

Although large letters are the most legible when read singly, they by no means favour quick reading or quick comprehension when used for consecutive prose[.]

Sir Cyril Burt

A Psychological Study of Typography
(1959)

TUGBOAT

COMMUNICATIONS OF THE **TEX** USERS GROUP

EDITOR BARBARA BEETON

VOLUME 11, NUMBER 1

PROVIDENCE

RHODE ISLAND

APRIL 1990

U.S.A.

TUGboat

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

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

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

Submitting Items for Publication

The deadline for submitting items for Vol. 11, No. 2, is April 10, 1989; the issue will be mailed in July. (Deadlines for future issues are listed in the Calendar, page 122.)

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 Charlotte Laurendeau at the TUG office.

***TUGboat* Editorial Committee**

Barbara Beeton, *Editor*

Ron Whitney, *Production Assistant*

Helmut Jürgensen, *Associate Editor, Software*

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

Don Hosek, *Associate Editor, Output Devices*

Jackie Damrau, *Associate Editor, IATEX*

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

See page 3 for addresses.

Other TUG Publications

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

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

Trademarks

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

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

APS μ5 is a trademark of Autologic, Inc.

METAFONT is a trademark of Addison-Wesley Inc.

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

PostScript is a trademark of Adobe Systems, Inc.

TeX is a trademark of the American Mathematical Society.

UNIX is a trademark of AT&T Bell Laboratories.

Addresses

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

TeX Users Group Office

Karen Butler

Charlotte Laurendeau

P. O. Box 9506
Providence, RI 02940-9506
or

653 North Main Street
Providence, RI 02904
401-751-7760
Fax: 401-751-1071
TUG@Math.AMS.com

Nelson H. F. Beebe

Center for Scientific Computing and
Department of Mathematics
South Physics Building
University of Utah
Salt Lake City, UT 84112
801-581-5254
beebe@science.utah.edu

Barbara Beeton

American Mathematical Society
P. O. Box 6248
Providence, RI 02940
401-455-4014
bnb@Math.AMS.com
TUGboat@Math.AMS.com

Johannes Braams

PTT Research Neher Laboratories
P. O. Box 421
2260 AK Leidschendam
The Netherlands
JL_Braams@pttrnl.nl

Peter Breitenlohner

Max-Planck-Institut für Physik
(Werner-Heisenberg-Institut)
Foehringer Ring 6
D-8000 München 40
Federal Republic of Germany
(089) 32 308-412
Bitnet: PEB@DMOMPI11

Nicolas Brouard

Institut National d'Etudes
Démographiques
27, rue du Commandeur
75675 Paris Cedex 14, France
33 (1) 43 20 13 45
E-mail: brouard@frined51.bitnet

Lance Carnes

c/o Personal TeX
12 Madrona Avenue
Mill Valley, CA 94941
415-388-8853

S. Bart Childs

Dept of Computer Science
Texas A & M University
College Station, TX 77843-3112
409-845-5470
bart@cssun.tamu.edu
Bitnet: Bart@TAMLSR

Malcolm Clark

Imperial College Computer Centre
Exhibition Road
London SW7 2BP, England
Janet: texline@uk.ac.ic.cc.vaxa

John M. Crawford

Computing Services Center
College of Business
Ohio State University
Columbus, OH 43210
614-292-1741
crawford-j@osu-20.ircc.ohio-state.edu
Bitnet: CRAW4D@OHSTVMA

Jackie Damrau

Superconducting Supercollider
Laboratory
Stoneridge Office Park, Suite 260
250 Beckleymeade Avenue
Dallas, TX 75237
214-708-6048
damrau@sscvx1.ssc.gov
Bitnet: damrau@sscvx1

Michael DeCorte

2300 Naudain St. "H"
Philadelphia, PA 19146
215-546-0497
mrd@sun.soe.Clarkson.edu
Bitnet: mrd@clutx

Luzia Dietsche

Rechenzentrum der Univ. Heidelberg
Im Neuenheimer Feld 293
D-6900 Heidelberg 1
Federal Republic of Germany
Bitnet: X68@HDURZ1

Lincoln K. Durst

46 Walnut Road
Barrington, RI 02806
LKD@Math.AMS.com

Allen R. Dyer

13320 Tridelphia Road
Ellicott City, MD 21043
301-243-0008 or 243-7283

Victor Eijkhout

Department of Mathematics
University of Nijmegen
Toernooiveld 5
6525 ED Nijmegen
The Netherlands
080-613169
Bitnet: U641000@HNYKUN11

Michael J. Ferguson

INRS - Télécommunications
Université du Québec
3 Place du Commerce
Verdun (H3E 1H6), Québec Canada
514-765-7834
mike@inrs-telecom.quebec.ca

Jim Fox

Academic Computing Center HG-45
University of Washington
3737 Brooklyn Ave NE
Seattle, WA 98105
206-543-4320
fox@uwavm.acs.washington.edu
Bitnet: fox7632@uwacd.c

David Fuchs

1775 Newell
Palo Alto, CA 94303
415-323-9436

Richard Furuta

Department of Computer Science
University of Maryland
College Park, MD 20742
301-454-1461
furuta@mimsy.umd.edu

Bernard Gaulle

91403 Orsay Cedex, France
Bitnet: UCIR001@FR0RS31

Regina Girouard

American Mathematical Society
P. O. Box 6248
Providence, RI 02940
401-455-4000
RMG@Math.AMS.com

Raymond E. Goucher

TeX Users Group
P. O. Box 9506
Providence, RI 02940-9506
401-751-7760
REG@Math.AMS.com

Roswitha Graham

K.T.H. Royal Institute of Technology
DAB
100 44 Stockholm, Sweden
46 (08) 7906525
uucp: roswitha@admin.kth.se

Dean Guenther

Computing Service Center
Washington State University
Pullman, WA 99164-1220
509-335-0411
Bitnet: Guenther@WSUVM1

Hope Hamilton

National Center for
Atmospheric Research
P. O. Box 3000
Boulder, CO 80307
303-497-8915
Hamilton@MM.M.UCAR.Edu

Brian Hamilton Kelly
 School of Electrical Engineering &
 Science
 Royal Military College of Science
 Shrivenham, Swindon SN6 8LA, U.K.
 +44-793-785252
 Janet: tex@uk.ac.cranfield.rmc

Yannis Haralambous
 101/11, rue Breughel
 59650 Villeneuve d'Ascq, France
 Bitnet: yannis@frcit171
 (haralamb@frcit181 after June 1)

Doug Henderson
 Blue Sky Research
 534 SW Third Ave
 Portland, OR 97204
 800-622-8398; 503-222-9571;
 TLX 9102900911

Alan Hoenig
 17 Bay Avenue
 Huntington, NY 11743
 516-385-0736

Don Hosek
 Platt Campus Center
 Harvey Mudd College
 Claremont, CA 91711
 DHosek@HMCVAX.Claremont.Edu

Patrick D. Ion
 Mathematical Reviews
 416 Fourth Street
 P. O. Box 8604
 Ann Arbor, MI 48107
 313-996-5273
 ion@Math.AMS.com

Calvin W. Jackson, Jr.
 1749 Michelorena St.
 Los Angeles, CA 90026
 818-356-6245
 CALVIN@CSVAX.Caltech.Edu

Helmut Jürgensen
 Department of Computer Science
 University of Western Ontario
 London N6A 5B7, Ontario, Canada
 519-661-3560
 Bitnet: helmut@uvovax
 uucp: helmut@julian

David Kellerman
 Northlake Software
 812 SW Washington
 Portland, OR 97205
 503-228-3383
 Fax: 503-228-5662
 uucp: uunet!nls!davek

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

David H. Kratzer
 Los Alamos National Laboratory
 P. O. Box 1663, C-10 MS B296
 Los Alamos, NM 87545
 505-667-2864
 dhk@lanl.gov

Gerard Kuiken
 Kuiken VAS-Consulting
 P. O. Box 65791
 2506EB The Hague
 The Netherlands
 Bitnet: WBAHKUI@HDETUD1

C.G. van der Laan
 Rekencentrum
 Universiteit Groningen
 WSN-gebouw Postbus 800
 9700 AV Groningen
 The Netherlands
 Bitnet: CGL@HGRRUG5

Joachim Lammarsch
 Research Center
 Universität Heidelberg
 Im Neuenheimer Feld 293
 6900 Heidelberg
 Federal Republic of Germany
 Bitnet: X92@DHDURZ1

Pierre A. MacKay
 Northwest Computer Support Group
 University of Washington
 Mail Stop DW-10
 Seattle, WA 98195
 206-543-6259; 206-545-2386
 MacKay@June.CS.Washington.edu

Frank Mittelbach
 Electronic Data Systems
 (Deutschland) GmbH
 Eisenstraße 56
 D-6090 Rüsselsheim
 Federal Republic of Germany
 Bitnet: pzf5hz@drueds2

David Ness
 803 Mill Creek Road
 Gladwyne, PA 19035
 215-649-3474

Hubert Partl
 EDP Center
 Technical University Vienna
 Wiedner Hauptstraße 8-10
 A-1040 Wien, Austria
 Bitnet: Z3000PA@AWITUW01

Mitch Pfeffer
 Suite 90
 148 Harbor View South
 Lawrence, NY 11559
 516-239-4110

Lee S. Pickrell
 Wynne-Manley Software, Inc.
 1094 Big Rock Loop
 Los Alamos, NM 87544
 PICKRELL%LSN.MFENET@CCC.NMFEC.C.GOV

Craig Platt
 Department of Math & Astronomy
 Machray Hall
 University of Manitoba
 Winnipeg R3T 2N2, Manitoba, Canada
 204-474-9832
 platt@ccm.UManitoba.CA
 Bitnet: platt@uofmcc

Nico Poppelier
 TEXnique
 Washingtondreef 153
 3564 KD Utrecht, The Netherlands
 Poppelier@nutruu53.bitnet

David Salomon
 Computer Science Department
 School of Engineering and Computer
 Science
 California State University
 18111 Nordhoff Street
 Northridge, CA 91330
 818-885-3398
 dxs@mx.csun.edu

Rainer Schöpf
 Institut für Theoretische Physik der
 Universität Heidelberg
 Philosophenweg 16
 D-6900 Heidelberg
 Federal Republic of Germany
 Bitnet: BK4@DHDURZ1

Joachim Schrod
 c/o TH Darmstadt
 Institut für Theoretische Informatik
 Alexanderstraße 10
 D-6100 Darmstadt, West Germany
 Bitnet: XITIJSCH@DDATHD21

Georgia K.M. Tobin
 The Metafoundry
 OCLC Inc., MC 485
 6565 Frantz Road
 Dublin, OH 43017
 614-764-6087

Ron Whitney
 TEX Users Group
 P. O. Box 9506
 Providence, RI 02940-9506
 TUGboat@Math.AMS.com

Michael Wichura
 Department of Statistics
 Computation Center
 University of Chicago
 5734 University Avenue
 Chicago, IL 60637
 wichura@galton.uchicago.edu

Hermann Zapf
 Seitersweg 35
 D-6100 Darmstadt
 Federal Republic of Germany

General Delivery

From the President

Nelson H.F. Beebe

I have just received *TUGboat* 10, no. 4, which is the Proceedings of TUG's Tenth Anniversary Conference; what a fine issue! And 320 pages long, too. This is the first Proceedings that has appeared as a regular issue of *TUGboat*, a practice that we hope to continue. The TUG community is an exciting one to be a part of.

It is really satisfying to see such a broad range of topics covered, from fractals to Thai typesetting, from Icelandic dictionaries to the Encyclopedia of Mexico. I'm sure Don Knuth is amused that *TeX* is even being used to write computer games.

Malcolm Clark's survey of "Olde Worlde" (European) *TeX* activities is a welcome one. The French-speaking group, GUTenberg, has published several issues of *Cahiers GUTenberg*, and will host its third annual conference in Toulouse in May, 1990. The German-speaking group, DANTE, has just released the first issue of its newsletter, *Die TEKnische Komödie*.

I'm happy to report that, under the capable editorship of Dr. A. van Roggen, the journal *IEEE Transactions on Electrical Insulation* is now set entirely with *LATeX*. The 330-page issue that I have just received is set in two-column format with, it seems, about one figure per column, and sometimes several. The formatting requirements often raise difficult questions, and the editor has kindly provided me with a long list of comments that I hope can be addressed in the *LATeX* developments discussed below.

In early January, I undertook a substantial rearrangement of the *TeX* directories on our local facility, which includes TOPS-20, VAX VMS, UNIX (several flavors), IBM PC DOS, and Apple Macintosh systems. My motivation for this was to remove variations in directory structure, so that our users can easily find files in the *TeX* tree. With the help of TOPS-20 FTP and several UNIX utilities, it has been possible to preserve exact time stamps across most of these systems, greatly easing the job of determining when files are out-of-date with respect to the master archives.

The largest impact this reorganization had was on UNIX *TeX*, and on the font directories. UNIX *TeX* has traditionally dumped *TeX*, *LATeX*, *AMS*,

TeX, and *BIBTeX* input files all into a single directory, despite the fact that the implementation has always supported environment variable search paths. After the reorganization, there are now separate subdirectories for each of these systems. Fonts had been stored in a single directory too, which was probably all right when we had only Computer Modern fonts. After the reorganization, I now have separate directories for AMS, Computer Modern, Concrete, Euler, Greek, Imagen, music, Pandora, and AP-*TeX* font families.

It seems to me that it would be worthwhile to try to develop a recommended *TeX* file tree organization that would promote standardization in the entire community. To that end, I invite site coordinators in particular to send their comments to me, and to supply me with a summary of the *TeX* file tree layouts that they are currently distributing.

Kinch Computer Company announced at the Stanford TUG meeting the availability of PostScript fonts in 300-dpi pk file format, which are tentatively being called AP-*TeX* fonts. We licensed a copy of these, and I have developed a set of style files for *TeX* and *LATeX* that make it trivial for our users to switch from Computer Modern to some other font family by means of a single *TeX* \input statement or a *LATeX* \documentstyle option. After a suitable test period, I expect that these style files can become generally available.

Note that having PostScript fonts in pk file format means that they can be used with almost any DVI driver and any output device (assuming that fonts of the required resolution are available), including screen previewers like *xvdi* in the X Window system on workstations. This solves an important problem of how to proof phototypesetter output on low-resolution devices when fonts other than Computer Modern are required.

The 29 January 1990 issue of *InfoWorld* on pages 46-47 carried an advertisement for a series of controller boards for IBM PC and Macintosh II systems that can turn 300-dpi Hewlett-Packard LaserJet Series II and Apple LaserWriter II printers into 400-dpi, 600-dpi, 800-dpi, and 1000-dpi PostScript printers that the vendor terms "plain paper typesetters". If these are found to work satisfactorily, it seems to me that this announcement represents a very important advance for the *TeX* community. I would very much like to hear from anyone who tries any of these products. I am curious about the extent to which toner granularity and paper roughness impact the apparent output resolution.

In *TUGboat* 10, no. 3, I outlined some projects that I'd like to see accomplished during my time in

TUG office. The first of these was the improvement of electronic retrieval of **TeX**ware, particularly for the many users who have electronic mail access, but no Internet FTP access. As the new books of Frey and Adams [2] and of Quarterman [3] show, electronic mail can now reach several countries in Asia, and even a few in Africa. I hope that the recent political changes in Eastern Europe open paths there as well.

I have implemented a modified version of the NETlib system (see [1]). It is tentatively called TUGlib, and is operational on a local machine, providing access to the **TeX** collections on **science.uta.utah.edu**; it is not yet available to the public. TUGlib could be moved to the home of the **TeX** master archives. Michael DeCorte at Clarkson University has independently implemented an excellent e-mail server for the Clarkson **TeX**ware collection. Each of these systems has features the other lacks, and further investigation is needed before we make a choice that will be offered to the public.

I expect to soon have the TUG address list for extraction of electronic mail addresses to facilitate constructing an automatic forwarding mechanism for TUG members, permitting mail to be sent through a standard site when you do not know the exact electronic mail address of a TUG member.

The second issue was that of trademarks, which impacts us now that Intel Corporation has claimed a trademark on the name **DVI**, which they think stands for "Digital Video Interface", but of course, **TeX** users know what it really means. We have now initiated legal negotiations with Intel about this.

The third issue was the relationship of national or language user groups to TUG. Malcolm Clark's survey cited above should be helpful in understanding the background here.

The sixth issue was the future of **LATEX**. The outstanding work of Frank Mittelbach and Rainer Schöpf reported in the Proceedings has been followed by their establishment of an electronic mailing list intended to reach a small number of interested people. This forum should provide a means where the critical issues in **LATEX** development can be discussed by geographically distributed experts and wizards.

I expect that 1990 will be an interesting year for **TeX** users, with these conferences scheduled so far: a DANTE meeting in March; the GUTenberg meeting in May; the annual TUG meeting in June at College Station, **TeXas**; an NTG-SGML meeting in August; TUG's first meeting outside North America in Cork in September; and another DANTE meeting in October. See the calendar for details.

References

- [1] Jack J. Dongarra and Eric Grosse. Distribution of mathematical software via electronic mail. *Comm. ACM*, 30(5):403–407, May 1987.
- [2] Donnalyn Frey and Rick Adams. **!%@: A Directory of Electronic Mail Addressing and Networks**. O'Reilly & Associates, Inc., 981 Chestnut Street, Newton, MA 02164, 1989. ISBN 0-937175-39-0.
- [3] John S. Quarterman. **The Matrix**. Digital Press, 1989. ISBN 1-55558-033-5.

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

From the Past President and Annual Meeting Host

Bart Childs

We have finally received numbers 3 and 4 of volume 10. They are beautiful and well done. Please don't be critical of our staff and volunteers because it is often our procrastination and the dealing with the real world that cause the delays.

Barbara has done her usual excellent job with the regular issues and Christina's job in getting the Proceedings of the Annual Conference out as number 4 was frustrating to her but a great contribution to us. We can never say thanks enough. [Editor's note: Barbara wishes to acknowledge the skillful contribution of Ron Whitney to this whole enterprise. Without his assistance, everything would take much longer and be much less well organized. Thanks, Ron!]

