or
\pickansform{#2}{#1}{#4}{#3}{#5}\or \pickansform{#4}{#3}{#2}{#1}{#5}\or
\pickansform{#3}{#5}{#1}{#2}{#4}\or \pickansform{#2}{#4}{#3}{#1}{#5}\or

```

```

\pickansform{#3}{#4}{#5}{#1}{#2}\or \pickansform{#4}{#5}{#1}{#2}{#3}\or
\pickansform{#5}{#4}{#3}{#2}{#1}\else
\pickansform{#1}{#3}{#4}{#2}{#5}
\fi
% put right answer into correct slot for this question
\ifcase\questionno\or \global\ansi=\anscor\or
\global\ansii=\anscor\or \global\ansiii=\anscor\or
\global\ansiv=\anscor\or \global\ansv=\anscor\or
\global\ansvi=\anscor\or \global\ansvii=\anscor\or
\global\ansviii=\anscor\or \global\ansix=\anscor\or
\global\ansx=\anscor\or
\fi
% we need the next few statements so that each test starts with page 1
\global\advance\questionno by 1 \ifnum\pageno>\oldpagenumber
\global\advance\oldpagenumber by 1
\global\advance\testpageno by 1
\fi} % end of the answer scrambler

% this zeros the answers
\def\settozero{\ansi=0 \ansii=0 \ansiii=0 \ansiv=0 \ansv=0
\ansvi=0 \ansvii=0 \ansviii=0 \ansix=0 \ansx=0}

\def\printst{*}\def\noprintst{}
\def\**{\ifprintstar\printst\else\noprintst \fi}

\def\LETTERPUT{\ifcase\letternum A\or B\or
C\or D\or E\or F\or G\or H\or % I\or % don't have seats with 'I'
J\or K\or L\or M\or N\or O\or P\or Q\or
R\or S\or T\or U\or V\or W\or X\or Y\or Z\fi}

% it is useful to print "more" at the bottom of all but the last page
\def\footlinea{\rm ---more---\hss ---more---\hss ---\number\testpageno
---\hss ---more--- \hss ---more---}
% and "end" at the bottom of the last page
\def\footlineb{\rm ---end---\hss ---end---\hss ---\number\testpageno
---\hss ---end---\hss ---end---}

\newcount\outputlinecounter\outputlinecounter=0

% this formats a copy of the test
\def\doacopy{\settozero \questionno=1 \testpageno=0
\footline=\{\footlinea\}
\phantom{.}
\ifprintseatnumber
\vskip-12mm
\line{\hfil\bf \LETTERPUT-\number\seatcounter } }
\vskip+12mm
\else
\vskip-12mm
\line{\hfil\bf \phantom{\LETTERPUT-\number\seatcounter } }
\vskip+12mm
\fi
{\getthetop {\getthetest }
\footline=\{\footlineb\}

```

```
\vfil\eject
\ifnum\outputlinecounter=0 \writeitout \fi
\ifnum\alternationnumber=1 \global\advance\outputlinecounter by 1 \fi}
% end of def of doacopy

% modular arithmetic is useful for computing the version number
\newcount\aaqqq\newcount\bbqqq\newcount\ccqqq\newcount\ddqqq
\newcount\modcounter
\def\modulo#1#2{\modcounter=0
\aaqqq=#1 \bbqqq=#2 \ccqqq=#1
\divide \ccqqq by \bbqqq \multiply \ccqqq by \bbqqq
\ddqqq=\aaqqq \advance \ddqqq by -\ccqqq
\global\modcounter=\ddqqq}

\advance\lastseatnum by 1 \advance\letternum by -1 % stop loops correctly
% Finally we actually DO something!
\versionend=\alternationnumber
\ifcase\numberofcopies\or
\doacopy      % numberofcopies=1
\or          % numberofcopies=2, single copy of each version
\multiply\versionend by \alternationnumber
\loop\ifnum\version<\versionend
\doacopy
\advance\version by 1
\repeat
\or      % a whole set of tests
% make many copies of the test with scrambled answers
\loop\seatcounter=\firstseatnum\ifnum\letternum<\lastseatletter
  {\loop
    \modulo{\seatcounter}{\alternationnumber}
    \testitone=\modcounter
    \modulo{\letternum}{\alternationnumber}
    \testittwo=\modcounter
    \multiply \testittwo by \alternationnumber
    \version=\testitone \advance \version by \testittwo
    \ifnum\seatcounter<\lastseatnum
    {\doacopy}
    \advance\seatcounter by 1
    \repeat} % close seatcounter loop
\advance\letternum by 1
\repeat % close letternumber loop
\fi % close \ifcase\numberofcopies
\end
```

A Sample of the Output**A-2**

NAME _____

PHYSICS 102**FIRST EXAMINATION****SPRING 1990****TEST CODE NUMBER 0**

This test consists of 4 multiple-choice questions. Answer each question in the space provided on the answer sheet. There is one 'best' answer to each question. Also **PLEASE FILL IN YOUR NAME** (last name first) on this sheet. Remember to darken the appropriate circles for your name. At the top of this test is a code number. **Fill in this number on your answer sheet. It must be put in the first column of SPECIAL CODES.** Do not consult with anyone during the test.

Good luck! _____

1. If a man and a half can dig a hole and a half in a day and a half, how many holes can three men dig in three days?
 a) 3 b) 6 c) 2.25 d) 4 e) 4.5

2. TEX is
 a) a very small mouse. b) a great big dog. c) a lonesome cowboy.
 d) a state of the union. e) a state of mind.

3. A good way to recall the spectral classes of main sequence stars is to remember the phrase
 a) All Cows Eat Grass. b) Spectroscopy Prevents Daddy From Going Home.
 c) Old MacDonald Had A Farm EIEIO. d) Oh Be A Fine Girl, Kiss Me.
 e) Twinkle Twinkle Little Star.

4. We need a question and some answers that each fill up more than one line. In order to accomplish this we will ask the following question: "What are the basic food groups?"
 a) Sugar, salt, caffeine and fat.
 b) Burger, fries, coke and a candy bar.
 c) Fruit, cereal, meat or fish and dairy products.
 d) Nuts (eeech!), berries (eeech!), honey (mmmmm!) and anything we can beg, borrow or steal from the picnic basket of an unsuspecting camper.
 e) A book of verses underneath the Bough, a Jug of Wine, a Loaf of Bread—and Thou Beside me, singing in the Wilderness—Oh, Wilderness were Paradise enow!

◊ Don De Smet
 University of Alabama
 Department of Physics and
 Astronomy
 Box 870324
 Tuscaloosa, AL 35487-0324
 DDESMET@ua1vm.ua.edu

Getting \answers in TeX

Jim Hefferon

The tutorial *Long-winded Endnotes and Exercises with Hints or Solutions* by Lincoln Durst (TUGboat 11, no. 9, pp. 580–588) gives a clear explanation of why setting up a format like

```
\begin{exercises}
  :
  \item ...
    \answer{-text-}
  :
\end{exercises}
```

is hard. The problem is that writing *-text-* to an auxiliary file is more complex than it seems. Briefly, TeX grabs the argument to `\answer`, expands all tokens fully (looking for things like page numbers that must be evaluated now), and then writes the result as a single—possibly very long—line to that auxiliary file.

Besides being hard to edit, those long lines may choke TeX's input mechanism since `buf_size` may be only a thousand characters.

Long-winded ... details a fascinating trick to try to work around this problem. But for me this approach has two problems. First, I don't entirely understand it (so if it breaks I can't fix it). Second, it won't take line returns inside of braces.

Just as I don't understand the macro, I don't entirely understand the macro's problems, so I am

reluctant to rely on it. This note is to point out that an obvious kludge has some advantages.

I defined `\answer` to write the number of the current exercise to the file `exnos.tex`:

```
\newwrite\ansnos
\immediate\openout\ansnos=exnos
\outer\long\def\answer#1{%
  \immediate\write\ansnos{\thelabeledtext} }
```

(`labeledtext` counts the formal parts of my document). A typical output file looks as follows.

```
1.2.1.3
1.2.1.4
1.2.2.7
  :
1.2.2.13
```

Then I wrote an editor macro to find instances of `\answer{-text-}`, match them with lines from `exnos.tex`, and write the result to `answers.tex`. After executing the macro, `answers.tex` looks like Figure 1. (The lines of the answers are, comments and all, as I typed them.)

My editor macro, listed in the Appendix, is written in REXX (specifically Personal REXX for the Kedit editor on an IBM PC).

This solution may involve leaving TeX but it has the advantages that I can modify it to suit my needs, and that it has no obscure restrictions. Stretching TeX is fun, but maybe it just isn't right for this job.

Figure 1. `answers.tex`

```
\par\noindent{\bf 1.2.1.3}
\par{Known as Fermat's Last Theorem, \% !!index this
this result is easy when
\((n)\) is infinite or \((2)\), but is harder
for intermediate \((n)\).}

\par\noindent{\bf 1.2.1.4}
  :
```

Appendix

```
/* ANSWER.kex  THIS IS A REXX PROGRAM          */
/* Used to separate answers from the file in TeX */
/* Expects the counters for those answers to be   */
/* in the file texauxfile, one to a line.         */
/* Kills .w and .z.                            */
/* jim hefferon St. Mike's College. 91-1-21      */
answerfilename='answers.tex'
texauxfile='exnos.tex'
```

```

prefixstring='\\par\\noindent{\\bf '
postfixstring=' }\\par\\nobreak '

/* trace ?i */
'set msgmode off' /* Matches the "on" below. */

/* Remember the filename and where we are. */
'extract /fileid'
'set point .z'
'top'
'/{\\answer{}' /* find string like this */
failedtofind=rc
'set point .w' /* .w is last place found answer */

/* Go to output file. Add header (announce the new page). */
'k 'answerfilename
'bot'
'sos addline'
'text %ANSWERS!!'

/* Enter main loop. */
do while failedtofind=0
  'k 'answerfilename      /* Open the answerfile */
  'bot'                  /* and insert any pre- */
  'sos addline'          /* number info. */
  'text '||prefixstring /* */
  'k 'texauxfile         /* Open the counter */
  'down'                 /* file and take the */
  'reset block'          /* next line. */
  'mark line'            /* */
  'k 'answerfilename     /* Reopen the answer */
  'bot'                  /* file, copy next line*/
  'copy block'           /* and put in post- */
  'sos addline'          /* number info */
  'text '||postfixstring /* */
  'k 'fileid.1           /* Go to main (TeX) */
  'locate .w'             /* file and stream */
  'extract /curline/'    /* mark the answer, */
  parse value curline.3 with s1 '\\answer{' rest /* e.g., */
  'extract /line/'        /* \\answer{Hello, this */
  'cursor file 'line.1 length(s1)\\answer{}' /* ~~~~~ */
  'mark stream'          /* is it.} More stuff. */
  'cmatch'                /* ~~~~~ */
  'sos makecurr'          /* */
  'mark stream'          /* */
  'k 'answerfilename     /* Transfer to answer */
  'bot'                  /* file. */
  'sos addline'          /* */
  'copy block'           /* */
  'k 'fileid.1           /* Any more answers? */
  'reset block'          /* */
  '/{\\answer{}'          /* */
  failedtofind=rc          /* */
  'set point .w'          /* */

```

```

end

/* Cleanup. */
'k 'texauxfile
'qquit'

/* File the answer file. */
'k 'answerfilename
'file '
/* Move back to where we were when the macro was called. */
'k 'fileid.1
'set point .w off'
'locate .z'
'set point .z off'
'set msgmode on' /* Matches the "off" above. */
'msg Answers appended to '||answerfilename||'.

```

◊ Jim Hefferon
 Mathematics
 St. Michael's College
 Colchester, VT 05439
 BITnet: hefferon@smcvax

Oral **T****E****X**

Victor Eijkhout

T**E****X** knows two sorts of activity: those actions that can be classified under ‘execution’, and those that fall under ‘expansion’. The first class comprises everything that gives a typeset result, or that alters the internal state of **T****E****X**. Examples of this are control sequences such as `\vskip`, macro definitions, and all assignments.

Expansion activities are those that are performed by what is called the mouth of **T****E****X**. The most obvious example is macro expansion, but the command `\the` and evaluation of conditionals are also examples. The full list can be found on pages 212–215 of the **T****E****X**book [1].

In this article I will give two examples of complicated macros that function completely by expansion. Some fancy macro argument delimiting occurs, and there are lots of applications of various conditionals. For a better understanding of these I will start off with a short section on the expansion of conditionals.

About conditionals

For many purposes one may picture **T****E****X**’s conditionals as functioning like conditionals in any other

programming language. Every once in a while, however, it becomes apparent that **T****E****X** is a macro processor, absorbing a stream of tokens, and that conditionals consist of nothing more than just that: tokens.

Consider the following example:

```

\def\bold#1{{\bf #1}}
\def\slant#1{{\sl #1}}
\ifsomething \bold \else
    \slant \fi {word}

```

If the ‘something’ condition is true, the whole `\if ... \else ... \fi {word}` sequence is *not* replaced by `\bold {word}`; instead **T****E****X** will start processing the ‘true’ part of the conditional. It expands the `\bold` macro, and gives it the first token in the stream as argument. Thus the argument taken will be `\else`. **T****E****X** will only make a mental note that when it first encounters – more precisely: expands – an `\else` it will skip everything up to and including the first `\fi`¹.

¹ The reader may enjoy figuring out why in spite of the apparent accident in this example the ‘word’ will still be bold, and why **T****E****X** will report that ‘end occurred inside a group at level 1’ at the end of the job.

For this sort of problem there are (at least) two solutions. One solution is:

```
\ifsomething \let\next=\bold
  \else \let\next=\slant \fi
  \next {word}
```

Note that this uses more than just the mouth of TeX, because `\let` statements are executed, not expanded.

Second solution:

```
\ifsomething \expandafter\bold
  \else \expandafter\slant \fi {word}
```

The `\expandafter` commands will let TeX find the delimiting tokens of the conditional before it starts expanding for instance `\bold`. When this command is finally expanded, the irrelevant parts of the conditional will have been removed.

The reader may not see the point of this second solution at first, and indeed, the first solution is more natural in a sense. However, there is a very important advantage to the second. Primitive conditionals of TeX are fully expanded, for instance inside an `\edef`. Example:

```
\edef\savelastskip
  {\ifhmode \hskip \else
   \vskip \fi \the\lastskip}
```

will expand to

macro: $\rightarrow \hskip \dots$

or

macro: $\rightarrow \vskip \dots$

depending on the mode. If you are implementing a test that should function in a context where there is only expansion `\let` cannot be used, so a number of `\expandafter` commands will probably be needed.

Application 1: string comparison

Suppose we would like to have a macro that tests for equality of two strings, and it should be useable as if it were a conditional:

```
\ifsamestring{some}{other}
  \message{yes!}\else
  \message{no!}\fi
```

A simple construction exists for this:

```
\def\ifsamestring
  #1#2{\def\testa{#1}\def\testb{#2}%
        \ifx\testa\testb}
```

however, this suffers from the objection mentioned above: it uses more than just the expansion performed in TeX's mouth. Thus, the following call

```
\message{\ifsamestring
  {some}{other}yes\else no\fi!}
```

gives a, probably, somewhat unexpected result:

```
! Undefined control sequence.
\ifsamestring #1#2->\def \testa
  ...
  ...
```

Solutions using only expansion are possible, but they are more complicated. The reader may want to try solving this before looking at the solution below. Keep in mind that we want something that behaves like a conditional: the final result should be allowed to be followed by

```
... \else ... \fi
```

The solution given here is not the only possible one: variations may exist. However, there is probably only one basic principle, which is to compare the strings one character at a time.

Here is the first part:

```
\def\ifsamestring
  #1#2{\ifallchars#1$ \are#2$ \same}
```

The strings are terminated by a dollar character, which we suppose not to appear in the string.

Next the routine `\ifallchars` will be used recursively. At first it tests if either of the two strings has run out. If some incarnation of this routine finds that both strings are empty the initial strings must have been equal, if exactly one is empty the initial strings were of unequal length, thus unequal; if neither is empty another routine should check if their leading characters are the same, and if so, do a recursive call to `\ifallchars` to see if the rest is also the same.

```
\def\ifallchars#1#2\are#3#4\same
  {\if#1$ \if#3$ \say{true}%
   \else \say{false}\fi
   \else \if#1#3\ifrest#2\same#4\else
         \say{false}\fi\fi}
```

The `\say` macro is something of a trick; we'll get to that. Let's first consider the last clause: the test `\if#1#3` checks if the leading characters of the strings are equal; if so, the remainder should be tested for string equality.

In the previous section I showed that a `\fi` is just a token standing in the input stream. Standing behind the call to `\ifrest` there are two such tokens, and somehow they are to be removed. Unfortunately the `\expandafter` method of the previous section cannot be used here, as there may be an indefinite number of tokens between the `\ifrest` and the `\fi` tokens.

One solution here is to let the final argument of `\ifrest` be delimited by the whole closing sequence:

```
\def\ifrest#1\same#2\else#3\fi\fi
  {\fi\fi \ifallchars#1\are#2\same}
```

A trick if ever there was one. When the `\if#1#3` test turns out false the `\ifrest` call is skipped, and the `\fi\fi` sequence delimits both conditionals that are open at that moment. When `\if#1#3` is true, everything up to and including `\fi\fi` is scooped up as part of the parameter text. The delimiting `\fi\fi` sequence is not expanded in this process, so the conditionals must still be closed; this is done by the `\fi\fi` sequence with which the replacement text starts. Thus this construction effectively lifts the relevant part of the macro outside the boundaries of the conditional.

Now for the `\say` macro. What we want to accomplish by it is this: the call `\say{true}` should put the primitive `\iftrue` outside all conditionals that are active at the moment, and similarly for `\say{false}`. The implementation of `\say` given here is not very general: it uses the fact that all three calls are nested two conditionals deep, and that the final boundary is the `\fi\fi` sequence.

```
\def\say#1#2\fi\fi
{\fi\fi\csname if#1\endcsname}
```

All tokens in between the first argument of `\say` and the delimiting `\fi\fi` are lumped together in #2 and they are never used. The reason that `\say` is necessary at all, is that `\TeX` would misinterpret the occurrence of `\iftrue` and `\iffalse` in clauses that are skipped.

There is a, maybe somewhat surprising, second possible implementation of `\say`. Inside a `\csname ... \endcsname` sequence all expandable tokens are expanded, and in particular `\expandafter`. This means that one may alter the state of the input stream after `\endcsname` by putting `\expandafter` directly in front of it. I leave it to the reader to figure out what are the exact mechanisms of this second implementation:

```
\def\say#1{\csname if#1\expandafter
\expandafter\expandafter\endcsname}
```

Note that the calls to `\say` always occur directly in front of an `\else` or a `\fi`.

Once the reader understands this trick, the following routine for alphabetical comparison of two strings will not present insurmountable problems.

```
\let\xp=\expandafter
\def\ifbefore
#1#2{\ifallchars#1$\are#2$\before}
\def\ifallchars#1#2\are#3#4\before
{\if#1$\say{true}\xp\else
\if#3$\say{false}\xp\xp\xp\else
\ifnum'#1' #'3 \say{false}%
\xp\xp\xp\xp\xp\xp\else
\ifrest#2\before#4\fi\fi\fi}
```

```
\def\ifrest#1\before#2\fi\fi\fi
{\fi\fi\fi
\ifallchars#1\are#2\before}
\def\say#1{\csname if#1\endcsname}
```

Lexicographic comparison is done here by numerical comparison of character codes. The `\say` macro now occurs on three different levels, so the number of `\expandafter` commands needed to remove various amounts of `\else` and `\fi` tokens is 1, 3, and 7. The first two clauses of the test take care of the case where strings are of different lengths.

A comment about the principle underlying these macros. Every step replaces one `\if...` command by another, until finally only `\iftrue` or `\iffalse` results. All the magic with `\expandafter` and delimiting with `\fi` is necessary, because we can't deliver these final conditionals as a result of other conditionals. However, we can let `\TeX` deliver some tokens that give a true or false test. The `\ifsamestring` test can for instance be implemented as²

```
\def\saytrue{0=0 } \def\sayfalse{0=1 }
\def\ifsamestring#1#2%
{\ifnum \allchars#1$\are#2$\same}
\def\allchars#1#2\are#3#4\same
{\if#1$\if#3$\saytrue\else\sayfalse\fi
\else \if#1#3\allchars#2\are#4\same
\else\sayfalse
\fi
\fi
}
\def\ifsamestring#1#2%
{\ifnum \allchars#1$\are#2$\same}
```

Now there is an outer `\ifnum` test, and `\TeX` will expand tokens after that test until two numbers and a relation remain.

Application 2: implementing the Lisp backquote macro

Coming (partly) from a Lisp programming background, I can't help being reminded of the Lisp backquote macro when using `\edef`. The backquote macro [2] is something like a reverse `\edef`: it expands nothing, unless you explicitly order it to. And, relishing a good `\TeX` hack, I was wondering if I could write something like that in `\TeX`. The answer turned out to be: yes. But it wasn't particularly easy.

Another incentive than sheer curiosity was the fact that, in a certain application, I was writing things like

```
\edef\act{\noexpand\af\noexpand\b}
```

² This implementation was suggested to me by Marc van Leeuwen.

```
{\noexpand\c\noexpand\d{\e}}}}}
\act
```

and I was getting tired of typing all the `\noexpand` commands. I wanted to tell TeX: ‘expand this’, instead of having to point out everything that shouldn’t be expanded.

My solution to this problem takes the form of a ‘backquoting definition’ `\bdef`, which, by the way, defines only macros without parameters. The basic principle is to traverse the replacement text, to reproduce everything in it, but to expand everything that has `\expand` in front of it.

The macro `\bdef` is just an `\edef` in disguise: it appends a terminator and installs a routine that will eat its way through the argument list.

```
\def\bdef#1#2{\edef#1{\TakeItem#2\Stop}}
```

As before, I save myself a lot of typing by putting `\let\xp=\expandafter`

For this macro I wanted to allow conditionals to be part of the argument. This meant being careful sequences such as

```
\if something #1 \else #2 \fi
```

are completely misunderstood by TeX if either of the arguments is some `\if...`, `\else`, or `\fi`. Therefore I allowed macro arguments to appear only in the tests themselves, and outside conditionals.

The first test is easy: if we have found the terminator we can stop.

```
\def\TakeItem#1%
  {\ifx\Stop#1\xp\StopTesting \else
   \xp\GroupTest\fi #1\stop\Stop}
\def\StopTesting#1\stop\Stop{}
\def\Stop{1}\def\stop{0}
```

If not, we have now one argument. This can be a single token, or it can be a group that was enclosed in `{...}`. We have to test for this distinction.

Note how the argument occurs only in the test, and is then reproduced outside the conditional for the benefit of the `\GroupTest` macro. The other macro, `\StopTesting` doesn’t need the argument, so it has to remove it. This slight overhead (also in most of the following macros) ensures that we will not have conditional tokens inside a conditional.

Now the macro `\GroupTest` receives as argument a string of tokens, delimited by `\stop\Stop`. If the argument of `\TakeItem` was a single token, `\GroupTest` will find `\stop` as its second argument, and it will invoke a routine that handles single tokens; otherwise the argument of `\TakeItem` must have been a group, and it will invoke a routine that handles groups.

```
\def\GroupTest#1#2#3\Stop
```

```
{\ifx#2\stop \xp\TakeToken
 \else \xp\TakeGroup\fi #1#2#3\Stop}
```

Single tokens can be `\stop` in which case the end of a group has been reached, and intake of tokens on this level can stop; otherwise it is a token that must be expanded or must be reproduced without expansion.

```
\def\TakeToken#1\stop\Stop
```

```
{\ifx#1\stop
 \xp\RemoveToken \else
 \xp\MaybeExpand \fi #1}
```

```
\def\RemoveToken#1{}%get rid of a \stop
```

Groups are handled by putting a left brace (recall that braces around macro arguments are removed), tackling in succession all tokens of the group, putting a right brace, and continuing with the items after the group.

```
\def\TakeGroup#1\Stop
```

```
{\leftbrace\TakeItem#1\rightbrace
 \TakeItem}
```

In between braces there is now a sequence delimited by `\stop`.

The following macros yield a left and right brace respectively:

```
\def\leftbrace{\iftrue\else\fi}
\def\rightbrace{\iffalse\else\fi}
```

which is based on the fact that the nesting structures of groups and conditionals are independent.

Now for the single tokens. If the token is `\expand` we have to expand the token following it, otherwise we reproduce the token without expansion.

```
\def\MaybeExpand#1{\ifx#1\expand
 \else \xp\id \fi #1}
\def\id#1{\noexpand#1\TakeItem}
```

If parameter 1 is `\expand` we let it stand; this has the effect of applying the macro `\expand` (see below) to what follows. Otherwise we apply `\id`, which has the effect of simply reproducing its argument; however, as we are still in the context of an `\edef`, this argument has to be prefixed with `\noexpand`.

Expansion of a token is a tricky activity. Merely reproducing a token will cause it to be fully expanded, as we are still inside an `\edef`. However, once we abandon control, we cannot get it back, so we will have to do all expansion ourselves.

At first I had here

```
\def\expand#1{\expandafter\TakeItem#1}
```

which is in the spirit of the Lisp backquote macro. However, it will not expand completely the way it is done inside an `\edef`. The solution I found to

this problem necessitated me to put a delimiter after the string to be expanded, instead of having it only prefixed. Tokens in between

```
\expand ... \endexpand
```

delimiters will be fully expanded. I don't believe solutions are possible without this closing delimiter.

As `\expandafter` is the only mechanism by which the user can explicitly force expansion, I arrived at the following idea. If a token is to be expanded, store a copy for comparison, hit the original over the head with an `\expandafter`, see if it still moves (that is, if it is not equal to the comparison copy), and if so, repeat this algorithm. Crude but effective³.

First the simple part: if we have found the closing `\endexpand` delimiter, we remove it and go on absorbing tokens after it.

```
\def\expand#1{\ifx#1\endexpand
  \xp\TakeFirst\xp\TakeItem
  \else \xp\fullexpand \fi #1}
\def\endexpand{2}% just to have it defined
Otherwise we compare the token to its expansion:
```

```
\def\fullexpand#1%
  {\xp\maybeexpandfurther\xp#1#1}
```

Comparison can be done by `\ifx`, which is able to handle both characters and control sequences.

```
\def\maybeexpandfurther#1#2%
  {\ifx#1#2%
   \xp\TakeFirstAndExpandOn \else
   \xp\TakeFirst\xp\expand \fi #1#2}
\def\TakeFirstAndExpandOn#1#2{#1\expand}
```

If the two parameters are the same we stop; otherwise we again `\expand`. Note that the call to `\TakeFirst` has the effect of removing the `#1` after the conditional; the `#2` is the expanded token, and it should be expanded further.

Now that we have all pieces together we can perform a small test: I put some conditionals in the test to make sure it would be hard.

```
\def\aa#1#2{\ifnum\count0>0
  \twice{#1}\else \thrice{#2}\fi}
\def\twice#1{#1#1} \def\thrice#1{#1#1#1}
\count0=0
\bdef\tmp{\a{bc}\fi\iftrue\b
  {\expand\a{fg}\hj\endexpand}z\else}
\show\tmp
```

which gives

³ Well ... Cases like `\ifnum ... \ifnum ... \fi\fi`, where expansion of one token yields the same token, go wrong.

```
> \tmp=macro:
-> \a {bc}\fi \iftrue \b {hjhjhj}z\else .
and
```

```
\count0=1
\bdef\tmp{\a{bc}\fi\iffalse\b
  {\expand\a{fg}\hj\endexpand}z\else}
\show\tmp
```

which gives

```
> \tmp=macro:
-> \a {bc}\fi \iffalse \b {fgfg}z\else .
```

The above implementation has a slight shortcoming, as it cannot distinguish between a single token and that same token with braces around it⁴. Both

```
\bdef\tmp{\a{b}}
```

and

```
\bdef\tmp{\a b}
```

give

```
>\tmp=macro:
```

```
-> \a b
```

Of course, shortcoming or not, this whole section is of rather academic value: `\bdef` is so much slower than `\edef` in execution that I've reconciled myself with writing lots of `\noexpand` tokens. But I do hope that these farfetched examples give inspiration to macro writers. Because `TEX`'s mouth does lend itself to useful purposes [3,-4]. And to loads of fun.

References

- [1] Donald Knuth, *The T_EXbook*, Addison-Wesley Publishing Company, 1984.
- [2] Guy L. Steele jr., Common Lisp, the language, Digital Press 1990.
- [3] Alan Jeffrey, Lists in T_EX's mouth, *TUGboat*, 11(1990), no. 2, 237-245.
- [4] Sonja Maus, An expansion power lemma, *TUGboat*, 12(1991), no. 2, 277.

◊ Victor Eijkhout
 Center for Supercomputing
 Research and Development
 University of Illinois
 305 Talbot Laboratory
 104 South Wright Street
 Urbana, Illinois 61801-2932, USA
 eijkhout@csrd.uiuc.edu

⁴ Also, it cannot cope with macros that expand to a space token or to nothing. The second objection can probably be repaired; the first is inherent to T_EX's parameter mechanism.

An Expansion Power Lemma

Sonja Maus

Most applications of the famous Power Theorem (*The TeXbook*, p. 202) use expansion of tokens in TeX's "mouth", and some primitive commands; the latter (in particular assignments) are done in TeX's "stomach" and can influence subsequent expansion. As an example of

Lemma 2. *TeX's expansion alone is also powerful*, the macro `\Copies` makes any number of copies of an argument by *expansion*. Here is the definition, to be read when @ is a letter:

```

1. \def\beforefi#1\fi{\fi#1}
2. \def\h@lve#1#2#3{\ifcase#1#2 0\or0\or
3.   1\or1\or2\or2\or3\or3\or4\or4\or5\or
4.   5\or6\or6\or7\or7\or8\or8\or9\fi
5.   \ifx#3:\else\expandafter\h@lve\number
6.   0\ifodd#2 1\fi\beforefi\space#3\fi}
7. \def\copies#1.#2{\ifodd#1 #2\fi
8.   \ifnum#1>@ne\expandafter\copies
9.   \number\h@lve0#1:\beforefi.{#2#2}\fi}
10. \def\nocopies#1.#2{}
11. \def\CCopies#1{\ifx#1-%
12.   \expandafter\nocopies\else
13.   \beforefi\copies#1\fi}
14. \def\Copies#1{\expandafter\CCopies
15.   \number#1.}

```

Examples of how to use `\Copies` and `\copies`:

```

16. \chardef\n=27 % or \newcount\n \n=...
17. \edef\asts{\Copies\n*}

```

is another solution of the `\asts` problem, see *The TeXbook*, Appendix D, section 1.