I was particularly pleased with Nelson Beebe's article. He addressed several areas that we should consider for the future of TUG. I think this shows the value of change in such societies. I encourage each of you to be more involved in TUG; your membership and active participation helps each and every one.

We Aggies are looking forward to hosting the Annual Meeting here in June. Ray has scheduled a Board of Directors meeting for Sunday. This should allow us to get most (hopefully all) of our business done and then use the lunch time for interaction with the membership. We have been frustrated and rightly criticized by not being available to meet with the rest of the membership.

Sunday evening we are going to have a TEXas ethnic feast. The highlight will be TEXas BBQ, TEXMEX, something to drink, Czech pastries for dessert, and Vrazel's Polka Band. The first two items of food are not 3-alarm type and should not cause anybody a problem. We plan to have some *jalapeños* available for anybody who wants to show (make your appropriate selection) their:

- fondness for hot peppers,
- virility,
- lack of taste buds, or
- lack of judgment.

The musical group plays music from *Urban Cowboy* type to ballroom. Most of it is in between and comes from the rich local heritage that is based upon immigration from western Europe, particularly Czech and German. We will have several couples of designated dancers who will dance with willing brave souls. They do a large number of specialty dances with titles like: the *Herr Schmidt* which is also known as the *Mexican Hat Dance*, the *Chicken Dance* which we will do especially for Ray Goucher, *Schottische*, *Cotton Eyed Joe*, It will be a great time for visiting as well because they use their amplifiers with discretion.

I hope to participate in at least three birds-of-a-feather sessions. The topics that I think need addressing include:

1. What should be in a TEX distribution, especially which tools, fonts, magnifications, METAFONT,
2. The driver standards committee should have an open discussion of printer limitations.
3. What have been the effects of the latest changes in TEX and what additional effects should be expected.

For those who will remain in town on Wednesday evening, we are planning to have a Dutch-treat outing at a local Country and Western dance hall which is like *Gilley's* that was immortalized in the movie *Urban Cowboy*.

College Station and Bryan are adjoining cities. The university and the hotels for the conference are at the border between the cities. We are about 90 miles northwest of Houston's Intercontinental

Airport, 110 miles from Houston's Hobby Airport, 170 miles south of Dallas-Ft. Worth airport, and 100 miles east of Austin's airport. NASA-Houston is 15 miles south of Houston's Hobby Airport. San Antonio is about 170 miles away. We are served by commuter airlines from Dallas (American and Delta) and Houston (Continental). We are on the Greyhound Bus route and Amtrak routes between Dallas and Houston. We have public transportation but it does not run with the frequency that some would like.

We will make every effort to have a wonderful conference and I hope we have a large attendance. Our weather will not be as good as Stanford's, and we may not be on the main line, but we are aggressive and happy TEXers. We feel you will have a wonderful time. The annual meeting brochure has information on how to get more info about vacations in the area, and there are many fun ones to be had. (Contact the TUG office if you have not received a copy of this brochure.)

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

Donald E. Knuth Scholarship

Frank Mittelbach was honored at the Tenth Annual Meeting, Stanford University, as the 1989 Knuth Scholarship Winner. His article "An environment for multicolumn output" (*TUGboat* 10, no. 3) was the basis for his selection. Frank has volunteered to serve on the 1990 selection committee.

We are pleased to announce the Fifth Annual "Donald E. Knuth Scholarship" competition. The award is an all-expense-paid trip to either 1990 meeting and one of the Short Courses immediately following the meeting selected. This year's meetings are at Texas A&M University (June 18-22) and the University College Cork (Ireland) (September 10-15). The competition is open to all 1990 TUG members holding support positions that are secretarial, clerical or editorial in nature.

To enter the competition, applicants should submit to the Scholarship Committee by April 30, 1990, the input file and final TEX output of a

project that displays originality, knowledge of *TeX*, and good *TeXnique*. The project may make use of a macro package, either a public one such as *LATEX* or one that has been developed locally; such a macro package should be identified clearly. Such features as sophisticated use of math mode, of macros that require more than "filling in the blanks", or creation and use of new macros will be taken as illustrations of the applicant's knowledge. Along with the *TeX* files, each applicant should submit a letter stating his/her job title, with a brief description of duties and responsibilities, and affirming that he/she will be able to attend the meeting of his/her choice, specifying meeting and course preference.

Selection of the scholarship recipient will be based on the *TeX* sample. Judging will take place April 30 - May 20, and the winner will be notified by mail after May 20.

All applications should be submitted to the Scholarship Committee at the appropriate address:
Submissions from North America:

TeX Users Group
Attn: Knuth Scholarship Competition
658 North Main St
P. O. Box 9506
Providence, RI 02940
email: TUG@math.ams.com

Submissions from other countries:

Frank Mittelbach
Electronic Data Systems GmbH
Eisenstraße 56
D-6090 Rüsselsheim
Fed Republic of Germany
email: pzf5hz@drueds2.bitnet

Coordination of Non-English Use of *TeX*

Michael J. Ferguson

As indicated in Barbara Beeton's column in *TUGboat* 10, no. 3, I have volunteered to collect hyphenation patterns for "all" languages. In fact, I have volunteered to act as a coordinator of the non-English use of *TeX*. This includes the collection of patterns, special macros, special *LATEX* styles, fonts, lists of inadequacies, and lists of things that *TeX* does better than any, or at least most, other systems. Since we are entering a new era, with *TeX*

2.993... to become *TeX* 3.0 in the near future, all of the information to be distributed will be compatible with the new version. I am also especially interested in the rationale for the various national typesetting conventions, including, of course, hyphenation. I would like to eventually create a compendium for the various languages and national groups. An interesting example of an unresolved multilingual problem is whether a phrase in a second language should be hyphenated according to the rules of a that language or of the language of the reader, most likely the first language.

The exact form of the distribution of the special language-dependent information is still to be determined. I expect to include a referral service to the principals in the various countries that have supplied the information. Along with patterns for French that are part of the current Multilingual *TeX* distribution, I have received additional patterns for Danish, Dutch, German, Icelandic, Portuguese, Russian, and Swedish. For Icelandic, Jorgen Pind also supplied a set of fonts. Both Danish and Swedish patterns should be easily adaptable to the new *TeX*. The Russian patterns assume Cyrillic fonts that are apparently still in development. It is a good beginning but we still have a long way to go.

Since the new *TeX* is fully 8-bit, the patterns should include all the national characters that are needed for typesetting. Unfortunately, the *TeX* community has not agreed on a standard way of encoding these characters. For pattern and information interchange, the currently most universal encoding in the *TeX* community is *TeX*'s backslash encoding. Thus a "é" would be encoded \e and a "À" would be \AA. Some simple macros can be written that will allow *TeX*'s \pattern primitive to accept these sequences and replace them with a single 8-bit character. To exploit the new *TeX*, we need to have all the national characters, at least for the "Latin" European languages, in (hopefully) a single integrated font. Thus we need fully 8-bit fonts as soon as possible. When this is accomplished, the need for national fonts should disappear. Along with the letter forms, an agreement on the 8-bit code is needed. I would suggest a basis for that encoding could be the ISO-Latin 1 encoding scheme. This encoding is virtually identical to the Digital (DEC) Multinational character codes. ISO also has a number of other encoding schemes for other character sets. We should give strong consideration to following these where possible.

Unfortunately, until the new fonts arrive, it will be very difficult to use the new *TeX* in anything other than the old way. However, it is possible

to test patterns for the new \TeX using the current Multilingual \TeX . Patterns compatible with Multilingual \TeX can be used, almost untouched, in the new \TeX . I have done so with the current French patterns, but needed to make a small modification to the new \TeX to do so. The basic idea of the modification is to use, internally, the 8-bit equivalent of a character such as "î" which is ee in hex, and to replace it with the explicitly accented character just before it is sent to the .dvi file. Although currently the modification allows only two letter sequences such as 'e and uses \TeX 's standard accent algorithm for the character rebuild, the idea is quite powerful and could be easily enhanced if necessary. Since I consider it really only an interim "fix" for the new \TeX , I doubt that this will be either necessary or desirable. This modification has allowed me to use the new \TeX with the current fonts. I firmly believe that it will permit us to "root out and destroy the obsolete 7-bit systems" by allowing us to work while waiting for the new fonts. The modification is quite benign and should be easily incorporated in any port.

As part of the coordination function of the non-English use of \TeX , I have also agreed to act as coordination point for the Philological Group, originally organized by Reinhard Wonnenberger, and most recently led by Dominik Wujastyk. I would hope to obtain many important suggestions from this group.

Please send contributions, and requests for the "character substitution" change file to me at

`mike@inrs-telecom.quebec.ca`

◊ Michael J. Ferguson
 INRS - Télécommunications
 Université du Québec
 3 Place du Commerce
 Verdun H3E 1H6
 Québec, Canada
`mike@inrs-telecom.quebec.ca`

Using \TeX 3 in a Multilingual Environment — Some Ideas

Peter Breitenlohner

As a physicist at a research institute I have been using \TeX (2.x) in a bilingual (german + english) environment for many years. In the past we had for each format package (plain, L \TeX , A \TeX - \TeX , ...) two versions, one with english and one with german hyphenation patterns. With the new \TeX 3 this will probably change: we will use versions of the format packages containing hyphenation patterns for both languages and the user will have to select the appropriate language (globally at the beginning of his input and maybe locally if his document contains parts from the other language).

This seems to be the right time to consider how the new multilingual \TeX 3 can be used in a way which is first of all portable, then user friendly, and finally efficient (in that order).

I strongly advocate that in the near future *all* macro packages be rewritten to support multilingual environments in a uniform way.

In the following I will try to discuss what should and what can be done for such a situation.

1. Language specific aspects of \TeX

1.1. Assigning numbers to the languages.

Recently there have been various proposals (e.g. in \TeX hax) how to assign numbers to the various languages. In my opinion it is completely unnecessary to universally assign fixed numbers to languages. Numbers should be assigned to languages by a \newlanguage macro in the same way as box register values are assigned by \newbox and should be referred to by their symbolic names. When I first proposed to Don Knuth that such a \newlanguage macro should become part of plain.tex he was very reluctant, but since then he has included a \newlanguage macro in the latest version of plain.tex.

In a multilingual environment one may or may not want to include the english patterns, and in a monolingual non-english version of \TeX the english patterns should certainly be excluded. In order not to change plain.tex one should therefore have a dummy file hyphen.tex, and the original (english) hyphen.tex could be renamed hyphen.eng (or similar).

Thus as absolute minimum one could specify the following INITEX job:

```
\input plain
\newlanguage\english \language\english
  \input hyphen.eng
\newlanguage\german \language\german
  \input hyphen.grm
\dump
```

The user can then, in his *TeX* job, select his language with the command `\language\english`.

1.2. Setting up an environment for each language. From past experience we know that more needs to be done. For german texts one would probably want a larger value for `\tolerance` than the value (200) chosen in `plain.tex` for english texts. Other languages might want to change the values for `\lefthyphenmin` and/or `\righthyphenmin` or similar.

In addition one would want to make " an active character for german and define it such that "a, "s, "ck, "ll, etc., expand to something like `\a, \ss, \discretionary{k-}{}{c}k,` `\discretionary{l-}{}{}l` and so on. (The actual definitions used for german are different, but this is the basic idea.)

1.3. Language specific texts. Macro packages designed for multiple languages should support the possibility of selecting texts, e.g., for headings, in a way that depends on the language currently selected. (I strongly advocate that all format packages should be redesigned in this sense in the near future.)

Here we find a further complication. Both german and austrian (the german language as it is used in Austria) use the same patterns for hyphenation but they use different names e.g., for the month January: Januar vs. Jänner. Technically I would like to distinguish languages using different patterns and within each language different dialects which differ in their language specific texts to a lesser or larger extent.

2. The problem

In the past such things have often been put on top of the macro package (e.g., `german.sty` on top of `latex.tex`). With *TeX* 3 and its capability to handle multiple patterns it seems natural that the basic mechanisms for language dependencies should be defined when the hyphenation patterns are digested, i.e., immediately after `plain.tex` (or `lplain.tex`) but before further macro definitions (`latex.tex`, `amstex.tex` or whatever).

There are (at least) three different aspects of the language dependencies in *TeX*:

2.1. The installation. At a particular installation (i.e. computer) there will exist for each macro package one or more `.fmt` files, each one supporting one or more languages and/or dialects. On a mainframe computer one will probably try to support all languages in one `.fmt` file; on a PC with its limited memory one might prefer a different `.fmt` file for each language and/or dialect.

2.2. The author of a macro package. The author of a macro package should define all texts through control sequences which expand to a text that depends on the language currently chosen by the user.

2.3. The user. At the start of his input the user has to specify the language of his text (among the languages and/or dialects supported by the `.fmt` file). If his documents contain parts in a different language he might want to locally change the current language.

3. Proposal for a design

First of all, the technical difference between language and dialect should be of no concern to the user or to the author of a macro package. Moreover the decisions made by the installation (choice of supported languages) should be decoupled from those made by the author (definition of language dependent texts) and by the user (selection of the current language).

3.1. The installation. Immediately after reading `plain.tex` (or `lplain.tex`) but before reading further macro definitions, the installation should define the supported languages and/or dialects. This could be done with macros `\deflanguage` (which uses `\newlanguage`) and `\defdialect` as follows:

```
% define and select \english
\deflanguage\english{eng}{eng_setup}%
{eng_reset}
\input hyphen.eng
% define and select \german
\deflanguage\german{grm}{grm_setup}%
{grm_reset}
\input hyphen.grm
% define and select \austrian
\defdialect\ austrian{...}{...}
...
```

where the first four would probably be contained in a file `language.eng`, the next six in a file `language.grm`, etc., i.e. the actual input would probably be

```
% definitions for languages
% (\deflanguage etc.)
\input language
% definitions for english
\input language.eng
% definitions for german (and austrian)
\input language.grm
...

```

The macro `\deflanguage` would first of all assign language numbers to `\english` and `\german` (via `\newlanguage`) and would define macros `\s@tenglish`, `\res@tenglish`, `\s@tgerman` and `\res@tgerman` to be invoked by `\setlanguage` in a suitable way. `\setlanguage\german` (or `\setlanguage{german}`) would invoke `\s@tgerman` which includes in its definition the `grm_setup` code which would make " an active character and add it to the `\dospecials` list (needed e.g., for `\verbatim`), would remember the current value of `\tolerance`, set a new value for `\tolerance` and set `\language=\german`. All this would remain in effect until either the current group ends or a different language is selected through `\setlanguage` which would first invoke `\res@tgerman` which includes the `grm_reset` code to cancel the effect of the `grm_setup` code. Similarly `\setlanguage\ austrian` would invoke first `\s@tgerman` and then `\s@taustrian`, set `\language=\german` and `\chardef\dialect=\austrian`.

The current values of `\language` (and `\dialect`) will be used in the expansion of the macros for language dependent texts. Here the idea is that a dialect uses the texts for its language unless a different text is specified explicitly.

Since the difference between language and dialect should be irrelevant to both user and author it must be equally possible to specify that one language uses the texts of another language as defaults. This might, e.g., be important if at some future time a dialect is promoted to a language (with its own hyphenation patterns).

The names for languages and/or dialects (as well as the special setup for each of them) should be coordinated by TUG and/or the national TeX users groups.

Note added in Proof: It would be convenient to assign unique ASCII codes in the range 128–255 to all the special characters used by various non-english languages (compare Yannis Haralambous in *TUGboat* 10, no. 3 (1989), pp. 342–345). These characters should, however, be created through suitably defined language specific active characters (such as the german ") and not as ligatures. In

addition one should use standardized virtual fonts (see Don Knuth in this issue of *TUGboat*, p. 13) to construct them from characters present in the standard Computer Modern fonts.

3.2. The user. The user simply specifies `\setlanguage\english` or `\setlanguage\ austrian` which either does everything required for the language and/or dialect desired or, if the desired language is not supported by the .fmt file, gives an error message.

3.3. The author of a macro package. First of all the author of a macro package must define all language dependent texts through control sequences. These control sequences should be defined such that their expansion (technically expansion of expandable tokens and nothing else!) yields the desired text depending on the current values of `\language` (and `\dialect`).

There is, however, a problem. Defining, e.g.,

```
\def\m@nthjan{%
  \ifnum\language=\english January\else
  \ifnum\language=\german
  \ifnum\dialect=\austrian J"anner\else
  Januar\fi\else
  \ifnum\language=\italian Gennaio\else
  ... \fi\fi\fi}
```

is no solution for several reasons. First this leads to an error if any of the language/dialect names: `\english`, `\german`, `\austrian`, `\italian`, ... is undefined (although this could be avoided somehow). Next this would require updating these definitions each time a new language is added. Finally, for an installation which uses just one language, such a definition wastes token memory (and time).

Here I would like to propose a different scheme. Somewhere in his file (probably at the beginning or end) the author of the macro package lists all the macros which need to be defined in a language dependent way (`\m@nthjan` of the example above and all the other ones) and maybe supplies default values in case no language specific values are found. To be specific let us assume all this happens in the file ‘`abcdef.tex`’. Then the author would use a special macro `\makelanguage` to define the language dependent texts:

```
\makelanguage{abcdef}%
{\m@nthjan{...}\m@nthdec{...}...}
```

which inputs a file `abcdef.eng` (specifying the english texts) if and only if english is among the supported languages, a file `abcdef.grm` (specifying the german texts) if and only if german is among

the supported languages, and so on. Through this process it would be possible to end up with macro definitions for `\m@nthjan` etc. which are tailored to the set of languages supported by the `.fmt` file.

Clearly all this requires some rather complicated macros to do the job. In addition these macros would not be terribly fast but they would be invoked only once for each set of macros with language dependent text. Here I think portability and flexibility are more important than efficiency! The suffixes `.eng` for english or `.grm` for german are supplied as an argument of `\deflanguage`. Unfortunately these suffixes should be restricted to three characters because some systems (DOS and VMS) allow only that much, otherwise the full names (e.g., `english`) would be preferable.

The precise form of the macro definitions manufactured by `\makelanguage` should be of little interest to author or user. They are somewhat analogous to the PASCAL and TEX code produced by the WEB system programs TANGLE and WEAVE. These macro definitions should, however, be optimized whenever several languages use the same text. In particular, if all languages use the same text (or there is only one language defined) the replacement text for the macro should be simply this text.

Sometimes a user wants a different text (for one or several languages) than what is supplied by a macro package with its files for language dependent texts (`abcdef.eng` and `abcdef.grm` in the example above). Without precise knowledge of the macro definitions constructed by `\makelanguage` this requires a macro `\changelanguage` which could be used, e.g., in the form

```
\changelanguage
\somemacro{\somelanguage{...}...}
```

to change the replacement texts of `\somemacro` for the languages `\somelanguage`,

4. Protection

Some of the macro definitions discussed above, in particular `\setlanguage`, certainly must be protected against expansion if they are, e.g., written to an external file. Here I would, however, propose a slight deviation from LATEX's scheme. `\setlanguage` should be defined via

```
\global\defprotect\setlanguage\setl@ng
with the definitions
```

```
\def\defprotect#1#2{\def#1{\pr@tect#1#2}}
\def\doprotect{\let\pr@tect\d@protect}
\def\noprotect{\let\pr@tect\n@protect}
\def\n@protect#1{}
```

```
\def\d@protect#1{\noexpand#1}
\noprotect % normally, no protection
```

Thus the sequence

```
{\doprotect
\immediate\write{\setlanguage}}
```

would write the string '`\setlanguage`' (not '`\setl@ng`') which would still be protected when read in and written again.

5. Summary

In the preceding I have discussed the design of a scheme to handle multiple languages in TEX 3. I have intentionally left out almost all details of how such a scheme can be implemented. (At present a preliminary version of all the required macros is being tested.) For the moment it seems more urgent to agree on a design (including the user interface) than on details of how such a design can be realized through macro definitions.

◇ Peter Breitenlohner
 Max-Planck-Institut für Physik
 München
 Bitnet: `PEB@DMOMPI11`

Software

Erratum:

The New Versions of TEX and METAFONT
TUGboat Vol. 10, No. 3

Donald E. Knuth

Editor's note: The following should replace the second full paragraph on page 326, column 1:

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

Virtual Fonts: More Fun for Grand Wizards

Donald Knuth

Many writers to *TeXhax* during the past year or so have been struggling with interfaces between differing font conventions. For example, there's been a brisk correspondence about mixing oldstyle digits with a caps-and-small-caps alphabet. Other people despair of working with fonts supplied by manufacturers like Autologic, Compugraphic, Monotype, etc.; still others are afraid to leave the limited accent capabilities of Computer Modern for fonts containing letters that are individually accented as they should be, because such fonts are not readily available in a form that existing *TeX* software understands.

There is a much better way to solve such problems than the remedies that have been proposed in *TeXhax*. This better way was first realized by David Fuchs in 1983, when he installed it in our DVI-to-APS software at Stanford (which he also developed for commercial distribution by ArborText). We used it, for example, to typeset my article on Literate Programming for *The Computer Journal*, using native Autologic fonts to match the typography of that journal.

I was expecting David's strategy to become widely known and adopted. But alas—and this has really been the only significant disappointment I've had with respect to the way *TeX* has been propagating around the world—nobody else's DVI-to-X drivers have incorporated anything resembling David's ideas, and *TeXhax* contributors have spilled gallons of electronic ink searching for answers in the wrong direction.

The right direction is obvious once you've seen it (although it wasn't obvious in 1983): All we need is a good way to specify a mapping from *TeX*'s notion of a font character to a device's capabilities for printing. Such a mapping was called a "virtual font" by the AMS speakers at the TUG meetings this past August. At that meeting I spoke briefly about the issue and voiced my hope that all dvi drivers be upgraded within a year to add a virtual font capability. Dave Rodgers of ArborText announced that his company would make their WEB routines for virtual font design freely available, and I promised to edit them into a form that would match the other programs in the standard *TeXware* distribution.

The preparation of *TeX* Version 3 and MF Version 2 has taken me much longer than expected, but at last I've been able to look closely at the

concept of virtual fonts. (The need for such fonts is indeed much greater now than it was before, because *TeX*'s new multilingual capabilities are significantly more powerful only when suitable fonts are available. Virtual fonts can easily be created to meet these needs.)

After looking closely at David Fuchs's original design, I decided to design a completely new file format that would carry his ideas further, making the virtual font mechanism completely device-independent; David's original code was very APS-specific. Furthermore I decided to extend his notions so that arbitrary dvi commands (including rules and even specials) could be part of a virtual font. The new file format I've just designed is called vf; it's easy for dvi drivers to read vf files, because vf format is similar to the pk and dvi formats they already deal with.

The result is two new system routines called VFtoVP and VPtoVF. These routines are extensions of the old ones called TFtoPL and PLtoTF; there's a property-list language called VPL that extends the ordinary PL format so that virtual fonts can be created easily.

In addition to implementing these routines, I've also tested the ideas by verifying that virtual fonts could be incorporated into Tom Rokicki's dvips system without difficulty. I wrote a C program (available from Tom) that converts Adobe afm files into virtual fonts for *TeX*; these virtual fonts include almost all the characteristics of Computer Modern text fonts (lacking only the uppercase Greek and the dotless j) and they include all the additional Adobe characters as well. These virtual fonts even include all the "composite characters" listed in the afm file, from 'Aacute' to 'zcaron'; such characters are available as ligatures. For example, to get 'Aacute' you type first 'acute' (which is character 19 = ^S in Computer Modern font layout; it could also be character 194 = Meta-B if you're using an 8-bit keyboard with the new *TeX*) followed by 'A'. Using such fonts, it's now easier for me to typeset European language texts in Times-Roman and Helvetica and Palatino than in Computer Modern! [But with less than an hour's work I could make a virtual font for Computer Modern that would do the same things; I just haven't gotten around to it yet.]

[A nice ligature scheme for dozens of European languages was just published by Haralambous in the November *TUGboat*. He uses only ASCII characters, getting Aacute with the combination <A. I could readily add his scheme to mine, by adding a

few lines to my *vpl* files. Indeed, multiple conventions can be supported simultaneously (although I don't recommend that really).]

Virtual fonts make it easy to go from *dvi* files to the font layouts of any manufacturer or font supplier. They also (I'm sorry to say) make "track kerning" easy, for people who have to resort to that oft-abused feature of lead-free type.

Furthermore, virtual fonts solve the problem of proofreading with screen fonts or with lowres laserprinter fonts, because you can have several virtual fonts sharing a common *tfm* file. Suppose, for example, that you want to typeset camera copy on an APS machine using Univers as the ultimate font, but you want to do proofreading with a screen previewer and with a laserprinter. Suppose further that you don't have Univers for your laserprinter; the closest you have is Helvetica. And suppose that you haven't even got Helvetica for your screen, but you do have cmss10. Here's what you can do: First make a virtual property list (*vpl*) file *univers-aps.vpl* that describes the high-quality font of your ultimate output. Then edit that file into *univers-laser.vpl*, which has identical font metric info but maps the characters into Helvetica; similarly, make *univers-screen.vpl*, which maps them into cmss10. Now run VPtoVF on each of the three *vpl* files. This will produce three identical *tfm* files *univers.tfm*, one of which you should

put on the directory read by *TEX*. You'll also get three distinct *vf* files called *univers.vf*, which you should put on three different directories—one directory for your DVI-to-APS software, another for your DVI-to-laserwriter software, and the third for the DVI-to-screen previewer. Voilà.

So virtual fonts are evidently quite virtuous. But what exactly are virtual fonts, detailwise? Appended to this message are excerpts from *VFtoVP.web* and *VPtoVF.web*, which give a complete definition of the *vf* and *vpl* file formats.

I fully expect that all people who have implemented *dvi* drivers will immediately see the great potential of virtual fonts, and that they will be unable to resist installing a *vf* capability into their own software during the first few months of 1990. (The idea is this: For each font specified in a *dvi* file, the software looks first in a special table to see if the font is device-resident (in which case the *tfm* file is loaded, to get the character widths); failing that, it looks for a suitable *gf* or *pk* file; failing that, it looks for a *vf* file, which may in turn lead to other actual or virtual files. The latter files should not be loaded immediately, but only on demand, because the process is recursive. Incidentally, if no resident or *gf* or *pk* or *vf* file is found, a *tfm* file should be loaded as a last resort, so that the characters can be left blank with appropriate widths.)

An Excerpt from VFtoVP.web

6. Virtual fonts. The idea behind VF files is that a general interface mechanism is needed to switch between the myriad font layouts provided by different suppliers of typesetting equipment. Without such a mechanism, people must go to great lengths writing inscrutable macros whenever they want to use typesetting conventions based on one font layout in connection with actual fonts that have another layout. This puts an extra burden on the typesetting system, interfering with the other things it needs to do (like kerning, hyphenation, and ligature formation).

These difficulties go away when we have a "virtual font," i.e., a font that exists in a logical sense but not a physical sense. A typesetting system like *TEX* can do its job without knowing where the actual characters come from; a device driver can then do its job by letting a VF file tell what actual characters correspond to the characters *TEX* imagined were present. The actual characters can be shifted and/or magnified and/or combined with other characters from many different fonts. A virtual font can even make use of characters from virtual fonts, including itself.

Virtual fonts also allow convenient character substitutions for proofreading purposes, when fonts designed for one output device are unavailable on another.

7. A VF file is organized as a stream of 8-bit bytes, using conventions borrowed from DVI and PK files. Thus, a device driver that knows about DVI and PK format will already contain most of the mechanisms necessary to process VF files. We shall assume that DVI format is

understood; the conventions in the DVI documentation (see, for example, *TEX: The Program*, part 31) are adopted here to define VF format.

A preamble appears at the beginning, followed by a sequence of character definitions, followed by a postamble. More precisely, the first byte of every VF file must be the first byte of the following “preamble command”:

pre 247 *i*[1] *k*[1] *x*[*k*] *cs*[4] *ds*[4]. Here *i* is the identification byte of VF, currently 202. The string *x* is merely a comment, usually indicating the source of the VF file. Parameters *cs* and *ds* are respectively the check sum and the design size of the virtual font; they should match the first two words in the header of the TFM file, as described below.

After the *pre* command, the preamble continues with font definitions; every font needed to specify “actual” characters in later *set_char* commands is defined here. The font definitions are exactly the same in VF files as they are in DVI files, except that the scaled size *s* is relative and the design size *d* is absolute:

fnt_def1 243 *k*[1] *c*[4] *s*[4] *d*[4] *a*[1] *l*[1] *n*[*a* + *l*]. Define font *k*, where $0 \leq k < 256$.

fnt_def2 244 *k*[2] *c*[4] *s*[4] *d*[4] *a*[1] *l*[1] *n*[*a* + *l*]. Define font *k*, where $0 \leq k < 65536$.

fnt_def3 245 *k*[3] *c*[4] *s*[4] *d*[4] *a*[1] *l*[1] *n*[*a* + *l*]. Define font *k*, where $0 \leq k < 2^{24}$.

fnt_def4 246 *k*[4] *c*[4] *s*[4] *d*[4] *a*[1] *l*[1] *n*[*a* + *l*]. Define font *k*, where $-2^{31} \leq k < 2^{31}$.

These font numbers *k* are “local”; they have no relation to font numbers defined in the DVI file that uses this virtual font. The dimension *s*, which represents the scaled size of the local font being defined, is a *fix_word* relative to the design size of the virtual font. Thus if the local font is to be used at the same size as the design size of the virtual font itself, *s* will be the integer value 2^{20} . The value of *s* must be positive and less than 2^{24} (thus less than 16 when considered as a *fix_word*). The dimension *d* is a *fix_word* in units of printer’s points; hence it is identical to the design size found in the corresponding TFM file.

define *id_byte* = 202

{ Globals in the outer block 7 } ≡

vf_file: packed file of 0 .. 255;

See also sections 10, 12, 20, 23, 26, 29, 30, 37, 42, 49, 51, 54, 67, 69, 85, 87, 111, and 123.

This code is used in section 2.

8. The preamble is followed by zero or more character packets, where each character packet begins with a byte that is < 243 . Character packets have two forms, one long and one short:

long_char 242 *pl*[4] *cc*[4] *tfm*[4] *dvi*[*pl*]. This long form specifies a virtual character in the general case.

short_char0 .. *short_char241* *pl*[1] *cc*[1] *tfm*[3] *dvi*[*pl*]. This short form specifies a virtual character in the common case when $0 \leq pl < 242$ and $0 \leq cc < 256$ and $0 \leq tfm < 2^{24}$.

Here *pl* denotes the packet length following the *tfm* value; *cc* is the character code; and *tfm* is the character width copied from the TFM file for this virtual font. There should be at most one character packet having any given *cc* code.

The *dvi* bytes are a sequence of complete DVI commands, properly nested with respect to *push* and *pop*. All DVI operations are permitted except *bop*, *eop*, and commands with opcodes ≥ 243 . Font selection commands (*fnt_num0* through *fnt4*) must refer to fonts defined in the preamble.

Dimensions that appear in the DVI instructions are analogous to *fix_word* quantities; i.e., they are integer multiples of 2^{-20} times the design size of the virtual font. For example, if the virtual font has design size 10 pt, the DVI command to move down 5 pt would be a *down* instruction with parameter 2^{19} . The virtual font itself might be used at a different size, say 12 pt; then that *down* instruction would move down 6 pt instead. Each dimension must be less than 2^{24} in absolute value.

Device drivers processing VF files treat the sequences of *dvi* bytes as subroutines or macros, implicitly enclosing them with *push* and *pop*. Each subroutine begins with $w = x = y = z = 0$, and with current font f the number of the first-defined in the preamble (undefined if there's no such font). After the *dvi* commands have been performed, the h and v position registers of DVI format and the current font f are restored to their former values; then, if the subroutine has been invoked by a *set_char* or *set* command, h is increased by the TFM width (properly scaled)—just as if a simple character had been typeset.

```

define long_char = 242 { VF command for general character packet }
define set_char_0 = 0 { DVI command to typeset character 0 and move right }
define set1 = 128 { typeset a character and move right }
define set_rule = 132 { typeset a rule and move right }
define put1 = 133 { typeset a character }
define put_rule = 137 { typeset a rule }
define nop = 138 { no operation }
define push = 141 { save the current positions }
define pop = 142 { restore previous positions }
define right1 = 143 { move right }
define w0 = 147 { move right by  $w$  }
define w1 = 148 { move right and set  $w$  }
define x0 = 152 { move right by  $x$  }
define x1 = 153 { move right and set  $x$  }
define down1 = 157 { move down }
define y0 = 161 { move down by  $y$  }
define y1 = 162 { move down and set  $y$  }
define z0 = 166 { move down by  $z$  }
define z1 = 167 { move down and set  $z$  }
define fnt_num_0 = 171 { set current font to 0 }
define fnt1 = 235 { set current font }
define xxx1 = 239 { extension to DVI primitives }
define xxx4 = 242 { potentially long extension to DVI primitives }
define fnt_def1 = 243 { define the meaning of a font number }
define pre = 247 { preamble }
define post = 248 { postamble beginning }
define improper_DVI_for_VF ≡ 139, 140, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253,
254, 255

```

9. The character packets are followed by a trivial postamble, consisting of one or more bytes all equal to *post* (248). The total number of bytes in the file should be a multiple of 4.

And Here's an Extract from VPtoVF.web

5. **Property list description of font metric data.** The idea behind VPL files is that precise details about fonts, i.e., the facts that are needed by typesetting routines like **TEX**, sometimes have to be supplied by hand. The nested property-list format provides a reasonably convenient way to do this.

A good deal of computation is necessary to parse and process a VPL file, so it would be inappropriate for **TEX** itself to do this every time it loads a font. **TEX** deals only with the compact descriptions of font metric data that appear in TFM files. Such data is so compact, however, it is almost impossible for anybody but a computer to read it.

Device drivers also need a compact way to describe mappings from **TEX**'s idea of a font to the actual characters a device can produce. They can do this conveniently when given a packed sequence of bytes called a VF file.

The purpose of VPtoVF is to convert from a human-oriented file of text to computer-oriented files of binary numbers. There's a companion program, VFtoVP, which goes the other way.

< Globals in the outer block 5 > ≡

vpl_file: text;

See also sections 21, 24, 27, 29, 31, 36, 44, 46, 47, 52, 67, 75, 77, 82, 86, 89, 91, 113, 123, 138, 143, 147, 158, 161, 167, and 175.

This code is used in section 2.

6. < Set initial values 6 > ≡

reset(vpl_file);

See also sections 22, 26, 28, 30, 32, 45, 49, 68, 80, 84, and 148.

This code is used in section 2.

7. A VPL file is like a PL file with a few extra features, so we can begin to define it by reviewing the definition of PL files. The material in the next few sections is copied from the program PLtoTF.

A PL file is a list of entries of the form

(PROPERTYNAME VALUE)

where the property name is one of a finite set of names understood by this program, and the value may itself in turn be a property list. The idea is best understood by looking at an example, so let's consider a fragment of the PL file for a hypothetical font.

```
(FAMILY NOVA)
(FACE F MIE)
(CODINGScheme ASCII)
(DESIGNSIZE D 10)
(DESIGNUNITS D 18)
(COMMENT A COMMENT IS IGNORED)
(COMMENT (EXCEPT THIS ONE ISN'T))
(COMMENT (ACTUALLY IT IS, EVEN THOUGH
          IT SAYS IT ISN'T))
(FONTDIMEN
  (SLANT R -.25)
  (SPACE D 6)
  (SHRINK D 2)
  (STRETCH D 3)
  (XHEIGHT R 10.55)
  (QUAD D 18)
  )
(LIGTABLE
  (LABEL C f)
  (LIG C f O 200)
  (SKIP D 1)
  (LABEL O 200)
  (LIG C i O 201)
  (KRN O 51 R 1.5)
  (/LIG C ? C f)
  (STOP)
  )
(CHARACTER C f
  (CHARWD D 6)
  (CHARHT R 13.5)
  (CHARIC R 1.5)
  )
```

This example says that the font whose metric information is being described belongs to the hypothetical NOVA family; its face code is medium italic extended; and the characters appear in ASCII code positions. The design size is 10 points, and all other sizes in this PL file are given in units such that 18 units equals the design size. The font is slanted with a slope of -.25 (hence the letters actually slant backward—perhaps that is why the family name is NOVA). The normal space between words is 6 units (i.e., one third of the 18-unit design size), with glue that shrinks by 2 units or stretches by 3. The letters for which accents don't need to be raised or lowered are 10.55 units high, and one em equals 18 units.