```
18. \message{\copies 79.-}
```

makes a row of 79 minus signs on the screen.

```

19. $$\chardef\n=4 {1+\sqrt{5} \overline{2}}=
20. \Copies{n}{1+\bgroup 1\overline{2}\egroup}\ldots
21. \Copies{n}{\egroup};.$$

```

displays the continued fraction

$$\frac{1 + \sqrt{5}}{2} = 1 + \frac{1}{1 + \frac{1}{1 + \dots}}$$

```

22. \newcount\m \newcount\n
23. \m=\dimen0 \divide\m by\baselineskip
24. \advance\m by1 \n=\m \advance\n by1
25. \parshape\n
26. \Copies{m}{0.5\hsize} Opt \hsize

```

defines a paragraph shape (*The TeXbook*, p. 101) which leaves space for a half-column picture of height `\dimen0`.

27. `\Copies{\Copies{10.1}()}`

keeps TeX's jaw muscles busy for a few seconds and expands to 1111111111 copies of nothing.

The general syntax is

```
\Copies<(number)><argument>
or \copies<integer constant>.12<argument>
```

with `(number)` and `<integer constant>` as in *The TeXbook*, p. 269. A single-token `(number)` does not need the braces. `<argument>` is an argument for an undelimited macro parameter: that is a sequence of tokens in explicit braces, or one token. The `(number)` will be expanded after `\Copies` has seen it, whereas the `<integer constant>.12` must be explicit before `\copies` is expanded. The `<argument>` (with braces stripped off as usual) is copied as many times as the `(number)` or `<integer constant>` says; a negative `(number)` counts 0.

Although the macros are hard to read, the way they work is easy to understand. With `\Copies`, the `(number)` is expanded by `\number` (15.), and `\CCopies` reads the first digit or minus sign and rules out negative numbers. The remaining tokens match the syntax for `\copies`. The expansion of `\copies` is best illustrated by some intermediate steps for (17.):

```
\copies 27.*
*\copies 13.{**}
***\copies 6.{****}
***\copies 3.{*****}
*****\copies 1.{*****}
*****\copies 1.{*****}
```

Here the desired number of '*'s is composed of powers of two. Division by 2 in this algorithm is done by `\number\h@lve0#1:` (9.), stepping through the (decimal) digits from left to right and carrying down a 1 for an odd digit (the #1 (2.) will always be 0 or 1). In (6,9.) `\beforefi` removes a `\fi` (tail recursion, see *The TeXbook* p. 219). When `\number` (9.) is complete, `\copies` (8.) expands again. Eventually, `\ifnum#1>@ne` (8.) turns false, and expansion finishes.

"Mouth & stomach" macros are usually simpler and more versatile than "pure expansion" macros. The latter are independent of grouping (20-21.) and can work in a context where commands cannot be executed (17.,18.,26.). For such rare occasions, Lemma 2 can be applied successfully.

◊ Sonja Maus
Memelweg 2
5300 Bonn 1
Germany

Generating \n asterisks

George Russell

At the start of Appendix D of *The T_EXbook*, Donald Knuth considers the “toy problem” of defining a macro `\asts` which contains just `\n` asterisks (where `\n` denotes one of T_EX’s count registers). For simplicity I intend to assume that `\n` is non-negative from now on, though ideally we should check that `\n` is a count register with non-negative contents before executing the macros.

The first solution by Knuth uses the following operations:

- (a) we can make `\asts` null with `\xdef\asts{}`;
- (b) we can add one asterisk to it with
`\xdef\asts{\asts*}`.

Therefore we just make `\asts` null and then add one asterisk to it `\n` times. Unfortunately T_EX needs $O(t)$ time to execute step (b) when `\asts` contains t tokens, so the whole method takes $O((\n)^2)$ time in total.

However it would be much nicer to have a solution which took time proportional to `\n` rather than its square. Knuth gives one. This works by building up a definition for `\asts` on the T_EX save stack using `\aftergroup`. But the problem with this solution is that as we rarely require a large save stack, most implementations only have a small one, so Knuth’s solution probably will not work for large `\n` (on the T_EX I usually use, it fails for `\n` bigger than 170 or so*).

Therefore I propose two refinements (in the order they occurred to me) which are both linear and allow `\n` to get quite large.

The first refinement can be thought of as follows. If we look again at the list of operations which were used for Knuth’s first solution, we see that there is a third useful operation which can be used to increase the size of `\asts`; namely:

- (c) we can *double* `\asts` using
`\xdef\asts{\asts\asts}`.

Thus if we want a 1000-asterisk macro, we can generate it by generating in turn 0-, 1-, 2-, 3-, 6-, 7-, 14-, 15-, 30-, 31-, 62-, 124-, 125-, 250-, 500- and 1000-asterisk macros, obtaining each from the previous one by adding an asterisk with (b) or doubling with (c) (this method is analogous to the algorithm for taking powers by repeated

* The T_EX implementation I usually use has a save stack of 600 words (the distribution default) and a main memory with 65535 (the default is about 30000).

squaring). Here therefore is my first solution to Knuth’s problem, which (as the reader can satisfy himself) is linear in `\n`.

```
\def\makeasts#1{\% Function is to make
%   \asts contain \n asterisks.
% \count0=#1 % Put argument into a
% register so we can do arithmetic on it.
% (it is OK to use \count0 and \count2
% as scratch registers as no output is
% generated in the macro so they will
% not be referenced by \output.)
\xfnum\count0=0 %
\xdef\asts{}% operation (a)
\else
\count2=\count0\divide\count2 by 2
      Set \count2 to half \count0.
%
\makeasts{\count2}%
\xdef\asts{\asts\asts}% operation (c)
\xifodd\count0
\xdef\asts{\asts*}\fi% operation (b)
\fi}
\makeasts{\n}
```

This solution is reasonably fast, and works with `\n` as large as 39800 on my local implementation (because it is limited by the size of T_EX’s main memory rather than the size of its save stack). Only a maniac would want more asterisks than that! Of course it will fall over for smaller values of `\n` if we have other stuff occupying the main memory (the figures given here were obtained on a version of T_EX with only the Plain T_EX macros loaded). So I was reasonably satisfied, until Chris Thompson, the local T_EX wizard, asked me the following question: is there a set of T_EX commands which expands to `\n` asterisks without using any primitive commands? (The tokens T_EX understands can be divided into those which reach its stomach, like the “typeset character” commands and `\def` (described in chapters 24–26 of *The T_EXbook*) and those which are expanded and removed in its mouth, like macros, `\if` and `\the` (described in chapter 20)). It looked as if the answer to Chris Thompson’s question was almost certainly ‘no’, because we are not even allowed to use the arithmetic operations. But `\the` provides a loop-hole, since `\the\n` expands to the digits of `\n`, and we can then operate on them. It took me several hours to produce a solution along these lines, which was ugly and slow (but it wasn’t a waste of time, since I learnt a lot about T_EX’s expansion mechanisms in the process). However I did eventually think of a much neater way. We need some initial definitions (which don’t overwrite any macros in Plain T_EX):

```
\def\f{\def\0{\sts\p}\def\1{\sts*\p}
\def\2{\sts**\p}\def\3{\sts***\p}
\def\4{\sts****\p}\def\5{\sts*****\p}
\def\6{\sts*****\p}\def\7{\sts*****\p}
\def\8{\sts*****\p}
\def\9{\sts*****\p}\def\q{}}
\def\sts#1\p#2\q#3{\csname#3\endcsname
#1#2#2#2#2#2#2#2\q}
```

After these, to set \sts to contain just `\n` asterisks you just have to type

```
\xdef\sts{\expandafter\sts
\expandafter\p\expandafter\q
\the\n}
```

I have deliberately left the above uncommented as I hope some readers will enjoy working out for themselves how the macros work. Note the use of `\csname...\endcsname` to provide a look-up table; a trick that every TeXhacker should know, though I used it here because I wanted to eliminate conditional commands since I regard them as “almost” primitive commands. The macro works for bigger `\n` than the previous one. I have used to it to produce 54250 asterisks. Furthermore it seems to be marginally faster on the local implementation. I shall be interested to see whether anyone can find a still faster macro!

Exercise for METAFONT hackers: Appendix D of *The METAFONTbook* begins with the problem of defining a macro containing exactly `n` asterisks. Rewrite the above bits of TeX in METAFONT to solve this as well.

◊ George Russell
Mathematics
Trinity College
Cambridge, England
GER11@uk.ac.cam.phx

TeX and Envelopes

Dimitri Vulis

I have revised and improved the LATEX envelope macros that I posted to TeXHAX some years ago. Using them may save money.

Why bar codes on envelopes and other USPS gossip

It is reported that recently the United States Postal Service board of governors approved the 27-cent “public automation rate” for first-class mail whose envelopes are pre-printed with a ZIP+4 code¹ and a Postnet code, the bar code often found in lower right corner of business reply and courtesy envelopes, saving 2 cents off the new 29-cent rate for first-class mail. In the past, organizations simultaneously mailing 10 pieces in the same ZIP code, or mailing 250 and even 500 pieces pre-sorted by ZIP code were given discounts; now the discount may extend to single letters.

The existing Post Office sorting machines read the bar code placed in the lower right corner of a letter-sized envelope, but the new wide-area scanners, to be installed in the spring of 1991, will read the bar code virtually anywhere on the envelope, and it will be possible to bar code larger letters, magazines, and catalogs—so called flats.

USPS optical scanners already generate Postnet bar codes while processing envelopes with address legible enough for the optical character reader (i.e., not handwritten), but the Post Office would prefer to deal with letters already with a Postnet bar code. USPS expects to save \$40 to \$80 million on every 1% of mail that is sent “pre-bar-coded”, and it passes a part of that saving back to the senders.

When a letter without a Postnet code is processed by the Post Office, an attempt is first made to feed it to an optical character reader (OCR) machine; if it succeeds in reading the address, it attempts to look up the ZIP+4 code in a database, sprays the Postnet code on the envelope, and from then on the envelope is handled automatically by bar code sorters (BCSs) at several points, until it reaches the destination post office; only then does a letter carrier read the address once again. However the OCR machines are known to be very finicky and it’s very difficult to print an address that will be reliably scanned. The OCR machines want the

¹ The system of 9-digit numeric codes developed by the United States Postal Service that identifies small groups of delivery addresses.

address to be printed in one of the few pre-approved fonts, in all capital letters and devoid of punctuation (the machines don't register differences in size, and may mistake a comma for a '9', or a period for an 'O').

A letter that is mailed already preprinted with the Postnet code and a facing identification mark (FIM—a series of longer bars on the top of the envelope, two inches from the right edge) bypasses the OCR and goes directly to the BCSSs. While the OCR scanning doesn't take much time, pre-printing the bar codes eliminates the possibility that the address will not be readable by the OCR, and that the address would have to be read by Post Office personnel several times during manual sorting, with a greater chance of human error, just as the case would be with a handwritten address.

Vendors of commercial bar code printing programs claim that putting a Postnet code and a FIM on an envelope speeds up delivery, but the Post Office denies this. My limited tests seem to indicate that there is no speed-up for mail sent locally in New York City, but non-local mail is sometimes delivered a day faster if it has pre-applied bar codes. The time saving is reported to be greater during Christmas season, when the postal system is flooded with letters with handwritten addresses.

The Post Office does say that using postage meter imprints, which don't need to be faced and canceled, instead of adhesive stamps, speeds up mail processing, and also saves time for the sender, who does not have to apply stamps.

By 1993 the Post Office plans to implement an 11-digit system, where the additional two digits will be used to sequence the mail in route delivery order. The *TeX* macros presented here should be compatible with this change.

TeX is very good at making bar codes. Some years ago I used METAFONT to make 3 of 9 bar codes, and Peter Flynn pointed out that *TeX* is very good with rules and can be used to make bar codes by itself, so I wrote these macros.

Unfortunately, mail with FIMs, ZIP+4 codes, and bar codes looks like junk mail, and recipients may throw it away without opening.

A solution to the problem of having to print the address on both the envelope and the letter inside it that is gaining popularity is to print the address, with the Postnet code beneath it, on the right side of the letter, and to fold and insert the letter into an envelope with a large transparent window, so that the address and the bar code are visible in the correct area on the envelope. Although envelopes with windows are more expensive, their cost is

offset by not having to print the address on the envelope and to match letters and envelopes. But there is a difference between what one wants to put on the letter and what the Post Office wants to see on the envelope to make it easier to read by OCR machines and letter carriers. In addition to specifying a visually unattractive typeface, all capitals and no punctuation, the Post Office wants the sender to simplify the OCR's and letter carrier's lives by using the standard, unattractive, 4-letters-or-less abbreviations for street suffixes:

Alley	ALY	Light	LGT
Annex	ANX	Loaf	LF
Apartment	APT	Locks	LCKS
Arcade	ARC	Lodge	LDG
Avenue	AVE	Loop	LOOP
Bayou	BYU	Mall	MALL
Beach	BCH	Manor	MNR
Bend	BND	Meadows	MDWS
Bluff	BLF	Mill	ML
Bottom	BTM	Mills	MLS
Boulevard	BLVD	Mission	MSN
Branch	BR	Mount	MT
Bridge	BRG	Mountain	MTN
Brook	BRK	Neck	NCK
Burg	BG	North	N
Bypass	BYP	Northeast	NE
Camp	CP	Northwest	NW
Canyon	CYN	Orchard	ORCH
Cape	CP	Oval	OVAL
Causeway	CSWY	Park	PARK
Center	CTR	Parkway	PKY
Circle	CIR	Pass	PASS
Cliffs	CLFS	Path	PATH
Club	CLB	Pike	PIKE
Corner	COR	Pines	PNES
Corners	CORS	Place	PL
Course	CRSE	Plain	PLN
Court	CT	Plains	PLNS
Courts	CTS	Plaza	PLZ
Cove	CV	Point	PT
Creek	CRK	Port	PRT
Crescent	CRES	Prairie	PR
Crossing	XING	Radial	RADL
Dale	DL	Ranch	RNCH
Dam	DM	Rapids	RPDS
Divide	DV	Rest	RST
Drive	DR	Ridge	RDG
East	E	River	RIV
Estates	EST	Road	RD
Expressway	EXPY	Row	ROW
Extension	EXT	Run	RUN
Fall	FALL	Room	RM
Falls	FLS	Shoal	SHL
Ferry	FRY	Shoals	SHLS
Field	FLD	Shore	SHR
Fields	FLDS	Shores	SHRS

Flats	FLT	South	S
Ford	FRD	Southeast	SE
Forest	FRST	Southwest	SW
Forge	FRG	Spring	SPG
Fork	FRK	Springs	SPGS
Forks	FRKS	Spur	SPUR
Fort	FT	Square	SQR
Freeway	FWY	Station	STA
Gardens	GDNS	Stravenue	STRA
Gateway	GTWY	Stream	STRM
Glen	GLN	Street	ST
Green	GRN	Suite	STE
Grove	GRV	Summit	SMT
Harbor	HBR	Terrace	TER
Haven	HVN	Trace	TRCE
Heights	HTS	Track	TRAK
Highway	HWY	Trail	TRL
Hill	HL	Trailer	TRLR
Hills	HLS	Tunnel	TUNL
Hollow	HOLW	Turnpike	TPKE
Inlet	INLT	Union	UN
Island	IS	Valley	VLY
Islands	ISS	Viaduct	VIA
Isle	ISLE	View	VW
Junction	JCT	Village	VLG
Key	KY	Ville	VL
Knolls	KNLS	Vista	VIS
Lake	LK	Walk	WALK
Lakes	LKS	Way	WAY
Landing	LNDG	Wells	WLS
Lane	LN	West	W

For example, OCR machines prefer to read addresses of this form:

JOHN DOE
11 PINE ST APT 4
ANYTOWN CA 10101-1000

On the other hand, the recipients prefer their address on the letter inside the envelope to be in mixed case, in the same typeface as the body of the letter, with punctuation, and spelled out:

John Doe
11 Pine Street, Apt. 4
Anytown, Calif. 10101

One can use control sequences \St, \Ave, et al., that expand to full words in the letter and to abbreviations on the envelope, to achieve this. One can also bypass the OCR machines by pre-printing the Postnet code.

Finding out ZIP+4 codes of your correspondents

In a few years we can expect that giving one's ZIP+4 code as part of one's postal address will be as common as giving the regular ZIP code, or the area code of the telephone number. Not giving

someone your ZIP+4 code will mean that that person either won't be able to take advantage of the lower postage or will have to make an extra effort to find out the full ZIP+4 code. Users of these macros are also likely to be faced with the problem of adding the 4-digit suffix to existing address files.

The Post Office says it will add ZIP+4 codes to a printout of an address list for free. I have not tested this service, but it says one can print out the contents of one's address book, send it to the Post Office, and get it back with the ZIP+4 codes added, which one can then add to one's file. The Post Office also says it would accept a mailing list (between 350 and 50,000 entries) on a floppy disk (in ASCII format with fixed length fields), and update it with ZIP+4 codes for free. (Ask for form 5603, "Request for ZIP+4 coding of address files on diskettes".)

There is a public-access terminal in New York Central Post Office which allows one to key in one's addresses and obtain the corresponding ZIP+4 codes. I've used it update much of my address book to ZIP+4 codes. However, I found that for many people I correspond with the only kind of postal address I have is of the form

Prof. Chaim R. Shafarevich
Department of Algebraic Masonry
Emanuelle University
Smalltown, NY 10101

The public access terminal calls such an address insufficient (indeed, it lacks the street address!), and refuses to give the ZIP+4 code. Experience shows that mail addressed in this manner still gets delivered, although 'Emanuelle University' is not what the Post Office calls a valid delivery address — the street address of its mailroom would be one. The 10101 post office in Smalltown knows where Emanuelle University is. Presumably, the same software is used by the Post Office for its free service, so such addresses won't be automatically ZIP+4-coded.

Luckily, both the street address and the ZIP+4 code are often found on preprinted departmental stationery, whose return address was presumably composed by Emanuelle University's mailroom in cooperation with the 10101 post office. Occasionally, distinct departments have distinct ZIP+4 extensions. The stationery also sometimes includes an internal mail stop; putting that on the envelope may make internal delivery faster after the Post Office delivers the letter to Emanuelle University's mailroom.

One should be wary of databases that match the 5-digit ZIP code with a city and state. The correct

name for 11375 (my home post office) is Forest Hills. One database lists most ZIP codes starting with 113 as "Flushing" (a different community a few miles away from Forest Hills), and another database lists it as "Forrest (sic) Hills".

Laying out the envelope: the macros

First the copyright notice...

```
% Copyright 1988, 1991 by Dimitri Vulis.  
% All rights reserved.
```

A Postnet bar code consists of 52 long and short bars. The code always starts and ends with long frame bars. Bars 2–46 represent the nine digits of the ZIP+4 code. Each digit is represented by five bars: two long and three short. The value of the digit equals the sum of the "weights" of the positions of the two long bars, assigned to be 7, 4, 2, 1, and 0 starting from the left. For example, 6 = 4 + 2 is represented by long bars in positions 4 and 2. The digit 0 is the sole exception, being represented by long bars in positions 7 and 4. The last digit of the code (bars 47–51) is the "correction digit" chosen so that the sum of all digits is divisible by 10. The following example shows all 10 digits:



Here the "f"s are the framing bars and the "c" is the correction digit, 5 in this example. The *TEX* commands I used to generate the code follow:

```
\newbox\PostNetBox  
\newbox\ZipBarL  
\newbox\ZipBarS  
\setbox\ZipBarL\hbox{\vrule \height.125in  
 \width.020in\hskip.0276in}  
\setbox\ZipBarS\hbox{\vrule \height.05in  
 \width.020in\hskip.0276in}  
  
% counters used to compute the checksum  
\newcount\ZipBarn  
\newcount\ZipBarm  
\chardef\ten=10  
  
\def\ZipBar@@@#1#2{  
 \expandafter\def\csname ZipBar@@#1\endcsname  
 {#2\advance\ZipBarn#1\relax}}  
  
\ZipBar@@@0{\copy\ZipBarL\copy\ZipBarL  
 \copy\ZipBarS\copy\ZipBarS\copy\ZipBarS}  
\ZipBar@@@1{\copy\ZipBarS\copy\ZipBarS  
 \copy\ZipBarS\copy\ZipBarL\copy\ZipBarL}  
\ZipBar@@@2{\copy\ZipBarS\copy\ZipBarS  
 \copy\ZipBarL\copy\ZipBarS\copy\ZipBarL}  
\ZipBar@@@3{\copy\ZipBarS\copy\ZipBarS  
 \copy\ZipBarL\copy\ZipBarL\copy\ZipBarS}
```

```
\ZipBar@@@4{\copy\ZipBarS\copy\ZipBarL  
 \copy\ZipBarS\copy\ZipBarS\copy\ZipBarL}  
\ZipBar@@@5{\copy\ZipBarS\copy\ZipBarL  
 \copy\ZipBarS\copy\ZipBarL\copy\ZipBarS}  
\ZipBar@@@6{\copy\ZipBarS\copy\ZipBarL  
 \copy\ZipBarL\copy\ZipBarS\copy\ZipBarS}  
\ZipBar@@@7{\copy\ZipBarL\copy\ZipBarS  
 \copy\ZipBarS\copy\ZipBarS\copy\ZipBarL}  
\ZipBar@@@8{\copy\ZipBarL\copy\ZipBarS  
 \copy\ZipBarS\copy\ZipBarL\copy\ZipBarS}  
\ZipBar@@@9{\copy\ZipBarL\copy\ZipBarS  
 \copy\ZipBarL\copy\ZipBarS\copy\ZipBarS}  
  
\def\ZipBar@@#1{\csname ZipBar@@#1\endcsname}  
  
\def\ZipBar@#1{  
 \ifx#1\null  
   \let\next\relax  
 \else  
   \ZipBar@@{#1}%  
   \let\next\ZipBar@  
 \fi  
 \next}  
  
\def\ZipBar#1{  
 \gdef\zipcode{#1}%  
 \ifx\zipcode\empty\else  
 \QFIMAttrue  
 \setbox\PostNetBox\hbox{  
 \copy\ZipBarL % start with a frame bar  
 \ZipBarn\z@  
 \ZipBar@#1\null  
 \ZipBarn\ZipBarn  
 \divide\ZipBarn\ten  
 \multiply\ZipBarn\ten  
 \advance\ZipBarn-\ZipBarn  
 \ifnum\ZipBarn<0  
   \advance\ZipBarn\ten  
 \fi  
 \ZipBar@@{\the\ZipBarn}% correction digit  
 \copy\ZipBarL % end with a frame bar  
 }%  
 \fi % if empty  
 }  
 }
```

The macros place a FIM on the envelope if and only if the Postnet bar code is also present. The FIM tells the optical scanner where the front top edge of the envelope is, and hence where to look for the postage, the address, and the bar code. There are four kinds of FIM marks, and for our purposes we are only concerned with FIM-A, which looks like this:



and I make it so:

```

\newbox\fimaBox
\setbox\fimaBox\vbox{%
\hrule\@height.625in\@width.031in}
\setbox\fimaBox\hbox{\copy\fimaBox
\hskip.0315in\copy\fimaBox\hskip.1565in
\copy\fimaBox\hskip.1565in
\copy\fimaBox\hskip.0315in\copy\fimaBox}
\newif\if@fima

```

Now we will arrange these bar codes and the two addresses on the envelope. The numerous boxes, some of which might be extraneous, are meant to insure that the bar codes start and end in the locations where the present scanners will look for them. I use envelopes with a pre-printed return address, and only the name needs to be filled in. Users of these macros may want to move the \FromBox around if they use blank envelopes. If one wishes to add a company logo, or other rasterized graphics, on the envelope using a \special, the Post Office recommends that it go to the left of the return address. If your device driver doesn't allow mixing of graphics and rotated output, this can be circumvented by rotating the picture in a paint program before including it.

```

\newbox\AddressBox
\newbox\FromBox

\setbox\FromBox\null

% font used for printing the address
\let\addressfont\twlsf

\def\EnvMakeBox#1#2{
\setbox#1\vbox{
\parindent0pt \leftskip0pt
\lineskip1pt \baselineskip14pt
\rightskip\@flushglue
\addressfont #2
}

\def\envaddress#1{\EnvMakeBox\AddressBox{#1}}
\def\from#1{\EnvMakeBox\FromBox{#1}}

\def\envelope{
\@fimafalse
\gdef\zipcode{\empty}
\setbox\PostNetBox\null
\setbox\AddressBox\null
}

\def\endenvelope{
\newpage
\if@fima
\ vbox to Opt{
\hbox to \hsize{\hfill\copy\fimaBox\hskip2in}
\vss}
\fi

```

```

% These magic numbers are for my stationery
\vbox{\vskip.435in
\hbox{\hskip.32in\copy\FromBox}}
\vfill
\vbox to0pt{\vss
% Address is 1 inch from the left
% and 1 inch from the bottom
\hbox{\hskip1in\box\AddressBox}
\vskip.375in}
\vbox to .625in{
\vfill
\hbox to \hsize{%
\hfill
\hbox to 3.875in{%
\unhbox\PostNetBox\hfill}}}
\vskip.25in
}}
```

For mail going outside of the US, the country name in capital letters should be the only information present on the last line of the address. Endorsements for special services (restricted delivery, do not forward, registered mail, forwarding and address correction requested, etc.) should be placed above and flush left with the delivery address. All these data can be given as an argument to the \envaddress macro.

Finally, here is a L^AT_EX file which actually produces an envelope.

```

\documentstyle[env]{article}
\nofiles
\pagestyle{empty}
\textwidth9.5in
\textheight4.125in
I hard-coded Commercial #10 size; other common envelope sizes are Monarch, 3.875in × 7.5in, and in Europe: DL, 110mm × 220mm and C5, 162mm × 229mm. Calling the macros is simple:
\begin{document}

\from{Dimitri Vulis}

\begin{envelope}
\ZipBar{10036-8099}
\envaddress{
Dimitri Vulis \\
Department of Mathematics/Box 330 \\
Graduate School & University Center \\
City University of New York \\
33 West 42 Street \\
New York, New York\quad\zip\zipcode
}
\end{envelope}
\end{document}

```

The argument of \ZipBar is saved in \zipcode and may be reused, but it may be preferable to use

only the 5 digit ZIP code in `\envaddress` so as to make the envelope look less like junk mail:

New York, New York\quad10036

It's possible to print several envelopes at once by repeatedly calling `\begin{envelope}` and `\end{envelope}`.

There are absolutely no conventions about feeding envelopes into various printers and telling the device driver to print landscape. With Eberhard Mattes' excellent `dvihplj` driver² and my unusual printer I seem to achieve reasonable results with the options `/tr3 /l-1in /t3.5in` (rotate 90 degrees clockwise, and change offsets). It's desirable to have the FIM facing the inside, since most printers can't print on the outside edge. One needs to discover the correct options and envelope feeding procedure for one's driver/printer combination by trial and error.

Managing address files and the future

The layout of database tables for mass mailing lists and for one's personal correspondence is similar: each row should contain a unique identifier for joining with other data tables, an optional salutation, and the complete address (preferably, with ZIP+4 code and carrier route, and room for ZIP+6). But the operations are somewhat different for the two applications. For a mass mailing list, it's desirable to detect similar entries (i.e., slight variations of the same addressee) to avoid duplicate entries, and to be able to select a random sample of a specified size (so-called *n*th sample, used for tests); while for a personal address file it would be convenient to have a tool similar in spirit to BIBTEX, where the user would reference only a name tag, or a name and a department tag, in the `TEX` file, and one or more clever programs would pull the missing address information from the address table and complete the letter and the envelope, just like BIBTEX retrieves references from tags in a `TEX` file. Oren Patashnik suggested that BIBTEX itself may be capable of doing this.

I may eventually write such a program. Alas, few people now write `TEX`ware and distribute it

² I use em`TEX`, the excellent implementation of `TEX` for MS-DOS, also by Eberhard Mattes, that's available for free; SBT`EX` is another such implementation, by Wayne Sullivan. Anecdotal evidence suggests that, regrettably, many *The TEXbook* readers are unaware that it's not necessary to spend hundreds of dollars to obtain `TEX`, and that free implementations are usually just as good as the commercial ones.

freely with source code. If I do write such a program, it will be distributed in the same spirit as `TEX` itself.

These macros have not been certified by the Post Office, and are not warrantied to do anything at all. You may use them at your own risk. The certification process costs \$375, and I'm not making any money by giving them away for free. The macros are copyrighted, though, and I intend to defend them strenuously against unauthorized commercial use.

◊ Dimitri Vulis
 Dept. of Mathematics / Box 330
 City University of New York
 Graduate Center
 33 West 42nd Street
 New York, NY 10036-8099
 Internet:
 dlv@cunyvms1.gc.cuny.edu

LATEX

The LATEX Column

Jackie Damrau

There has been much discussion as to what this column tries to cover. I would like to provide a few comments of what I view this column's purpose to be. I certainly welcome any comments from others on what they would like to see in this column as well.

I see my role as Associate Editor of this column to put forth questions, answers, and macros that other users—especially those of the novice class—would find most useful. However, what may work at one particular site may not always work the same at another. Bear in mind that most users can obtain an idea of how to create macros from what has been accomplished by others in creating their own macros. These are what I try to print in the LATEX column. These may not be the most sophisticated nor the most clever solutions to problems or situations. When I put forth a question that has been sent to me, I do not always provide an answer. I feel that anyone who would like to submit an answer to be published in the next issue deserves a chance to do so. It also helps to solicit answers from users who

are not necessarily available electronically and reach out to those who are advancing through the ranks from novice to L^AT_EXpert.

One goal I try to achieve when printing a solution in this column is not to imply this is the only solution. There are always many ways to solve a problem; however, the trick is to find the solution which is understandable and useful to the user and not necessarily to the half-dozen experts to whom the user has NOT turned. Users do not always wish to have the most detailed/clever/astute/elegant solution. The *role* of the T_EX or L^AT_EX consultant in this case *is* to be the person who provides a solution which is adequate to the problem and which is most useful and educational to the user.

Another goal I try to strive for is to reach out to novice users and invite them to submit their questions. We all began somewhere—some with advanced knowledge in computer programming; some just beginning to learn programming; and some with no programming background at all. There is a vast pool of knowledge in the T_EX community that needs to be shared and we should never treat a beginner's question—no matter how many times asked—like "That has been answered before, why ask again?".

We should strive to invite the novice user into our community in a friendly fashion. If someone repeats a question, we should provide them with some helpful suggestions and a possible place where relevant answers can be found (e.g., T_EXhax, *TUGboat*). In doing this, we all add enrichment to the T_EX community in our teaching, to our own knowledge, and we have gained one more contact in our everlasting networking scheme.

◊ Jackie Damrau
 SSC Laboratory
 Mail Stop 1011
 2550 Beckleymeade Avenue
 Dallas, TX
 email: damrau@sscvx1.ssc.gov

A comment on the L^AT_EX column

Nico Poppelier

1 Double spacing

In the L^AT_EX column of *TUGboat* 1, no. 1(4) (1990), Jackie Damrau presented two macros for changing from single line-spacing to double line-spacing. Everyone who has ever tried this knows that the simpler macros do not do what the author, Josephine Colmenares, claims them to do. The correct answer was already given by Jackie in her column in *TUGboat* 1, no. 1(1) (1990), namely

```
\newcommand{\single}{%
  \renewcommand{\baselinestretch}{1}%
  \large \normalsize}
\newcommand{\double}{%
  \renewcommand{\baselinestretch}{1.5}%
  \large \normalsize}
```

I will try to explain why only this works under *all* circumstances.

In the comment in *lfonts.tex* (version of 10 April 1989) we can read¹

```
% A SIZE COMMAND is something like
% \normalsize that defines a type size.
% It is defined by the document style.
% However, \normalsize is handled
% somewhat differently because it is
% called so often--e.g., on every
% page by the output routine. The
% document style defines \@normalsize
% instead of \normalsize.
```

The command \normalsize checks whether the current size is \normalsize.

- If it is, it switches to the \rm font
- If it isn't, it switches the new size by means of a call to the kernel macro \@setsizes

Somewhere else in *lfonts.tex* we find

```
% Each size command \SIZE
% executes the command
% \@setsizes\SIZE{BASELINESKIP}
%           \FONTSIZE\@FONTSIZE
% which does the following.
...
% 3. Sets \baselineskip to
%     \baselinestretch * BASELINESKIP
...
```

In other words, to switch between single and double spacing you have to do the following:

1. you start by re-defining \baselinestretch

¹ The comment is reformatted to fit into the narrow columns.

2. you make a switch to an arbitrary, *but really different*, font size
3. you switch back to `\normalsize`.

2 Footnotes in a minipage

I should also like to answer the minipage footnote problem Jackie reported. The answer is given on pages 91 and 99 of the L^AT_EX user's guide and reference manual. Alternatively, you can delve into the comment of `latex.tex`—a bit more effort—but the answer remains the same.

In the comment in `latex.tex` (version of 13 June 1989) we can read

```
% \minipage :  
% similar to parbox, except it also  
% ...  
% changes footnotes by redefining:  
% \@mpfn == mpfootnote  
% \thempfn == \thempfootnote
```

Inside a `minipage` environment L^AT_EX doesn't use the `footnote` counter, but the `mpfootnote` counter. And so a solution is

```
\renewcommand{\thempfootnote}{\arabic{mpfootnote}}
```

An example to prove that this solution works. This table is formatted without any re-definitions.

<code>\arabic</code>	arabic numerals ^a
<code>\roman</code>	roman numerals (lower case) ^b
<code>\alph</code>	lower-case letters (1–26)
<code>\fnsymbol</code>	symbols (1–9) ^c

^a Nice and simple.

^b I don't like 'mcmlxxviii'.

^c Old-fashioned.

This table is formatted with the re-definition I described above.

<code>\arabic</code>	arabic numerals ¹
<code>\roman</code>	roman numerals (lower case) ²
<code>\alph</code>	lower-case letters (1–26)
<code>\fnsymbol</code>	symbols (1–9) ³

¹ Nice and simple.

² I don't like 'mcmlxxviii'.

³ Old-fashioned.

◊ Nico Poppelier
 Elsevier Science Publishers
 Academic Publishing Division
 R&D Department
 Sara Burgerhartstraat 25
 Amsterdam, The Netherlands
 n.poppelier@elsevier.nl

L^AT_EX Tree Drawer

Glenn L. Swonk

Abstract

Today, many software systems are analyzed, designed and developed in a top-down hierarchical manner. This is especially apparent with Object Oriented Programming and Design. Diagramming a hierarchical relationship can be cumbersome in T_EX or L^AT_EX. This paper describes a MS-DOS tool that can be used to simplify the diagramming process by taking a simple input format and producing a L^AT_EX picture environment source which can be *input* in a L^AT_EX document.

Introduction

While trying to document a hierarchical directory structure, I manually placed the directory nodes' coordinates in a source file from calculations made from a rough sketch. Not only was this a tedious operation, it was prone to error and required many iterations before the exact result was achieved.

What I really wanted was a simple way specify the hierarchical relationship and an automated way to generate the output L^AT_EX file. Using the UNIX `find(1)` utility, it was easy to generate a list of files or directories that were to be plotted in a L^AT_EX picture environment. In the simplest case, `find(1)` can be used to generate a list of all files *under* a specified directory using the command `find <dirname> -print`.

What resulted from this problem is a MS-DOS tool which I call the L^AT_EX Tree Drawer (LTD). It takes input from a file in the form of a `find(1)` output and generates a L^AT_EX picture output that can be used as input to a L^AT_EX document. This paper describes the implementation and use of this tool.

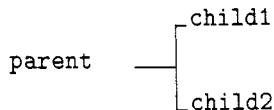
Samples

To best illustrate what LTD can do, a few samples are in order.

Example 1 illustrates the simplest example—a single parent node and its two children. From the input file in figure 1, the resultant picture is shown in figure 2.

```
parent
parent/child1
parent/child2
```

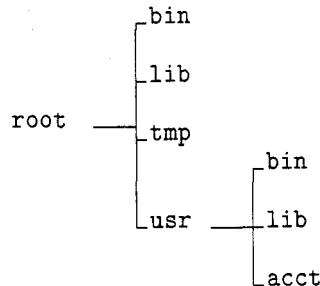
Figure 1: Example 1 Input File

**Figure 2:** Example 1 Output

The second example illustrates a partial UNIX-like directory filesystem. From the input file shown in figure 3, the diagram in figure 4 is the final output.

```

root
root/bin
root/lib
root/tmp
root/usr
root/usr/bin
root/usr/lib
root/usr/acct
  
```

Figure 3: Example 2 Input File**Figure 4:** Example 2 Output

The third example illustrates a shape class hierarchy.

From the input file in figure 5, the diagram in figure 6 is the final output. Note the *boxing* of the nodes. LTD can box all or either branch or leaf nodes.

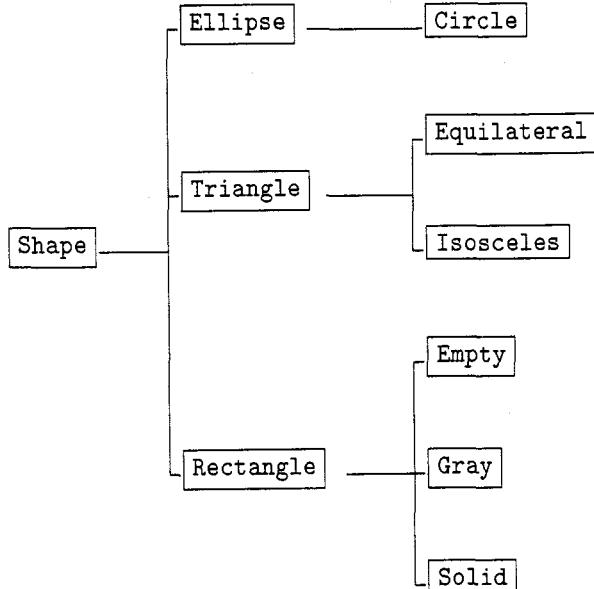
Implementation

The LTD implementation comprises four basic processing steps. The following provides a short description of each.

Input

The input phase of the program interprets the command line parameters specified by the user and sets the appropriate internal variables. It then reads in the nodes from the specified file and stores the nodes in an internal data structure representing its child/parent/sibling relationship. All input param-

Shape
Shape/Ellipse
Shape/Ellipse/Circle
Shape/Triangle
Shape/Triangle/Equilateral
Shape/Triangle/Isosceles
Shape/Rectangle
Shape/Rectangle/Empty
Shape/Rectangle/Gray
Shape/Rectangle/Solid

Figure 5: Example 3 Input File**Figure 6:** Example 3 Output

eter values are assumed to be specified in points (1/72").

Layout

The layout mechanism is based on an algorithm developed by Sven Moen from Brown University and is described in his paper in the July 1990 issue of IEEE Software. The layout algorithm uses information from the command line parameters and node information (such as width, height, and border) to produce a tree layout based on the node and sub-tree contours. The layout algorithm produces horizontal trees with the following characteristics:

- A parent is drawn to the left of its children.
- Trees are drawn from left to right to accommodate varying length strings.
- Subtrees look the same, regardless of position.
- Left children are drawn above the parent, right children below.

Planting

After the layout process, each node contains relative positional information that needs to be converted to absolute positions that can be used to generate the output code. The planting process performs this on the nodes to produce absolute positions compatible with the L^AT_EX picture environment (x increasing left to right, y increasing bottom to top).

Output

Up to now, all processing has been device independent (with the exception of assuming a point size coordinate system). The output process now traverses the internal data structures to produce the specified output code. The output code also contains additional information such as the command line parameters, node width and height, and source input lines that can be used for debugging.

Currently two types of output formats are implemented, L^AT_EX and raw data.

The L^AT_EX data can be redirected to a file and *input* into a document via the \input{} command.

The raw data format may be useful to users who wish to write their own output drivers for displaying the node layout information.

Node Layout Specification

To understand how the parameters from the LTD command line affect the layout, the picture in figure 7 defines the dimensions used in the layout algorithm. The dimensions are defined by the following:

- *w* – Width of the node. This dimension is calculated from the length of the string times the nominal font width parameter specified by -w <n>.
- *h* – Height of the node. This dimension is specified from the -h <n> command line parameter.
- *b* – Border of the node. This dimension is specified from the -b <n> parameter. It is used to provide the *spacing* around a node from its parent, siblings and children. By varying this parameter you can get the layout process to spread the height of the diagram. Negative values can be used to provide *very tight* spacing.
- *pd* – Parent Distance (not shown). This dimension is specified from the -pd <n> parameter. It is a global parameter to the layout algorithm for the layout process. By varying this parameter you can get the layout process to spread the width of the diagram.

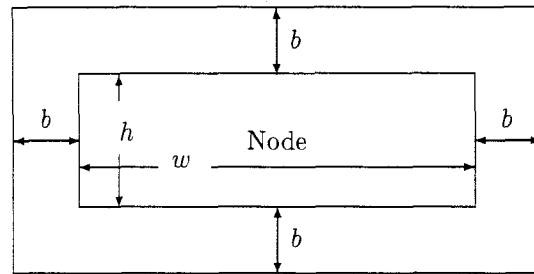


Figure 7: Node Dimensions

Command Line Parameters

The following list describes the command line parameters that can be used to modify the behavior of the LTD.

- -h <height> – Height of font, used for layout.
- -b <border> – Border distance, used for layout.
- -w <width> – Width of a single character, used for layout.
- -pd <parent_distance> – Minimum distance needed between a child and its parent, used for layout.
- -fix – A flag that can be used to create fixed sized widths for each level. Typically creates more visually pleasing diagrams.
- -f <filename> – Filename used for input.
- -o <filename> – Filename used for output.
- -ba – Switch to form box around all nodes.
- -bb – Switch to form box around branch nodes.
- -bl – Switch to form box around leaf nodes.
- -tex – Generate L^AT_EX output format (default).
- -xy – Generate XY output format.
- -help – Print help text of usage to stderr.
- -xo <xoffset> – a parameter that is used to shift the x position of the picture.
- -yo <yoffset> – a parameter that is used to shift the y position of the picture.
- -xs <xsize> – a parameter that is used to adjust the x size of the output picture.
- -ys <ysize> – a parameter that is used to adjust the y size of the output picture.

Other parameters are available in the most recent version of LTD and should be defined in the help text. The actual parameters used are printed out as comments for debugging and repeatability.

Using LTD

Invoking LTD

Typically, LTD is invoked from the DOS command line with a filename and any parameters to modify its default configuration. In order to use the output, you need to redirect the output to a filename which is eventually used for the picture environment.

In most cases, following is the simplest form:

```
ltd -f test.fil >test.ltd
```

Using LTD's Output

Once you have a file with the plotted data, you must *input* the file into a LATEX file. I suggest using a base file that includes the output file similar to the following form:

```
%  
% base.tex  
%  
\documentstyle{report}  
\begin{document}  
  
\begin{figure}[h]  
\begin{center}  
\fbox{ \tt \input{test.ltd} }  
\end{center}  
\caption{LaTeX Tree Drawer Demo}  
\end{figure}  
  
\end{document}  
%% ///////////////////////////////////////////////////////////////////  
%% /// eof ///////////////////////////////////////////////////////////////////  
%% ///////////////////////////////////////////////////////////////////
```

Note that you have full control of the font type and size that is used for the picture. By varying the font size with the command line parameters, you can vary the appearance of the final picture to provide the most aesthetic diagram.

Note that the **fbox** command is optional and is used to create an enclosure for the final output. This provides the user with an idea of the extent of the diagram. The box that borders the diagram can be shifted/sized by the optional command line parameters.

Limitations

The LTD has the following limitations:

- The input file must define the parent nodes before the child nodes can be defined. If this is not the case, some nodes may not be displayed properly.
- The '/' character is assumed to be the separator character.

- An input file character that is interpreted as a control sequence to **TeX** or **LATEX** can cause problems, specifically the '_' (underscore) character.
- Since the font that is being used is completely under the control of the user, we can only *guess* at the width of the parent node. Therefore, some of the connectors to the parent node may fall short or exceed the desired point. To solve this, it may be necessary to specify the **-w** parameter and/or use a fixed width font. Some hand tweaking may also be required to get the optimal results.
- The program is written to use defaults for a 12pt character font.
- Maximum number of nodes is ≈ 500 , but will vary depending on size of the strings.
- The size of the picture is calculated based on the min/max of the x and y dimensions. The width of the string is not taken into consideration, therefore some shifting of the picture may be necessary to center the picture.

Availability

Version 1.0 of LTD is in the public domain. I hope to submit the DOS executable to an archive server in uuencoded format for general availability.

Any ideas or general comments should be directed to the author at `uunet!toshais!swonk`.

References

- [1] Moen, Sven. "Drawing Dynamic Trees". IEEE Software (July 1990).
- [2] Lamport, Leslie. **LATEX**: A Document Preparation System. Reading, Mass.: Addison-Wesley, 1986.

◊ Glenn L. Swonk
25302 Fairgreen
Mission Viejo, CA 92692
`uunet!toshais!swonk`

“See also” indexing with makeindex

Harold Thimbleby

Makeindex (a program by Pehong Chen[1]) allows an author to construct an index by writing `\index{}` entries in his L^AT_EX document. Thus, writing `\index{LISP}` in the source file gets (by juggling through makeindex) an index entry of the form, “LISP, 23”—that is, if the index entry for LISP happens to land on page 23 when the document is typeset.

In many indices, it is useful to have index entries that redirect the reader to an alternative or more appropriate heading. Makeindex provides a simple facility for this. For example, by writing, `\index{artificial intelligence|see{AI}}`, makeindex inserts `\see{AI}` just in front of the page number.

If `\see` is defined to gobble up its second argument, then the index ends up with just the “*see*”: for example,

```
\index{artificial intelligence|see{AI}}
```

results in an index entry:

“artificial intelligence, *see* AI”.

Now if an index entry has both page numbers and a “*see*” entry, two things go wrong. First, it is more appropriate to say “*see also*”, and secondly makeindex puts the “*see*” entry in the wrong place—that is, so far as it is concerned, in the ‘right’ place corresponding to the gobbled-up page number, which we’re not interested in.

The following macros provide the appropriate facility, and also indent the “*see also*” neatly as if was a subitem. (By using `!zzzzz`, the subitem is guaranteed to come at the end of the index entry—unless you have some obscure entries that come alphabetically after `zzzzz`!)

```
\def\subsee#1#2{\{ \em see also\}/} #1
%   the #2 consumes a comma
\def\nosee#1{}
%   consume the page number
\def\sealso#1#2{%
  \index{#1!zzzzz@\subsee{#2}\nosee{}}}
```

Given these definitions, and supposing index entries for “Scheme” occur on pages 147 and 401, this is how `\sealso{Scheme}{LISP}` would end up in the index:

Scheme, 147, 401
see also LISP

As with the normal makeindex `see{}` construction, it does not matter where in your document `\sealso` is used: it will get placed at the end of

the index entry. So you can place it anywhere convenient.

Reference

- [1] Pehong Chen and Michael Harrison, “Automating Index Preparation”, Tech. Report No. 87/347, Computer Science Division, University of California, Berkeley, March 1987.

◊ Harold Thimbleby
 Stirling University
 Stirling
 Scotland, FK9 4LA

Babel, a multilingual style-option system for use with L^AT_EX's standard document styles*

Johannes Braams

Abstract

The standard distribution of L^AT_EX contains a number of document styles that are meant to be used, but also serve as examples for other users to create their own document styles. These styles have become very popular among L^AT_EX users. But it should be kept in mind that they were designed for American tastes and contain a number of hard-wired texts. This article describes a set of document-style options that can be used in combination with the standard styles, which makes the latter adaptable to other languages.

1 Introduction

Although Leslie Lamport has stated [5] that one should not try and write *one* document-style option to be used with *all* the standard document styles of L^AT_EX, that is exactly what I have done with this system of style options. The reasons for this approach will be explained in section 2.

A lot of the ideas incorporated in this set of files come from the work of Hubert Partl [4], *german.tex*. Some parts in the implementation are different, others are the same. It will be shown that *german.tex* can be modified to fit into this scheme of style options.

2 Why Babel?

When I first started using L^AT_EX I was very happy with just the style files that are distributed with the standard distributions of T_EX and L^AT_EX. That means, as long as I made texts in English I was happy. Then as other users found out about L^AT_EX and its advantages, they started using it for texts in languages other than English. As I was the most experienced L^AT_EX user at the time, they came to me and asked me ‘When I’m writing a report in Dutch I don’t want chapters to be named “Chapter”, I want them to be named “Hoofdstuk”, how do you change that?’ At that time I didn’t know, but I soon found out. The first thing I found was that Leslie Lamport states [2, pages 85–86] that you have to redefine the command \chapapp to get the desired result. This looked rather promising to me, so I had a look at the style files to find out how other such strings as “Figure” might be redefined. It was then that I found out that \chapapp is the *only* string defined this way, whereas all others are hard-wired into the style.

My first solution to this problem was to create a new document style file called *artikel.sty* as a “Dutch” counterpart to *article.sty*. The same was done for *report.sty*. This is exactly what Leslie Lamport suggests [5]. This approach has one major drawback however: you get two copies of basically the same file to maintain. This was discovered when newer releases of the styles reached our site. The standard styles had to be replaced *and* edited all over again to get the “Dutch” versions back. About the same time, in early 1988, a discussion on this subject appeared in T_EXhax. One of the persons commenting was Hubert Partl. The method he suggested was to modify the standard document styles by replacing the hard-wired texts by macros such as \chapapp. This led me to my second attempt at a solution. I modified the standard styles (all four of them) as suggested, but while doing that added an option, implemented like the option *draft*, by defining a command \ds@dutch.

* During the development ideas from Nico Poppelier and Piet van Oostrum have been used.

This command would set a variable to indicate which language was requested. This variable I used later on in a `\case` statement. In this `\case` statement a choice is made between English, Dutch and possibly other languages for texts such as "Figure" and "Contents". Unfortunately, some of this implied changing the secondary style files `xxx10.sty`, `xxx11.sty` and `xxx12.sty`. This was unfortunate because one of the research groups in our laboratories complained their document style didn't work properly. It turned out that their style was a modified `article.sty` that had been given a different name, but it still loaded `art10.sty` etc. I found a temporary solution, but I still wasn't exactly happy with the situation. Besides this, the drawback of replacing the document styles with newer versions still existed.

When after a while a new version of the L^AT_EX distribution arrived at our site, I began to think about a different way to solve the problem. In the meantime Hubert Partl had his `german.sty` published in *TUGboat* [4]. His article pointed the way to a different solution. Triggered by the discussion in T_EXhax in early 1989 about how to detect which is the main (primary) style when processing a document, I started work on what is now available as `dutch.sty` version 1.0, dated May 1989¹. While working on this style option I discovered that some parts could be borrowed from `german.sty`. This 'discovery' and some discussions I had with others at EuroTEX89, the fourth European T_EX Conference, held in September 1989 in Karlsruhe, led me towards a more universal approach. The basic idea behind it was, starting from the algorithm to detect the main style, to design an approach with one common file that contained macro definitions needed by a number of language-specific style options. Users specify the name of any of these language-specific options as an option to the `\documentstyle` command, and internally the common file is read.

3 L^AT_EX and document-style files

Before I discuss some of the code in the `babel` system I would like to discuss the document-style mechanism used by L^AT_EX. Every L^AT_EX document should start with a line like:

```
\documentstyle[opt1,opt2,...]{docstyle}
```

This line of code instructs L^AT_EX to first load the file `docstyle.sty`. When that is done the 'options' are processed *in the order specified*, by reading the files `opt1.sty`², `opt2.sty`, etc. This implies that definitions made in the file `docstyle.sty` can be overridden in one of the option files. It is even possible to redefine code from the very kernel of L^AT_EX, but you have to know what you are doing.

Some care has to be taken in writing document-style options, because a number of problems can occur. First of all, a document-style option should be modest in size; if it tries to redefine most of the code in `docstyle.sty` I think you should write (and maintain) your own, complete, document style. Next, as it was possible to override definitions from the main file in an option file, it is of course also possible to override definitions made in another option file. When this happens, your document might depend on the order in which you have specified your document-style options.

This mechanism of overriding definitions from the main document style is exploited in the `babel` system. The macros that contain the hard-wired texts are redefined in the common part of `babel`, replacing each of these texts with a unique macro. These macros have to be defined in the language-specific files.

4 L^AT_EX and multilingual documents

In a european environment it sometimes happens that one wants to write a document that contains more than *one* language. I have an example of a document, published

¹ This file is available from `listserv@hearn.bitnet` as file `dutch.old`.

² Except when the `\documentstyle` defines the control sequence `\ds@{opt1}`; in that case this control sequence will be executed.

by the EEC, that contains 9 (nine) different languages. Also in linguistics one can find documents written in more than one language, i.e. to compare two languages.

If you have to write such a multilingual document you should try to conform to the typographical conventions in use for each language. A well known example is the type of quotation marks used. *TeX* supplies the user with "quoted text", but a Dutch user might want to have „quoted text”, whereas a German text should contain „quoted text“ and a Frenchman would perhaps like to see something like «quoted text». These language-specific conventions should be implemented in a document-style option file for each language. These files should then be useable with *all* document styles.

In such a multilingual document a user would specify the languages used as options to the \documentstyle command. He would also want a mechanism to be able to switch between these languages in a simple way. When he would use *TeX* version 3.0 for processing his document, he would also want the hyphenation to come out right for the different languages.

5 Overview of the *babel* solution

5.1 The core of the system

The problems described in sections 3 and 4 can be solved using the *babel* system of document-style options.

The core of this system currently performs three functions.

1. It defines a user interface for switching between languages;
2. It contains code to dynamically load several sets of hyphenation patterns;
3. It ‘repairs’ the document styles provided in the standard distribution of *LATEX*.

Obviously part 2 can only be used while running *iniTeX* to create a new format, whereas part 3 should *not* be read by *iniTeX*. Part 3 should even disappear when *LATEX* version 3.0 arrives, as the style files supplied with the new *LATEX* will no longer be language specific. Part 1 can either be loaded into the format with multiple hyphenation patterns, or it can be read while processing a document.

For this reason the core of the *babel* system is stored in two separate files, *babel.switch*, containing parts 1 and 2, and *babel.sty* which contains part 3. The file *babel.sty* will instruct *LATEX* to load *babel.switch* if necessary; the file *babel.switch* checks the format to see if hyphenation patterns *can* be loaded.

5.2 Language specifics

The language switching mechanism contains a couple of hooks for the developers of language-specific document-style options.

First of all the macro \originalTeX should be defined. Its function is to disable special definitions made for a language to bring *TeX* into a ‘defined’ state. A language-specific document-style option might, for example, introduce an extra active character. It would then also modify the definitions of \dospecials and \Qsanitize. Such an option would then define a macro to restore the original definitions of these macros and restore the extra active character to its normal category code. It would then \let \originalTeX to this ‘restoration’ macro.

To enable the language-specific definitions three macros are provided in the switching mechanism, \captions(*language*), \date(*language*) and \extras(*language*).

The macro \captions(*language*) should provide definitions for the macros that replaced the hard-wired texts in the document style and the macro \date(*language*) should provide a definition for \today. The real fun starts with the macro \extras(*language*). This macro should activate all definitions needed for (*language*).

6 The user interface

The user interface to the *babel* system is quite simple. He should specify the languages he wants to use in his document in the list of document-style options. For instance,

for a document in which both the English and the Dutch language are used, the first line could read:

```
\documentstyle[a4,dutch,english]{artikel1}
```

Please note that in this case the Dutch-specific definitions are inactive when L^AT_EX has finished processing document-style option files.

If the user then wants to switch from English to Dutch he would include the command

```
\selectlanguage{dutch}
```

before starting to write Dutch.

If a user wants to write a document-style option of his own he might like to define a macro that checks which language is in use at the time the macro is executed. For this purpose the macro `\iflanguage{language}{then-clause}{else-clause}` is available.

7 Implementation of the core of the system

In this section I would like to discuss some parts of the implementation of the `babel` system. Not all code will be shown, because some parts of it are just series of slightly modified code from the standard document styles. The files are fully documented and interested readers can print them if they have access to the `doc` option, described by Frank Mittelbach.

The description of the macros that follows is based on an environment using T_EX 3.x, together with a version of `lplain.tex` based on `plain.tex` version 3.x. The actual implementation allows for other situations as well, i.e. a version of `babel.sty` for T_EX 2.x will be available.

7.1 Switching languages

For each language to be used in a document a control sequence of the form `\l@{language}` has to be defined. This will be done either while loading hyphenation patterns or while loading the language-specific file. The implementation of `\selectlanguage{language}` and `\iflanguage{language}{then-clause}{else-clause}` is based on the existence of `\l@{language}`.

```
\def\selectlanguage#1{%
  \ifundefined{\l@#1}
    {\@nolanerr{\l@#1}}
    {\originalTeX
      \language=\expandafter\csname \l@#1\endcsname\relax
      \expandafter\csname captions#1\endcsname
      \expandafter\csname date#1\endcsname
      \expandafter\csname extras#1\endcsname
      \gdef\originalTeX{\expandafter\csname noextras#1\endcsname}
    }
}
```

Figure 1: The definition of `\selectlanguage`.

To switch from one language to another the macro `\selectlanguage` is available. Its definition can be seen in figure 1. The first action it takes is to check whether the `{language}` is known, if it is not an error is signalled. If the language is known `\originalTeX` is called upon to reset any previously set language-specific definitions. Next the register `\language` is updated and the three macros that should activate all language-specific definitions are executed. Finally the macro `\originalTeX` receives a new replacement text in order to be able to deactivate the definitions just activated.