The example ligature table is a bit trickier. It specifies that the letter f followed by another f is changed to code '200, while code '200 followed by i is changed to '201; presumably codes '200 and '201 represent the ligatures 'ff' and 'ffi'. Moreover, in both cases f and '200, if the following character is the code '51 (which is a right parenthesis), an additional 1.5 units of space should be inserted before the '51. (The 'SKIP D 1' skips over one LIG or KRN command, which in this case is the second LIG; in this way two different ligature/kern programs can come together.) Finally, if either f or '200 is followed by a question mark, the question mark is replaced by f and the ligature program is started over. (Thus, the character pair 'f?' would actually become the ligature 'ff', and 'ff?' or 'f?f' would become 'fff'. To avoid this restart procedure, the /LIG command could be replaced by /LIG>; then 'f?' would become 'ff' and 'f?f' would become 'fff'.)

Character f itself is 6 units wide and 13.5 units tall, in this example. Its depth is zero (since CHARDP is not given), and its italic correction is 1.5 units.

8. The example above illustrates most of the features found in PL files. Note that some property names, like FAMILY or COMMENT, take a string as their value; this string continues until the first unmatched right parenthesis. But most property names, like DESIGNSIZE and SLANT and LABEL, take a number as their value. This number can be expressed in a variety of ways, indicated by a prefixed code; D stands for decimal, H for hexadecimal, O for octal, R for real, C for character, and F for "face." Other property names, like LIG, take two numbers as their value. And still other names, like FONTDIMEN and LIGTABLE and CHARACTER, have more complicated values that involve property lists.

A property name is supposed to be used only in an appropriate property list. For example, CHARWD shouldn't occur on the outer level or within FONTDIMEN.

The individual property-and-value pairs in a property list can appear in any order. For instance, 'SHRINK' precedes 'STRETCH' in the above example, although the TFM file always puts the stretch parameter first. One could even give the information about characters like 'f' before specifying the number of units in the design size, or before specifying the ligature and kerning table. However, the LIGTABLE itself is an exception to this rule; the individual elements of the LIGTABLE property list can be reordered only to a certain extent without changing the meaning of that table.

If property-and-value pairs are omitted, a default value is used. For example, we have already noted that the default for CHARDP is zero. The default for every numeric value is, in fact, zero, unless otherwise stated below.

If the same property name is used more than once, VPtoVF will not notice the discrepancy; it simply uses the final value given. Once again, however, the LIGTABLE is an exception to this rule; VPtoVF will complain if there is more than one label for some character. And of course many of the entries in the LIGTABLE property list have the same property name.

9. A VPL file also includes information about how to create each character, by typesetting characters from other fonts and/or by drawing lines, etc. Such information is the value of the

'MAP' property, which can be illustrated as follows:

```
(MAPFONT D 0 (FONTNAME Times-Roman))
(MAPFONT D 1 (FONTNAME Symbol))
(MAPFONT D 2 (FONTNAME cmr10)(FONTAT D 20))
(CHARACTER O 0 (MAP (SELECTFONT D 1)(SETCHAR C G)))
(CHARACTER O 76 (MAP (SETCHAR O 277)))
(CHARACTER D 197 (MAP
    (PUSH)(SETCHAR C A)(POP)
    (MOVEUP R 0.937)(MOVERIGHT R 1.5)(SETCHAR O 312)))
(CHARACTER O 200 (MAP (MOVEDOWN R 2.1)(SETRULE R 1 R 8)))
(CHARACTER O 201 (MAP
    (SPECIAL ps: /SaveGray currentgray def .5 setgray)
    (SELECTFONT D 2)(SETCHAR C A)
    (SPECIAL ps: SaveGray setgray)))
```

(These specifications appear in addition to the conventional PL information. The MAP attribute can be mixed in with other attributes like CHARWD or it can be given separately.)

In this example, the virtual font is composed of characters that can be fabricated from three actual fonts, 'Times-Roman', 'Symbol', and 'cmr10 at 20\u' (where \u is the unit size in this VPL file). Character '0' is typeset as a 'G' from the symbol font. Character '76' is typeset as character '277' from the ordinary Times font. (If no other font is selected, font number 0 is the default. If no MAP attribute is given, the default map is a character of the same number in the default font.)

Character 197 (decimal) is more interesting: First an A is typeset (in the default font Times), and this is enclosed by PUSH and POP so that the original position is restored. Then the accent character '312' is typeset, after moving up .937 units and right 1.5 units.

To typeset character '200' in this virtual font, we move down 2.1 units, then typeset a rule that is 1 unit high and 8 units wide.

Finally, to typeset character '201', we do something that requires a special ability to interpret PostScript commands; this example sets the PostScript "color" to 50% gray and typesets an 'A' from cmr10 in that color.

In general, the MAP attribute of a virtual character can be any sequence of typesetting commands that might appear in a page of a DVI file. A single character might map into an entire page.

10. But instead of relying on a hypothetical example, let's consider a complete grammar for VPL files, beginning with the (unchanged) grammatical rules for PL files. At the outer level, the following property names are valid in any PL file:

CHECKSUM (four-byte value). The value, which should be a nonnegative integer less than 2^{32} , is used to identify a particular version of a font; it should match the check sum value stored with the font itself. An explicit check sum of zero is used to bypass check sum testing. If no checksum is specified in the VPL file, VPToVF will compute the checksum that METAFONT would compute from the same data.

DESIGNSIZE (numeric value, default is 10). The value, which should be a real number in the range $1.0 \leq x < 2048$, represents the default amount by which all quantities will be scaled if the font is not loaded with an 'at' specification. For example, if one says '\font\A=cmr10 at 15pt' in TeX language, the design size in the TFM file is ignored and effectively replaced by 15 points; but if one simply says '\font\A=cmr10' the stated design size is used. This quantity is always in units of printer's points.

DESIGNUNITS (numeric value, default is 1). The value should be a positive real number; it says how many units equals the design size (or the eventual 'at' size, if the font is being scaled). For example, suppose you have a font that has been digitized with 600 pixels per em, and the design size is one em; then you could say '(DESIGNUNITS R 600)' if you wanted to give all of your measurements in units of pixels.

CODINGScheme (string value, default is ‘UNSPECIFIED’). The string should not contain parentheses, and its length must be less than 40. It identifies the correspondence between the numeric codes and font characters. (**T_EX** ignores this information, but other software programs make use of it.)

FAMILY (string value, default is ‘UNSPECIFIED’). The string should not contain parentheses, and its length must be less than 20. It identifies the name of the family to which this font belongs, e.g., ‘HELVETICA’. (**T_EX** ignores this information; but it is needed, for example, when converting DVI files to PRESS files for Xerox equipment.)

FACE (one-byte value). This number, which must lie between 0 and 255 inclusive, is a subsidiary identification of the font within its family. For example, bold italic condensed fonts might have the same family name as light roman extended fonts, differing only in their face byte. (**T_EX** ignores this information; but it is needed, for example, when converting DVI files to PRESS files for Xerox equipment.)

SEVENBITSAFEFLAG (string value, default is ‘FALSE’). The value should start with either ‘T’ (true) or ‘F’ (false). If true, character codes less than 128 cannot lead to codes of 128 or more via ligatures or charlists or extensible characters. (**T_EX82** ignores this flag, but older versions of **T_EX** would only accept TFM files that were seven-bit safe.) **V_PtoVF** computes the correct value of this flag and gives an error message only if a claimed “true” value is incorrect.

HEADER (a one-byte value followed by a four-byte value). The one-byte value should be between 18 and a maximum limit that can be raised or lowered depending on the compile-time setting of *max_header_bytes*. The four-byte value goes into the header word whose index is the one-byte value; for example, to set *header*[18] ← 1, one may write ‘(HEADER D 18 0 1)’. This notation is used for header information that is presently unnamed. (**T_EX** ignores it.)

FONTDIMEN (property list value). See below for the names allowed in this property list.

LIGTABLE (property list value). See below for the rules about this special kind of property list.

BOUNDARYCHAR (one-byte value). If this character appears in a **LIGTABLE** command, it matches “end of word” as well as itself. If no boundary character is given and no **LABEL BOUNDARYCHAR** occurs within **LIGTABLE**, word boundaries will not affect ligatures or kerning.

CHARACTER. The value is a one-byte integer followed by a property list. The integer represents the number of a character that is present in the font; the property list of a character is defined below. The default is an empty property list.

11. Numeric property list values can be given in various forms identified by a prefixed letter.

C denotes an ASCII character, which should be a standard visible character that is not a parenthesis. The numeric value will therefore be between ‘41’ and ‘176’ but not ‘50’ or ‘51’.

D denotes an unsigned decimal integer, which must be less than 2^{32} , i.e., at most ‘D 4294967295’.

F denotes a three-letter Xerox face code; the admissible codes are MRR, MIR, BRR, BIR, LRR, LIR, MRC, MIC, BRC, BIC, LRC, LIC, MRE, MIE, BRE, BIE, LRE, and LIE, denoting the integers 0 to 17, respectively.

O denotes an unsigned octal integer, which must be less than 2^{32} , i.e., at most ‘O 3777777777’.

H denotes an unsigned hexadecimal integer, which must be less than 2^{32} , i.e., at most ‘H FFFFFFFF’.

R denotes a real number in decimal notation, optionally preceded by a '+' or '-' sign, and optionally including a decimal point. The absolute value must be less than 2048.

12. The property names allowed in a FONTDIMEN property list correspond to various **TEX** parameters, each of which has a (real) numeric value. All of the parameters except SLANT are in design units. The admissible names are SLANT, SPACE, STRETCH, SHRINK, XHEIGHT, QUAD, EXTRASPACE, NUM1, NUM2, NUM3, DENOM1, DENOM2, SUP1, SUP2, SUP3, SUB1, SUB2, SUPDROP, SUBDROP, DELIM1, DELIM2, and AXISHEIGHT, for parameters 1 to 22. The alternate names DEFAULTRULETHICKNESS, BIGOPSPACING1, BIGOPSPACING2, BIGOPSPACING3, BIGOPSPACING4, and BIGOPSPACING5, may also be used for parameters 8 to 13.

The notation 'PARAMETER *n*' provides another way to specify the *n*th parameter; for example, '(PARAMETER D 1 R -.25)' is another way to specify that the SLANT is -0.25. The value of *n* must be positive and less than *max-param-words*.

13. The elements of a CHARACTER property list can be of six different types.

CHARWD (real value) denotes the character's width in design units.

CHARHT (real value) denotes the character's height in design units.

CHARDP (real value) denotes the character's depth in design units.

CHARIC (real value) denotes the character's italic correction in design units.

NEXTLARGER (one-byte value), specifies the character that follows the present one in a "charlist." The value must be the number of a character in the font, and there must be no infinite cycles of supposedly larger and larger characters.

VARCHAR (property list value), specifies an extensible character. This option and **NEXTLARGER** are mutually exclusive; i.e., they cannot both be used within the same CHARACTER list.

The elements of a VARCHAR property list are either TOP, MID, BOT or REP; the values are integers, which must be zero or the number or a character in the font. A zero value for TOP, MID, or BOT means that the corresponding piece of the extensible character is absent. A nonzero value, or a REP value of zero, denotes the character code used to make up the top, middle, bottom, or replicated piece of an extensible character.

14. A LIGTABLE property list contains elements of four kinds, specifying a program in a simple command language that **TEX** uses for ligatures and kerns. If several LIGTABLE lists appear, they are effectively concatenated into a single list.

LABEL (one-byte value) means that the program for the stated character value starts here.

The integer must be the number of a character in the font; its CHARACTER property list must not have a **NEXTLARGER** or **VARCHAR** field. At least one **LIG** or **KRN** step must follow.

LABEL BOUNDARYCHAR means that the program for beginning-of-word ligatures starts here.

LIG (two one-byte values). The instruction '(**LIG** *c r*)' means, "If the next character is *c*, then insert character *r* and possibly delete the current character and/or *c*; otherwise go on to the next instruction." Characters *r* and *c* must be present in the font. **LIG** may be immediately preceded or followed by a slash, and then immediately followed by > characters not exceeding the number of slashes. Thus there are eight possible forms:

LIG /**LIG** /**LIG**> **LIG**/ **LIG**/> /**LIG**/ /**LIG**/> /**LIG**/>>

The slashes specify retention of the left or right original character; the > signs specify passing over the result without further ligature processing.

KRN (a one-byte value and a real value). The instruction '(**KRN** *c r*)' means, "If the next character is *c*, then insert a blank space of width *r* between the current character and *c*; otherwise go on to the next instruction." The value of *r*, which is in units of the design size, is often negative. Character code *c* must exist in the font.

STOP (no value). This instruction ends a ligature/kern program. It must follow either a **LIG** or **KRN** instruction, not a **LABEL** or **STOP** or **SKIP**.

SKIP (value in the range 0 .. 127). This instruction specifies continuation of a ligature/kern program after the specified number of **LIG** or **KRN** has been skipped over. The number of subsequent **LIG** and **KRN** instructions must therefore exceed this specified amount.

15. In addition to all these possibilities, the property name **COMMENT** is allowed in any property list. Such comments are ignored.

16. So that is what **PL** files hold. In a **VPL** file additional properties are recognized; two of these are valid on the outermost level:

VTITLE (string value, default is empty). The value will be reproduced at the beginning of the **VF** file (and printed on the terminal by **VFToVP** when it examines that file).

MAPFONT. The value is a nonnegative integer followed by a property list. The integer represents an identifying number for fonts used in **MAP** attributes. The property list, which identifies the font and relative size, is defined below.

And one additional "virtual property" is valid within a **CHARACTER**:

MAP. The value is a property list consisting of typesetting commands. Default is the single command **SETCHAR** *c*, where *c* is the current character number.

17. The elements of a **MAPFONT** property list can be of the following types.

FONTNAME (string value, default is **NULL**). This is the font's identifying name.

FONTAREA (string value, default is empty). If the font appears in a nonstandard directory, according to local conventions, the directory name is given here. (This is system dependent, just as in **DVI** files.)

FONTCHECKSUM (four-byte value, default is zero). This value, which should be a nonnegative integer less than 2^{32} , can be used to check that the font being referred to matches the intended font. If nonzero, it should equal the **CHECKSUM** parameter in that font.

FONTAT (numeric value, default is the **DESIGNUNITS** of the present virtual font). This value is relative to the design units of the present virtual font, hence it will be scaled when the virtual font is magnified or reduced. It represents the value that will effectively replace the design size of the font being referred to, so that all characters will be scaled appropriately.

FONTDSIZE (numeric value, default is 10). This value is absolute, in units of printer's points. It should equal the **DESIGNSIZE** parameter in the font being referred to.

If any of the string values contain parentheses, the parentheses must be balanced. Leading blanks are removed from the strings, but trailing blanks are not.

18. Finally, the elements of a **MAP** property list are an ordered sequence of typesetting commands chosen from among the following:

SELECTFONT (four-byte integer value). The value must be the number of a previously defined **MAPFONT**. This font (or more precisely, the final font that is mapped to that code number, if two **MAPFONT** properties happen to specify the same code) will be used in subsequent **SETCHAR** instructions until overridden by another **SELECTFONT**. The first-specified **MAPFONT** is implicitly selected before the first **SELECTFONT** in every character's map.

SETCHAR (one-byte integer value). There must be a character of this number in the currently selected font. (**VPToVF** doesn't check that the character is valid, but **VFToVP** does.) That character is typeset at the current position, and the typesetter moves right by the **CHARWD** in that character's **TFM** file.

SETRULE (two real values). The first value specifies height, the second specifies width, in design units. If both height and width are positive, a rule is typeset at the current position. Then the typesetter moves right, by the specified width.

MOVERIGHT, **MOVELEFT**, **MOVEUP**, **MOVEDOWN** (real value). The typesetter moves its current position by the number of design units specified.

PUSH The current typesetter position is remembered, to be restored on a subsequent **POP**.

POP The current typesetter position is reset to where it was on the most recent unmatched **PUSH**. The **PUSH** and **POP** commands in any **MAP** must be properly nested like balanced parentheses.

SPECIAL (string value). The subsequent characters, starting with the first nonblank and ending just before the first ')' that has no matching '(', are interpreted according to local conventions with the same system-dependent meaning as a 'special' (*xxx*) command in a DVI file.

SPECIALHEX (hexadecimal string value). The subsequent nonblank characters before the next ')' must consist entirely of hexadecimal digits, and they must contain an even number of such digits. Each pair of hex digits specifies a byte, and this string of bytes is treated just as the value of a **SPECIAL**. (This convention permits arbitrary byte strings to be represented in an ordinary text file.)

- 19.** Virtual font mapping is a recursive process, like macro expansion. Thus, a **MAPFONT** might specify another virtual font, whose characters are themselves mapped to other fonts. As an example of this possibility, consider the following curious file called **recurse.vpl**, which defines a virtual font that is self-contained and self-referential:

```
(VTITLE Example of recursion)
(MAPFONT D 0 (FONTNAME recurse)(FONTAT D 2))
(CHARACTER C A (CHARWD D 1)(CHARHT D 1)(MAP (SETRULE D 1 D 1)))
(CHARACTER C B (CHARWD D 2)(CHARHT D 2)(MAP (SETCHAR C A)))
(CHARACTER C C (CHARWD D 4)(CHARHT D 4)(MAP (SETCHAR C B)))
```

The design size is 10 points (the default), hence the character A in font **recurse** is a 10×10 point black square. Character B is typeset as character A in **recurse** scaled 2000, hence it is a 20×20 point black square. And character C is typeset as character B in **recurse** scaled 2000, hence its size is 40×40 .

Users are responsible for making sure that infinite recursion doesn't happen.