```
\def\iflanguage#1#2#3{%
  \Qifundefined{l@#1}
  {\Qnolanerr{#1}}
  {\ifnum\language=\expandafter\csname l@#1\endcsname\relax
   #2 \else #3
   \fi}
}
```

Figure 2: The definition of `\iflanguage`

The macro `\iflanguage` (see figure 2) will issue a warning when its argument is an ‘unknown’ language. It then goes on to compare the value of `\language` and `\l@⟨language⟩` and executes either its second or third argument.

7.2 Dynamically loading patterns

With the advent of TEX 3.0 it has become possible to build a format with more than one hyphenation pattern preloaded. The core of the babel system provides code, to be executed by iniTEX *only*, to dynamically load hyphenation patterns. The only restriction is that the implementation of TEX that you use has to have rather high settings of `trie_size` and `trie_op_size` to actually load several hyphenation patterns.

For the purpose of dynamically loading hyphenation patterns a ‘configuration file’ has to be introduced. This file will be read by iniTEX. Each line should contain either a comment, nothing or the name of a language and the name of the file that contains the hyphenation patterns for that language. In figure 3 an example of such a file, instructing iniTEX to load patterns for three languages, English, Dutch and German.

```
% File      : language.dat
% Purpose   : tell iniTeX what files with patterns to load.
english    english.hyphenations

dutch      hyphen.dutch % Nederlands
german    hyphen.ger
```

Figure 3: An example configuration file

The configuration file will be read line by line using TEX’s `\read` primitive. Because the name of a file might be followed by a space-token and comment (as in the example) a macro to process each line is needed. The definition of this macro, `\process@language`, can be found in figure 4. As can be seen in the definition of this macro, its second argument *always* has to be followed by a space-token. The effect of this is that any trailing spaces are removed. The macro strips all spaces following its arguments. Its first argument is used to define `\l@⟨language⟩`. The macro `\addlanguage` is basically a non-outer version of the plain TEX macro `\newlanguage`. The second argument of `\process@language` is the name of the file containing the

```
\def\process@language#1 #2 {%
  \expandafter\addlanguage\csname l@#1\endcsname
  \expandafter\language\csname l@#1\endcsname
  \input #2}
```

Figure 4: The definition of `\process@language`.

following its arguments. Its first argument is used to define `\l@⟨language⟩`. The macro `\addlanguage` is basically a non-outer version of the plain TEX macro `\newlanguage`. The second argument of `\process@language` is the name of the file containing the

hyphenation patterns. Before the file can be read, the register `\language` has to be updated.

The configuration file is read in a `\loop` (see figure 5). When a record is read from the input file a check is done whether the record was empty. If it was not, a space token is added to the end of the string of tokens read. The reason for this is that we have to be sure there always is at least *one* space token present. When that has been taken care of the data just read can be processed. The last thing to do is to check the status of the input file, in order to decide whether TeX has to continue processing the `\loop`. When all patterns have been processed the value of `\language` is restored.

```
\loop
  \read1 to \@config@line
  \ifx\@config@line\empty
  \else
    \edef\@config@line{\@config@line\space}
    \expandafter\process@language\@config@line
  \fi
  \ifeof1 \qmorefalse \fi
  \if\qmore\repeat
\language=0
```

Figure 5: Reading the configuration file line by line

7.3 ‘Repairing’ LATEX’s standard document styles

A large part of the core of the babel system is dedicated to ‘repair’ the standard document styles. This means redefining the macros in table 1.

macro	article	report	book	letter
<code>\fnum@figure</code>	×	×	×	×
<code>\fnum@table</code>	×	×	×	×
<code>\tableofcontents</code>	×	×	×	
<code>\listoffigures</code>	×	×	×	
<code>\listoftables</code>	×	×	×	
<code>\thebibliography</code>	×	×	×	
<code>\theindex</code>	×	×	×	
<code>\abstract</code>	×	×	×	
<code>\part</code>	×	×	×	
<code>\chapter</code>		×	×	
<code>\appendix</code>		×	×	
<code>\cc</code>				×
<code>\encl</code>				×
<code>\ps@headings</code>				×

Table 1: macros that need to be redefined for the four standard document styles.

As an example of the way the macros have to be redefined, the redefinition of `\tableofcontents` is shown in figure 6.

The standard styles can be distinguished by checking the existence of the macros `\chapter` (not in `article` and `letter`) and `\opening` (only in `letter`). The result of these checks is stored in the macro `\doc@style`. When `\doc@style` already exists (which is the case when for instance `artikel1.sty` is used [7]) it is not superseded (see figure 7).

```
\@ifundefined{contentsname}
  {\def\tableofcontents
    {\section*{\contentsname
      \Gmkboth{\uppercase\expandafter{\contentsname}}
      {\uppercase\expandafter{\contentsname}}}}
  }
  {\@starttoc{toc}}
}
{}
```

Figure 6: An example of redefining a command

```
\@ifundefined{doc@style}
  {\def\doc@style{0}
   \@ifundefined{opening}
     {\@ifundefined{chapter}
       {\def\doc@style{1}}
       {\def\doc@style{2}}
     }{\def\doc@style{3}}
   }\relax
}
```

Figure 7: Determining the main document style

8 Implementing a language-specific document-style option file

To illustrate the way a language-specific file can be implemented, the file `dutch.sty` is discussed here. Note that not all of the code contained in the file `dutch.sty` is shown here; only those parts of interest for the scope of this article are included. If the reader would like to see the complete code, he can print all files in the `babel` system, using the file `doc.sty`, described by Frank Mittelbach in [6].

8.1 Compatibility with plain \TeX

The file `german.tex` [4] was written in such a way that it can be used by both plain \TeX users and \LaTeX users. This seemed a good idea, so all files in the `babel` system can be processed by both plain \TeX and \LaTeX . But some of the “useful hacks” from \LaTeX are used, so for a plain \TeX user they have to be defined. For this purpose the format is checked at the start of a language-specific file. If the format is `plain` an extra file, called `latehxhax.tex`, is read.

```
{\def\format{plain}
\ifx\fmtname\format
  \expandafter\ifx\csname @ifundefined\endcsname\relax
    \gdef\next{latehxhax.sty}
    \aftergroup\input\aftergroup\next
  \fi
}\fi}
```

Figure 8: Conditional loading of `latehxhax.sty`

This file should be read only once, so another check is done on the existence of one of the commands defined there.

A new group is started to keep the definition of the macro `\format`, which is used in the following if statement, local. When the current format turns out to be plain \TeX

the file `latexhax.sty` has to be read. But the definitions in that file should remain valid after the group is closed. This could be accomplished by making all definitions `global`, but another solution is to tell `TEX` to process the file `latexhax.sty` *after* the current group has been closed. The command `\aftergroup` puts the next token on a list to be processed after the group.

8.2 Switching to the Dutch language

In section 7.1 the names of macros needed to switch to a language have been described. In figure 9 these macros and their definition are shown for the Dutch language.

```
\def\captionsdutch{\gdef\refname{Referenties}%
    \gdef\abstractname{Samenvatting}%
    \gdef\bibname{Bibliografie}%
    ...
    \gdef\pagename{Pagina}%

\def\datedutch{%
    \gdef\today{\number\day`-\ifcase\month\or
        januari\or februari\or maart\or april\or
        mei\or juni\or juli\or augustus\or
        september\or oktober\or november\or december\fi
        \space \number\year}}
\begin{group} \catcode`\\"=active

\gdef\extrasdutch{%
    \global\let\dospecials\dutch@dospecials
    \global\let\@sanitize\dutch@sanitize
    \catcode`\\"=active
    \gdef"\{\protect\dutch@active@dq\}
    \gdef\"{\protect\@umlaut}
}\end{group

\def\noextrasdutch{%
    \catcode`\\"=12
    \global\let\dospecials\original@dospecials
    \global\let\@sanitize\original@sanitize
    \global\let\"=\dieresis
}
```

Figure 9: The macros needed to switch to the Dutch language

The definitions of `\captionsdutch` and `\datedutch` are pretty straightforward and need not be discussed. The macro `\extrasdutch` will be discussed in some more detail.

First, because for Dutch (as well as for German) the " character is made active, the `LATEX` macros `\dospecials` and `\@sanitize` have to be redefined to include this character as well. The new definitions are implemented as two special commands, so we globally `\let` the originals to their new versions. Then the " character is made active and is defined. Then, to prevent an error when \" appears in a moving argument, the macro \" is redefined and made robust. All this is done inside a group to keep the category code change for the " character local.

The macro `\extrasdutch` has a counterpart, `\noextrasdutch`, that cancels the extra definitions made by `\extrasdutch`. It changes the `\catcode` of the " character back to 'other' and globally `\lets` the macros `\dospecials` and `\@sanitize` to their original definitions. The original definition of \" is restored as well.

In figure 10 the code needed to redefine `\dospecials` and `\@makeother` is shown.

```
\begingroup
  \def\do{\noexpand\do\noexpand}%
  \xdef\dutch@dospecials{\dospecials\do"}%
  \expandafter\ifx\csname @sanitize\endcsname\relax
  % do nothing if \@sanitize is undefined...
  \else
    \def\@makeother{\noexpand\@makeother\noexpand}%
    \xdef\dutch@sanitize{\@sanitize\@makeother"}%
  \fi
\endgroup

\global\let\original@dospecials\dospecials
\global\let\original@sanitize@\sanitize
```

Figure 10: Code needed for the redefinition of `\dospecials` and `\@makeother`.

8.3 An extra active character

All the code discussed so far is necessary because we need an extra active character. This character is then used as indicated in table 2. One of the reasons for this is that in the Dutch language a word with an umlaut can be hyphenated just before the letter with the umlaut, but the umlaut has to disappear if the word is broken between the previous letter and the accented letter.

"a	\"a which hyphenates as -a; also implemented for the other letters.
"	disable ligature at this position.
"-	an explicit hyphen sign, allowing hyphenation in the rest of the word.
"`	lowered double left quotes (see example below).
"'	normal double right quotes.
\-	like the old \-, but allowing hyphenation in the rest of the word.

Table 2: The extra definitions made by `dutch.sty`

In [3] the quoting conventions for the Dutch language are discussed. The preferred convention is the single-quote Anglo-American convention, i.e. 'This is a quote'. An alternative is the slightly old-fashioned Dutch method with initial double quotes lowered to the baseline, „This is a quote”, which should be typed as "‘This is a quote’".

8.3.1 Supporting macro definitions

The definition of the active " character needs a couple of support macros. The macro `\allowhyphens` is used make hyphenation of word possible where it otherwise would be inhibited by `TEX`. Basically its definition is nothing more than `\nobreak \hskip Opt plus Opt`.

```
\gdef\allowhyphens{\penalty\@M \hskip\z@skip}
```

Then a macro is defined to lower the Dutch left double quote to the same level as the comma. It prepares a low double opening quote in box register 0. This macro was copied from `german.tex`.

```
\gdef\set@low@box#1{%
```

```
\setbox\tw@ hbox{,} \setbox\z@ hbox{\#1}
\dimen\z@\ht\z@ \advance\dimen\z@ -\ht\tw@
\setbox\z@ hbox{\lower\dimen\z@ \box\z@}
\ht\z@\ht\tw@ \dp\z@\dp\tw@}
```

The macro `\set@low@box` is used to define low opening quotes. Since it may be used in arguments to other macros it needs to be protected.

```
\gdef\dlqq{\protect\@dlqq}
\gdef\@dlqq{%
  \ifhmode
    \edef\QSF{\spacefactor\the\spacefactor}
  \else
    \let\QSF\empty
  \fi
  \leavevmode\set@low@box{''}
  \box\z@\kern-.04em\allowhyphens\QSF\relax}}
```

For reasons of symmetry we also define '''. This command is defined similar to `\dlqq`, except that the quotes aren't lowered to the baseline.

```
\gdef\@drqq{%
  \ifhmode
    \edef\QSF{\spacefactor\the\spacefactor}
  \else
    \let\QSF\empty
  \fi
  ''\QSF\relax}}
```

The original double quote character is saved in the macro `\dq` to keep it available.

```
\begingroup \catcode`\"12
\gdef\dq{"}
\endgroup
```

The original definition of `"` is stored as `\dieresis`. The reason for this is that if a font with a different encoding scheme is used the definition of `"` might not be the plain T_EX one.

```
\global\let\dieresis"
```

In the Dutch language vowels with a dieresis or umlaut accent are treated specially. If a hyphenation occurs before a vowel-plus-umlaut, the umlaut should disappear. To be able to do this, the hyphenation break behaviour for the five vowels, both lowercase and uppercase, could be defined first in terms of `\discretionary`. But this results in a large `\if`-construct in the definition of the active `"`.

As both Knuth and Lamport have pointed out, a user should not use `"` when he really means something like '''. For this reason no distinction is made between vowels and consonants. Therefore one macro, `\@umlaut`, specifies the hyphenation break behaviour for all letters.

```
\def\@umlaut#1{%
  \allowhyphens
  \discretionary{-}{#1}{\dieresis #1}%
  \allowhyphens}
```

The last support macro to be defined is `\dutch@active@dq`.

```
\gdef\dutch@active@dq#1{%
  \if$string#1'\@dlqq{}%
  \else\if$string#1'\@drqq{}%
  \else\if$string#1-\allowhyphens-\allowhyphens
```

```
\else\if\string#1|\discretionary{-}{}{\kern.03em}%
\else\if\string#1i\allowhyphens\discretionary{-}{i}{\dieresis{i}}%
\allowhyphens
\else\if\string#1j\allowhyphens\discretionary{-}{j}{\dieresis{j}}%
\allowhyphens
\else \@umlaut{#1}\fi\fi\fi\fi\fi\fi
```

The macro reads the next token and performs some appropriate action. If no special action is defined, it will produce an umlaut accent on top of argument 1.

The last definition needed is a replacement for \-. The new version of \- should indicate an extra hyphenation position, while allowing other hyphenation positions to be generated automatically. The standard behaviour of *TeX* in this respect is very unfortunate for languages such as Dutch and German, where long compound words are quite normal and all one needs is a means to indicate an extra hyphenation position on top of the ones that *TeX* can generate from the hyphenation patterns.

```
\def\-{ \allowhyphens\discretionary{-}{}{}\allowhyphens}
```

8.4 Activating the definitions

The last action that should be performed by a language-specific file, is activating its definitions. Before doing that the macro \originalTeX should be defined.

```
\@ifundefined{originalTeX}{\let\originalTeX\relax}{}%
```

Also, the macro \l@*language* should be defined. If it hasn't already been defined, this means that no hyphenation patterns were loaded for this language.

```
\@ifundefined{l@dutch}{\addlanguage{dutch}}{}%
\selectlanguage{dutch}
```

9 Conclusion

In this article a system of document-style option files has been presented that supports the multilingual use of *LATeX*. Some of the code involved has been discussed. The actual files will be made available through the international networks. They will be stored in the files server in the Netherlands (address: LISTSERV@HEARN.BITNET), the file *babel readme* will explain what you need to get to be able to use the system. The system was developed using the doc option, so the files available are fully documented.

References

- [1] Donald E. Knuth, *The TeXbook*, Addison-Wesley, 1986.
- [2] Leslie Lamport, *LATeX, A document preparation System*, Addison-Wesley, 1986.
- [3] K.F. Treebus. *Tekstwijzer, een gids voor het grafisch verwerken van tekst*. SDU Uitgeverij ('s-Gravenhage, 1988). A Dutch book on layout design and typography.
- [4] Hubert Partl, *German TeX*, *TUGboat*, 9 (1988) no. 1, pp. 70–72.
- [5] Leslie Lamport, in: *TeXhax Digest*, Volume 89, #13, 17 February 1989.
- [6] Frank Mittelbach, *The doc-option*, *TUGboat* 10 (1989) no. 2, pp. 245–273.
- [7] Johannes Braams, Victor Eijkhout and Nico Poppelier, *The development of national LATeX styles*, *TUGboat* 10 (1989) no. 3, pp. 401–406.
- [8] Joachim Schrod, *International LATeX is ready to use*, *TUGboat* 11 (1990) no. 1, pp. 87–90.

◇ Johannes Braams
PTT Research Neher Laboratories
P.O. Box 421
2260 AK Leidschendam
JL_Braams@pttrnl.nl

Queries

Public Domain SGML Tools Wanted

I'm interested in determining what *TeX*-to-SGML (and vice-versa) translators are available in the public domain, or any other public domain tools that could serve as a starting point for such translators (i.e., *TeX*-to-troff?). I'm familiar with some of the commercial packages (i.e., ArborText's Publisher) providing document import/export capability, but am specifically seeking public domain materials.

Jeff Lankford
 Northrop Research and
 Technology Center
 Palos Verdes, CA 90274
 jlankford@
 nrtc.nrtc.northrop.com

Editor's note: A public domain parser has been developed under the auspices of the National Institute of Standards and Technology (NIST), but at this time it is incomplete; it also does not understand *TeX*. No other prospects had turned up by press time. However, a group known as the "Text Encoding Initiative" is composed largely of humanities scholars who are interested in such tools, and any information on the subject is likely to be made public on their discussion list, TEI-L@UICVM.Bitnet.

Reporting *TeX*'s Hyphenations

Is there a macro (or a version of *TeX*) which can create a file containing a complete list of hyphenated words in a document? For work with rather long texts it would be desirable and it also should help in improvements of national hyphenation tables.

Jiří Veselý
 Sokolovská 83
 CS-186 00 Prague 8
 Czechoslovakia
 Bitnet: UMMJV@CSEARN

Editor's note: To our knowledge, there isn't any version of *TeX* that will produce such a list automatically, and though it's undoubtedly possible

to create a macro to do this, none is known at present. However, we discovered quite by accident a technique that may be of help: setting `\hsize` to some ridiculously small value will have the effect of reporting the entire text to consist of overfull boxes. Since the hyphenations are shown in such reports, the information sought will be included. Unfortunately, some words will be broken at these hyphenation points, and would have to be reconstructed. Another technique that might be worth tinkering with is to save up each paragraph as a token string and send it through `\showhyphens{...}`; we didn't have time to try it out, though.

A technique for printing out a word with hyphens assigned by *TeX* is given by the macro `\printhyphens` in the file for the *TUGboat* hyphenation list report; this file, *TBOHYF.TeX*, can be found at *labrea.Stanford.edu* in the directory */tex/tugboat* and also at other *TeX* archives.

Letters

Response to Victor Eijkhout

I thank Victor Eijkhout for his knowledgeable and generally favorable review of the book *TeX for the Impatient* that I wrote with Karl Berry and Kathryn Hargreaves (*TUGboat* 11(4), Nov 1990, p. 572). I particularly appreciate his comments about the book being written very clearly and the information in it being readily accessible, since those were primary goals for us when we wrote it. I'm also pleased that he feels that the book would make it easier for new users to learn *TeX*. I do want to respond, however, to a few of his comments.

He mentions being perturbed by small errors in the book and cites two of them:

1. "*The delimiters around \..withdelims commands don't grow as the authors claim.*" In fact, we made no such claim; our description on page 201 simply states that a construct is surrounded by delimiters. It says nothing about how big they are.
2. "*The remarks about the depth (height) of a \vbox (\vtop) on pages 52 and 161/2 are at odds.*" True, but the discrepancy is pretty minor; the difference shows up only in the case

where the first or last item in the vertical list is a whatsit. *TeX* would probably be better off, in fact, if a `\write` at the start of the vertical list of a `\vtop` had no effect on the height of the `\vtop`.

TeX for the Impatient has a great many references to particular pages of *The TeXbook*. Eijkhout concludes from this that *TeX for the Impatient* is not aimed at aspiring hackers. I'd agree that the book is not sufficient by itself for *TeX* hackers—we didn't intend it to be. We expect that hackers will use it in conjunction with *The TeXbook*, and we say as much in our preface.

Used this way, *TeX for the Impatient* largely compensates for *The TeXbook*'s well-recognized shortcomings. *The TeXbook* is indispensable as the definitive source of information on *TeX*, but a big problem with *The TeXbook* is that it's very hard to retrieve information from it. Essential topics appear in odd places, e.g., the description of registers in the middle of the chapter on *How TeX Makes Lines into Pages*. The index has so many references under each entry that it's hard to locate the definitive one, and the one that's marked as definitive often gives just the syntax of a command, not its semantics. Part of our reason for often referring to *The TeXbook* was to make *TeX for the Impatient* useful as a guide to *The TeXbook* as well as a guide to *TeX*.

In my own work I use both books. I refer to *TeX for the Impatient* for quick answers to most questions and to *The TeXbook* for things such as the details of particular algorithms and the definitions of the plain *TeX* macros. Even though I was the principal author of *TeX for the Impatient*, I don't claim to remember everything that's in it!

Paul Abrahams
214 River Road
Deerfield, MA 01342
`abrahams%wayne-mts@um.cc.umich.edu`

Response to Paul Abrahams

Paul Abrahams mentions two instances where my review cited errors in *TeX for the Impatient*; his explanations do not remove the problems.

Regarding whether or not delimiters grow, he is correct that no such claim is made on page 201; instead, it appears on page 58 with the definition of "delimiter": "However, *TeX* performs the adjustment only if the delimiter appears in a 'delimiter context' ... (see pp. 201, 204)." Maybe "Even though I was the principal author of *TeX for the Impatient*, I don't claim to remember everything that's in it!" can be used as attenuating circumstance.

As to differences between the depth (height) of a `\vbox` (`\vtop`), the discrepancy ("the difference shows up only in the case where the first or last item in the vertical list is a whatsit") is minor but important. I might not have known this fact if some rather befuddled user hadn't asked me what on earth was going on when he wrote

```
\def\caption#1{\hbox{\hbox{...}\vtop{#1}}}
```

and subsequently used this as

```
\caption{\label{...} ...}
```

and found the text dropping a line because of the `\label` command. So it does occur in practice.

Victor Eijkhout
`eijkhout@csrd.uiuc.edu`

TeX in Schools: Why Not?

In the March issue of *TUGboat*, Konrad Neuwirth claims that there is no place for *TeX* in schools. He claims, "No, there is no place where *TeX* fits into schools. It is too big, too powerful."

My experience is that nothing is too powerful for schools. In fact, one of the biggest problems with schools in America (and, it seems, worldwide) is that people underestimate the intellectual capabilities of young people.

We have been using *TeX* at Woburn High School in Woburn, Massachusetts, for about five years, and its popularity is growing with both faculty and students. Before describing what we do with *TeX* let me first say that we *don't* write programs in it. I don't ask my students to write a *TeX* program to filter primes from a list. Neither do I ask them to write a Logo program to format a paper. Different tools have different uses.

Here's how *TeX* has been used by the faculty:

- Two of us put together a series of over 100 problem sets in linear algebra. We took years of handwritten and poorly typed worksheets, and, at a rate of one per day, formatted them in \TeX . These sets were bound and passed out to students as workbooks, and the reaction from students has been quite positive. Not having to decipher our handwriting is one less hurdle for them.
- We conducted a \TeX based course for the faculty. The goal of the course was simple: Each week, teachers brought an exam or problem set to format, and we developed the \TeX constructions as they were needed (we used the $\mathcal{AM\&S}\text{-}\text{\TeX}$ macros). The teachers produced some very nice materials.

Students have also been using \TeX :

- I teach a directed study course in mathematics. One of the goals of the course is for students to write up their work and put together a journal that contains their papers. Last year we used \TeX for the first time, and the students were quite proud of the appearance of the journal. This year's students have already begun writing up their work.
- This year we initiated a technical word processing course. Students studied a variety of systems, including a WYSIWYG system for mathematical typing, and they studied \TeX . Next year the course plans to spend even more time on \TeX .

Is \TeX too hard for students? Of course not. Using the $\mathcal{AM\&S}\text{-}\text{\TeX}$ macros, students quickly get used to the rhythm of composing formulas. Here's what one of my students, Jason Gerry, has to say:

During the past school year I have been working with \TeX . I have found \TeX to be a very interesting and exciting program. I feel that learning the basics of \TeX does not take that much time and I see no reason for \TeX not to be used in a school environment. \TeX may not be appropriate for all areas of school or for all students. Just as you do not teach assembly language programming to a beginning programmer, you would not teach \TeX to a beginning math student or a beginning word processor.

In our school, we have taken \TeX in two routes. The first is offered to the advanced mathematics students involved in the independent study course here at Woburn High School. Here, we use \TeX in the generation of reports

and papers that need the high quality appearance of \TeX . The students in this course are very adept at learning new things and \TeX is a program that is understood by these students very quickly. Another route we are taking is through the Business Education Department at Woburn High School. Here students learn \TeX along with Word Perfect and TechWriter. By doing this they get to see the high quality output of \TeX compared to other systems and can appreciate the work they put in a \TeX document better. Last semester in this course we concentrated on imitating pages from engineering textbooks using \TeX . We got used to using some of the easier to moderate \TeX codes and using special \TeX formats, including the $\mathcal{AM\&S}\text{-}\text{\TeX}$ macro package and \LaTeX . This course was set up so that you could go at your own pace which made many students very proficient at \TeX .

\TeX does take more time than an ordinary word processor, and that is a problem with our 43 minute period system here at Woburn High, but the creation of many batch files and easy to follow menus set up by our teachers have taken some of the time out of \TeX ing a document.

I feel that \TeX does have a place in school, if and only if you take it for what it is, a tool for creating high quality documents.

Don't forget, young people take to programming languages much faster than adults. I even have an untested conjecture that the kind of mental visualization required to envision a typeset page from \TeX code helps students with mathematical abstractions.

Many of us who teach and study in schools are growing tired of the attitude that we should confine ourselves to toy environments. We are quite capable of reading real books, of doing real mathematics, of using real computers, and of formatting our work with real typesetting systems.

Al Cuoco
Mathematics Dept.
Woburn High School
Woburn, MA 01801
U.S.A.
alc@media-lab.media.mit.edu

Abstracts

Abstracts of Cahiers GUTenberg # 7 (November 1990)

Editorial committee

Title: *Éditorial : GUTenberg et l'Europe de 92*

Author: Jacques ANDRÉ

A presentation of this issue with a few comments about Europe.

Title: *Logiciels TEX sur serveurs*

Author: Peter FLYNN

A translation (by Christophe DE MONCUIT) of a document published on the networks by Peter FLYNN about the (LA)TEX servers in the world (how to access, how to use, how to retrieve files ...).

Title: *Le gestionnaire BIBLIOTEX*

Author: Dr J.-F. VIBERT

BIBLIOTEX is a package to manage a bibliographic file and to produce specific BIBTEX or LATEX output files. This paper gives functional and usage descriptions.

Title: *Traitement d'index avec LATEX*

Author: Philippe LOUARN

As LATEX handles structured documents, it is easy to create some features, such as table of contents or bibliography. But it is more difficult to create an index. This article attempts to evaluate several methods of index generation utilizable with LATEX. A real example will illustrate our purpose.

Title: *Fontes latines européennes et TEX 3.0*

Author: Michael J. FERGUSON

A translation of the Report on the Extended TEX Font Encoding scheme which has been officially approved by TUG.

Title: *"La typographie" « entre guillemets »*

Author: Fernand BAUDIN

This article is a discussion about the different glyphs used in Europe to bracket quotations, called "guillemets".

Title: *TEX90, Fáilte i gColáiste na hOllscoile, Corcaigh*

Author: Denis MÉGEVAND

A report on the European meeting in Cork.

Title: *Ballades irlandaises*

Author: Bernard GAULLE & Olivier NICOLE

During the Cork euro-TEX conference two important meetings were planned: a western and eastern european TEX users groups leaders summit and the meeting of the TUG board of directors. This paper gives a report of these two events.

Title: *Comment le faire en LATEX*

Author: Responsable de la rubrique :

Eric Picheral

This new regular chronicle will give (more or less dirty) tricks doing specific things in LATEX that are not always known and can help the user. In this issue we have the opportunity to learn more about usage of special characters, inserting a listing in a verbatim environment and also defining a new description environment.

Title: *Les fiches cuisine d'Onc' PostScript*

Fiche N° 4 : TEX et PostScript vont

chez l'imprimeur

Author: Bruno BORGUI

This column will teach us regularly how to use PostScript for different tasks. The author proposes here a technique to produce good PostScript output for the print shop.

◊ Editorial committee
Cahiers GUTenberg
C/o IRISA
Campus de Beaulieu
F-35042 Rennes Cedex, FRANCE
[<gut@irisa.fr>](mailto:gut@irisa.fr)

Abstracts of Cahiers GUTenberg # 8 (March 1991)

Editorial committee

Title: *L'avenir de TEX et METAFONT*

Author: D.-E. KNUTH

French translation of TUGboat Vol. 11 # 4, p489.

Title: *La composition typographique*

Author: Alan MARSHALL

Computers have radically altered the organisation of editorial and typographical production processes. However, the basic activities involved have remained largely untouched by technical change. This article describes the principal typographical task which has

to be carried out, whether it be with traditional or new technologies.

Title: *La typographie et la loi*

Author: André R. BERTRAND

This article is a re-print of one published in *Caractères* # 297 and gives the status of the laws, especially in France but also at the european level, regarding typography: protection of characters, fonts, glyphs, etc.

Title: *Serveurs de fichiers et de fontes pour T_EX*

Author: Raymond TRÉPOS and Philippe LOUARN

This article presents computer networks, software available with networking (mail, news and file transfer) and the way to connect to the french net (Fnet). A special case for T_EX users is shown.

Title: *Métrique des fontes PostScript*

Author: Jacques ANDRÉ and Justin BUR

This note is concerned with PostScript font metrics. Bounding boxes and the Adobe Font Metric file are presented, as well as how to use them for justification, pair and track kerning, etc.

Title: *Analyses bibliographiques*

Author: Olivier NICOLE and Jacques ANDRÉ

The major part of this article is a discussion about a new edition of the "*Lexique des règles typographiques en usage à l'imprimerie nationale*" which is considered an indispensable tool for authors. Following are a few other new books, catalogs, proceedings or journals which are commented on by the editorial board.

Title: *Comment le faire en L_AT_EX*

Author: Éric PICHERAL

A few solutions to common L_AT_EX problems are given in this issue. The first is a solution to suppress page numbering in L_AT_EX and another one to obtain indentation on the first line of the first paragraph of a section or any other subsection. The last item is devoted to the *verse* environment which is often unknown or unused.

Title: *Le coin des gourous*

Author: Georges WEIL

This regular column proposing macros to do basic things in plain T_EX offers here a few macros for printing complex matrices.

Title: *Sommaire général*

The editor presents here a table of contents of all articles published since number zero of April 1988. An index of authors is also provided.

Title: *Les distributions GUTenberg*

The association GUTenberg explains how to obtain its public domain distributions for DOS or Unix systems.

Title: *Journée SGML*

Here are given details about the one-day course organized by GUTenberg about SGML in May (teacher: Éric van Herwijnen).

◊ Editorial committee
 Cahiers GUTenberg
 C/o IRISA
 Campus de Beaulieu
 F-35042 Rennes Cedex, FRANCE
 <gut@irisa.fr>

Calendar

1991

**Sam Houston State University,
Huntsville, Texas**

- Jun 10–14 Advanced \TeX /Macro Writing
Jun 10–14 Intensive \LaTeX

- Jul 8–12 Advanced \TeX /Macro Writing,
Northeastern University, Dedham
Campus, Dedham, Massachusetts

- Jul 11–13 AMS- \LaTeX , American Mathematical
Society, Providence, Rhode Island

**TUG91 Conference
Dedham, Massachusetts (suburban Boston)**

- Jul 8–11 METAFONT
Jul 11–12 SGML
Jul 13 \TeX for Publishers
Jul 14 Publishing Books with DTP Tools
Jul 15–18 **TUG's 12th Annual Meeting**
Jul 19 Introduction to Typography
Jul 19–20 Output Routines
Jul 22–25 Intensive \LaTeX

**Northeastern University, Dedham Campus,
Dedham, Massachusetts**

- Jul 22–23 Macro Writing
Jul 24–26 PostScript

- Jul 25 **TUGboat Volume 12,
Proceedings issue:**
Deadline for receipt of news items,
reports.

- Jul 29– Aug 2 Intensive Beginning/Intermed. \TeX ,
Providence College,
Providence, Rhode Island

**Florida State University,
Tallahassee, Florida**

- Aug 5–9 \LaTeX Style Files
Aug 12–16 Intensive Beginning/Intermed. \TeX

- Aug 12–16 Intensive Beginning/Intermed. \TeX ,
California State University,
Northridge, California

- Aug 10–12 Intensive Beginning/Intermed. \TeX ,
University of Houston, Clear Lake,
Houston, Texas

- Aug 13 **TUGboat Volume 12,
2nd regular issue:**
Deadline for receipt of technical
manuscripts.

University of Illinois, Chicago, Illinois

- Aug 12–16 Intensive \LaTeX
Aug 19–23 Intensive Beginning/Intermed. \TeX

- Aug 26–30 Advanced \TeX /Macro Writing,
University of Maryland,
College Park, Maryland

- Sep 10 **TUGboat Volume 12,
2nd regular issue:**
Deadline for receipt of news items,
reports.

- Sep 23–25 **6th European \TeX Conference**
Paris, France. For information,
contact GUTenberg, 6th European
 \TeX Conference, B.P. 21, 78354
JOUY EN JOSAS cedex, France.
Phone: +33 1 34 65 22 32;
Fax: +33 1 34 65 20 51;
E-mail: gut@irisa.irisa.fr.
(See also page 309.)

- Sep 26 GUTenberg'91 Congress, Paris,
France. "Technical and scientific
edition". For information, same as
6th European \TeX Conference.
(See also page 310.)

- Sep 30 – Oct 3 Computer Publishing Conference.
San Jose Convention Center and
Fairmont Hotel, San Jose, California.
For information, contact Seybold
Publications, (213-457-5850).

	1992
Oct 1-3	Desktop Publishing in Astronomy and Space Sciences. Strasbourg Astronomical Observatory, Strasbourg, France. For information, contact Dr. André Heck (Bitnet: Heck@FRCCSC21, or +33-88.35.82.22). (See also page 311.)
Oct 15-16	RIDT 91, The second international workshop on raster imaging and digital typography, Boston, Massachusetts. For information, contact Robert A. Morris (ridt91-request@cs.umb.edu, or 617-287-6466). (See also <i>TUGboat</i> 11, no. 4, page 668.)
Nov 19	TUGboat Volume 13, 1st regular issue: Deadline for receipt of technical manuscripts (tentative).
Nov 21	NTG Fall Meeting, "Fun with TEX", Technische Universiteit te Eindhoven, The Netherlands. For information, contact Piet Tutelaers (rcpt@URC.TUE.NL).
Dec 17	TUGboat Volume 13, 1st regular issue: Deadline for receipt of news items, reports (tentative).
	Feb 18 TUGboat Volume 13, 2nd regular issue: Deadline for receipt of technical manuscripts (tentative).
	Mar 17 TUGboat Volume 13, 2nd regular issue: Deadline for receipt of news items, reports (tentative).
	Apr 7-10 EP'92 Swiss Federal Institute of Technology, Lausanne, Switzerland. For information, contact ep92@eldi.epfl.ch (see also page 311).
	May NTG Spring Meeting, "Science with TEX", CWI, Amsterdam, The Netherlands. For information, contact Gerard van Nes (vannes@ECN.NL).
	Aug 18 TUGboat Volume 13, 4th regular issue: Deadline for receipt of technical manuscripts (tentative).
	Sep 15 TUGboat Volume 13, 4th regular issue: Deadline for receipt of news items, reports (tentative).

For additional information on the events listed above, contact the TUG office (401-751-7760) unless otherwise noted.

News & Announcements

6th European TeX Conference & GUTenberg'91

Official announcements

The 6th European TeX Conference, organized by GUTenberg, will be held in Paris from September 23rd to 25th. This conference will be followed on the 26th by the GUTenberg'91 conference. During the preceding week four 4-day courses are organised.

6th European TeX Conference

Paris

23rd–25th September 1991

This major conference has support from:

Imprimerie Louis-Jean
TeX Users Group
Institut Blaise Pascal
École Normale Supérieure (ULM)

Programme Committee

Bernard GAULLE, Chairman (GUTenberg, France)
Chris ROWLEY (Open Univ., United Kingdom)
Kees van der LAAN (NTG, the Netherlands)
Joachim LAMMARSCH (Heidelberg Univ., Germany)
Roswitha GRAHAM (KTH, Sweden)
Michael FERGUSON (INRS-Télécom., Canada)
Nicolas BROUARD (INED, France)
Pierre DAGNÉLIE (Sc. Agro., Belgium)
Maurice LAUGIER (Imprimerie Louis-Jean, France)
Didier COLLIN (Toulouse, France)
Denis MÉGEVAN

(Observatoire Genève, Switzerland)

Organisation Committee GUTenberg

Olivier NICOLE, Chairman (INRA)
Jacques ANDRÉ (INRIA-IRISA)
Philippe LOUARN (INRIA-IRISA)
Malcolm CLARK, tutorials Organiser (PCL)
André DESNOYERS (Institut Blaise Pascal)
Jacques BEIGBEDER (École Normale Supérieure)
Michel BLANCHARD (Université d'Orléans)
Alain POSTY (INRA)

Preliminary programme

- . Russian TeX (A. SAMARIN)
- . TeX and Africa (J. KNAPPEN)
- . TeX: the limit of multilingualism (M. FANTON)

- . Towards an arabicized version of TeX (M. FANTON & al.)
- . TeXniques in Siberia (Th. JURRIENS)
- . Babel, a multilingual style options system (J. BRAAMS)
- . An internat. version of MakeIndex (J. SCHROD)
- . AsTeX: An integrated and customisable multi-window environment (M. LAVAUD)
- . Managing the interface to TeX (S. LARSEN)
- . Shell for TeX (B. MALYSHEV)
- . ColourTeX (Ch. CERIN)
- . GWEZ builds tree structures and print them (B. LEGUY)
- . Form letters in LATEX with mailing label (J. DAMRAU)
- . Conversion of MicroSoft WORD into LATEX (P. BACSICH & al.)
- . ScholarTeX: last enhancements (Y. HARALAMBOUS)
- . TeX dismembered (A. WITTBECKER)
- . Maths into BLUES (K. VAN DER LAAN)
- . Organising a large collection of style files for different TeX macro packages (A. BINDING)
- . Typesetting SGML documents using TeX (A. DOBROWOLSKI)

Other talks are under consideration at the time we publish this announcement.

Official languages will be English & French and the conferences will be simultaneously translated.

Panels such as:

- . Teaching (IA)TeX to a diverse audience
 - . (IA)TeX in Europe
- will be chaired by well-known panelists.

A Question Time will also take place during the congress.

BoFs will be set up on topics to be decided upon during the conferences.

Vendors will exhibit their latest products. A commercial session will be organised so they can talk freely about their products.

A Special Event Dinner

A dinner will be held on Tuesday evening at the restaurant of the Villette center of Sciences and Industry. The dinner will follow the showing of a film using hightech sound and light systems on the géode's 1,000 m² hemispheric screen. Special effects take the viewer on an imaginary journey to another place and time. A coach tour of Paris by night will conclude the evening.

All those taking part in the congress will be admitted free. Guests are welcome to join in the evening's entertainment for a FF 270 charge.

Conference fees

The registration fees for the 6th European conference are FF 1700. But, please, **notice** that a reduction of FF 200 applies for all TUG or European **T_EX** users groups members. A surcharge of 10% must be added if registration is made after June 30th.

Tutorials/Courses

*Paris
17th–20th September 1991*

Each course is 4 days long. Instructors will teach in english.

Tutorial I – Advanced T_EX and T_EX Macros

(Philip TAYLOR, RHBNC)

This course is intended to cover the more advanced features of **T_EX** macro programming, including such esoteric aspects as `\afterassignment`, `\futurelet`, `\expandafter`, `\uccode`, etc. Some time will be devoted to the area of cross-referencing, both via external files and via control sequences, and the manipulation of `\catcodes` will be covered in some detail.

Tutorial II – L_AT_EX Style Files

(Chris ROWLEY, Open University)

Ever wondered how to make L_AT_EX do what you wanted? This course will help you understand how to modify L_AT_EX style files to give quality typesetting, and may even give you the confidence to write your own. Some experience with L_AT_EX and **T_EX** would be an advantage.

Tutorial III – METAFONT

(Doug HENDERSON, Blue Sky Research and
Yannis HARALAMBOUS, Université de Lille)

The course will cover the basics of METAFONT, but should provide participants with enough information to generate their favourite CM fonts at any design size, tinker with the multitude of parameters, and even start to design their own characters.

Tutorial IV – Beginning and Intermediate L_AT_EX

(Malcolm CLARK, PCL)

Become proficient in L_AT_EX in just four days! The course covers the fascinating world of L_AT_EX, concentrating on the joys of structured documentation, whilst trying to maintain a facade of quality typesetting. Suitable for technical and non-technical alike.

Tutorial fees

Registration fees for one course are FF 2500. Please **notice** that a discount of FF 500 applies to the reg-

istration for the 6th European **T_EX** Conference for the participants in a tutorial.

GUTenberg'91

*Paris
September 26th, 1991*
Technical and Scientific Edition

The day will be devoted more specifically to issues connected with the French language; selected topics will cover French publishing problems. The following topics are among the planned speeches:

- *T_EX V3 ou M_IT_EX?*
- *Les disquettes GUTenberg'91* (N. BROUARD)
- *L'impression des formules chimiques* (M. LAUGIER)
- *Un programme pour modifier les .dvi* (M. LAUGIER & al.)

Other presentations are under consideration but have not been yet finalised.

Conferences will be simultaneously translated.

A “Question Time” about the networks is planned on this day.

GUTenberg'91 fees

The registration fees are FF 700 (minus FF 200 in case of joint registration with the 6th European **T_EX** conference). Other discounts apply. Call us for registration and accommodation forms.

Contacts:

GUTenberg
6th European **T_EX** Conference
BP 21
F-78354 Jouy en Josas Cedex
France

Telephone: +(33 1) 34 65 22 32

Fax: +(33 1) 34 65 22 28

E-mail: on@jouy.inra.fr

Call and ask us for your Registration and Accommodation forms

Desktop Publishing in Astronomy and Space Sciences

Strasbourg Astronomical Observatory
1-3 October 1991

Strasbourg Observatory/Astronomical Data Centre will be organizing from Oct. 1 to 3, 1991, a meeting on 'Desktop Publishing in Astronomy and Space Sciences'.

Desktop publishing is widespread nowadays and a number of packages are used by astronomers, space scientists, engineers and technicians for producing their papers, reports, etc., as well as their everyday mail (typically Word, TeX, L^AT_EX, ...). The motivations behind the choice of a given package are varied and not always rational ones (such as availability, financial constraints, mouth-to-ear recommendations).

There is in any event an experience to be shared openly for the benefit of everybody and it would certainly be useful to confront performance, capabilities, as well as possible complementarities of the text processing software packages that are presently most frequently used in astronomy and space sciences. There is most likely no 'best' system, but it might be possible to get a digest of the best parts of the major ones without having to review computer journals at length.

Not only the point of view of the authors, writers or scientific editors should matter here, but also the reasons behind the choices that a few publishers have already made (typically Springer TeX Macros, ...). On this side, the advantages are obvious: the manuscripts are delivered directly by the authors/scientific editors in a standardized machine-readable way (saving time and money) and the final appearance of the publications is substantially enhanced, be it only through its harmonization.

What happens in other communities of related fields will also be investigated. Some publishers represent up to 500 scientific journals. It will be interesting to listen to their explanations as to how their choices have been made and for them to hear what scientists have to say in that respect.

Another aspect of this colloquium is related to the developments carried out by auxiliary software companies or individuals. They are producing self-sustained packages, complementary tools and/or utilities to be plugged into already well-established text processing systems. Here again scientists should express their views, needs and wishes.

The meeting is timely as desktop publishing reaches such a level of development that it would be appropriate now for our scientific communities and for publishers to issue recommendations for standardization, compatibility and/or complementarity from the software producers.

Sessions will be organized in such a way that each of the parties will be able to present their viewpoints on the advantages of specific packages, the constraints they have to comply with, the requirements they have for further developments. We shall also attempt to set up exhibitions and/or demonstrations.

The meeting will also aim at issuing recommendations for publishing standards, as well as for compatibility and/or complementarity from the software producers. A special session involving the editors of the major scientific journals is being set up by James Lequeux (aanda@frmeu51.bitnet).

Proceedings will be published and distributed free of charge to the persons who actually attend the meeting.

If you are interested in attending this colloquium, please request a registration form in order to receive additional information (list of hotels, preliminary programme, and so on). As the audience might have to be limited, we advise you to do it as soon as possible; the deadline for registering will be

15 July 1991.

Dr. André HECK
Observatoire Astronomique
11, rue de l'Université
F-67000 Strasbourg
France

telephone: +33-88.35.82.22
telex: 890506 starobs f
telefax: +33-88.25.01.60
EARN/BITNET: Heck@FRCCSC21

Call for papers: EP92
**International conference on electronic
publishing, document manipulation, and
typography**

Lausanne, Switzerland, 7-10 April 1992

An International Conference on Electronic Publishing, Document Manipulation and Typography

will be held in Lausanne, Switzerland, at the Swiss Federal Institute of Technology, on April 7–10, 1992.

This conference will be the fourth in a series of international conferences organised to promote the exchange of novel ideas in the area of computer manipulation of documents.

The first two conferences in the series, EP86 held in Nottingham, England, and EP88 in Nice, France, concentrated mainly on the specific aspect of the production of documents by computer, from composition to printing. EP90, which was held in Washington, adopted a broader definition of the term Computer Assisted Publication, and accordingly included more materials on new topics such as the application of data-base techniques to document handling, hypertext and hypermedia systems, and document recognition and analysis.

EP92 will follow this trend. Its objective will be to present the state of the art by reporting original and recent contributions to this area. The conference proceedings will be published by Cambridge University Press and will be available at the conference.

A day of tutorials is planned for April 7. Suggestions for contributions with introductory and survey topics are requested.

Topics

- Modelling and representation of documents
 - Document structures
 - Integration of text, images and graphics
 - Integration of document manipulation systems with other software tools
 - Standards: evaluation and implementation
 - Object oriented approaches
- Documents management
 - Document preparation systems
 - Hypertext: production, editing and visualisation
 - Large text data-bases
 - Distributed documents: parallel algorithms, multi-user documents
 - User-machine interfaces
- Document recognition and interpretation
 - Structural recognition of documents
 - Filtering and image handling techniques
 - Multi-lingual documents
 - Semantic text structures
 - Indexing techniques

- Typography and graphics
 - Character design
 - Use of gray levels and colour technology
 - Pagination and layout
- Document manipulation and education
 - Computer assisted document production
 - Experiences in teaching EP
 - The place of EP in Computer Science curricula

Main deadlines

Today. Send for information. Write to the Conference Secretariat either by post or e-mail, at

EPFL – EP92
 Département d'Informatique
 IN (Ecublens)
 CH – 1015 Lausanne
 Switzerland
 Phone: (41) 21 693 25 75
 Fax: (41) 21 693 52 63
 E-mail: ep92@eldi.epfl.ch

August 15, 1991. The full paper should be received by the Conference Secretariat.

October 31, 1991. Notification of acceptance or rejection.

December, 1991. Distribution of the program.

December 15, 1991. Final version of the accepted paper should be received by the Conference Secretariat.

April 7–10, 1992. EP92 Conference.

Organization

The Conference Chair is Christine Vanoirbeek of the Swiss Federal Institute of Technology, Lausanne. The Program Committee Chair is Giovanni Coray, also of the Swiss Federal Institute of Technology.

The following are some members of the Program committee whose names should be familiar to the TUG community: Jacques André (INRIA/IRISA, Rennes, France), Charles Bigelow (Stanford University), Richard Furuta (University of Maryland), Roger D. Hersch (Swiss Federal Institute of Technology, Lausanne), Brian Kernighan (AT&T Bell Laboratories), Dario Lucarella (University of Milan), Pierre MacKay (University of Washington), and Robert A. Morris (University of Massachusetts, Boston).

Late-Breaking News

Fixed-Point Glue Setting: Errata

Donald E. Knuth

I thank Eberhard Mattes for calling my attention to an error in the demonstration WEB program I published in *TUGboat* 3,1 (March 1982), 10-27. After looking more closely at that program, I noticed that it actually contains at least *two* errors. I should have known better than to rush into print with the second draft of a program on which I had spent only a few hours of time; but I was just beginning to learn how to program in WEB, and the new methodology had lulled me into thinking that I understood what I was doing.

The most serious error occurs in line 9 of the program in section 14. That line should be:

```
if  $a + b + k - h = 15$  then
   $c \leftarrow (q + 1) \text{ div } 2$  {  $l = 16 - k$  }
```

This error caused the answers for test data set 4, on page 23 of the article, to be only about half as big as they should have been; the correct value of c for that data set is 30670, not 15335.

The other error arises when the number b in the algorithms turns out to be greater than 30. I believe the best way to correct it is to replace the four lines beginning with ‘**if** $b < 0$ ’ in section 12 by the following:

```
if  $(b < 0) \vee (b > 30)$  then
  if  $b < 0$  then write_ln(`!_Excessive_glue.`);
    { error message }
   $b \leftarrow 0$ ;  $c \leftarrow 0$ ; { make  $f(x)$  identically zero }
end
```

After writing that article I learned that standard Pascal also wants the specification ‘: 0’ to become ‘: 1’ in *write* and *write_ln* statements (sections 21, 24, and 25; five places altogether).

I also learned that the language is properly called Pascal, not PASCAL; and I began calling scaled points ‘sp’ instead of ‘spt’.

Although the comments in section 1 state that my algorithm uses only 16-bit multipliers and divisors, the truth is that it also might use larger multipliers and divisors that happen to be powers of 2. Such multiplications and divisions can be replaced by binary shifts, in a language like C.

I overlooked a few typographic errors: ‘process or’ should be ‘processor’ on the first line of page 11, and ‘as’ should be ‘that’ on line 8 from the end

of that page; ‘hve’ should be ‘have’ in section 4, line 20, and ‘[$2^{-a}x]$ ’ should be ‘[$2^{-a}x_i]$ ’, a few lines further down. Also ‘ x_1 ’ should be ‘ x_i ’ in line 5 of section 21.

Finally, I should have used the same notation in the program of sections 12-14 as I used in the theoretical discussion of section 4.

A corrected version of the program, incorporating the remarks above and a few other things, is now available in standard TEX electronic archives under the file name *glue.web*.

Looking on the bright side, I’m pleased to report that TEX now processes the entire woven program in only 10 seconds on my home computer; according to the article, the same task took 40 seconds in 1981, using the KL10 mainframe on which I did all the development of TEX.

And oh yes, one further correction is necessary: The amount of time spent proofreading and debugging, mentioned on page 11 of my article, should now be increased from ‘about two hours’ to ‘about six hours’.

Production Notes

Barbara Beeton

Input and input processing

Electronic input for articles in this issue was received by mail and on floppy disk. Most articles 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). Several authors requested copies of the macros (which we were happy to provide); however, the macros have also been installed at labrea.stanford.edu and the other archives, and an author retrieving them from an archive will most likely get faster service. Of course, the TUG office will provide copies of the macros on diskette to authors who have no electronic access.

The number of articles in this issue was split about evenly between “plain” and LATEX; pages too were about evenly divided. In organizing the issue, attention was given to grouping bunches of plain or LATEX articles, to yield the smallest number of separate typesetter runs, and the least amount of handwork pasting together partial pages. This also affected the articles written or tagged

by the staff, as the conventions of *tugboat.sty* or *ltugboat.sty* would be chosen depending on what conventions were used in the preceding and following articles; no article was changed from one to the other, however, regardless of convenience.

This issue was mercifully free of insidious redefinitions of macros that already exist in the styles used; either authors were more careful, or the mechanism that isolates one article from another when several are being combined into a single run is finally working at that level. We had some problems of interaction within the *TUGboat* macros, however, that more than made up for the lack of author-provided problems.

As has been customary for the past few issues, several articles required font work; these included the articles by Alan Jeffrey (p. 227) and Yannis Haralambous (p. 224). The latter rooted out some bugs in the laser printer device driver used at AMS (it does not deal well with 256-character fonts) and forced a delay when no one was available during a weekend to process proof from the typesetter.

One article, by Malyshев, Samarin and Vulis (p. 212), required the new *LATEX* font selection scheme; another, by Hefferon (p. 270), required the multicols option. Other *LATEX* articles were processed with whatever version was convenient.

The following articles were prepared using *LATEX*.

- Nelson Beebe, *President's introduction*,
page 205.

- Barbara Beeton, *Editorial comments*,
page 208.
- Walter Obermiller, *TEX in Germany*,
page 211.
- Michel Goossens, *LATEX meeting in London*,
page 212.
- Reinhard Fößmeier, *X bitmaps in TEX*,
page 229.
- Nico Poppelier, a book review, page 235.
- Victor Eijkhout, three articles, pages 253, 260,
and 272.
- Jim Hefferon, *Getting \answers*, page 270.
- all items in the *LATEX* section, pages 284 ff.
- abstracts of the *Cahiers GUTenberg*,
page 305.
- announcement of *TEX91* and *GUTenberg'91* in
Paris, page 309.

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 Ultrix on a DECsystem 5000. Output was typeset on an APS-μ5 at the AMS using resident CM fonts and additional downloadable fonts for special purposes.

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

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

Coming Next Issue

Inside Type & Set

Graham Asher describes the Type & Set system, which consists of *T_EX*, several *T_EX* macro packages, a suite of C programs including a style sheet editor, an automatic page make-up system that replaces *T_EX*'s output mechanism, and a family of device drivers. This system was developed to overcome problems which make *T_EX* difficult to use for commercial journal and book publishers.

Invisibility using virtual fonts

Sebastian Rahtz proposes an alternate method for generating "invisible" fonts as used by *SLiT_EX*. This method makes it possible to use the standard Post-Script fonts in place of Computer Modern. [Delayed by technical problems.]

Arrows for technical diagrams

David Salomon, requiring arrows of more varieties than are available in unextended (I_A)*T_EX*, has created a font of arrowheads. Since *T_EX* does not have diagonal rules, only horizontal and vertical arrowheads were developed. However, the methods used can easily be extended for diagonal arrowheads.

Some Basic Control Macros for *T_EX*

Jonathan Fine uses techniques that require only *T_EX*'s mouth to define and describe macros *\break*, *\continue*, *\switch*, *\return*, *\exit*, *\chain*, and labels *\end* and *::* that make it easier to write *T_EX* macros.

TUG Business

Financial Reports of the *T_EX* Users Group

Ron Whitney, Acting Business Manager

Following are the TUG balance sheets along with revenue and expense statements for the years 1989 and 1990. In years prior to 1990, our accountant has performed financial *reviews*, as opposed to full *audits*. We felt that it was time the organization underwent an audit of its books and so went through this closer inspection at the end of year 1990. Also included below is the 1991 budget approved by the TUG Board in March of this year.

These are difficult times for TUG, as they are for many similar organizations. Although we had been operating at close to a breakeven level for several years through 1989, there was a considerable operating deficit ($\approx \$93,000$) for 1990. Roughly half of this total was due to a general decrease in sales and attendance at meetings and courses, the other half being involved with changes in administration within the TUG office (see the prototype issue of *T_EX and TUG News* for a full description of those changes). Our assets entering 1991 were equivalent to about 40% of the operating budget.

The 1991 year-to-date figures will appear in the next regular issue of *TUGboat* (12#4). The 1991 budget forecasts a small surplus, but at this point we believe we will be looking at deficit figures by the end of the year. The cause again is due to a decrease in traffic for sales and courses, and perhaps also, in membership.

If you wish to see more detailed information or have any further questions, please write or call the TUG office at

P. O. Box 9506
Providence, RI 02940
Phone: (401) 751-7760

TeX Users Group

Balance Sheet and Revenue & Expense Statements

December 31, 1989 (unaudited) and 1990 (audited)

Prepared by Michael D. Aaronson and Associates C.P.A.

1991 budget, approved by the TUG Board, March 1991

ASSETS	1989	1990
<i>Current assets</i>		
Cash	\$ 143 575	\$ 85 610
Certificates of deposit	100 000	100 600
Accounts receivable	43 799	13 644
Inventory	44 043	61 448
Prepaid expenses	<u>7 022</u>	<u>6 146</u>
Total current assets	<u>338 439</u>	<u>267 448</u>
<i>Property and equipment</i>		
Vehicle	5 150	
Office furniture and equipment	13 578	13 578
Computer software and equipment	<u>28 316</u>	<u>31 156</u>
	47 044	44 734
Less accumulated depreciation	<u>(14 674)</u>	<u>(22 059)</u>
Net property and equipment	<u>32 370</u>	<u>22 675</u>
<i>Other assets</i>		
Rent and utility deposits	<u>1 200</u>	<u>1 100</u>
Total other assets	<u>1 200</u>	<u>1 100</u>
Total assets	<u>\$ 372 009</u>	<u>\$ 268 548</u>

LIABILITIES AND FUND BALANCES

Current liabilities	1989	1990
Accounts payable	\$ 26 493	\$ 29 499
Accrued payroll and payroll taxes payable	3 284	10 172
Accrued vacation pay	7 955	3 614
Accrued pension expense	1 514	4 293
Other payables	7 486	9 431
Deferred income	<u>81 736</u>	<u>83 534</u>
Total current liabilities	<u>128 468</u>	<u>140 543</u>
Total liabilities	<u>128 468</u>	<u>140 543</u>
<i>Fund balances</i>		
Property and equipment	32 370	22 675
Unrestricted funds	<u>211 171</u>	<u>128 005</u>
Total fund balances	<u>243 541</u>	<u>150 680</u>
Total liabilities and fund balances	<u>\$ 372 009</u>	<u>\$ 291 223</u>

REVENUES	1989	1990	1991
<i>Membership income</i>			
Individual	\$ 114 495	\$ 120 186	\$ 167 303
Institutional—educational	33 422	30 636	37 637
Institutional—noneducational	19 084	20 146	24 146
	<u>167 001</u>	<u>170 968</u>	<u>229 086</u>
<i>Annual meeting and course income</i>			
Meeting	95 319	42 836	68 450
Regional courses	70 002	93 820	74 485
In-house courses	56 828	10 000	10 000
Meeting and courses, Europe		59 782	
	<u>222 148</u>	<u>206 438</u>	<u>152 935</u>
<i>Sales income</i>			
Resale of books	120 358	82 461	86 609
Resale of software	32 237	46 674	49 008
In-house publications and software	39 987	31 104	32 659
Video tape rental	800	0	0
Back issues	19 146	16 935	17 782
Shipping and handling fees	7 725	9 508	9 982
	<u>220 253</u>	<u>186 681</u>	<u>196 040</u>
<i>Other income</i>			
Advertising	36 935	32 980	34 629
Mailing lists	1 667	1 322	1 388
Contributions	8 222	9 157	9 157
Promotional items	720	2 348	2 466
Interest	21 578	13 175	8 775
Other	0	647	
	<u>69 122</u>	<u>59 629</u>	<u>56 415</u>
Total revenue	\$ 678 525	\$ 623 718	634 476

EXPENSES	1989	1990	1991
<i>TUGboat expenses</i>			
Editorial	\$ 25 155	\$ 24 274	\$ 27 486
Computing	9 531	1 569	2 000
Printing	49 587	38 598	42 458
Mailing	30 575	26 192	35 789
Other	0	1 093	0
	<u>114 848</u>	<u>91 725</u>	<u>107 733</u>
<i>Newsletter</i>			
printing and mailing			5 000
<i>Annual meeting and course expenses</i>			
Meeting	46 269	29 908	40 850
Regional courses	44 147	37 041	35 013
In-house courses	28 043	6 251	6 000
Meeting and course expense, Europe		76 493	5 000
Knuth scholarship			2 000
	<u>118 459</u>	<u>147 693</u>	<u>88 863</u>

EXPENSES (cont'd)	1989	1990	1991
<i>Sales expenses</i>			
Cost of goods sold (resale)	94 555	64 156	67 436
Cost of goods sold (software)	11 540	29 254	30 295
Cost of goods sold (in-house)	17 224	12 028	11 343
Promotional merchandise		298	313
Bankcard processing fees	5 756	3 900	4 095
Bad debt expense	1 263	1 797	
Shipping	7 726	9 553	10 031
	<u>138 064</u>	<u>117 086</u>	<u>123 513</u>
<i>Personnel expenses</i>			
Salaries and wages	146 003	154 877	129 050
Payroll taxes	12 984	12 596	12 205
Health insurance	8 829	12 975	11 510
Pensions	15 658	12 238	10 969
Other employee benefits	1 110	660	700
Consultants and temporary services	0	10 240	11 300
	<u>184 585</u>	<u>203 586</u>	<u>175 734</u>
<i>Operational expense</i>			
Rent	10 809	10 900	11 800
Telephone	4 685	6 885	7 885
Utilities	3 125	3 019	3 170
Office supplies	8 196	5 737	4 920
Postage and mailing	17 954	13 014	21 514
Printing/photocopying	12 803	1 629	4 629
Business insurance	4 020	3 616	4 576
Legal and accounting fees	2 627	3 888	6 888
Off-site computer usage	1 445	2 500	2 750
Cleaning and maintenance		1 500	1 500
Equipment repairs and maintenance		3 997	1 997
Equipment depreciation	9 017	9 445	10 103
Car expense		2 955	
Other	3 862	1 775	
	<u>78 543</u>	<u>68 360</u>	<u>81 732</u>
<i>Other expenses</i>			
TUG Committees	10 266	12 114	15 803
TeXhax	11 995	11 460	11 460
Radel collection			1 600
BIBTEX project	10 000	0	0
Staff training	3 507	1 211	2 351
Exhibits/meetings	11 791	6 765	3 265
Graphic design, promotion	0	3 938	13 000
	<u>47 559</u>	<u>41 866</u>	<u>47 479</u>
<i>Sub-total expenses</i>	<u>682 057</u>	<u>670 339</u>	<u>630 054</u>
<i>Extraordinary expense</i>			
Expense associated with departure of ED		44 295	
Transfer of interest earned on funds held for employee		<u>1 945</u>	
		<u>46 240</u>	
Total expenses	\$ 682 057	\$ 716 579	\$ 630 054
Net income	\$ (3 532)	\$ (92 861)	\$ 4 422

Institutional Members

The Aerospace Corporation,
El Segundo, California

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

American Mathematical Society,
Providence, Rhode Island

ArborText, Inc.,
Ann Arbor, Michigan

ASCII Corporation,
Tokyo, Japan

Belgrade University,
Faculty of Mathematics,
Belgrade, Yugoslavia

Brookhaven National Laboratory,
Upton, New York

CERN, *Geneva, Switzerland*

Brown University,
Providence, Rhode Island

California Institute of Technology,
Pasadena, California

Calvin College,
Grand Rapids, Michigan

Carleton University,
Ottawa, Ontario, Canada

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

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

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

Construcciones Aeronauticas, S.A.,
CAE-Division de Proyectos,
Madrid, Spain

DECUS, Electronic Publishing
Special Interest Group,
Marlboro, Massachusetts

Department of National Defence,
Ottawa, Ontario, Canada

E. S. Ingenieros Industriales,
Sevilla, Spain

Edinboro University
of Pennsylvania,
Edinboro, Pennsylvania

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

European Southern Observatory,
*Garching bei München,
Federal Republic of Germany*

Fermi National Accelerator
Laboratory, *Batavia, Illinois*

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

Fordham University,
Bronx, New York

General Motors
Research Laboratories,
Warren, Michigan

Grinnell College,
Computer Services,
Grinnell, Iowa

GTE Laboratories,
Waltham, Massachusetts

Hatfield Polytechnic,
Computer Centre,
Herts, England

Hughes Aircraft Company,
Space Communications Division,
Los Angeles, California

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

IBM Corporation,
Scientific Center,
Palo Alto, California

Institute for Advanced Study,
Princeton, New Jersey

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

Intevep S. A., *Caracas, Venezuela*

Iowa State University,
Ames, Iowa

The Library of Congress,
Washington D.C.

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

Louisiana State University,
Baton Rouge, Louisiana

MacroSoft, *Warsaw, Poland*

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

Mathematical Reviews,
American Mathematical Society,
Ann Arbor, Michigan

Max Planck Institut
für Mathematik,
Bonn, Federal Republic of Germany

McGill University,
Montréal, Québec, Canada

Michigan State University,
Mathematics Department,
East Lansing, Michigan

NASA Goddard
Space Flight Center,
Greenbelt, Maryland

National Institutes of Health,
Bethesda, Maryland

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

Naval Postgraduate School,
Monterey, California

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

Northrop Corporation,
Palos Verdes, California

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

Pennsylvania State University,
Computation Center,
University Park, Pennsylvania

Personal TeX, Incorporated,
Mill Valley, California

Princeton University,
Princeton, New Jersey

Purdue University,
West Lafayette, Indiana

- Queens College,
Flushing, New York
- Rice University,
Department of Computer Science,
Houston, Texas
- Roanoke College,
Salem, VA
- Rogaland University,
Stavanger, Norway
- Rutgers University, Hill Center,
Piscataway, New Jersey
- St. Albans School,
*Mount St. Alban, Washington,
D.C.*
- Sandia National Laboratories,
Albuquerque, New Mexico
- Smithsonian Astrophysical
Observatory, Computation Facility,
Cambridge, Massachusetts
- Software Research Associates,
Tokyo, Japan
- Space Telescope Science Institute,
Baltimore, Maryland
- Springer-Verlag,
*Heidelberg, Federal Republic of
Germany*
- Springer-Verlag New York, Inc.,
New York, New York
- Stanford Linear
Accelerator Center (SLAC),
Stanford, California
- Stanford University,
Computer Science Department,
Stanford, California
- Talaris Systems, Inc.,
San Diego, California
- Texas A & M University,
Department of Computer Science,
College Station, Texas
- UNI-C, *Aarhus, Denmark*
- University of Alabama,
Tuscaloosa, Alabama
- University of British Columbia,
Computing Centre,
*Vancouver, British Columbia,
Canada*
- University of British Columbia,
Mathematics Department,
*Vancouver, British Columbia,
Canada*
- University of Calgary,
Calgary, Alberta, Canada
- University of California, Berkeley,
Space Astrophysics Group,
Berkeley, California
- University of California, Irvine,
Information & Computer Science,
Irvine, California
- University of California, Los Angeles, Computer
Science Department Archives,
Los Angeles, California
- University of Canterbury,
Christchurch, New Zealand
- Universidade de Coimbra,
Coimbra, Portugal
- University College,
Cork, Ireland
- University of Crete,
Institute of Computer Science,
Heraklio, Crete, Greece
- University of Delaware,
Newark, Delaware
- University of Exeter,
Computer Unit,
Exeter, Devon, England
- University of Glasgow,
Department of Computing Science,
Glasgow, Scotland
- University of Groningen,
Groningen, The Netherlands
- University of Heidelberg,
Computing Center Heidelberg,
Germany
- University of Illinois at Chicago,
Computer Center,
Chicago, Illinois
- University of Kansas,
Academic Computing Services,
Lawrence, Kansas
- Universität Koblenz-Landau,
*Koblenz, Federal Republic of
Germany*
- University of Maryland,
Department of Computer Science,
College Park, Maryland
- University of Maryland
at College Park,
Computer Science Center,
College Park, Maryland
- University of Massachusetts,
Amherst, Massachusetts
- University of Oslo,
Institute of Mathematics,
Blindern, Oslo, Norway
- University of Ottawa,
Ottawa, Ontario, Canada
- University of Salford,
Salford, England
- University of Southern California,
Information Sciences Institute,
Marina del Rey, California
- University of Stockholm,
Department of Mathematics,
Stockholm, Sweden
- University of Texas at Austin,
Austin, Texas
- University of Washington,
Department of Computer Science,
Seattle, Washington
- University of Western Australia,
Regional Computing Centre,
Nedlands, Australia
- Uppsala University,
Uppsala, Sweden
- Vereinigte Aluminium-Werke AG,
Bonn, Federal Republic of Germany
- Villanova University,
Villanova, Pennsylvania
- Vrije Universiteit,
Amsterdam, The Netherlands
- Washington State University,
Pullman, Washington
- Widener University,
Computing Services,
Chester, Pennsylvania
- Worcester Polytechnic Institute,
Worcester, Massachusetts
- Yale University,
Department of Computer Science,
New Haven, Connecticut

BYLAWS of the TeX Users Group ("TUG")

Article I PURPOSES, POWERS AND NON-PROFIT STATUS

Section 1. Purposes. The TeX Users Group (the "Corporation") has been formed exclusively for charitable, educational and scientific purposes as such terms are defined in Section 501(c)(3) of the Internal Revenue Code of 1986, or the corresponding provision of any future United States internal revenue law (hereinafter the Internal Revenue Code of 1986), and specifically to identify, develop, operate, fund, support, promote and encourage charitable, educational and scientific programs and projects which will stimulate those who have an interest in systems for typesetting technical text and font design; to exchange information of same and associated use of computer peripheral equipment; to establish channels to facilitate the exchange of macro packages, etc., through publications and otherwise; and to develop, implement and sponsor educational programs, seminars and conferences in connection with the foregoing and for any lawful purpose or purposes permitted under the Rhode Island Non-profit Corporation Act.

Section 2. Powers. The Corporation shall have the power, directly or indirectly, either alone or in conjunction or cooperation with others, to do any and all lawful acts and things and to engage in any and all lawful activities which may be necessary, or convenient to effect any or all of the purposes for which the Corporation is organized, and to aid or assist other organizations whose activities are such as to further accomplish, foster, or attain any of such purposes. The power of the Corporation shall include, but not be limited to, the acceptance of contributions in cash, in kind or otherwise from both the public and private sectors. Notwithstanding anything herein to the contrary, the Corporation shall exercise its powers only in furtherance of exempt purposes as such terms are defined in Section 501(c)(3) of the Internal Revenue Code of 1986 and the regulations from time to time promulgated thereunder.

Section 3. Non-Profit Status. The Corporation shall be nonprofit and shall not have or issue

shares of capital stock, and shall not declare or pay dividends. No part of the net income or profit of the Corporation shall inure to the benefit of any member, director, officer, or other individual, or to the benefit of any organization not qualified for tax exemption under Section 501(c)(3) of the Internal Revenue Code except as permitted by law. No substantial part of the activities of the Corporation shall be carrying on propaganda, or otherwise attempting to influence legislation (except as otherwise provided by Internal Revenue Code Section 501(h)), or participating in, or intervening in (including the publication or distribution of statements), any political campaign on behalf of any candidate for public office. Upon the dissolution of this organization, assets shall be distributed for one or more exempt purposes within the meaning of Section 501(c)(3) of the Internal Revenue Code or corresponding Section of any future Federal tax code, or shall be distributed to the Federal Government, or to a state or local government, for a public purpose.

Article II OFFICES

The Corporation will have offices at such places both within and without the State of Rhode Island as may from time to time be determined by the board of directors.

Article III MEMBERS

Section 1. Constitution. The members of the Corporation will be such persons, natural or legal, who will meet such qualifications and requirements (including without limitation payment of initiation fees and dues) as from time to time may be established by the board of directors. The board of directors will be the sole judge of the qualifications of the members and its determination as to whether a person is or is not a member will be final. The board of directors may, in its discretion, create different classifications of members and prescribe different rights, privileges, qualifications or requirements for each class.

Section 2. Place of Meetings. All annual meetings of the members and all special meetings of the members called by the president or the board of directors will be held at such place, either within or without the State of Rhode Island, as will be stated in the notice of meeting.

Section 3. Annual Meetings. Meetings of the members will be held in conjunction with TUG conferences. Such conferences will normally be held annually; otherwise, an annual meeting of the members will be held on the first Monday of August in each year if not a legal holiday in the place where it is to be held, and, if a legal holiday, then on the next day following which is not a legal holiday, beginning at 10:00 a.m. or at any other time designated in the notice of the meeting. At each annual meeting, the members will transact such business as may properly come before the meeting. In the event of the failure to hold said annual meeting at any time or for any cause, any and all business which might have been transacted at such meeting may be transacted at the next succeeding meeting, whether special or annual.

Section 4. Special Meetings. A special meeting of the members, for any purpose or purposes, may be called by the President or by the Board of Directors. Any such call will state the purpose or purposes of the proposed meeting.

Section 5. Notice of Meetings. Written notice of each annual or special meeting stating the place, day and hour of the meeting (and the purpose or purposes of any special meeting) will be given by or at the direction of the president, the secretary or the person or persons calling the meeting to each member entitled to vote at such meeting not less than ten nor more than sixty days before the meeting. Business transacted at any special meeting of members will be limited to the purposes stated in the notice of the meeting or any written waiver thereof.

Section 6. Quorum. Fifty (50) members present in person, will constitute a quorum at all meetings of the members. If, however, such quorum will not be present at any such meeting, the members entitled to vote thereat will have power to adjourn the meeting from time to time, without notice other than announcement at the meeting, until a quorum will be present. At such adjourned meeting at which a quorum will be present any business may be transacted which might have been transacted at the meeting as originally called. If adjournment is for more than thirty days, a notice of the adjourned

meeting will be given to each member entitled to vote at the meeting. When a quorum is present at any meeting, the vote of the holders of a majority of the votes entitled to be cast and present in person will decide any question brought before such meeting, unless the vote of a greater number is required by law. A voice vote will normally be considered sufficient for business actions. A show of hands may be requested when the outcome is in doubt.

Section 7. Access to Documents. Nothing in these bylaws shall be construed to limit the access of TUG members to TUG documents. Members requesting copies of any TUG document may be charged a reasonable copying fee and members requesting publications or mailing lists presented to the public for sale may be charged the same fee as the general public. Members requesting copies of documents to be used in performance of TUG related duties may request that the copying fee be waived. TUG documents include, but are not limited to: contracts, Board minutes, Executive Committee minutes, Finance Committee minutes, office procedure manuals, IRS filings, and written communications from or to the TUG office. This section does not authorize the release of any information that federal or state law protects from disclosure.

Article IV DIRECTORS

Section 1. Powers. The affairs of the Corporation will be managed by the Board of Directors.

Section 2. Number. The number of directors will be not more than thirty. Under very special circumstances, particularly deserving individuals may be designated as permanent honorary members of the Board, without vote, and without being included in the number of members specified in this section.

Section 3. Composition. The Board of Directors will consist of the TUG President, Elected Board Members, Honorary Members, and Non-elected Vice Presidents. Prior to the first election of Elected Board Members, the Board of Directors will consist of the Finance Committee, Site Coordinators, Wizards and other active members nominated by the Board of Directors.

Section 4. Honorary Members. The Grand Wizard, Donald E. Knuth, the Wizard of Fonts, Hermann Zapf, and the Founding Executive Director,

Raymond Goucher, are designated as permanent honorary members of the Board.

Section 5. Non-elected Vice Presidents. The leaders of other T_EX user groups may be appointed to the Board with the title of vice president. An increase in the number of members on the Board shall be made as appropriate.

Section 6. Nominations, Election and Term. Any member may have their name placed in nomination for election to the board by submitting a petition, signed by two (2) other members, to the TUG Office at least thirty (30) days prior to the election. In addition, any member may be nominated for the board during the annual business meeting. Members nominated at the annual business meeting shall have seven (7) days to notify the TUG Office that they accept the nomination in writing. Election of the directors shall be by written mail ballot of the entire membership. Each director will hold office for a term of two (2) years. Directors may be re-elected for successive terms. Directors need not be members of the Corporation or residents of the State of Rhode Island.

Section 7. Meetings. The board of directors may hold meetings, both regular and special, either within or without the State of Rhode Island. The first meeting of each newly elected board of directors will be held at such time and place as will be specified in a notice delivered as hereinafter provided for special meetings of the board of directors, or as will be specified in a written waiver signed by all of the directors. Regular meetings of the board of directors may be held without notice at such time and at such place as will from time to time be determined by the board of directors. Special meetings of the board of directors may be called by the president on two days' notice to each director, either personally or by mail or by telegram. Special meetings will be called by the president in like manner and on like notice on the written request of two directors. Meetings of the directors may be held by means of a telephone conference circuit and connection to such circuit will constitute presence at such meeting.

Section 8. Vacancies. Any vacancy occurring on the board of directors may be filled by the President. A director appointed to fill a vacancy will be appointed for the unexpired term of his or her predecessor in office. Any place on the board to be filled by reason of an increase in the number of directors may be filled by the President for a term of office continuing only until the next appointment of directors.

Section 9. Quorum. At all meetings of the board of directors, twenty-five (25%) percent of the number of directors fixed pursuant to Section 2 of this Article will constitute a quorum for the transaction of business, and the act of a majority of the directors present at a meeting at which a quorum is present will be the act of the board of directors, unless the act of a greater number is required by the Rhode Island non-profit corporation act or by the articles of incorporation.

Section 10. Directors' Consent Vote. Any action required or permitted to be taken at a meeting of the board of directors or of any committee thereof may be taken without a meeting by instead taking a vote by mail, according to the following procedure. Any board member may submit a motion in writing. Any other board member may second the motion. Amendments to the motion are allowed, but not amendments to amendments. From the point that the motion on the floor has been seconded, there shall be a two-week period of discussion regarding the motion. After the discussion period, there shall be an active voting period of two weeks, after which voting shall be terminated. The votes shall then be tallied, counting the number of yeas, nays, and abstentions. The total number of votes cast must be at least 50% of total members permitted to vote, otherwise the motion fails. When the 50% requirement is met, a motion shall pass when two-thirds of votes cast are in the affirmative. Members may use standard mail, electronic mail, or facsimile to cast a written vote. Upon approval of the motion, the entire board or committee shall be notified by standard mail of the results of the vote.

Section 11. Committees of Directors. The board of directors may, by resolution adopted by a majority of the board, designate one or more committees, including an executive committee, each committee to consist of two or more directors appointed by the board. The board may appoint one or more directors as alternate members of any committee, who may replace any absent or disqualified member at any meeting of the committee. Except as otherwise provided by the Rhode Island non-profit corporation act or these bylaws, any such committee, to the extent provided in the resolution, will have and may exercise all the authority of the board of directors; provided, however, that in the absence or disqualification of any member of such committee or committees, the member or members thereof present at any meeting and not disqualified from voting, whether or not he or she or they constitute a quorum, may unanimously appoint another member

of the board of directors to act at the meeting in the place of any such absent or disqualified member. Such committee or committees will have such name or names as may be determined from time to time by resolution adopted by the board of directors. Each committee will keep regular minutes of its proceedings and report the same to the board of directors when required.

ARTICLE V COMMITTEES

Section 1. Executive Committee. There will be established an Executive Committee which will consist of the President, plus three other board members. It will be the responsibility of the Executive Committee to adopt interim procedures and policies when necessary on behalf of the Corporation, subject to the ultimate approval of the Board of Directors.

Section 2. Technical Council. There will be established a Technical Council which will consist of Site Coordinators, Wizards and other active TeX contributors. The initial members will be appointed by the Board of Directors. After the initial appointment of Council members by the Board, the Technical Council shall determine its own composition and operating procedures. The purposes and goals of the Technical Council shall be determined by the Technical Council; however, the purposes and goals shall be consistent with the purposes, powers, and non-profit status of the TeX Users Group. The Grand Wizard and the Wizard of Fonts shall be permanent honorary members of the Technical Council. The Technical Council shall designate a representative to attend meetings of the TUG board in an advisory capacity.

Section 3. Planning Committee. There will be established a Planning Committee responsible for establishing, with approval by the Board, TUG's strategic goals for recommending to the Board a three- (or more) year strategic plan to implement these goals. Members of the Planning Committee will be appointed by the President with the approval of the Board.

Section 4. Nominating Committee. Prior to the annual meeting, a Nominating Committee will be appointed by the Board for the purpose of suggesting candidates to fill those offices. This committee shall nominate at least one member to fill each office up for election.

Section 5. Ad Hoc Committees. The Board of Directors may from time to time, by resolution adopted by a majority of the Board, appoint one or more Ad Hoc Committees to perform such functions as may be designated in said resolution.

Article VI NOTICES

Section 1. How Delivered. Whenever under the provisions of the Rhode Island non-profit corporation act or of the articles of incorporation or of these bylaws written notice is required to be given to any person, such notice may be given by mail, addressed to such person at his or her address as it appears in the records of the Corporation, with postage thereon prepaid, and such notice will be deemed to be delivered, if mailed, at the time when the same will be deposited in the United States mail. Notice may also be given by telegram or personally to any director.

Section 2. Waivers of Notice. Whenever any notice is required to be given under the provisions of the Rhode Island non-profit corporation act or the articles of incorporation or these bylaws, a waiver thereof in writing, signed by the person or persons entitled to such notice, whether before or after the time stated therein, will be deemed equivalent to the giving of such notice. Attendance of a person at a meeting will constitute a waiver of notice of such meeting, except when the person attends a meeting for the express purpose of objecting to the transaction of any business because the meeting is not lawfully called or convened.

Section 3. Specification of Business. Neither the business to be transacted at, nor the purpose of, any meeting of the members of the Corporation or of a committee of the board of directors of the Corporation need be specified in any written waiver of notice except as otherwise herein expressly provided.

Article VII OFFICERS

Section 1. Number. The officers of the Corporation will be a president, a vice president, a secretary, and a treasurer. The board of directors may from time to time elect or appoint such other officers including more vice presidents and assistant officers, as it may deem necessary. Any two or more offices may be

held by the same person with the exception of the offices of president and secretary.

Section 2. Eligibility for Nomination. Any member may be nominated for TUG President, and any board member may be nominated for the other board offices.

Section 3. Nomination Procedure. Any member may have their name placed in nomination for election to the office of TUG President by submitting a petition, signed by two (2) other members, to the TUG Office at least thirty (30) days prior to the election. In addition, any member may be nominated for the office of TUG President during the annual business meeting. Members nominated at the annual business meeting shall have seven (7) days to notify the TUG Office in writing that they accept the nomination.

Section 4. Election and Term. During the 1991 annual meeting, officers will be selected by the Board of Directors. Thereafter, the president will be elected by the general membership in accordance with the election procedures.

All Officers shall be selected for a term not to exceed two years. Officers other than the President shall be appointed by the Board of Directors. Any officer may be removed by the Board of Directors whenever, in its judgment, the best interests of the Corporation will be served thereby.

Section 5. Election Procedures. All elections will be conducted in accordance with election procedures approved by the Board of Directors. The election of the president shall be by written mail ballot of the entire membership. The candidate receiving the most votes will be elected.

Section 6. President. The President will preside at meetings of the General Membership, the Board of Directors and the Executive Committee.

Section 7. Vice President. The Vice President will serve in the absence of the President and will undertake other administrative duties as designated by the President.

Section 8. Secretary. The Secretary will maintain the records of the Corporation and see that all notices are duly given in accordance with the provisions of these Bylaws or as required by law. The Secretary will also conduct Corporate correspondence.

Section 9. Treasurer. The Treasurer will serve as chief financial officer and in general, will perform all of the duties incident to the office of Treasurer

and such other duties as from time to time may be assigned to him by the President or Board of Directors.

Section 10. Vacancies. When an office becomes vacant for any reason, the President will appoint a member to serve out the remainder of that term. When the office of the President becomes vacant, the Vice President will become President for the remainder of the President's term and will then, as President, appoint a member to serve as Vice President.

Section 11. Signing of Instruments. All checks, drafts, orders, notes and other obligations of the Corporation for the payment of money, deeds, mortgages, leases, contracts, bonds and other corporate instruments may be signed by such officer or officers of the Corporation or by such other person or persons as may from time to time be designated by general or special vote of the board of directors.

Section 12. Voting of Securities. Except as the board of directors may generally or in particular cases otherwise specify, the president or the treasurer may on behalf of the Corporation vote or take any other action with respect to shares of stock or beneficial interest of any other corporation, or of any association, trust or firm, of which any securities are held by the Corporation, and may appoint any person or persons to act as proxy or attorney-in-fact for the Corporation, with or without power of substitution, at any meeting thereof.

Article VIII EXECUTIVE DIRECTOR

Section 1. Duties. The Board of Directors shall select and employ an Executive Director who shall be responsible for the general administration of the Corporation's activities.

Section 2. Immediate Supervision. The Executive Director shall work under the immediate direction of the Executive Committee. The Executive Director shall attend meetings of the Executive Committee, the Finance Committee, and the Board of Directors, but shall not be a member of any of these bodies. The presiding officer of any of these meetings may request the absence of the Executive Director.

Article IX SEAL

The corporate seal will have inscribed upon it the name of the Corporation and such other appropriate

language as may be prescribed by the Rhode Island non-profit corporation act or from time to time by the board of directors.

Article X FISCAL YEAR

The fiscal year of the Corporation will be determined by the board of directors and in the absence of such determination will be the calendar year.

Article XI INDEMNIFICATION

Section 1. Agreement of Corporation. In order to induce the directors and officers of the Corporation to serve as such, the Corporation adopts this Article and agrees to provide the directors and officers of the Corporation with the benefits contemplated hereby.

Section 2. Acceptance of Director or Officer. This Article will apply, and the benefits hereof will be available, to each director and officer of the Corporation who executes and delivers to the Secretary of the Corporation a written statement to the effect that the director or officer accepts the provisions of this Article and agrees to abide by the terms contained herein.

Section 3. Definitions. As used herein, the following terms will have the following respective meanings:

“Covered Act” means any act or omission by the Indemnified Person in the Indemnified Person’s official capacity with the Corporation and while serving as such or while serving at the request of the Corporation as a member of the governing body, officer, employee or agent of another corporation, partnership, joint venture, trust or other enterprise.