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

Additional Hyphenation Patterns

Gerard Kuiken

The following set of patterns may be added to Frank Liang's standard set to overcome many of the English exceptions published in *TUGboat*, Volume 10 (1989), No. 3, pp. 337–341, and a number of other badly hyphenated words not published. The patterns generate no new bad hyphenations as far as we know. These patterns (plus Liang's) fit in the standard hyphenation pattern memory of 8000, and thus can be used with all standard implementations of *TeX*. The size of the hyphenation trie becomes 6661 with 229 ops.

A larger list of patterns which find all admissible hyphenation points of the English words in the Exception List is under development. At this time, when these patterns are used with Liang's, the size of the hyphenation trie becomes 7192 with 377 ops and this exceeds the standard hyphenation memory size. These patterns generate no bad hyphenations of which we are currently aware.

The two lists will be available at the standard *TeX* archives under the names *ushyphen.add* and *ushyphen.max*. These may be used freely for non-commercial purposes. Copyright 1990 G.D.C. Kuiken. Patterns made March 1, 1990.

Editor's note: The state of the Exception List after the following patterns have been added will be published in an issue of *TUGboat* later this year.

The list *ushyphen.add*

.driv4	.sem4ic	anti3re
.eth1y6l1	.semid6	a4peable
.eu4ler	.semip4	ar4range
.ev2	.semir4	as5ymptot
.ga4som	.sem6is4	at6tes.
.ge4ome	.semiv4	auth4tl
.he3p6a	.sph6in1	av3iou
.in5u2t	.to6pog	ba6r1onie
.kil2ni	.to2q	bbi4t
.ko6rte	.vi4car	be4vie
.le6ice	.we2bil	bid4if
.me4gal	.ree4c	bil4lab
.met4ala	a4cabl	bio5m
.mimi4cr	af6fish	bi3orb
.mis4ers	anal6ys	bio3rh
.ne6o3f	anoa4c	blan4d
.non1e2m	anti3de	blin4d
.semi5	anti3n2	blon4d

bo4tul	geo3d
brus4q	ght3we
bus6i2er	g3lead
bus6i2es	4gilish
buss4in	g4nac
but4ed.	g4nor.
but4ted	g4nores
cad5em	4go4niza
catas4	griev3
4chs.	ha4parr
chs3hu	hatch3
ciga3r	hex2a
cin4q	hipela4
cle4ar	ho6r1ic.
co6ph1o3n	h4teou
cous4ti	id4ios
cri5tie	ign4it
cro5e4co	i4jk
c2tro3me6c	im5peda
cu4ranc	infras4
4d3alone	i5nitely.
data3b	irre6v3oc
d2dib	i5tesima
de4als.	ithi4l
de2clin	janu3a
de5fin5iti	ke6ling
de4mos	ki5netic
des3ic	k3sha
de4tic	la4cie
dic5aid	lai6ness
dif5fra	l3chai
di4ren	l3chil6d
di4rer	lea4sa
4dlead	lecta6b
4dli4e	litho5g
do5word	ll1fl
drif4ta	l4lish
d3tab	lo4ges.
ea4nies	l3tea
e3chas	lthi4ly
edg3l	lue1p
eli4tis	man3u1sc
e3loa	mar3gin
en3dix	medi4c
e6pineph	medi5cin
e4rian.	medio6c
espac6i	me5gran
ethy6lene	mi3dab
eu4clid1	mil4lag
feb3ru	milli5li
fermi3o	mi6n3is.
3fich	min4ute
flag6el	m3mab
g6endre.	5maphro

5mocrat	ono4ton	radi1o6g	spi4cil	4tian.
moe4las	o3nou	r4amen	spokes5w	ti4nom
mole5c	op3ism.	ra4me3triz	sports5c	tli4er
mon4ey1l	or4tho3ni4t	ra3mou	sports5w	tot3ic
mono5ch	orth5ri	ran4has	s5quito	trai5tor
mo4no1en	o4spher	ra3or	s4sa3chu	tra5vers
moro6n5is	o6v3ian.	r3binge	ss3hat	treach5e
monos6	oxi6d2ic	re4cipr	s4sian.	tr4ial.
moth4et2	pal6mat	rec4ogn	s4tamp	5tro1leum
mou5sin	parag6ra4	rec5t6ang	s4tants	tro5fit
mshack4	par4ale	re2f1orma	star5tli	trop4is
mu4dro	param4	re4tribu	sta3ti	tropo5les
mul4ti5u	para5me	r3ial.	st5b	tropo5lis
nar4chs.	pee4v1	riv3ol	stor5ab	tropol5it
nch4est	phi213an	rk3ho	strat5ag	tsch5ie
ne5back	phi5late1	6rks.	strib5ut	ttribut5
4neski	pi4cad	ro3bot3	st5scr	turn5ar
nd3thr	pli4cab	roe4las	stupi4d	t1wh
nfin5ites	pli5nar	r3thou	styl5is	ty4pal
4nian.	1pole.	r3treu	su2perie6	u2ral.
nge5nes	poly1e	r3veil	3sync	v3ativ
ng3ho	prema5c	rz3sc	sythi4	va6guer
ng3spr	pre5neu	sales5c	swimm6	v5ereig
nk3rup	pres4pli	sales5w	ta3gon.	voice5p
n5less	pro4cess	scaf6p1er	talka5	waste3w6
nom1a61	proc5ity.	sca4tol	ta3min	waveg4
nom5eno	pro4ge	s4chitz	t6ap6ath	w3c
no5mist	3pseu4d	sci4utt	tar4rh	week3n
non5eq	pseud6od2	scy4th	tch3c	wide5sp
noni4so	pseud6of2	se1mi6t1ic	tch5ed	wo4ken
no5vemb	ptomat4	sep5temb	tch5ier	wrap5aro
ns5ceiv	p5trol	shoe5st	t1cr	x1q
ns4moo	pubes5c	side5st	tele4g	xquis3
ntre3p	quain4te	side5sw	tele1r6o	y5ched
obli2g1	qu6a3si3	sky3sc	tess4es	ym5etry
o3chas	quasir6	s4ogamy	tha4lam	y3stro
oerst4	quasis6	s04lute	tho5don	z2z3w
oke3st	quiv4ar	s4pace	tho5geni	
olon4om	r5abolic	s4pacin	thok4er	
o3mecha6	ra3chu	spe5cio	thy4lan	
o5norma	r3a3dig	spher5o	thy3sc	

◊ Gerard Kuiken
 Kuiken VAS-Consulting
 P.O. Box 65791
 2506EB The Hague
 The Netherlands
 Bitnet: WBAHKUI@HDETUD1

Fonts

Typesetting Modern Greek – An Update

Yannis Haralambous

I would like to announce that, as of March 1, version 1.1 of the reduced greek fonts and macros (cf. *TUGboat* 10, no. 3 (1989), 354–359) is available. New features include

- hyphenation patterns for modern greek following the rules mentioned in *op. cit.*,
- some refinements of the fonts,
- an italics font,
- a new version of `greekmacros.tex`, and finally
- a BONUS: an extended `logo10.mf` file for writing the METAFONT logo in greek!

This work has been done on a Mac Plus using OzTEX and MacMETAFONT. Many thanks to Peter Abbott and Anestis Antoniadis for their kind help.

The RGR PACKAGE 1.1 is (or will soon be) available at the Aston Archive (UK) and from myself on Bitnet, at `yannis@frcit171` (after June 1, at `haralamb@frcit181`). To obtain a Macintosh version (fonts for Apple *Textures*), send a 3.5" diskette ("Coupons de Réponse Internationaux" would be appreciated) to

Yannis Haralambous
101/11, rue Breughel
59650 Villeneuve d'Ascq
France

Graphics

Combining Graphics with TeX on IBM PC-Compatible Systems and LaserJet Printers

Lee S. Pickrell

Abstract

We describe a method for including graphics in TeX documents created on IBM PC computer systems with HP LaserJet printers (and compatibles). Although TeX has suffered from a perception that it does not handle graphics well, the intrinsic graphics

ability of TeX is no different than that of any other word processing system. However, two particular aspects of TeX may exacerbate the perception of a graphics limitation: TeX is implemented over a broad range of computer platforms, and TeX files are explicitly processed in two distinct stages.

We maintain that TeX has an excellent intrinsic graphics capability, which has largely been unexploited. To demonstrate the graphics capability of TeX, we have chosen the IBM PC and the HP LaserJet as a natural configuration. Indeed, this article was produced using the PC/LaserJet combination, and includes graphics plots derived from several different sources. The caption of each plot explains how the graphics image was obtained. These figures were not "cut and pasted", rather they were included electronically on the device driver level.

After considering several possible methods for acquiring graphics, printer capture is selected because the LaserJet PCL language is well standardized [1]. Our premise is that it is wasteful to regenerate an image when many application programs generate document-quality graphics. This technique also provides a larger number of graphics sources than any other we considered. This article concludes with a detailed description of a graphics solution for the PC/LaserJet combination, that forms the basis for a new software product which we are introducing. The product name is CAPTURE, and it acquired all the graphics shown here. The intent of this article is to demonstrate, by example, that TeX is well suited for graphics.

1 The Perception of a Graphics Limitation in TeX

A common perception of TeX is that it is unable to incorporate graphics into typeset documents [2, 3, 4]. This belief is unfortunate, and is not true when taken in context. The technical problem of introducing graphics in TeX is no different than the problem facing any other word processing system.

Because a graphics image can take any form, its components cannot be standardized as, e.g., types of font are standardized. The lack of a universal graphics standard forces graphics inclusion to be done on the device driver level. Therefore, in order to be printed, a graphics image must be in a format compatible with the output device. This requirement is the same regardless of the document preparation system and applies equally to TeX and to other word processing systems.

Mixing graphics with text is done regularly on many PC-based word processing programs which are not perceived to have difficulty including graphics [5,

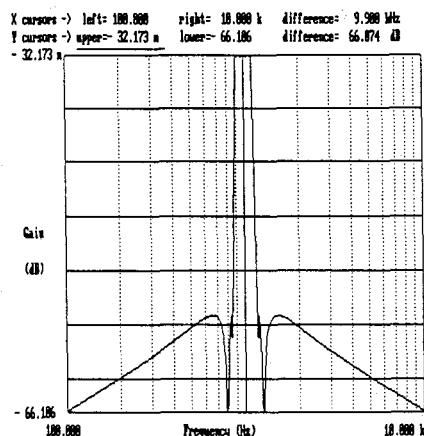


Figure 1: This is a plot from the Active Filter Design (AFD) Program by RLM Research, Boulder, Colorado.

6, 7, 8]. Conversely, the common belief is that \TeX has a graphics limitation. Curiously, the technical problem is the same in both cases. This apparent dichotomy probably has more to do with psychology than computer technology, although our analysis is speculative. The issue is worth considering because this perception affects how \TeX is used.

\TeX is unique from other word processing systems because it is implemented across a broad spectrum of computing platforms. There are now implementations of \TeX running on virtually all brands of mainframe computers and micro-computers. Conversely, most other word processing systems typically run on only one or two different computers. Take, for example, WordPerfect [8]. The original program only ran on IBM PCs, although it has recently been ported to the Macintosh. The specific graphics problem for the WordPerfect device driver only involves translating graphics images into the formats of the printers available on PCs. The available printers for the PC are quite limited in scope. Even with the addition of the Apple printer format, PostScript, the problem is still reasonable. Conversely, attempting to create a single program that would convert any graphics image to any of the printers used with \TeX is intractable.

A solution to the perceived graphics limitation of \TeX may not lie with any single piece of software or particular standard, but rather with an array of programs, each configured to the particular system on which \TeX is run. In this sense, the graphics solution for \TeX will be analogous to the development of device driver programs: a separate program for each computer/printer combination. This approach

is intrinsically no different than that for other word processor systems, the only difference is the larger scope reflected by the broader application of \TeX .

Another possible contribution to the psychology that \TeX has difficulty with graphics is the explicit separation of the device driver component. A separate device driver program emphasizes that the inclusion of graphics is not intrinsic to the \TeX program. Most other word processors use a single program which incorporates the device driver. However, functionally these programs must also have a distinct device driver component that is configured to the particular output device. Again consider the available PC-based word processors, for example, WordPerfect. During the installation the user must specify the output device [8]. This selection loads a particular code segment specific to that output device. The sections of code specific to other printers are retired. Functionally, a device driver program is chosen, although the operation is somewhat transparent to the user.

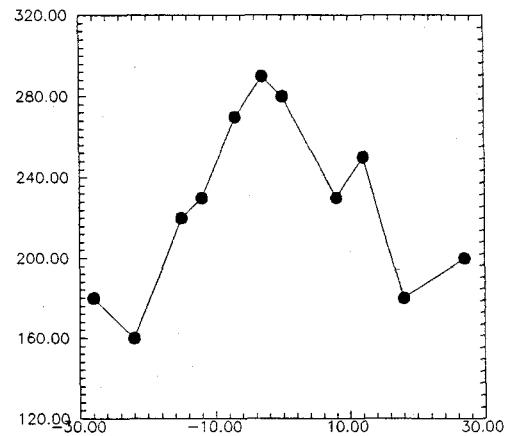


Figure 2: This is a plot from the popular scientific graphing program Grapher, by Golden Software, Inc.

2 Requirements for a General Graphics Solution for \TeX

There are two requirements for creating a general graphics solution for \TeX , and one has already been satisfied:

1. There must be a method, standardized in \TeX , which allows the \TeX program to communicate to the device driver. This communications path would enable \TeX to instruct the device driver that a graphics file should be included. This condition is satisfied by the \TeX

\special{} command. The argument of the \special{} command is passed verbatim to the device driver. When this argument is the name of a graphics file, the graphics image in the file is inserted into the output document. The procedure is functionally identical to that for any other word processor. For example, the other PC-based word processors also store the graphics images as separate files, and insert them at the device driver level. The only effective difference between TeX and these word processors is that historically it has been somewhat less convenient for TeX to communicate to the device driver.

2. There must be an array of software designed to create the graphics image files. This development would logically parallel the development of device drivers available for TeX. That is, a separate graphics program is needed for each computer/printer combination on which TeX is used.

The first condition is well satisfied by the \special{} command. Furthermore, more extensive TeX macros can be constructed which permit easy inclusion of the graphics files using the \special{} command as a root. Some of these are already available from TeX vendors. For example, Personal TeX, Inc., includes the file `setpcl.tex` with its product, which contains simple macro definitions for inserting graphics into TeX [3, 9].

We have also developed command definitions for including graphics in TeX, which are part of a new product called CAPTURE. The most primitive macro defines a command called \insertplot{}{}{} that creates an effective \hbox{} (or \mbox{} in LATEX). This box contains the graphics, and has the height and width of the image. The macro definition takes 3 parameters: the name of the graphics file, the vertical size of the graphics, and the horizontal size of the graphics. It can be treated as any other \hbox{} in TeX, (\mbox{} in LATEX). It inserts the graphics file and treats the image as a large letter of type. The definition of \insertplot{}{}{} is:

```
\def\insertplot#1#2#3{%
  \vbox to #2 true in{
    \vfill
    \hbox to #3 true in{
      {\special{pcl:#1} \hfill}
    }% End of vbox
  }%End of Definition
```

This particular macro definition works for the Personal TeX, Inc., drivers. Similar constructs have

been made for the other PC-based TeX programs. For example, to use the μ TEX or TEXPLUS drivers the "pcl:" must be replaced with "hp:" or "hp: 300 insert" respectively. Also, the \vfill and \hbox to ... may be reversed. However, the principal remains the same in all cases, an \hbox{} is created with the width and height of the graphics, and the graphics file is inserted.

We have also developed more extensive commands to be used in LATEX. These create plots that leave space for, position, caption, label, and insert graphical images. One particular macro is the \pfig{}{}{}{} command, which takes four parameters:

1. The name of the graphics file. This name is also used in a \label{} command so that the figure can be referenced by the file name using the \ref{} command.
2. The vertical size of the plot.
3. The horizontal size of the plot.
4. The text for the caption.

\pfig{}{}{}{} creates a floating block which has the dimensions of the graphics image. The plot is centered, captioned, and labeled with the file name. The plots used as examples in this article were all inserted using the \pfig{}{}{}{} command.

Using these simple command definitions, or others like them, mixing graphics with TeX is relatively simple, once the graphics image files are created. However, the problem has persisted of how to create the graphics image files to begin with.

3 The Problem of Choosing a Particular Processing Platform

The second condition which must be satisfied in order to mix graphics with TeX is the development of a broad base of graphics sources. Although a general solution to the TeX graphics limitation may require an array of programs, the proof-of-principle can be satisfied by demonstrating an effective graphics source on just one platform. Therefore, our first step was to choose the configuration we felt provided the largest opportunity for mixing graphics and TeX. This configuration is the IBM PC and the HP LaserJet printer. Our intent is not to establish the PC and LaserJet as the only configuration for including graphics in TeX, only that it is the best place to start.

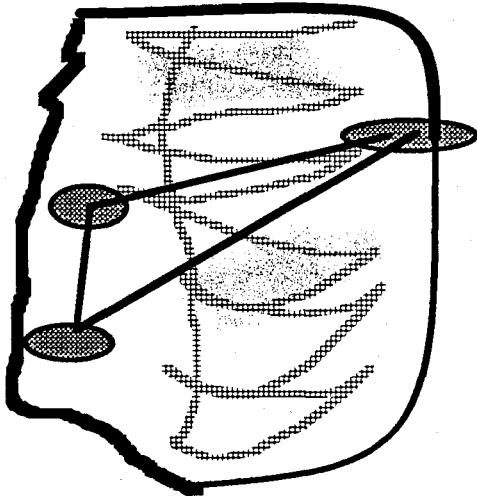


Figure 3: This is an utterly useless scribble done on PC Paintbrush. I did this just to prove I could capture the PC Paintbrush image.

3.1 Mainframe Compared to PC-Based *TEX*

The primary benefit of running *TEX* on a PC is the extensive source of graphics available on the PC, which far exceeds the possibilities from mainframes. Graphics is more common on PCs because the fast screen communications of microcomputers allow effective use of screen graphics. Graphics screens require very large communications bandwidths because each pixel (individual point of the image) is controlled by the processor. Conversely, the processor only needs to control characters to a text screen, because the local screen driver generates the fonts. Typically, the ratio of the graphics to text screen communications rate is about 500:1.