“Excluded Claim” has the meaning set forth in Paragraph 6, hereof.

“Expenses” means any reasonable expenses incurred by the Indemnified Person in connection with the defense of any claim made against the Indemnified Person for Covered Acts including, without being limited to, legal, accounting or investigative fees and expenses (including the expense of bonds necessary to pursue an appeal of an adverse judgment).

“Indemnified Person” means any director or officer of the Corporation who accepts election or appointment as a director or officer and agrees to serve as such in the manner provided in Paragraph 2 hereof.

“Loss” means any amount which the Indemnified Person is legally obligated to pay as a result of any claim made against the Indemnified Person for Covered Acts including, without being limited to, judgments for, and awards of, damages, amounts paid in settlement of any claim, any fine or penalty or, with respect to an employee benefit plan, any excise tax or penalty.

“Proceeding” means any threatened, pending or completed action, suit or proceeding, whether civil, criminal, administrative or investigative.

Section 4. Indemnification. Subject to the exclusions hereinafter set forth, the Corporation will indemnify the Indemnified Person against and hold the Indemnified Person harmless from any Loss or Expenses.

Section 5. Advance Payment of Expenses. The Corporation will pay the Expenses of the Indemnified Person in advance of the final disposition of any Proceeding except to the extent that the defense of a claim against the Indemnified Person is undertaken pursuant to any directors’ and officers’ liability insurance (or equivalent insurance known by another term) maintained by the Corporation. The advance payment of Expenses will be subject to the Indemnified Person’s first agreeing in writing with the Corporation to repay the sums paid by it hereunder if it is thereafter determined that the Proceeding involved an Excluded Claim or that the Indemnified Person was otherwise not entitled to indemnity under these Bylaws.

Section 6. Exclusions. The Corporation will not be liable to pay any Loss or Expenses (an “Excluded Claim”):

(a) With respect to a Proceeding in which a final non-appealable judgment or other adjudication by a court of competent jurisdiction determines that the Indemnified Person is liable to the Corporation (as distinguished from being liable to a third party) for: (i) any breach of the Indemnified Person’s duty of loyalty to the Corporation or its members; (ii) acts or omissions not in good faith or which involve intentional misconduct or knowing violation of law; or (iii) any transaction from which the Indemnified Person derived an improper personal benefit; or

(b) If a final, non-appealable judgment or other adjudication by a court of competent jurisdiction determines that such payment is unlawful.

Section 7. Notice to Corporation; Insurance. Promptly after receipt by the Indemnified Person of notice of the commencement of or the threat

of commencement of any Proceeding, the Indemnified Person will, if indemnification with respect thereto may be sought from the Corporation under these Bylaws, notify the Corporation of the commencement thereof. Failure to promptly notify the Corporation will not adversely affect the Indemnified Person's right to indemnification hereunder unless and only to the extent that the Corporation is materially prejudiced in its ability to defend against the Proceeding by reason of such failure. If, at the time of the receipt of such notice, the Corporation has any directors' and officers' liability insurance in effect, the Corporation will give prompt notice of the commencement of such Proceeding to the insurer in accordance with the procedures set forth in the policy or policies in favor of the Indemnified Person. The Corporation will thereafter take all the necessary or desirable action to cause such insurer to pay, on behalf of the Indemnified Person, all Loss and Expenses payable as a result of such Proceeding in accordance with the terms of such policies.

Section 8. Indemnification Procedures. (a) Payments on account of the Corporation's indemnity against Loss will be made by the Treasurer of the Corporation except if, in the specific case, a determination is made that the indemnification of the Indemnified Person is not proper in the circumstances because such Loss results from a claim which is an Excluded Claim. If the Corporation so determines that the Loss results from an Excluded Claim (although no such determination is required by the Corporation hereunder prior to payment of a Loss by the Treasurer), the determination shall be made:

(i) By the Board of Directors by a majority vote of a quorum consisting of directors not at the time parties to the Proceeding; or

(ii) If a quorum cannot be obtained for purposes of clause (i) of this subparagraph (a), then by a majority vote of a committee of the Board of Directors duly designated to act in the matter by a majority vote of the full Board (in which designation directors who are parties to the Proceeding may participate) consisting solely of three or more directors not at the time parties to the Proceeding; or

(iii) By independent legal counsel designated: (A) by the Board of Directors in the manner described in clause (i) of this subparagraph (a), or by a committee of the Board of Directors established in the manner described in clause (ii) of this subparagraph (a), or (B) if the requisite quorum of the full Board cannot be obtained therefor and a

committee cannot be so established, by a majority vote of the full Board (in which designation directors who are parties to the Proceeding may participate). If made, any such determination permitted to be made by this subparagraph (a) will be made within 60 days of the Indemnified Person's written request for payment of a Loss.

(b) Payment of an Indemnified Person's Expenses in advance of the final disposition of any Proceeding will be made by the Treasurer of the Corporation except if, in the specific case, a determination is made pursuant to Paragraph 8(a) above that indemnification of the Indemnified Person is not proper in the circumstances because the Proceeding involved an Excluded Claim.

(c) The Corporation will have the power to purchase and maintain insurance on behalf of any Indemnified Person against liability asserted against him or her with respect to any Covered Act, whether or not the Corporation would have the power to indemnify such Indemnified Person against such liability under the provisions of this Article. The Corporation will be subrogated to the rights of such Indemnified Person to the extent that the Corporation has made any payments to such Indemnified Person in respect to any Loss or Expense as provided herein.

Section 9. Settlement. The Corporation will have no obligation to indemnify the Indemnified Person under this Article for any amounts paid in settlement of any Proceeding effected without the Corporation's prior written consent. The Corporation will not unreasonably withhold or delay its consent to any proposed settlement. If the Corporation so consents to the settlement of any Proceeding, or unreasonably withdraws or delays such consent, it will be conclusively and irrebutably presumed for all purposes that the Loss or Expense does not constitute an Excluded Claim. If the Corporation reasonably withdraws its consent solely on the ground that the Proceeding constitutes an Excluded Claim, the Indemnified Person may accept the settlement without the consent of the Corporation, without prejudice to the Indemnified Person's rights to indemnification in the event the Corporation does not ultimately prevail on the issue of whether the Proceeding constitutes an Excluded Claim.

Section 10. Rights Not Exclusive. The rights provided hereunder will not be deemed exclusive of any other rights to which the Indemnified Person may be entitled under any agreement, vote of disinterested

directors or otherwise, both as to action in the Indemnified Person's official capacity and as to action in any other capacity while holding such office, and will continue after the Indemnified Person ceases to serve the Corporation as an Indemnified Person.

Section 11. Enforcement. (a) The Indemnified Person's right to indemnification hereunder will be enforceable by the Indemnified Person in any court of competent jurisdiction and will be enforceable notwithstanding that an adverse determination has been made as provided in Paragraph 8 hereof.

(b) In the event that any action is instituted by the Indemnified Person under these Bylaws, the Indemnified Person will be entitled to be paid all court costs and expenses, including reasonable attorneys' fees, incurred by the Indemnified Person with respect to such action, unless the court determines that each of the material assertions made by the Indemnified Person as a basis for such action was not made in good faith or was frivolous.

Section 12. Severability. If any provision of this Article is determined by a court to require the Corporation to perform or to fail to perform an act which is in violation of applicable law, this Article shall be limited or modified in its application to the minimum extent necessary to avoid a violation of law, and, as so limited or modified, this Article shall be enforceable in accordance with its terms.

Section 13. Successor and Assigns. The provisions of this Article will be (a) binding upon all successors and assigns of the Corporation (including any transferee of all or substantially all of its assets) and (b) binding on and inure to the benefit of the heirs, executors, administrators, and other personal representatives of the Indemnified Person.

Amendment. No amendment or termination of this Article will be effective as to an Indemnified Person without the prior written consent of that Indemnified Person and, in any event, will not be effective as to any Covered Act of the Indemnified Person occurring prior to the amendment or termination.

Article XII AMENDMENTS

The power to alter, amend or repeal the bylaws or to adopt new bylaws will be vested in the Board of Directors by affirmative vote of the directors in the manner provided in these bylaws.

PXLGen™

**Finally, you can use
superior MicroPress
scalable typefaces
with your version of
TeX.**

- Choose from 150+ quality typefaces
- Generate PXL/PK files in seconds
- Font effects: compressed/expanded fonts, shading, outline, smallcaps, and more
- Creates matching TFM files
- As low as \$10 per font

Contact MicroPress for pricing and availability for your CPU.

MICROPRESS INC.

68-30 HARROW STREET, FOREST HILLS, NY 11375
TEL: 718-575-1816 FAX: 718-575-8038



PXLGen is a trademark of MicroPress Inc. Other Products mentioned are trademarks of their respective companies.

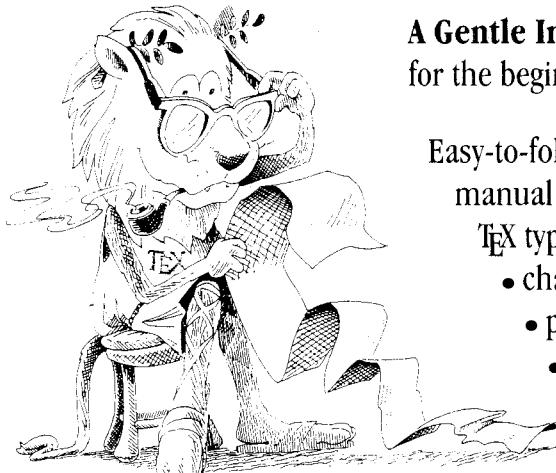
TAME THAT T_EX LION!

with



A Gentle Introduction to T_EX

A Manual for Self-study by Michael Doob



A Gentle Introduction to T_EX is perfect
for the beginning T_EX user.

Easy-to-follow and straightforward, this 89 page, softbound
manual reveals the basics behind
T_EX typesetting:

- characters and words
- paragraphs and pages
- mathematics
- simple tables
- simple macros

...with dozens of helpful examples and
solutions in 11 friendly chapters.

Special offer for TUG members \$10
10 or more copies (members) \$ 8
Regular price \$15

Shipping: U.S.: \$2 per copy

Outside of U.S.: \$3 per copy surface; \$4 air

Ask about TUG's other publications for beginning T_EXers



**Available now
from TUG**

T_EX Users Group

P. O. Box 9506

Providence, RI 02940 U. S. A.

Phone: (401) 751-7760

Fax: (401) 751-1071

Mastercard/Visa, checks and
money orders accepted

TEXniques

Publications for the T_EX Community

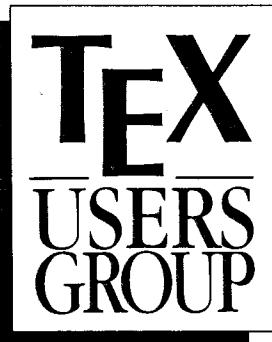
Available now:

1. VAX Language-Sensitive Editor (LSEDIT)
Quick Reference Guide for Use with the L_AT_EX Environment and L_AT_EX Style Templates by Kent McPherson
2. Table Making – the INRST_EX Method by Michael J. Ferguson
3. User's Guide to the IdxT_EX Program by R. L. Aurbach
4. User's Guide to the GloT_EX Program by R. L. Aurbach
5. Conference Proceedings, T_EX Users Group Eighth Annual Meeting, Seattle, August 24–26, 1987, Dean Guenther, Editor
6. The P_TC_TE_X Manual by Michael J. Wichura
7. Conference Proceedings, T_EX Users Group Ninth Annual Meeting, Montréal, August 22–24, 1988, Christina Thiele, Editor
8. A Users' Guide for T_EX by Frances Huth
9. An Introduction to L_AT_EX by Michael Urban
10. L_AT_EX Command Summary by L. Botway and C. Biemesderfer
11. First Grade T_EX by Arthur Samuel
12. A Gentle Introduction to T_EX by Michael Doob
13. METAFONTware by Donald E. Knuth, Tomas G. Rokicki, and Arthur Samuel

Coming soon:

14. A Permuted Index for T_EX and L_AT_EX by Bill Cheswick
15. EDMAC: A Plain T_EX Format for Critical Editions by John Lavagnino and Dominik Wujastyk

T_EX Users Group
P. O. Box 9506
Providence, R. I. 02940, U.S.A.



Complete and return this form with payment to:

TEx Users Group
 Membership Department
 P.O. Box 594
 Providence, RI 02901 USA
 Telephone: (401) 751-7760
 FAX: (401) 751-1071
 Email: tug@Math.AMS.com

Membership is effective from January 1 to December 31 and includes a subscription to *TUGboat, The Communications of the TEx Users Group*. 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
- More information on TEx

Individual Membership Application

Name _____

Institutional affiliation, if any _____

Position _____

Address (business or home (circle one)) _____

City _____

State or Country _____ Zip _____

Daytime telephone _____ FAX _____

Email addresses (*please specify networks, as well*) _____

I am also a member of the following other TEx organizations:

Specific applications or reasons for interest in TEx:

Hardware on which TEx is used:

Computer and operating system

Output device/printer

There are two types of TUG members: regular members, who pay annual dues of \$45; and full-time student members, whose annual dues are \$35. Students must include verification of student status with their applications.

Please indicate the type of membership for which you are applying:

- Regular @ \$45 Full-time student @ \$35

Amount enclosed for 1991 membership: \$ _____

(Prepayment in US dollars drawn on a US bank is required)

- Check/money order payable to TEx Users Group enclosed
- Charge to MasterCard/VISA

Card # _____ Exp. date _____

Signature _____

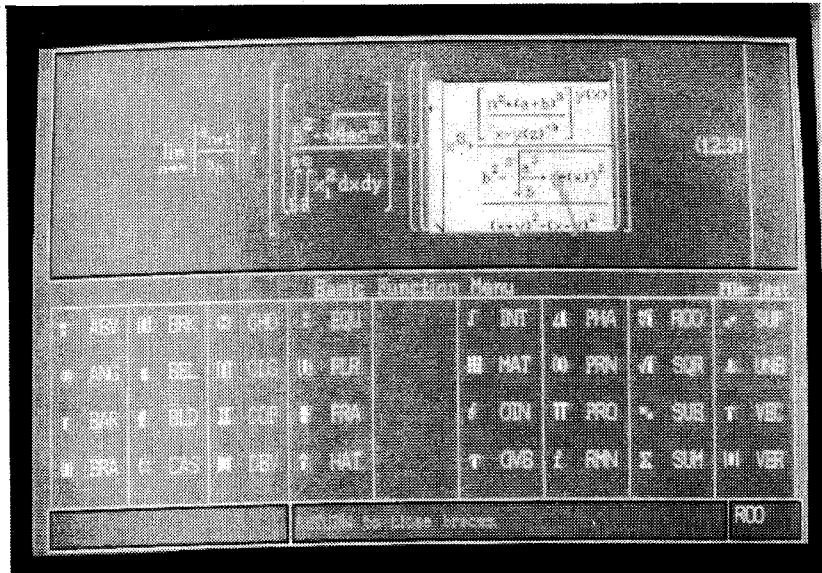
T_EX EDITION of MathEdit

This powerful new equation editor can
be used to create equations for T_EX.
For IBM PC

MathEdit

- * Easy to Use Version 2.0
- * Menu driven so no codes
need to be learned
- * High-quality T_EX printing

WYSIWYG ->
View your equation as
you create it. Then
insert into your T_EX
document with one
command.



T_EX Edition ONLY \$129.00

Professional Edition \$199.00

Shipping: \$4 (U.S.A.), \$25 (Canada), \$35 (Overseas)

VISA, Mastercard and University P.O.'s accepted.

K-TALK
COMMUNICATIONS

30 West First Avenue
Columbus, Ohio 43201
(614) 294-3535
FAX (614) 294-3704

Publishing Companion® translates

WordPerfect

to

T_EX or L_AT_EX

IN ONE EASY STEP!

With **Publishing Companion**, you can publish documents using T_EX or L_AT_EX with **little or no T_EX knowledge**. Your WordPerfect files are translated into T_EX or L_AT_EX files, so anyone using this simple word processor can immediately begin typesetting their own documents!

Publishing Companion translates EQUATIONS, FOOTNOTES, ENDNOTES, FONT STYLES, and much more!

Retail Price	\$249.00
Academic Discount Price	\$199.00

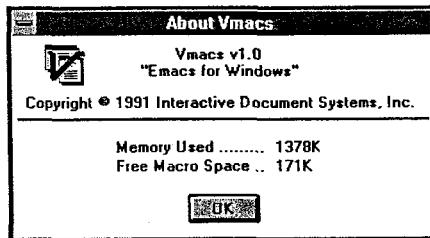
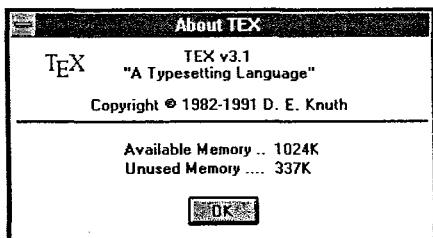
For more information or to place an order, call or write:



30 West First Ave, Suite 100
Columbus, Ohio 43201
(614)294-3535
FAX (614)294-3704

TYPESET QUALITY WITH THE EASE OF WORD PROCESSING

Finally, a *better* way to use **TEX**



TEX in a multitasking Windows™ environment gives you the freedom to compose long jobs without tying up your computer. The interactive previewer shows your documents before printing. And, Windows™ allows you to choose among a wide variety of output devices, from dot matrix printers to typesetters.

An enhanced Emacs-style editor helps you avoid common TEX errors. The integrated spell checker catches spelling errors before someone else does. And, the hypertext help system guides you directly to the information you need to get the job done.

It's About Time...AzTEX

Interactive Document Systems, Inc.
P.O. Box 25323 • Tempe, AZ 85285-5323 • (602) 967-2563 • FAX: (602) 780-2472



Publishing Services



From the Basic

The American Mathematical Society can offer you a basic TEX publishing service. You provide the DVI file and we will produce typeset pages using an Autologic APS Micro-5 phototypesetter. The low cost is basic too: only \$5 per page for the first 100 pages; \$2.50 per page for additional pages, with a \$30 minimum. Quick turnaround is important to you and us ... a manuscript up to 500 pages can be back in your hands in just one week or less.

To the Complex

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

Macro-Writing

TEX Problem Solving

Autologic Fonts

Keyboarding

Art and Pasteup

Camera Work

Printing

Binding

For more information or to schedule a job, please contact Regina Girouard, American Mathematical Society, P.O. Box 6248, Providence, RI 02940 or call 401-455-4060 or 800-321-4AMS in the continental U.S.

Do more and do it better with new PC_T_EX.

PC_T_EX, PC_T_EX/386 & Big PC_T_EX/386, Versions 3.0

Feature/Specification	PC _T _E X 2.93	PC _T _E X 3.0	PC _T _E X/386 3.0	Big PC _T _E X/386 3.0
Page & Memory Capacity You won't see "T _E X Capacity Exceeded"!	mem_max (1.00)	65534 131070 (Double!)	131070 131070 (Double!)	262140 262140 (Quadruple!)
Hyphenation Table Size Space for hyphenation patterns	trie_size (1.00)	15000 30000 (Double!)	30000 30000 (Double!)	60000 60000 (Quadruple!)
Trie Op Size Complexity of hyphenation patterns	trie_op_size (1.00)	255 1024 (4.02)	1024 1024 (4.02)	2048 2048 (8.03)
Maximum Trie Ops Per Language Especially important for Dutch and German hyphenation	N/A	512	512	512
Buffer Size Maximum # of characters on input lines	buf_size (1.00)	1024 1500 (1.46)	1500 1500 (1.46)	3000 3000 (2.93)
Stack Size Maximum # of simultaneous input sources	stack_size (1.00)	200 200 (1.00)	200 200 (1.00)	300 300 (1.50)
Maximum # of Strings	max_strings (1.00)	4500 5000 (1.11)	5000 5000 (1.11)	10000 10000 (2.22)
String Pool Maximum # of characters in strings	pool_size (1.00)	50000 50000 (1.00)	60000 60000 (1.20)	60000 60000 (1.20)
Save Size Space for saving values outside current group	save_size (1.00)	600 2000 (3.33)	2000 2000 (3.33)	4000 4000 (6.67)
Maximum # of T_EX Commands	hash_size (1.00)	3000 5000 (1.66)	5000 5000 (1.66)	10000 10000 (3.33)
Minimum Free RAM Required	385K (1.00)	385K 1.3M (1.00)	1.3M 1.3M (3.38)	1.3M 1.3M (3.38)
Minimum Free RAM Recommended Memory recommended for optimum performance	550K (1.00)	550K 1.3M (1.00)	550K 1.3M (2.36)	4.0M 4.0M (7.27)
Font Memory For TFM data storage	font_mem_size (1.00)	51199 65534 (1.28)	65534 65534 (1.28)	65534 65534 (1.28)
Maximum Fonts Per Job	font_max (1.00)	127 127 (1.00)	127 127 (1.00)	255 255 (2.00)
List Price Order yours today!	\$249 (1.00)	\$249 (1.00)	\$295 (1.18)	\$349 (1.40)
Upgrade Price From PC _T _E X 2.93 or earlier version	\$50 (1.00)	\$50 (1.00)	\$99 (1.98)	\$149 (2.98)

This all adds up to...

More power, greater performance, and increased memory capacity for the latest versions of popular macro packages like L_AT_EX and A_MS-T_EX. And all three new PC_T_EX products feature the character sets and hyphenation tables to handle even the most complex European languages.

Order today. Call (415) 388-8853.



12 Madrona Avenue
Mill Valley, CA 94941

IT'S BETTER THAN
WYSIWYG

AHH!....
WHAT YOU SEE IS
MATHEMATICS

Scientific Word - INERTIA.TEX

File Edit Insert View Style Tools Math

If E_1 and H_1 denote the energy of the body after the emission of light, as measured relative to the system (x, y, z) and (ξ, η, ζ) , respectively, we obtain, using the relation indicated above,

$$E_0 = E_1 + \left(\frac{L}{2} + \frac{L}{2} \right),$$

$$H_0 = H_1 + \left(\frac{L}{2} \frac{1 - \frac{v}{c} \cos \varphi}{\sqrt{1 - (\frac{v}{c})^2}} + \frac{L}{2} \frac{1 + \frac{v}{c} \cos \varphi}{\sqrt{1 - (\frac{v}{c})^2}} \right) = H_1 + \frac{L}{\sqrt{1 - (\frac{v}{c})^2}}$$

Clipboard

SH_0=H_1+Lefl(\frac{L}{2}\frac{1-\frac{v}{c}\cos\varphi}{\sqrt{1-(\frac{v}{c})^2}})+\frac{L}{2}\frac{1+\frac{v}{c}\cos\varphi}{\sqrt{1-(\frac{v}{c})^2}}=H_1+\frac{L}{\sqrt{1-(\frac{v}{c})^2}}

Actual Screen Image

**It doesn't take a genius
to comprehend**

SCIENTIFIC
wordTM

In fact, it's created by the same scientists who brought you T³_{tm}, TCI Software Research Inc.

Scientific Word_{tm} is the latest in PC word processing for Windows 3.0. The file storage format is T_EX. It's a full document editor, not a previewer. You compose and edit directly on the screen without being forced to think in T_EX.

Your input is mathematics, and your output is T_EX.

Discover the genius when you combine the power of T_EX with the simplicity of **Scientific Word**_{tm}.

To be a part of this exciting new discovery, contact TCI Software Research Inc. Call today, toll free **1-800-874-2383**, and ask to be put on the list for this summer's special **Scientific Word**_{tm} introductory offer.

TCI
SOFTWARE RESEARCH, INC.

1190 FOSTER ROAD
LAS CRUCES, NM
88001

TEL:(505)522-4600
FAX:(505)522-0116
TELEX: 317629

T³ and Scientific Word are trademarks of TCI Software Research Inc. T_EX is a trademark of the American Mathematical Society. Windows is a trademark of Microsoft.

TYPESETTING: JUST \$2.50 PER PAGE!

Send us your T_EX DVI files and we will typeset your material at 2000 dpi on quality photographic paper — \$2.50 per page!

Choose from these available fonts: Computer Modern, Bitstream Fontware™, and any METAFONT fonts. (For each METAFONT font used other than Computer Modern, \$15 setup is charged. This ad was composed with PCTEX® and Bitstream Dutch (Times Roman) fonts, and printed on RC paper at 2000 dpi with the Chelgraph IBX typesetter.)

And the good news is: just \$2.50 per page, \$2.25 each for 100+ pages, \$2.00 each for 500+ pages! Laser proofs \$.50 per page. (\$25 minimum on all jobs.)

Call or write today for complete information, sample prints, and our order form. **TYPE 2000, 16 Madrona Avenue, Mill Valley, CA 94941. Phone 415/388-8873.**

— TYPE —
2000

VECTOR TEX

B
T
w w
g g
C & H
E A
M

TEX FOR THE 90'S

**Are you still
struggling with
PXL's, PK's or GF's?
Move on to scalable
fonts:**

- Save megabytes of storage—entire VT_EX fits on one floppy.
- Instantly generate any font in any size and in any variation from 5 to 100 points.
- Standard font effects include compression, slant, smallcaps, outline, shading and shadow.
New: landscape. New: scalable graphics.
- Discover the universe of MicroPress Font Library professional typefaces: not available from any other T_EX vendor.

List price \$299

Includes the VT_EX typesetter (superset of T_EX), 10 scalable typefaces, VVIEW (arbitrary magnification on EGA, CGA, VGA, Hercules, AT&T), VLASER (HP LaserJet), VPOST (PostScript), VDOT (Epson, Panasonic, NEC, Toshiba, Proprinter, Star, DeskJet) and manuals.

S/H add \$5. COD add \$5.

WordPerfect Interface add \$100. Site licenses available.

Dealers' inquiries welcome. Professional typefaces available for older implementations of T_EX.



MicroPress Inc.

68-30 Harrow Street, Forest Hills, NY 11375
Tel: (718) 575-1816 Fax: (718) 575-8038

The Joy of T_EX

A Gourmet Guide to Typesetting with the A_MS-T_EX Macro Package, Second Edition

M. D. Spivak

This is the second edition of *The Joy of T_EX*, the user-friendly guide to A_MS-T_EX, which is a software package based on the revolutionary computer typesetting language T_EX. A_MS-T_EX was designed to simplify the typesetting of mathematical quantities, equations, and displays, and to format the output according to any of various preset style specifications. This second edition of *Joy* has been updated to reflect the changes introduced in Version 2.0 of the A_MS-T_EX macro package.

The first two parts of the manual, "Starters" and "Main Courses," teach the reader how to typeset the kind of text and mathematics one ordinarily encounters. "Sauces and Pickles," the third section, treats more exotic problems and includes a 60-page dictionary on special techniques. The manual also includes descriptions of conventions of mathematical typography to help the novice technical typist. Appendices list handy summaries of frequently used and more esoteric symbols.

This manual will prove useful for technical typists as well as scientists who prepare their own manuscripts. For the novice, exercises sprinkled generously throughout each chapter encourage the reader to sit down at a terminal and learn through experience.

1980 Mathematics Subject Classifications: 00, 68
 ISBN 0-8218-2997-1, LC 90-1082
 309 pages, Softcover, 1990
 List Price \$38, AMS institutional member \$34,
 AMS individual member \$30
 To order, please specify, JOYT/T

All prices subject to change. Free shipment by surface; for air delivery, please add \$6.50 per title. *Prepayment required.* **Order from** American Mathematical Society, P. O. Box 1571, Annex Station, Providence, RI 02901-1571, or call toll free 800-321-4AMS in the continental U.S. and Canada to charge with VISA or MasterCard.

For fast answers to common T_EX questions, you need ...

T_EX for the Impatient

a practical handbook for
new and current T_EX users

by Paul W. Abrahams
with Karl Berry and Kathryn A. Hargreaves

Clear, concise, and accessible, this handy reference manual is organized for easy retrieval of information. Thoroughly indexed and carefully designed, *T_EX for the Impatient* is packed with explicit instructions, useful tips, and a wealth of witty and illuminating illustrations.

384 pages include:

- Complete descriptions of T_EX commands, arranged for lookup either by function or alphabetically
- Clear definitions of essential T_EX concepts, collected in a separate chapter so that the command descriptions remain brief and accessible
- Explanations of common error messages and advice on solving problems that frequently arise
- A collection of useful macros

T_EX for the Impatient is for busy people who have discovered the unmatched benefits of T_EX, but who lack the time to master its intricacies. It is for everyone—new user to expert—who needs a quick reference guide to T_EX.

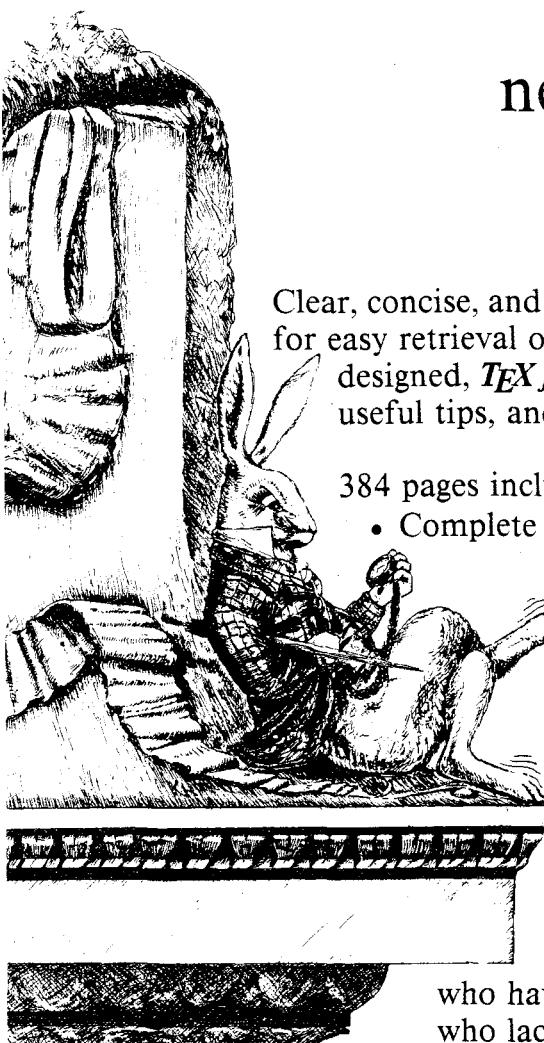
Available now from

T_EX Users Group

P. O. Box 9506
Providence, RI 02940, USA
Phone (401) 751-7760
FAX: (401) 751-1071
Mastercard/Visa, checks and money orders accepted

TUG Members . \$25
Regular price . . . \$27

Shipping in US: \$3/copy
Outside US: \$8/copy air





TM

The converter that produces resolution-independent PostScript output using outline fonts.

Now you can *really* exploit outline font technology in TeX...

A DVI-to-PS converter that uses outline fonts makes it possible to use fonts other than Computer Modern—and to use Computer Modern fonts in unusual sizes and orientations.

It is, of course, always possible to 'permanently' download outline fonts to a PostScript printer and then treat these fonts as if they were printer resident. Trouble is, typical outline fonts take over 50 kbytes of printer memory, and so only a handful can be loaded at a time. And this means that very many DVI files cannot be printed, since it is not uncommon for a DVI file to call for 30 or more fonts.

...without running out of printer memory,

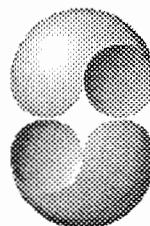
DVIPIPSONE's advanced font management techniques make it possible to use outline fonts as efficiently as one can use bitmapped fonts with older DVI-to-PS converters.

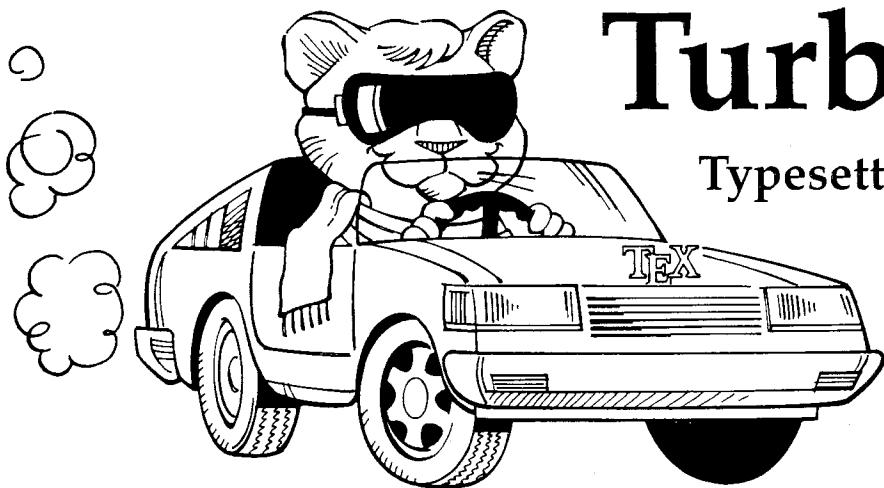
Disks no longer have to be cluttered up with those huge high-resolution bitmap font files. It is also no longer necessary to reconvert from DVI to PS format every time an output device of different resolution is used. The very same PostScript file used for proofing on a laser printer can be sent to a high resolution image setter. This greatly reduces the chance of 'surprises' when the final copy is run off.

...while enjoying considerable cost savings.

Many service bureaus will image resolution-independent PostScript files for a little over \$2 per page, while it can cost as much as four times that for someone knowledgeable about TeX to process DVI files.

DVIPIPSONE™ runs on IBM PC compatibles and can use any Adobe Type 1 outline font, including printer resident fonts, Computer Modern outline fonts, as well as any of the over 7,000 Type 1 fonts available from 30 vendors, including more than 1,000 in the Adobe Type Library alone.





TurboTEX

Typesetting Software

Executables \$150
With Source \$300

Now
Windows
Compatible!

NOW YOU CAN run the *TeX* typesetting system in the powerful and convenient graphical environment of Microsoft Windows, with the new Windows-compatible TurboTeX Release 3.1.

TurboTeX brings you the latest *TeX* 3.1 and METAFONT 2.7 standards and certifications: preloaded plain *TeX*, *LATeX*, *AMS-TeX* and *AMS-LATeX*, METAFONT, preview for EGA/VGA displays, Computer Modern and *LATeX* fonts, and printer drivers for HP LaserJet and DeskJet, PostScript, and Epson LQ and FX dot-matrix printers. This wealth of software runs on your IBM PC (MS-DOS, Windows, or OS/2), UNIX, or VAX/VMS system.

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

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

■ **Classic *TeX* for Windows.** Even if you have never used Windows on your PC, the speed and power of TurboTeX will convince you of the benefits. While the *TeX* command-

line options and *TeXbook* interaction work the same, you also can control *TeX* using friendly icons, menus, and dialog boxes. Windows protected mode frees you from MS-DOS limitations like DOS extenders, overlay swapping, and scarce memory. You can run long *TeX* formatting or printing jobs in the background while using other programs in the foreground.

■ **MS-DOS Power, Too:** TurboTeX still includes the plain MS-DOS programs. Even without expanded memory hardware, our virtual memory simulation provides the same sized *TeX* that runs on multi-megabyte mainframes, with capacity for large documents, complicated formats, and demanding macro packages.

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

■ **Availability & Requirements:** TurboTeX executables for IBM PC's include the User's Guide and require 640K, hard disk, and MS-DOS 3.0 or later. Windows extensions require Microsoft Windows 3.0. Order source code (includes Programmer's Guide) for other machines. On the PC, source compiles with Microsoft C 5.0 or later (and Windows SDK for Windows extensions), Watcom C 8.0, or Borland C++ 2.0; other op-

erating systems need a 32-bit C compiler supporting UNIX standard I/O. Media is 360K 5-1/4" or 720K 3-1/2" PC floppy disks (please specify).

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

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

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

Ordering TurboTeX

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

The Kinch Computer Company

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

AP-TEX Fonts

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

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

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

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

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

Avant Garde	Bold
Avant Garde	Bold Oblique
Avant Garde	Demiobold
Avant Garde	Demiobold Oblique
Bookman	Light
Bookman	Light Italic
Bookman	Demiobold
Bookman	Demiobold Italic
Courier	
Courier	Oblique
Courier	Bold
Courier	Bold Oblique
Helvetica	
Helvetica	Oblique
Helvetica	Bold
Helvetica	Bold Oblique
Helvetica Narrow	
Helvetica Narrow	Oblique
Helvetica Narrow	Bold
Helvetica Narrow	Bold Oblique
Schoolbook	New Century Roman
Schoolbook	New Century Italic
Schoolbook	New Century Bold
Schoolbook	New Century Bold Italic
Palatino	Roman
Palatino	Italic
Palatino	Bold
Palatino	Bold Italic
Times	Roman
Times	Italic
Times	Bold
Times	Bold Italic
Zapf Chancery	Medium Italic
Symbol	ΔΦΓΘΛΠΘ
Zapf Dingbats	❧❧❧□

For T_EX Users . . .

New Services and Prices from Computer Composition Corporation

We are pleased to announce the installation of several ***new output services*** now available to T_EX users:

1. High Resolution Laser Imaging (1200 dpi) from Postscript diskette files created on either Mac- or PC-based systems.
2. High Resolution Laser Imaging (960 dpi) from DVI magnetic tape or diskette files using a variety of typefaces in addition to the Computer Modern typeface family.
3. High quality laser page proofs at 480 dpi.
4. **NEW PRICING** for high resolution laser imaging:
 - a. From **Postscript text files** in volumes over 400 pages **\$2.00 per page**
 - b. From **Postscript text files** in volumes between 100 & 400 pages **\$2.25 per page**
 - c. From **Postscript text files** in volumes below 100 pages **\$2.40 per page**
 - d. From **DVI files** in volumes over 400 pages **\$2.15 per page**
 - e. From **DVI files** in volumes between 100 & 400 pages **\$2.30 per page**
 - f. From **DVI files** in volumes below 100 pages **\$2.45 per page**

NOTE: DEDUCT \$1.00 FROM THE ABOVE PRICES FOR HIGH QUALITY LASER PAGE PROOFS.

5. **All jobs shipped within 48 hours.**

Call or write for page samples or send us your file and we will image it on the output unit of your choice.



COMPUTER COMPOSITION CORPORATION

1401 West Girard Avenue • Madison Heights, MI 48071

(313) 545-4330 FAX (313) 544-1611

— Since 1970 —

Public Domain *TEX*

The public domain versions of *TEX* software are available from *Maria Code - Data Processing Services* by special arrangement with Stanford University and other contributing universities. The standard distribution tape contains the source of *TEX* and *METAFONT*, the macro libraries for *AMS-TEX*, *LATEX*, *SliTEX* and *HP TEX*, sample device drivers for a Versetec and LN03 printers, documentation files, and many useful tools.

Since these are in the public domain, they may be used and copied without royalty concerns. A portion of your tape cost is used to support development at Stanford University.

Compiled versions of *TEX* are available for DEC VAX/VMS, IBM CMS, IBM MVS and DEC TOPS systems. Systems using a standard format must compile *TEX* with a Pascal compiler.

TEX Order Form

TEX Distribution tapes:

- Standard ASCII format
- Standard EBCDIC format
- Special VAX/VMS format Backup
- Special DEC 20/TOPS 20 Dumper format
- Special IBM VM/CMS format
- Special IBM MVS format

Font Library Tapes (GF files)

- 300 dpi VAX/VMS format
- 300 dpi generic format
- IBM 3820/3812 MVS format
- IBM 3800 CMS format
- IBM 4250 CMS format
- IBM 3820/3812 CMS format

Tape prices: \$92.00 for first tape, \$72.00 for each additional tape. Postage: allow 2 lbs. for each tape.

Documents:

	Price \$	Weight	Quantity
<i>T<small>E</small>Xbook</i> (vol. A) softcover	30.00	2	_____
<i>T<small>E</small>X: The Program</i> (vol. B) hardcover	44.00	4	_____
<i>METAFONT</i> book (vol. C) softcover	22.00	2	_____
<i>METAFONT: The Program</i> (vol. D) hardcover ...	44.00	4	_____
Computer Modern Typefaces (vol. E) hardcover	44.00	4	_____
<i>L<small>A</small>T<small>E</small>X</i> document preparation system	30.00	2	_____
WEB language *	12.00	1	_____
<i>T<small>E</small>Xware</i> *	10.00	1	_____
<i>BibT<small>E</small>X</i> *	10.00	1	_____
Torture Test for <i>T<small>E</small>X</i> *	8.00	1	_____
Torture Test for <i>METAFONT</i> *	8.00	1	_____
<i>METAFONTware</i> *	15.00	1	_____
Metamarks *	15.00	1	_____

* published by Stanford University

Orders from within California must add sales tax for your location.

Shipping charges: domestic book rate-no charge, domestic priority mail-\$1.50/lb, air mail to Canada and Mexico-\$2.00/lb, export surface mail (all countries)-\$1.50/lb, air mail to Europe, South America-\$5.00/lb, air mail to Far East, Africa, Israel-\$7.00/lb.

Purchase orders accepted. Payment by check must be drawn on a U.S. bank.

**Send your order to: Maria Code, DP Services, 1371 Sydney Drive, Sunnyvale, CA 94087
FAX: 415-948-9388 Tel.: 408-735-8006.**

Finally . . .

... a complete \TeX solution that implements all the new \TeX 3.0 capabilities, virtual fonts, and the Extended Font standard adopted at the TUG '90 meeting in Cork.

ArborText put it all together. You don't have to!

\TeX 3.0 komplett. Die ArborText-Lösung.

ArborText a tout préparé pour vous. Profitez-en!

ArborText opracował wszystko. Wy nie musicie!

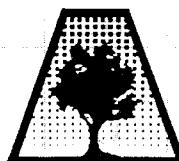
ArborText's \TeX 3.14 provides everything you need in a complete, ready-to-use package:

- Utilize the Extended \TeX Font Encoding capability with pre-built virtual fonts for Computer Modern and PostScript
- Use the conversion utilities we supply to make your own extended fonts from existing \TeX 2.0 style fonts
- Easily accent characters from your foreign language keyboard
- Create multi-language documents
- Choose from included hyphenation patterns for English, French, German, Dutch, Spanish, Portuguese, or add your own
- Use the extended version of Plain \TeX and $\text{L}\text{\AT}\text{\EX}$
- We've provided access to the New Extended Fonts directly—*macro source included!*

\TeX 3.14 and support software is available for Sun, Apollo, DEC/Risc-Ultrix, IBM PCs, HP 9000 Series 300 & 400.

*Coming Soon: \TeX 3.14 for IBM RS/6000,
Silicon Graphics, HP 9000 Series 700 & 800:*

535 W. William St. Ann Arbor, MI 48103 (313) 996-3566 FAX (313) 996-3573



ARBORTEXT INC.



Public Domain TExware

Now available through the TEx Users Group

This collection has been compiled over the past several years by Jon Radel and will continue to be updated by him in the future. It contains a wide variety of TEx and TEx-related programs including: front ends and editing tools, graphics support, drivers, fonts, style files and macros, and programming tools. It also contains *A Gentle Introduction to TEx* by Michael Doob, as well as compilations of various electronic TEx digests from around the world.

For a free catalogue contact:

TEx Users Group
P.O. Box 9506 Providence, RI 02940
Tel: (401) 751-7760 Fax: (401) 751-1071
Email: tug@math.ams.com

Prices:

US	Foreign
\$3.50 each	\$4.00 each
\$2.50 each	\$3.00 each

1-10 disks
11+ disks

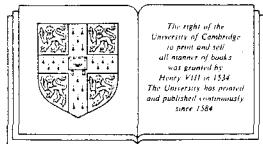
Postage & handling included

A new and unique service from the Printing Division of the Oldest Press in the World

TEx-to-TYPE

The CAMBRIDGE service that lets you and your publisher decide how your mathematical or scientific text will appear.

Monotype output in Times and Helvetica as well as a complete range of Computer Modern faces from your TEx keystrokes



For details contact

TECHNICAL APPLICATIONS GROUP CAMBRIDGE UNIVERSITY PRESS
UNIVERSITY PRINTING HOUSE SHAFESBURY ROAD CAMBRIDGE CB2 2BS ENGLAND

TELEPHONE (0223) 325070

VALUABLE ADDITIONS TO YOUR \TeX TOOLBOX

CAPTURE

Capture graphics generated by CAD, circuit design, data plotters, and other application programs that support the LaserJet. Make LaserJet images compatible with \TeX . Create pk files from pcl or pcx files.

\$115.00

texpic

With texpic graphics package, you have the tools to integrate simple graphics—boxes, circles, ellipses, lines, arrows—into your \TeX documents. Maintains output device independence.

\$79.00

Voyager

Macros to produce viewgraphs quickly and easily using \TeX . They provide format, indentation, font, and spacing control. Macros included to produce vertical and horizontal bar charts.

\$25.00



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

SPRINGER FOR \TeX

A Beginner's Book of \TeX

Raymond Seroul, Universite Louis Pasteur, Strasbourg, France;
Silvio Levy, Princeton, New Jersey

Many other \TeX books are simply manuals. This book is a friendly introduction to \TeX , the powerful typesetting system by Don Knuth. It is addressed primarily to beginners, but it contains much information that will be useful to aspiring \TeX wizards. Moreover, the authors kept firmly in mind the diversity of backgrounds that characterizes \TeX users: authors in the sciences and in the humanities, secretaries, and technical typists.

The book contains a careful explanation of all the fundamental concepts and commands, but also a wealth of commented examples and "tricks" based on the authors' long experience with \TeX . The attentive reader will quickly be able to create a table, or customize the appearance of the page, or code even the most complicated formula. The last third of the book is devoted to a Dictionary/Index, summarizing all the material the text and going into greater depth in many areas.

1991/283 pp./Softcover \$29.95/ISBN 0-387-97562-4

- Chapter 1: What is \TeX ?*
- Chapter 2: The characters of \TeX*
- Chapter 3: Groups and Modes*
- Chapter 4: The Fonts \TeX uses*
- Chapter 5: Spacing, glue and springs*
- Chapter 6: Paragraphs*
- Chapter 7: Page Layout*
- Chapter 8: Boxes*
- Chapter 9: Alignments*
- Chapter 10: Tabbing*
- Chapter 11: Typesetting Mathematics*
- Chapter 12: \TeX Programming*
- Chapter 13: Dictionary and Index*

Buy at your local technical bookstore.
To order, call 1-800-SPRINGER or
201-348-4033 in NJ. Please reference S838
to expedite your order. Or mail this ad to:

Springer-Verlag New York, Inc.

175 Fifth Avenue
New York, NY 10010
Attn.: John DiStefano
Add \$2.50 for shipping (CA, NJ & NY residents
please add sales tax.)

Name _____

Address _____

S838

City/State/Zip _____

491



Springer-Verlag

New York • Berlin • Heidelberg • Vienna • London • Paris • Tokyo • Hong Kong • Barcelona

Index of Advertisers

334, 339	American Mathematical Society	342, 343	Kinch Computer Company
346	ArborText	328, 338	MicroPress, Inc.
Cover 3	Blue Sky Research	348	Micro Programs, Inc.
347	Cambridge University Press	335	Personal \TeX Inc.
344	Computer Composition	348	Springer-Verlag
345	DP Services	336	TCI Software
334	IDS (Interactive Document Systems)	329, 330, 340, 347	\TeX Users Group
332, 333	K-Talk Communications	337	Type 2000
		341	Y&Y

Bugs in Computers & Typesetting

26 March 1991

This is a list of corrections made to *Computers & Typesetting*, Volumes A–E, since the last publication of the Errata and Changes list (21 September 1990). Corrections to the softcover version of *The TeXbook* are the same as corrections to Volume A; corrections to the softcover version of *The METAFONTbook* are the same as corrections to Volume C.

Page A137, lines 2 and 3 from the bottom	(11/9/90)
--	-----------

*and you shouldn't even be reading this manual,
which is undoubtedly all English to you.*

Page A141, line 15 from the bottom	(10/18/90)
------------------------------------	------------

Thus if you type ‘\$1\over2\$’ (in a text) you get $\frac{1}{2}$, namely style *S* over style *S'*;

Page A377, the bottom 14 lines	(3/26/91)
--------------------------------	-----------

ASCII (space); the macro also decides whether a space token is explicit or implicit.