Most mainframe terminals are too slow to handle graphics well. The communications bandwidth between a terminal and a mainframe computer is typically 9600 characters per second. Conversely, the display on an IBM PC is part of the processor memory map. Communications with the screen are done at memory bus speeds, which typically run at 10 megahertz for AT class machines, or about a thousand times the speed of the mainframe terminal.

The fast screen communications allow microcomputer systems to be graphics based. Some notable examples are the Apple computers MacIntosh and Apple II, the WINDOWS operating system for DOS, and the new OS/2 operating system for IBM PCs. All of these operating systems run entirely in graphics mode. Therefore, we selected the IBM PC

as the best choice for finding an abundant source of graphics.

3.2 Selecting from the Possible MicroComputer Printers

Because closely followed standards have evolved in the personal computer industry, there are only a few microcomputer printer standards which apply to this problem. In particular, there are three classes of printer which primarily support *TEX* on a PC: an Epson compatible dot matrix printer, a PostScript compatible laser printer, and a Hewlett-Packard LaserJet compatible laser printer.

PostScript systems (Apple or IBM) do not require a separate graphics program because PostScript-based *TEX* already supports a mixture of graphics and text. In one sense, PostScript is the ideal method for including graphics in *TEX*. PostScript is a complete page description language which defines both text and graphics, so the two can be mixed easily.

Unfortunately, PostScript is expensive, and its high cost has reduced the number of available applications. Adobe Systems charges large royalties on all applications which use PostScript as an output language, and on all of the laser printer manufacturers whose printers interpret the PostScript commands. PostScript laser printers typically cost a factor of two more than equivalent LaserJet compatible printers. Although laser printer prices have dropped in recent years, this ratio has remained constant. Also, PostScript based software applications typically cost more than those which do not support the language.

As a result of the high prices of PostScript, the number of PostScript applications are far fewer than those which support the LaserJet control language (PCL) [1]. PostScript has found a niche in the desktop publishing area, but this is a single application not necessarily well suited for scientists or engineers.

Another possible configuration is the IBM PC with an Epson compatible dot matrix printer. This combination is unsuitable because the dot matrix printer cannot switch modes between text and graphics as the laser printers can. Therefore, it is impossible to mix text and graphics, and the *TEX* dot matrix printer drivers do not support the \special{} command [3].

By elimination, we selected the HP LaserJet for the output device. The choice was fortuitous because the LaserJet (and equivalents) have become commonplace in the overall PC marketplace. This trend has accelerated because the prices of laser printers have dropped in recent years to levels equiv-

alent to the high-end dot matrix printers. Due to its wide use, almost all current applications now support the LaserJet PCL language in addition to the Epson. The LaserJet control language has become a de-facto standard in the PC industry [1].

4 Sources for the Graphics Image File

The next issue to be considered is the possible sources of graphics. We have identified three general methods for creating a graphics image file.

1. The program can be a file conversion utility. This method assumes that the source of the graphics will be one of the popular drawing or paint programs. The manufacturers of these programs publish the formats for the files which store the image. For example, the format for PC Paintbrush files is the PCX format, and the format for many image scanners is the TIFF format. Therefore, in order to use an image in one of these formats with **TEX**, all that is necessary is to convert from the format of the paint program to the format required by the LaserJet. Technically, this method provides a simple solution, but the range of graphics is severely limited. Although the file formats are published, only a few manufacturers use them and they are not standard. Even though typical file conversion programs attempt to cover many formats, the range of graphics is still limited.
2. The program can convert a graphics screen to the LaserJet control language. This approach is called screen capture, and is a popular method because it allows including graphics from any program which generates a graphics screen, whether or not the program supports the LaserJet. The disadvantage is that the image quality is poor. The best standard resolution for PC monitors is the 800×600 pixels VGA mode. This mode uses 800×600 pixels to generate the image across the entire 14 inch screen. By comparison, the LaserJet resolution is 300 pixels per inch, or about 7 times better. Screen capture offers the user two possibilities: a high resolution image that *at best* is about 2.5 inches wide on the paper, or a full size but very crude low resolution image.
3. The program can capture the printer output from the application program. This method has several benefits. First, the range of graphics is very large. Any program which supports the LaserJet can provide graphics for inclusion in a document. Because the LaserJet control language is a de-facto standard, almost all applications now support it [1]. Therefore, it is

very likely that most application programs can provide graphics for **TEX** documents.

Another benefit is that the full resolution of the LaserJet is used, and the graphics can be full sized. The quality of the graphics is much higher than for the screen capture method, and the range of graphics is much larger than for the file conversion method. As an example of the range and quality of graphics possible, we have included several examples in this document obtained with **CAPTURE**. The figure captions explain the source of each of the graphics plots. We have deliberately chosen application programs that are somewhat obscure to highlight the broad range of potential graphics sources.

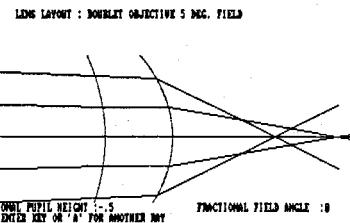


Figure 4: This is a plot of a badly aberrated, acromatic doublet lens at infinite conjugate. We used EZ-RAY, by Technical Software, Mountain View, CA., which is a ray tracing/lens design program. The screen image was converted to the LaserJet PCL language by GRAFLASR from Jewel Technologies, and captured by **CAPTURE**. It was then reduced to fit in the narrow columns of TUGboat.

5 Description of **CAPTURE**

The program which satisfies these conditions and can incorporate graphics into **TEX** documents is called **CAPTURE**. **CAPTURE** captures the output of any program which supports the LaserJet, writes the output to a file, and insures that the file can be inserted into **TEX** without disrupting the rest of the document.

The **CAPTURE** program consists of 4 files:

CPT.EXE This is the heart of the system. It captures the printer output from any application program which drives the HP LaserJet. Once the output has been captured to a disk file, it is automatically processed by the second program, **FIXPIC**.

FIXPIC.EXE This program processes the graphics file, so that it can be inserted into a **TEX** document. It removes all 28 control codes which

can disrupt the normal **T_EX** output and allows re-positioning of the graphics according to the option switches.

FIXPIC.OPT This is an options file which contains the defaults the user wishes **CAPTURE** to use. The user can create **FIXPIC** with an ordinary text editor.

PLOT.STY This is a **T_EX** style file which can be included in **T_EX** and **L^AT_EX** documents processed by any of the PC-based **T_EX** systems. It has macros which simplify the graphics insertion.

To create the graphics file for inclusion into a **T_EX** document, the command line for invoking the graphics software is prefixed with **CPT**. For example, say the user wishes to create a graphics file for **T_EX** using Z-Soft's PC Paintbrush program. Normally, the user would type:

```
C:> paint
```

The computer would run the batch file **paint.bat**, which invokes PC Paintbrush. After the picture was created, the user would print it on the LaserJet using the output utility in PC Paintbrush.

In order to create a file to include in **T_EX**, the user would do everything identically, except the command line would be prefixed with **CPT**. For example:

```
C:> cpt paint
```

After the image was created, the user would print it, just as he would otherwise. However, the output would not go to the printer, but would be written to a disk file. After the user exited PC Paintbrush, **FIXPIC** would automatically take over and sanitize the file. The sanitizing process would remove all the reset, positioning, and mode-changing commands, which would otherwise disrupt the rest of the **T_EX** document. In addition, some commands may be added to the file such as horizontal positioning. The result is a file which can be directly included into a **T_EX** document.

CAPTURE also calculates the width and vertical size of the graphics, which are printed at the end of the processing by **FIXPIC**. These numbers are used for positioning the graphics and leaving sufficient vertical space.

Using the printer capture, sanitizing, and macro definitions of the **CAPTURE** product, mixing graphics in **T_EX** documents is quite simple.

6 Conclusion

The intent of this article is not to suggest that **CAPTURE** is the final solution for graphics in **T_EX**. Rather, **CAPTURE** demonstrates a principle which can be applied more generally. Once graphics include

files are generated, mixing graphics with **T_EX** is relatively easy. Moreover, the generation of the graphics include files must be device specific. Just as separate device drivers are developed for each of the possible computer/printer combinations, so also must **CAPTURE**-like programs. We targeted **CAPTURE** for the IBM PC and the HP LaserJet because this configuration provides the largest possible spectrum of graphics sources at present and best illustrates the considerable graphics potential for **T_EX**.

References

- [1] *LaserJet series II User's Manual*. Hewlett Packard Corporation, Boise Division, P. O. Box 15, Boise, Idaho 83707, December 1986. Part No. 33440-90901.
- [2] Stephan Lindner and Lutz Birkhahn. "Towards a Complete and Comfortable **T_EX** System." *TUGboat*, 10(3):368 - 372, November 1989.
- [3] *Personal **T_EX** News*. Personal **T_EX**, Inc., 12 Madrona Ave., Mill Valley, CA 94941, Fall 1988. Volume 2, Number 2.
- [4] T. L. (Frank) Pappas. "TEXnology on the IBM PC." *Computer*, 111 - 120, August 1989. Product Reviews.
- [5] Kate Anderson. "Page-Layout Packages Boost Text Handling." *PC Week*, 6(51):63, December 1989.
- [6] Gus Venditto. "Pipeline." *PC Magazine*, 9(1):63, January 1990.
- [7] Edward Mendelson. "Two Aces and a King, The Big Three Word Processors Raise the Ante." *PC Magazine*, 8(20):97 - 120, November 1989.
- [8] *WordPerfect 5.0*. WordPerfect Corporation, 1555 N. Technology Way, Orem, Utah 84057, April 1989. ISBN 1-55692-200-0.
- [9] Michael Spivak. *PCTEX Manual*. Personal **T_EX**, Inc., 12 Madrona Ave., Mill Valley, CA 94941, November 1985.

◆ Lee S. Pickrell
Wynne-Manley Software, Inc.
1094 Big Rock Loop
Los Alamos, NM 87544
pickrell%lsn.mfenet@ccc.nmfcc.gov

Resources

Resources Available to TeX Users

Barbara Beeton and Ron Whitney

The resources available to users of TeX and its relations are many. The problem is not lack of sources, but knowing just where to look, depending on what methods of communication are available to you.

We attempt to provide here a summary of the places one can look or inquire after more information. This list is far from complete, and we welcome additions and corrections.

Archives with network connections

The main repositories of information, software, macros, and so forth, are the electronic archives that have been established at various sites around the world. We list here the archives that have been described already in *TUGboat*, or mentioned in the distributions of the various electronic mailing lists devoted to TeX arcana.

All the archives listed here support access either by anonymous FTP, by the Bitnet Listserv protocol, by a mail server, or by some combination of those facilities. When using anonymous FTP, it is usually advisable (and polite) to schedule large jobs for off hours, to avoid the heaviest network traffic. You should also choose the location nearest to you that supports the kind of access that is available to you, for the same reason. Some sites that support anonymous FTP try to keep track of their audience, and ask that you give your user name and host site (in the same form that it would be used as an e-mail address) as the password. This seems to be a good practice, so you should follow it unless some other requirement is specified.

Stanford. The canonical source of TeX is at Stanford. The Unix machine `labrea.stanford.edu` is the current site onto which new versions of TeX, METAFONT, etc., are placed when Donald Knuth makes changes. The directory `/tex` is the root node for the TeX collection. `labrea` supports anonymous FTP.

Several of the other archives check regularly and update their holdings as new versions of files appear at `labrea`. `labrea` is the primary resource for TeX implementors and the maintainers of other archives, but it should be only a secondary source

for ordinary users, as the other archives contain a greater selection of material.

Aston. The archive at Aston University, in the U.K., has been described in *TUGboat* 10, no. 1, pages 59–60, and *TUGboat* 10, no. 2, pages 194–195. Updates and changes are reported regularly in UKTeX; see below.

Clarkson. The holdings of the Clarkson archive are summarized regularly in *TUGboat*; see page 38 for the latest edition. These files can be obtained on magnetic tape as well as via the network, as described in the article.

Other sites maintain shadow archives of parts or all of the Clarkson collection. These include Texas A&M, Aston, a SPAN/DECNET depository in Italy, and a Canadian LATEX archive. See the above-mentioned article for details.

Harvey Mudd. An archive has been established at `hmvcvax.claremont.edu`, containing everything at `labrea`, plus. The relevant directories are headed by `TeX_Root`: on this VMS machine. A description of this archive is being sought for publication in a future issue.

DECUS. The DECUS TeX collection, described in *TUGboat* 10, no. 2, pages 195–196, is intended principally for distribution on magnetic tape. However, it has now been made available for anonymous FTP from `power.eee.ndsu.nodak.edu` (192.33.18.40). The root directory is `disk$ftp:[tex]`.

A vote of the DECUS membership for the Top 30 Public Domain programs for VMS was reported by Ted Nieland, compiler of the TeX collection, to the `info-vax` mailing list on 14 November 1989. Several familiar items were prominent in the list: `MAKEINDEX` (tied for 21), `SPELL` (number 9), and `TeX` (number 1). All three are included in the DECUS TeX collection.

Other archives. GUTenberg and DANTE (the French and German TeX user groups, respectively) maintain an archive at Heidelberg, accessible via `LISTSERV@HDURZ1` on Bitnet/EARN. NTG (in the Netherlands) also maintains an archive. Specific information is being sought for publication in a future issue.

Sources of software and macros for PC and Macintosh

Many, if not most, users of TeX on personal computers do not have access to any of the electronic networks, and must find distributors who provide

software and macros on diskettes. A number of vendors advertising *TeX*-related products in *TUGboat* supply them in this form. A quick glance through the ads of the most recent issues should uncover several sources.

In addition to the vendors, several individuals have packaged *TeX*, METAFONT, L^AT_EX macro collections, *TeXhax*, and other interesting material onto diskettes and offer copies of them for a nominal charge, usually no more than necessary to cover their expenses.

Jon Radel described his service in *TUGboat* 10, no. 2, page 202; he will also make available the *TEXT1* macro package from Washington State University, newly released to the public domain as described in this issue, page 54.

David Hopper will redistribute the L^AT_EX style files from Clarkson; see the article on the Clarkson archive, page 38, this issue.

For personal computer users with modems in Germany, a BBS-System called **The Insider** has message and file libraries devoted to *TeX*. This service was described in *TUGboat* 10, no. 3, pages 367-368.

Oz^{TeX}, a public-domain implementation for the Macintosh, is being distributed by a number of volunteers; see *TUGboat* 10, no. 2, pages 202-203.

Publications

The number of periodicals, formal and informal, devoted to *TeX* and related topics is growing, as is the collection of books, reports, and articles in various publications. This issue contains the beginnings of a *TeX* bibliography, starting on page 36.

TUGboat. It seems unnecessary to mention *TUGboat*, as you are reading this now, but we do so for the sake of completeness.

TeXline. The second oldest publication devoted to serving *TeX* users, after *TUGboat*, is *TeXline*, an independent publication produced by the independent-minded Malcolm Clark. A short history appeared in *TUGboat* 10, no. 2, pages 193-194.

Cahiers GUTenberg. The Cahiers GUTenberg, the official publication of GUTenberg, the French *TeX* users group, is now in its third year. See below, under Gutenberg in the section on user groups, for more information.

Die *TeXnische Komödie*. The first number (actually, No. 0) of *Die *TeXnische Komödie** has just been issued by DANTE, the users group for speakers of German. See below for more information.

Electronic discourse

TeXhax. The grand-daddy of the electronic "bulletin boards" for *TeX* is *TeXhax*. This began in 1985 and appears at irregular intervals in digest form. It is compiled at the University of Washington, with support from TUG. To subscribe from the Bitnet, send a one-line mail message to your local listserver (*LISTSERV@xxx*):

SUBSCRIBE *TEX-L* *<your name>*

From the Internet, send a similar one-line mail message to

TeXhax-request@cs.washington.edu

Janet users may choose to use

texhax-request@uk.ac.nsf

All submissions for publication in *TeXhax* should be sent to

TeXhax@cs.washington.edu

UKTeX. UKTeX, moderated by Peter Abbott, originates at Aston University. It frequently contains reports on activity in the archive there. Coverage sometimes overlaps that of *TeXhax*, but there is a great deal more, often amply demonstrating that England and the United States are "two countries separated by a common language". Send requests for subscriptions to

info-tex-request@uk.ac.aston

and submissions to

info-tex@uk.ac.aston

(From the U.S., messages from the Internet can be routed through *NSFnet-relay.ac.uk*.) If the traffic from the western side of the Atlantic increases as a result of publishing this information here, we shall try to assist Peter in finding a secondary distribution site in North America, to cut down network traffic across the ocean.

TeXMaG. *TeXMaG* is an "electronic magazine" created, edited and distributed by Don Hosek. Don described it in *TUGboat* 10, no. 2, page 192. It appears every two to four months. His address has changed since he wrote the article; it is now *DHosek@HMCVAX.Claremont.Edu*.

GUT. The GUT mailing list is a service of GUTenberg (see below). The official language is French. GUT is a two-pronged affair, with official, numbered messages sent out by the president, Bernard Gaulle, and other messages forwarded from the sender directly to the mailing list. One has to be on the mailing list in order to send messages to the list. To join the list, send a message to your local listserver as follows:

SUBSCRIBE GUT@FRULM11
 prenom NOM -ORGANISME (LIEU)-
 (This should be all on one line.)

TEX_D-L. This list is a service of DANTE, and is centered at DHDURZ1 at Heidelberg. We have not yet subscribed to this list, so cannot give any details except that it is in German. A message to your local listserver similar to the one for GUT, substituting

TEX_D-L@DHDURZ1

should put you on the list.

TEX-NL. The TeX group in the Netherlands sponsors this list, which produces some quite active and enthusiastic discussions, mostly in Dutch, of course.

TEX-NL is installed on the LISTSERV@HEARN on the Bitnet. To be added to the list, send a message there asking to subscribe.

TEX-EURO. This list was established to try to provide a focus for discussion of issues that affect all users in Europe (for example, A4 paper). A short introduction was provided in *TUGboat* 10, no. 3, pages 366-367 by Joachim Lammarsch. It is installed at the LISTSERV@DHDURZ1.

To subscribe, send the command

SUB TEX-EURO *(your name)*

to your nearest listserver or to LISTSERV@DHDURZ1.

TEX-Ed. This special-interest list, devoted to issues of TeX training, was described in *TUGboat* 10, no. 3, pages 360-361. In addition, the article gives useful information on dealing with a listserver. **TEX-ED** is installed at UICVM on the Bitnet; the archives are also available by anonymous FTP from uicvm.cc.uic.edu, the equivalent address on the Internet.

RUSTEX-L. This is another special-interest list, covering topics of interest to those who wish to use TeX for Russian and other Slavic languages. It was established by Dimitri Vulis, who described it briefly within a longer article on Russian TeX in *TUGboat* 10, no. 3, pages 332-336, especially page 336. RUSTEX-L is installed at the Bitnet node UBVM, which is the same as ubvm.cc.buffalo.edu on Internet.

Laser-Lovers. Laser-Lovers was established at about the same time as TeXhax to provide a forum on topics concerning laser and other printers suitable for use with TeX. It has evolved into a general discussion list about such printers, but still contains much of interest to TeX users looking for hardware-specific information. To subscribe, send a message to

laser-lovers-request@mimsy.umd.edu
 To submit a message to the list, send it to
 laser-lovers@mimsy.umd.edu

The moderator of Laser-Lovers is Rick Furuta.

Getting help on installing TeX systems

The TUG site coordinators are volunteers familiar with particular types of hardware and with TeX systems installed on that hardware. Very often, they have done the primary installation of TeX themselves, starting with TANGLE, creating the hardware-specific change file, and proceeding through validation by the TRIP test.

The site coordinators have agreed to answer questions related to hardware and operating systems and how these interact with TeX and related software. They are not prepared to teach the use of TeX.

The names and addresses of the TUG site coordinators are listed in the "official" addresses section at the beginning of the TUG membership list and supplements.

Reports from the site coordinators and other individuals on topics dealing with specific computer hardware are highlighted in the *TUGboat* tables of contents, either under the heading "Site reports", or more recently, "Resources", with the name of the computer shown in the left-hand column. (For example, this issue contains an article specific to IBM VM/CMS.)

Of course, if the implementation of TeX that you are using is a commercial one, you should request help from the originator of the implementation.

TeX printers. If you are looking for information on printers, the "Output devices" column in *TUGboat* identifies many printers known to be able to produce TeX output reliably, along with the sources of output driver software. This column is absent from the present issue, but is expected to return in *TUGboat* 11, no. 2.

Getting help in using TeX and METAFONT

Local specialists. Every organization installing TeX for general use should assign one or more individuals to be local specialists. These specialists should be the first source of help for users at such sites.

Courses. TUG and the regional TeX user groups offer courses in TeX, LATEX, METAFONT and related topics at varying levels. If you are expecting to

become a heavy user, attending these courses or others offered by such organizations as DECUS in their Pre-symposium Seminar program is a good idea.

Electronic sources of help. *TeXhax*, *UKTEX* and the other electronic mailing lists are read by an audience that is both knowledgeable and willing to answer questions. When posing a question to one of these lists, prepare your submission carefully, so that it is brief but complete, and give a subject heading that characterizes the problem succinctly.

Before sending an inquiry to one of the lists, check *The TeXbook* or the *LATeX* manual as appropriate, including the errata lists, to make sure you can't find the answer on your own. And if your problem is with *LATeX*, before sending out a general cry, check first with *LaTeX-help*.

LaTeX-help. This service was established by a group of volunteers who wanted to reduce the traffic on *TeXhax* of *LATeX* "beginners' questions". They have offered to answer questions about using *LATeX*, and will be keeping records about the kinds of questions being asked and building informative "packaged" answers to frequently-asked questions.

The service is described in *TUGboat* 10, no. 3, page 360.

Consultants. For really difficult problems, or very large tasks, free help will not suffice, and it may be worthwhile obtaining the services of a consultant. Consultants may register their availability with TUG; this information is published at the end of each annual membership list and supplement.

Inquiries to *TUGboat* and TUG. If you have an intractable problem and time is not of the essence, you can submit it to *TUGboat*. This, of course, will make it known to more and different people than those who read the lists; not everyone has access to network communications, though those of us who do often forget that fact. The *TUGboat* Associate Editors may also be able to help in their areas of expertise.

Don't underestimate the value of the TUG membership list. The annual list contains separate listings by institution, by geographical location, and by computer and output hardware in addition to the main alphabetical listing. The main listing shows everything that the TUG office knows about a member, including interest areas and how that member is actually using *TeX*. (A suggestion of long standing, that a listing by area of interest and use be added to the membership list, is being investigated for possible implementation next year.)

The user groups

Last, but not least, TUG and the regional and national user groups can often give assistance.

At the most recent TUG annual meeting, Malcolm Clark presented an excellent overview of the user groups in Europe and the concerns of *TeX* users in Europe. The paper can be found in the Proceedings, *TUGboat* 10, no. 4, pages 667-673; it also contains the names and addresses of the contacts for the European groups and a partial bibliography of *TeX*- and *LATeX*-related books that have been published in Europe.

Meetings of these groups are regularly announced in *TUGboat*, both in the calendar and in calls for papers, and reports are published as well. In addition to the groups listed below, we are aware that a Japanese group at least used to exist, and we have heard about efforts to start groups elsewhere (see the article by Hubert Partl on page 122 for news from a meeting in Czechoslovakia).

TUG. The address of the TUG office can be found on the inside front cover of every issue of *TUGboat*. The officers and board of directors are listed on the inside front cover of *TUGboat*. The first few pages of the annual membership list contain the names and addresses of the officers, board, site coordinators and committees. This information is repeated in the membership supplements printed in regular issues of *TUGboat* after the first one of the year.

The elected heads of most major regional groups have been designated vice-presidents of TUG; their names are included in the lists of board members on the inside front cover of *TUGboat* and at the beginning of the annual membership list and supplements.

In addition to keeping membership and subscription records, the TUG office keeps an inventory of publications and software of interest to *TeX* users. Specific information can be obtained from the TUG office.

DANTE. The Deutschsprachige Anwendervereinigung *TeX*, DANTE, was founded in 1989, but actually, groups of German-speaking *TeX* users have been meeting fairly regularly for much longer. (This issue's calendar lists the 9th and 10th annual meetings of the "Deutschsprachige *TeX*-Interessenten".)

DANTE maintains an archive with Listserver at the University of Heidelberg (DHDURZ1), and has just begun producing *Die TeXnische Komödie*, a "Bühnenstück [stage play] in (hoffentlich) vielen Folgen".

Joachim Lammarsch, the president of DANTE, is on the TUG board.

GUTenberg. Groupe (francophone) des Utilisateurs de *TeX* has been in existence since 1988, but like the German speakers, the French have been actively working together for much longer.

The Cahiers GUTenberg is the official publication. GUTenberg also maintains an archive, in coordination with DANTE, at Heidelberg, and official announcements are distributed via the electronic mailing list GUT. More information on GUTenberg and its services can be obtained from Bernard Gaulle, the president of GUTenberg and a member of the TUG board.

The Nordic *TeX* Group. This group watches over the interests of *TeX* users in Denmark, Norway, Sweden and Finland. The diversity of the languages used in these countries and other considerations have resulted in a much more loosely-structured organization than is the case with the other groups described here. A description of the challenges facing this group is included in the article by Malcolm Clark cited above.

Roswitha Graham, the chairperson of the Nordic group, is also a member of the TUG board.

NTG. The Nederlandse *TeX* Gebruikersgroep has an ambitious program and a number of working groups devoted to particular interest and problem areas. NTG is working actively with the Dutch SGML users group in areas of common interest, and the two groups are holding a joint meeting on August 31 (see page 126). NTG also maintains an archive and sponsors *TeX-NL*.

The chairman of NTG, Kees van der Laan, is also on the TUG board.

The UK *TeX* Group. This group has been formally organized for only about a year, and only since October 1989 has it had elected officers. Malcolm Clark is now chairman, and a member of the TUG board; he is also TUG's European Coordinator, a position which may predate the existence of the UK *TeX* group.

Summing up

This compendium is certainly incomplete, both because we have probably forgotten to mention resources that we take for granted and because we know that *TeX* is being used in areas that we've never heard of and that are likewise unaware of TUG's existence. This is partly a consequence of the *TeX* system being in the public domain.

If you have knowledge of any *TeX*-related organization, public mailing list or electronic discussion group, or publication that isn't identified here, please let us know about it. We would also like to keep track of commercial organizations providing *TeX* software or services, as vendors or consultants. The more information that we have, the better TUG will be able to provide support services to its members.

A Proto-TUG Bibliography

Barbara Beeton

The number of books and articles about *TeX*, *LATeX*, *WEB* and others of their friends, or prepared using one of them, is growing at an astounding rate. Copies of many of these publications actually come to my desk, and references to others find their way here as well. For a long time, *TeX* users have been asking whether a bibliography of these works exists, or suggesting that one be established. We are publishing here the beginning of such a TUG bibliography, starting with the accumulation from my in-box for about the past year.

I envision this bibliography growing, with help from the *TeX* community, and being made available in several ways. First, installments will be published regularly in *TUGboat*. Second, as we are compiling the information in *BIBTeX* form, the source of these partial listings will be consolidated and installed in the *TUGboat* area of the various electronic archives. Finally, when the size of the collection has reached a reasonable level of completeness, it should be neatly packaged by TUG and made available in print.

The scope is another area which is open to speculation. The following topics are what I would like to see covered in such a bibliography.

- Using *TeX*, *LATeX*, etc.
- Reviews of *TeX*-related products
- Items published by *TeX* user organizations
- Using *WEB*
- Literate programming
- Document design
- Typography
- Fonts
- *METAFONT*
- Structured documents
- SGML

- Graphics
- Printer support
- Postscript

As this project is just in its infancy, now is the time for suggestions on what direction you think it should take. And please send in your candidates for inclusion in future installments as well.

Here is the first installment, in two parts: publications about *TeX*, and publications prepared with *TeX*.

Publications about *TeX*

- A. Brüggemann-Klein and D. Wood. Drawing trees nicely with *TeX*. *Electronic Publishing—Origination, Dissemination and Design*, 2(2):101–115, July 1989.
- Anne Brüggemann-Klein. *Einführung in die Dokumenten-verarbeitung*. B.G. Teubner, Stuttgart, 1989. German. This book was prepared with *LATeX*.
- R. de Bruin, C.G. van der Laan, J.R. Luyten, and H.F. Vogt. Publiceren met *LATeX*. CWI Syllabus 19, Centrum voor Wiskunde en Informatica, Amsterdam, 1988. Dutch; copious examples. This report was prepared with *LATeX*.
- David J. Buerger. *LATeX for Engineers & Scientists*. McGraw-Hill Publishing Company, New York, 1990. This book was prepared with *LATeX*.
- Bill Cheswick. A permuted index for *TeX* and *LATeX*. Computing Science Technical Report 145, AT&T Bell Laboratories, Murray Hill, NJ, February 1990. Also published by TUG as *TEXniques* No. 14.
- Michael Doob. *A Gentle Introduction to *TeX**, No. 12 of *TEXniques*. *TeX* Users Group, Providence, 1990.
- Donald E. Knuth. *The *TeX*book*. ASCII Corporation, Tokyo, 1989. Japanese translation of the English edition.
- Donald E. Knuth, Tomas G. Rokicki, and Arthur L. Samuel. *METAFONTware*, No. 13 of *TEXniques*. *TeX* Users Group, Providence, 1990. Originally published as Stanford Computer Science Report STAN-CS-89-1255.
- C.G. van der Laan, D.C. Coleman, and J.R. Luyten. SGML-*LATeX* 1. Mathematical formulas. RC-Rapport 24, Rekencentrum der Rijksuniversiteit te Groningen, Groningen, 1989. (English version).
- C.G. van der Laan and J.R. Luyten. *LATeX* templates: Letter and article. RC-Rapport 27,

Rekencentrum der Rijksuniversiteit te Groningen, Groningen, November 1989.

- Saul Lubkin. Porting sophisticated programs to your Unix environment for free. *Computer Shopper*, pages 642–644, March 1990.
- Sebastian Rahtz. A survey of *TeX* and graphics. Technical Report CSTR 89-7, University of Southampton, Department of Electronics and Computer Science, Southampton SO9 5NH, October 1989.
- Raymond Seroul. *Le petit Livre de *TeX**. InterEditions, Paris, 1989. French.
- Wayne Sewell. *Weaving a Program: Literate Programming in WEB*. Van Nostrand Reinhold, New York, 1989.
- L. Steemers and C.G. van der Laan. Journal style guidelines. RC-Rapport 26, Rekencentrum der Rijksuniversiteit te Groningen, Groningen, 1989.
- Die *TeXnische Komödie*. 0:1–35, December 1989. German. This publication was prepared with *LATeX*.

Publications prepared with *TeX*

- Paul C. Anagnostopoulos. *VAX/VMS: Writing Real Programs in DCL*. Digital Press, 1989. This book was prepared with *LATeX*.
- Mathematical Sciences Education Board/Board on Mathematical Sciences/Committee on the Mathematical Sciences in the Year 2000, National Research Council. *Everybody counts: A report to the nation on the future of mathematics education*. Washington, D.C., 1989. This report was prepared with *TeX*; it does not have the traditional “*TeX*” appearance, but is an excellent example of modern graphical design.
- David F. Rogers and J. Alan Adams. *Mathematical Elements for Computer Graphics*. McGraw-Hill Publishing Company, second edition, 1990. This book was prepared with *TeX*.

Contents of the Clarkson Archive Server as of 26 January 1990

Michael DeCorte

As this is the first issue of the year it will contain a characterization of the complete archive, including lists of many of the files. Subsequent articles this year will only contain updates.

Many changes have been made to the archive since my last report. First, the software has been entirely rewritten. It is now possible for all users to obtain all files stored. You are no longer restricted to text files and there are no file size limits. The software has also allowed me to create a more logical and convenient structure for archive. Second, changes at Stanford are now reflected in the archive within a week of their posting.

The new software keeps full accounting information. The latest statistics indicate that the archive server is servicing about 20,000 mail requests per year, resulting in about one terabyte of outgoing mail.

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

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

For Internet users: how to ftp

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

Sample FTP session for Internet users

```
% ftp sun.soe.clarkson.edu
...
user: anonymous
password: <any non-null string>
ftp> cd pub/latex-style
ftp> ls
...
ftp> get Index
...
ftp> quit
```

(a.k.a. 128.153.12.3)
(general blurb)

(where the files are)
(to see what is there)
(lots of output)

(more blurb)

Non-Internet users: how to retrieve by mail

To conserve space I will briefly describe only the most common commands. Provisions have been added to respond to most operating systems in a logical manner. I strongly suggest that you request the help file to learn about these new commands.

To retrieve files, current indexes or help documentation, send mail to

`archive-server@sun.soe.clarkson.edu`

with the body of the mail message containing the command path and any of the commands `help`, `index` or `send`. The `send` command must be followed by the name of the archive and then the files you want. The `path` command must be followed by a path from Clarkson to you in domain style format. Domain style format means that your address must be routed through Internet, Bitnet or registered UUCP sites. Therefore `host!user` is guaranteed to fail, but `user@host.UUCP` should work. For example, this user should send

To: `archive-server@sun.soe.clarkson.edu`
Subject:
`help`
`path user@host.UUCP`
`send latex-style res.sty res-sample.tex`
`index latex-style tex-style`

Traffic on the network servers and gateways is usually very high. In order to provide improved service, local “slave” repositories are being maintained by volunteers. Since slave repositories are created to reduce network traffic, please obey any geographic or network restriction requested. The following areas will be covered by the volunteers listed.

- Bitnet users:

Texas A&M maintains a list-server and file-server which is already handling (with `TEX-L`) much of the Bitnet distribution of `TeXhax`. An

inquiry via listserv will retrieve a list of all **TEX**-related files:

```
tell listserv at tamvml get tex filelist
```

- UK users:

Aston University maintains a **TEX** archive covering all aspects of **TEX**, **LATEX**, **METAFONT** and ancillary software. **UKTEX** (like **TEXhax**) digests are distributed from Aston. For users with Colour book software, FTP access is available; for all users mail access is available. To use the mailserver send mail to

```
TeXserver@uk.ac.aston.tex
```

```
---
```

```
yourname%yoursite@relay  
help
```

- Italian users:

Marisa Luvisetto and Max Calvani maintain a SPAN/DECNET depository. They have software for redistribution including the **LATEX**-style collection, Beebe's driver family, the **TEXhax**, **TEXMAG** and **UKTEX** magazines, **dvitovdu**, **psprint**, **texsis**. For more info on what is available and how to get it, please send a mail message to 39947::luvisetto or 39003::fisica. American users can also contact Ed Bell at 7388::bell. Max Calvani's Internet address is **fisica@astrpd.infn.it**.

- Canadian users:

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

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

Additional volunteers for slave depositories should contact me.

Tape Distribution by USMail

To obtain large quantities of files please use USMail. To do so, mail to

- Rob Logan
ERC
Clarkson University
Potsdam NY 13676

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

Note: If the tape is not self-addressed we will keep the tape and use it for backups.

You may obtain an index of all the top level archives by sending a self-addressed stamped envelope to the same address.

For your information, a portion of these funds is used to pay a student to copy the tapes. The remainder is placed into an account to purchase a disk drive dedicated to the archive server. Contributions are strongly encouraged and are tax deductible.

Distribution for IBM PC and clone users

There are two sources.

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

has **LATEX** style files only. Send:

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

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

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

Listing of Directories and Files

AMSTEX Sources. Contains the *TEX* source needed to build *AMSTEX*. It is a duplicate of the directory *tex/amstex* on labrea. Files are located in *pub/amstex* for ftp users. Mail users should use *amstex*.