```
\newif\ifspace \newif\iffunny \newif\ifexplicit
\def\stest#1{\expandafter\s\the#1! \stest}
\def\s{\funnyfalse \global\explicitfalse \futurelet\next\ss}
\def\ss{\ifcat\noexpand\next\stoken \spacetrue
      \ifx\next\stoken \let\next=\ss \else\let\next=\ssss \fi
      \else \let\next=\ssss \fi \next}
\long\def\sss#1 #2\stest{\def\next{#1}%
  \ifx\next\empty \global\explicittrue \fi
\long\def\ssss#1#2\stest{\funnytrue {\uccode`#1=``%
  \uppercase{\ifcat\noexpand#1}\noexpand``% active funny space
  \else \escapechar=\if*#1?\else*\fi
    \if#1\string#1\global\explicittrue\fi \fi}}
\long\def\ssss#1\stest{\spacefalse}
```

Page A444, lines 15–26	(3/26/91)
------------------------	-----------

14. If the current item is an Ord atom, go directly to Rule 17 unless all of the following are true: The nucleus is a symbol; the subscript and superscript are both empty; the very next item in the math list is an atom of type Ord, Op, Bin, Rel, Open, Close, or Punct; and the nucleus of the next item is a symbol whose family is the same as the family in the present Ord atom. In such cases the present symbol is marked as a text symbol. If the font information shows a ligature between this symbol and the following one, using the specified family and the current size, then insert the ligature character and continue as specified by the font; in this process, two characters may collapse into a single Ord text symbol, and/or new Ord text characters may appear. If the font information shows a kern between the current symbol and the next, insert a kern item following the current atom. As soon as an Ord atom has been fully processed for ligatures and kerns, go to Rule 17.

Page C11, replacement for second quotation at bottom of page (9/27/90)

To anyone who has lived in a modern American city (except Boston) at least one of the underlying ideas of Descartes' analytic geometry will seem ridiculously evident. Yet, as remarked, it took mathematicians all of two thousand years to arrive at this simple thing.

— ERIC TEMPLE BELL, *Mathematics: Queen and Servant of Science* (1951)

Page C262, line 15 (3/26/91)

`string base_name, base_version; base_name="plain"; base_version="2.7";`

Page C262, lines 19–21 (11/9/90)

for commonly occurring idioms. For example, 'stop "hello"' displays 'hello' on the terminal and waits until ⟨return⟩ is typed.

`def upto = step 1 until enddef; def downto = step -1 until enddef;`

Page C271, line 17 from the bottom (3/26/91)

`currentpen_path shifted (z.t_) withpen penspeck enddef;`

Page C329, line 325 (12/29/90)

which can be used to specify a nonstandard file area or directory name for the gray

Page C347, left column (9/27/90)

Bell, Eric Temple, 11.

Page C347, Brontë entry (1/29/91)

[The accent was clobbered; her name should, of course, be Brontë. Fix the entries for Dürer, Möbius, and Stravinsky in the same way.]

Page C349, left column (9/27/90)

Descartes, René, 6, 11, 19.

Page C356, right column (9/27/90)

[remove the entry for Rex Stout.]

Page C358, right column (9/27/90)

[remove the entry for Nero Wolfe.]

Changes to the Programs and Fonts

26 March 1991

TeX

Changes subsequent to errata publication, 21 September 1990:

Note: When making change 376, I forgot to delete the redundant code in module 883, and I should also have changed the name of that module. These cosmetic changes (and some changes to the comments) were made in version 3.14, in addition to the following two changes.

393. Show unprintable characters in font id's (Wayne Sullivan, Dec 1990)

@x module 63
print(s);
@y
slow_print(s);
@z
@x module 262 can now be spruced up
else begin print_esc(""); slow_print(text(p));
@y
else begin print_esc(text(p));
@z
@x and module 263 likewise
else begin print_esc(""); slow_print(text(p));
end;
@y
else print_esc(text(p));
@z

394. Avoid range check if total_pages>65535 (Eberhard Mattes, Dec 1990)

@x module 642
dvi_out(total_pages div 256); dvi_out(total_pages mod 256);@/
@y
dvi_out((total_pages div 256) mod 256); dvi_out(total_pages mod 256);@/
@z

395. The absolutely final change (to be made after my death)

@x module 2
@d banner=='This is TeX, Version 3.14' {printed when \TeX\ starts}
@y
@d banner=='This is TeX, Version \$\\pi\$' {printed when \TeX\ starts}
@z

My last will and testament for TeX is that no further changes be made under any circumstances. Improved systems should not be called simply 'TeX'; that name, unqualified, should refer only to the program for which I have taken personal responsibility. -- Don Knuth

METAFONT

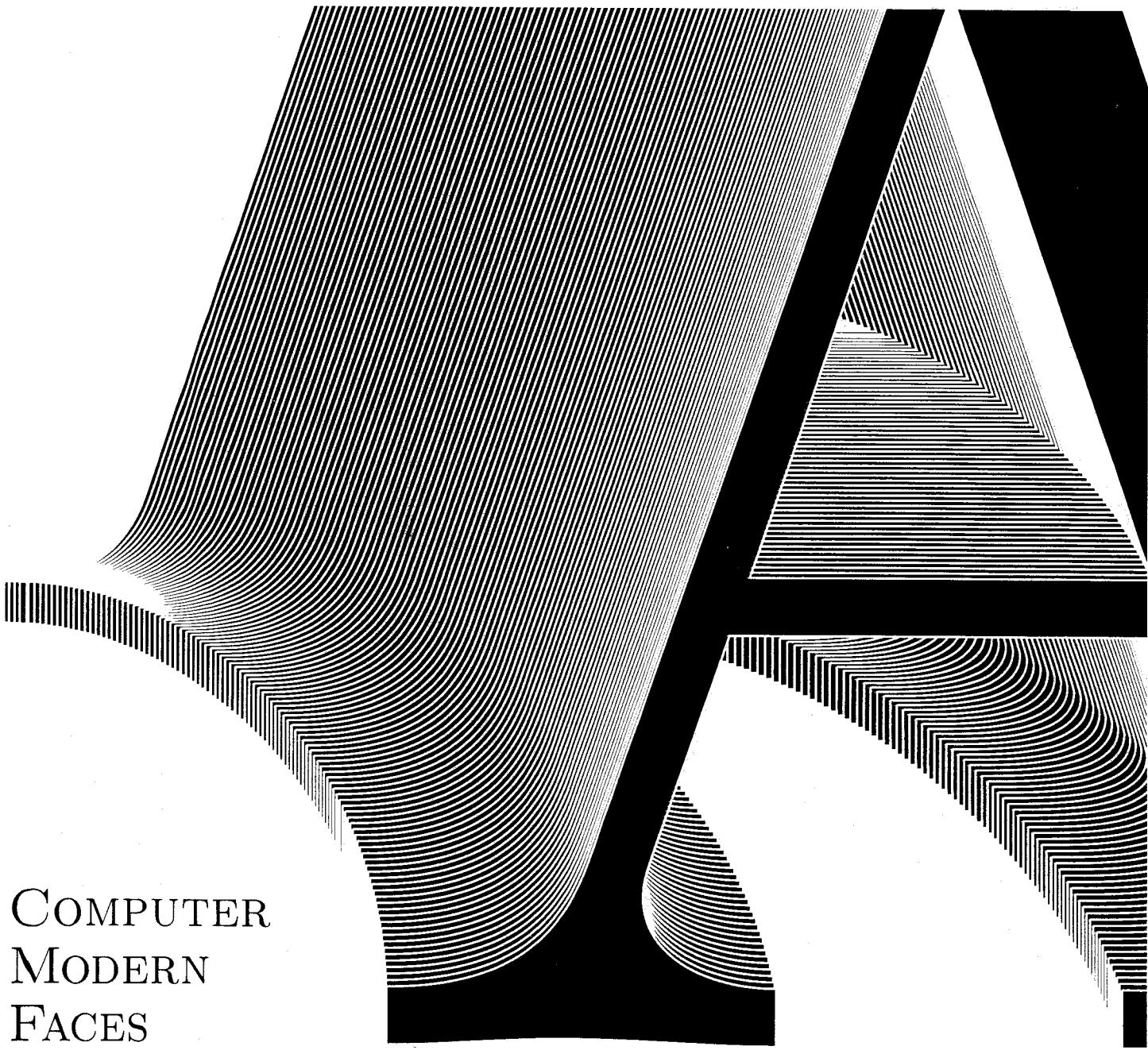
Changes since 21 September 1990.

557. The absolutely final change (to be made after my death)
@x module 2
@d banner==’This is METAFONT, Version 2.7’ {printed when \MF\ starts}
@y
@d banner==’This is METAFONT, Version \$e\$’ {printed when \MF\ starts}
@z
My last will and testament for METAFONT is that no further changes be made
under any circumstances. Improved systems should not be called simply
‘METAFONT’; that name, unqualified, should refer only to the program for which
I have taken personal responsibility. -- Don Knuth

Computer Modern fonts

Changes since 25 March 1990.

No current changes.



COMPUTER MODERN FACES

ADOBE
TYPE 1
POSTSCRIPT
FONTS

BLUE
SKY
RESEARCH

Forty faces of Computer Modern
designed by Donald Knuth
published in Adobe Type 1 format
compatible with
Adobe Type Manager
and all PostScript printers

\$345.00 Educational \$195.00
Macintosh or MS-DOS

Blue Sky Research
534 Southwest Third Avenue
Portland, Oregon 97204 USA
(800) 622-8398, (503) 222-9571
FAX (503) 222-1643

TUGBOAT

Volume 12, Number 2 / June 1991

	203 Addresses
General Delivery	205 President's introduction / Nelson H. F. Beebe
	208 Editorial comments / Barbara Beeton
	211 TeX in Germany / Walter Obermiller
Philology	212 Russian TeX / Basil Malyshev, Alexander Samarin and Dimitri Vulis
	215 Serbo-Croatian hyphenation: a TeX point of view / Cvetana Krstev
	224 On TeX and Greek... / Yannis Haralambous
	227 JemTeX 2.00 available for Japanese / François Jalbert
Fonts	227 Labelled diagrams in METAFONT / Alan Jeffrey
Graphics	229 X bitmaps in TeX / Reinhard Fößmeier
Output Devices	232 Report on the DVI Driver Standard / Joachim Schrod
Resources	233 Review of <i>3:16 Bible Texts Illuminated</i> / Karl Berry and Kathy Hargreaves
	235 Review of <i>LATEX for engineers and scientists</i> / Nico Poppelier
Questions	237 Just plain Q&A / Alan Hoenig
Tutorials	238 The \if, \ifx and \ifcat comparisons / David Salomon
	248 Some tools for making indexes: Part I / Lincoln Durst
	253 The structure of the TeX processor / Victor Eijkhout
Warnings	257 Initiation rites / Barbara Beeton
	259 Solution to the riddle from TUGboat 11, no. 4 / Frank Mittelbach
Macros	260 The bag of tricks / Victor Eijkhout
	261 TeX macros for producing multiple-choice tests / Don De Smet
	270 Getting \answers in TeX / Jim Hefferon
	272 Oral TeX / Victor Eijkhout
	277 An expansion power lemma / Sonja Maus
	278 Generating \n asterisks / George Russell
	279 TeX and envelopes / Dimitri Vulis
LATEX	284 The LATEX column / Jackie Damrau
	285 A comment on The LATEX column / Nico Poppelier
	286 LATEX tree drawer / Glenn L. Swonk
	290 "See also" indexing with Makeindex / Harold Thimbleby
	291 Babel, a multilingual style-option system for use with LATEX's standard document styles / Johannes Braams
Queries	302 Public domain SGML tools wanted / Jeff Lankford
	302 Reporting TeX's hyphenations / Jiří Veselý
Letters	302 Response to Victor Eijkhout / Paul Abrahams
	303 Response to Paul Abrahams / Victor Eijkhout
	303 TeX in schools: Why not? / Al Cuoco
Abstracts	305 Cahiers GUTenberg #7 and #8
News & Announcements	307 Calendar
	309 TeX91/Congres GUTenberg'91, Paris, 23–26 September 1991
	311 Desktop Publishing in astronomy and space sciences, Strasbourg, 1–3 October 1991
	311 Call for papers: EP92: International conference on electronic publishing, document manipulation, and typography, Lausanne, Switzerland, 7–10 April 1992
Late-Breaking News	313 Fixed-point glue setting: Errata / Donald Knuth
	313 Production notes / Barbara Beeton
	315 Coming next issue
TUG Business	315 TUG financial statements
	319 Institutional members
	321 TUG Bylaws
Forms	331 TUG membership application
Advertisements	348 Index of advertisers
Supplements	Computers & Typesetting: Errata and Changes, Supplement TUG Resource Directory