AMSTEX Style. Contains style files specific to *AMSTEX* users. Files are located in *pub/amstex-style* for ftp users. Mail users should use *amstex-style*.

<i>amstexsiam.sty</i>	<i>AMSTEX</i> style file for
<i>amstexsiam-doc.tex</i>	SIAM
<i>amstexsiam-sample.tex</i>	

<i>imappt.sty</i>	<i>AMSTEX</i> style file for
<i>imappt-doc.tex</i>	reports and preprints on
<i>imappt-sample.tex</i>	8.5 by 11 inch paper. It is presently used by the Institute of Mathematics and its Applications in Minneapolis; documentation and sample for above as well.

<i>mssymb.sty</i>	Definitions for symbols in the two extra symbols fonts created at the American Mathematical Society.
-------------------	--

BIBTEX Sources. Contains the *BIBTEX* style files and the *WEB* files need to build *BIBTEX*. It is a duplicate of the directory *tex/bibtex* on labrea. Files are located in *pub/bibtex* for ftp users. Mail users should use *bibtex*.

BIBTEX Style. Contains files that are specific to version 0.99 of *BIBTEX*. Many of these files are to be used with files in the *LATEX* Collection. Files are located in *pub/bibtex-style* for ftp users. Mail users should use *bibtex-style*.

<i>aaai-named bst</i>	AAAI conference
-----------------------	-----------------

<i>acm.bst</i>	ACM
----------------	-----

<i>apalike.bst</i>	American Psychological Association (<i>LATEX</i> style file in <i>latex-style</i>)
--------------------	---

<i>cpp.el</i>	C preprocessor written for GnuEmacs for use in generating <i>BIBTEX</i> style files from a master file.
---------------	--

<i>ieeetr.bst</i>	IEEE Transactions
-------------------	-------------------

<i>makebst.sh</i>	Shell script to make <i>BIBTEX</i> style files from a master <i>bst</i> file
-------------------	---

<i>named.bst</i>	For use with <i>ijcai89.sty</i>
------------------	---------------------------------

<i>physics.btx</i>	Various physics journals; must be run through <i>cpp.el</i> or <i>makebst.sh</i>
--------------------	---

<i>siam.bst</i>	SIAM (<i>LATEX</i> style in <i>latex-style</i>)
-----------------	--

BIBTEX 0.98 Style. Contains files that are specific to version 0.98 of *BIBTEX*. Many of these files are to be used with files in the *LATEX* Collection. Files are located in *pub/bibtex-style-0.98* for ftp users. Mail users should use *bibtex-style-0.98*.

<i>aaai-named.bst</i>	AAAI conference 1988 (<i>LATEX</i> style file in <i>latex-style</i>)
-----------------------	---

<i>btxbst.doc</i>	Master file for <i>BIBTEX</i> styles
<i>btxbst.readme</i>	with standard styles and some new ones.

<i>natsci.bst</i>	Natural sciences generic <i>BIBTEX</i> style (<i>LATEX</i> style file in <i>latex-style</i>)
-------------------	--

<i>walpha.bst</i>	Modified alphabetic <i>BIBTEX</i> style
-------------------	---

Canon 300. Contains the *pk* files at 300 dpi for canon laser printers. You should find this to be very complete. Files are located in *pub/canon300* for ftp users. Mail users should use *canon300*.

CM Fonts. Contains the *METAFONT* files needed to build the CM fonts. It is a duplicate of the directory *tex/cm* on labrea. Files are located in *pub/cm-fonts* for ftp users. Mail users should use *cm-fonts*.

DVI Driver Standards. Contains digests from the DVI Driver standards committee and articles about DVI standards. Files are located in *pub/dvi-standard* for ftp users. Mail users should use *dvi-standard*. Digests are named *driver.YY.MM* where YY is the year of the issue, MM is the month.

Errata. Contains the *TEX* error logs and some documentation. It is a duplicate of the directory *tex/errata* on labrea. Files are located in *pub/errata* for ftp users. Mail users should use *errata*.

FONTS 118. Contains the *pk* files at 118 dpi. Files are located in *pub/fonts118* for ftp users. Mail users should use *fonts118*.

Fonts 96. Contains the pk files at 96 dpi. Files are located in pub/fonts96 for ftp users. Mail users should use fonts96.

LATEX Sources. Contains the *TEX* files needed to build LATEX. It is a duplicate of the directory *tex/latex* on labrea. Files are located in pub/lamport for ftp users. Mail users should use lamport.

LATEX Style. Contains files that are specific to LATEX. Some of the files have supporting BIBTEX style files that are in **BIBTEX Style** or **BIBTEX 0.98 Style**. Files are located in pub/latex-style for ftp users. Mail users should use latex-style.

a4.sty	Sets page size to A4.
a4wide.sty	Sets page size to A4 with narrow margins.
a5.sty	Sets A5 page size (use only with 10pt).
a5comb.sty	Sets A5 page style, but for spirally-bound documents (bigger inner margins).
aaai.sty aaai-doc.tex	Style file for AAAI conference 1988 (BIBTEX style in bibtex-style and bibtex-style-0.98)
agugrl-sample.tex agugrl.sty	AGU Geophysical Research Letters style
agujgr-sample.tex agujgr.sty	AGU Journal of Geophysical Research style
aip.sty	For American Institute of Physics journals
album	Style for printing cassette labels
allegno.sty	Makes all displayed equations numbered by default
alltt.sty	Like verbatim, but permits other commands inside
amssymbols.sty	Loads AMS fonts
apalike.sty	For American Psychological Association. (BIBTEX style in bibtex-style)
apsabstract.sty	For American Physical Society meetings
biihead.sty	Underlined heading
boxedminipage.sty	Puts a box round a minipage

breakcites.sty	Allows citations to break across lines
bsf.sty	Provides access to bold san serif fonts in LATEX
captcont.sty	Captions in continuations of floats
catmac-doc.tex	Commutative diagram macros
catmac.sty	
cd.sty cd-doc.tex	More commutative diagram macros
changebar.sty	Changebars for LATEX
chapterbib.sty	Allows LATEX and BIBTEX to produce separate bibliographies for each chapter.
cite.sty	Compresses lists of numbers in citations to ranges, and allows linebreaks with difficulty.
cup.sty	For books in the Cambridge University Press, British Computer Society Workshop series.
cyrillic.sty	Load Cyrillic font
deproc-doc.tex	DECUS proceedings style
deproc.sty	
deproc.readme	
doublespace.sty	Double spacing in text
draft.sty	Draft option for documents for debugging
drafthead.sty	Prints DRAFT in heading
drop.sty	For making large dropped initials for starting paragraphs
eepic	Picture environment that used tpic specials
env.sty	To print on envelopes
epic	An extended picture environment
equations.sty	Macros to aid in constructing displayed equations
espo.sty	For Esperanto
fancyheadings.sty	To modify the headers and footers
farticle.sty	French version of article.sty
fixup.sty	Fixup plain's \bigl, etc. to track LATEX size changes
flowchart.sty	To write flow charts
fnpara.sty	Sets footnotes as paragraphs

format.sty	Prints FP numbers in fixed format	mitpress.sty	Simple MIT Press format
frenchponct.sty	French punctuation	mitthesis-sample.tex	Massachusetts Institute
frontiers.sty	For Frontiers '88 symposium and other IEEE conferences	mitthesis.sty	of Technology thesis format
fullpage.sty	Gets more out of a page	multibox.sty	Provides multiple boxes in pictures
geophysics.sty	Geophysics Journal style	named.sty	For use with named.bst
german.sty	For German	natsci.sty	Natural sciences style (BIBTEX file in bibtex-style-0.98)
hackalloc.sty	Makes allocation local	nl.sty	For Dutch
headerfooter.sty	Adds the capability of underlining the heading	nofm.sty	For "n of M" style pagination
icassp.sty	For the ICASSP '89 conference (and other IEEE conferences)	nopagenumbers.sty	Removes page numbers
ijcai89.sty	Conference Style for IJCAI-89	overcite.sty	Gives superscript numbers for citations, compressing lists to ranges, and moving past punctuation
ijcai89.tex		pagefoots.sty	Puts footnotes at the bottom of each page
insertplot.readme	For inserting PostScript	parskip.sty	Sets parindent to 0 and puts some glue into parskip to aid page breaks
insertplot.sty	in files printed with Arbortext's DVIPS.	portland.sty	Environments to switch between portrait mode and landscape mode
iso.sty	For ISO standards	pslatex	Macros for a LATEX that uses printer resident PostScript fonts. Requires dvi2ps that understands PostScript fonts.
iso9.sty		refman.sty	Document style for reference manuals similar to the PostScript manual
iso-doc.tex		remark.sty	Like newtheorem but no \it
ist21.sty	IST21 document style	res-sample.tex	Format for doing resumes by
jbs.sty	For the Journal of Business Strategies	res.sty	Michael DeCorte
jeep.sty	Useful modifications of the article	resume-sample.tex	Format for doing resumes by
jeep.tex	style	resume.sty	Stephen Gildea
layout.readme	Prints nice diagram showing page	romanneg.sty	Roman-numbered pages get negative page numbers (useful when selecting only part of a document to be printed)
layout.tex	parameters of LATEX	rotate.sty	Rotation of TeX output with TEXtures (and maybe other PostScript drivers)
lablst.sty	For printing label definitions at the end of the document	sc21.sty	ISO/TC97/SC21 document style
lcustom.tex	Useful macros and definitions	sc21-wg1.sty	option for cover page
lfonts_ams.readme	Use AMS symbols	schedule.sty	For generating schedule sheets
lfonts_ams.tex			
local-suppl.tex	Supplement to local guide that describes lcustom.tex, sfmtmac.sty, tgrind.sty, trademarks.sty, xxxcustom.tex, xxxslides.sty		
ltugboat.sty	For articles to <i>TUGboat</i>		
manual	Like book but for manuals.		
memo.sty	Memo and memo for the record		
mfr.sty	style option		
merge.sty	Form letter option to letter style		
mf.sty	Make METAFONT logos at all sizes		

screen.sty Helps create a document suitable for screen previewing
semitic.sty Used to set Semitic languages
sfwmac.sty Useful macros for Unix documentation
showlabels.sty Shows labels and references to them
siam SIAM style (BIBTEX style in bibtex-style)
slem.sty Change \sl to \em
spacecites.sty Gives spacing between citations
subeqn.sty Allows related equations to be numbered with the same number and further qualified by a,b,c etc.
subeqnarray.sty Allows related eqnarrays to be numbered with the same number and further qualified by a,b,c etc.
subfigure.sty Allows related figures to be numbered with the same number and further qualified by a,b,c etc.
supertab.sty Allows multipage tabulars
suthesis.sty Stanford U thesis style
svma.sty Style for Springer-Verlag reports,
svsa.sty single and multi-author
svma-doc.tex
tbls.sty Ruled and unruled tables made easy
tables-doc.tex Simulate minimum-lineskip glue
tables.sty in tabular environments
tape Files to setup tape libraries
texnames.sty Define a couple more TEX names
tgrind.sty Tgrind macros
threepart.sty Three part page headers
trademarks.sty Definitions of common trademarks
twoup.sty Changes the page sizes so that 2 pages can fit onto one with the help of dvidvi
ucthesis U of California thesis style
ukdate.sty Changes the \today command to UK style
ulem.sty Use underlining for emphasis
usenix.sty For Usenix conference proceedings

vdm.sty Vienna Development Method
vdm-doc.tex
verbatimfiles.sty Includes a file in a verbatim mode
xxxcustom.tex Supplementary macros for xxx-tex, for some xxx
xxsslides.sty Supplementary macros for SLiTeX, includes **slides.sty**

METAFONT Sources. Contains the WEB files needed to build METAFONT. It is a duplicate of the directory **tex/mf** on labrea. Files are located in **pub/mf** for ftp users. Mail users should use **mf**.

METAFONTware. Contains the WEB files for some of the METAFONT utilities. It is a duplicate of the directory **tex/mfware** on labrea. Files are located in **pub/mfware** for ftp users. Mail users should use **mfware**.

Misc Fonts. Contains miscellaneous METAFONT files. It is a duplicate of the directory **tex/local/cm** on labrea. Files are located in **pub/misc-fonts** for ftp users. Mail users should use **misc-fonts**.

PCTEX. Contains the source for PCTEX. Files are located in **pub/pictex** for ftp users. Mail users should use **pictex**.

Ricoh 300. Contains the pk files for Ricoh printers at 300 dpi. Files are located in **pub/ricoh300** for ftp users. Mail users should use **ricoh300**.

TEX files. Contains miscellaneous TEX and METAFONT files. It is a duplicate of the directory **tex/local/lib** on labrea. Files are located in **pub/tex-files** for ftp users. Mail users should use **tex-files**.

TEX Fonts. Contains the METAFONT files for user contributed fonts. Files are located in **pub/tex-fonts** for ftp users. Mail users should use **tex-fonts**.

apl	APL fonts and related macros
cmpica	CM Pica by Don Hosek
concrete	The fonts and macros for Concrete Mathematics
greek	For papers in Greek
hershey	Hershey fonts

barcodes.mf	To generate barcodes
logic	For logic diagrams
music	To typeset music
ocr	OCR-A fonts by Tor Lillqvist
sauter	John Sauter's reparameterized Computer Modern.
tengwar	The fonts used by Tolkien in <i>Lord of the Rings</i>
wujastyk.tex	Description of many METAFONT fonts
xhershey	Hershey script fonts and a program to convert vector fonts to METAFONT

□ **TeX lib.** Contains the TeX and METAFONT files used by the TeXbook. It is a duplicate of the directory `tex/lib` on labrea. Files are located in `pub/tex-lib` for ftp users. Mail users should use `tex-lib`.

□ **TeX man.** Contains manual pages to the TeX programs. It is a duplicate of the directory `tex/local/man1` on labrea. Files are located in `pub/tex-man` for ftp users. Mail users should use `tex-man`.

□ **TeX misc.** Contains miscellaneous TeX programs. It is a duplicate of the directory `tex/etc` on labrea. Files are located in `pub/tex-misc` for ftp users. Mail users should use `tex-misc`.

□ **TeX Programs.** Contains programs that are of general interest to TeX users. Files are located in `pub/tex-programs` for ftp users. Mail users should use `tex-programs`.

docsty	Program to convert .doc to .sty by stripping comments
dvidoc	DVI to character device filter for Unix BSD systems
dvidvi	Program to select pages from dvi files or print multiple pages on a single page
dvitty	Previewer for glass tty's
fig2epic	Converts fig code to epic or eepic files
lgraph	Data to graph command filter in Pascal

pcwritex.uue	PC-Write to TeX interface that has been uuencoded
schemetex	Simple support for literate programming in Lisp. A Unix filter that translates schemeTeX source into LATEX source
texindex	Style file and processor for index entries for VMS
tib	Bibliography and citation setter for TeX
wsltex	Wordstar to LATEX filter, C and Pascal versions

□ **TeX Sources.** Contains the WEB files needed to build TeX. It is a duplicate of the directory `tex/tex` on labrea. Files are located in `pub/tex-sources` for ftp users. Mail users should use `tex-sources`.

□ **TeX Style.** Contains style files for plain TeX. Files are located in `pub/tex-style` for ftp users. Mail users should use `tex-style`.

cell	Allows for both vertical and horizontal spans within a ruled table, and takes steps to prevent 'nubs' and 'gaps' when rules are used.
dayofweek.tex	Computes day of week and phase of moon; examples of how to use TeX arithmetic capabilities
declare.tex	Allocates local registers
epigram.tex	Prints text either centered or in a displayed paragraph
figplace.tex	Handles floating insertion
fnpara.tex	Sets footnotes as paragraphs
hyphen-nederlands.tex	Dutch hyphen.tex
ithyphen.tex	Italian hyphen.tex
mssymb.tex	Definitions for symbols in the two extra symbols fonts created at the AMS.
scorecard.tex	Prints a baseball scorecard for one team
select.tex	Selectively prints pages in a document
texinfo.tex	Handles Gnu texinfo files
texpictex.tex	tpic \special changes to PCTEX

□ TExhax Digests. Contains all of the back issues of TExhax. Files are located in pub/texhax for ftp users. Mail users should use texhax. Files are named texhax.YY.NNN where YY is the year of the issue and NNN is the issue number.

□ TEXMAG Digests. Contains all of the back issues of TEXMAG. Files are located in pub/texmag for ftp users. Mail users should use texmag. Files are named texmag.V.NN where V is the volume number and NN is the issue number.

□ TExware. Contains the WEB programs that manipulate the files used by TEx. It is a duplicate of the directory tex/texware on labrea. Files are located in pub/texware for ftp users. Mail users should use texware.

□ TFM. Contains the tfm files for most fonts. Files are located in pub/tfm for ftp users. Mail users should use tfm.

□ Transfig Sources. Contains the C source for Transfig; a program that converts Fig output to other forms such as PCTEX. Files are located in pub/transfig for ftp users. Mail users should use transfig.

□ TUGboat Files. Contains files related to TUGboat. It is a duplicate of the directory tex/tugboat on labrea. Files are located in pub/tugboat for ftp users. Mail users should use tugboat.

□ UKTEX Digests. Contains all of the back issues of UKTEX. Files are located in pub/uktex for ftp users. Mail users should use uktex. Files are named uktex.YY.NNN where YY is the year of the issue and NNN is the issue number.

□ WEB. Contains the WEB files to build the WEB system. It is a duplicate of the directory tex/web on labrea. Files are located in pub/web for ftp users. Mail users should use web.

□ WEB2c. Contains the compressed tar files for web2c. It is a duplicate of the directory tex/web2c on labrea. Files are located in pub/web2c for ftp users. Mail users should use web2c.

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

IBM VM/CMS Site Report

Dean Guenther

Effective January 1st, 1990 I will no longer be the VM/CMS site coordinator. I have handed the baton to Joachim Lammarsch. Joachim, the president of Germany's national TEx organization DANTE, has been in the TEx community for some time and is already distributing VM/CMS TEx in Europe. I will continue to offer phone consulting for VM/CMS users in North America, so that this set of users will not need to call trans Atlantic for help. Electronic mail requests for help should now be sent to Joachim instead of myself. Joachim's Bitnet address is X92@DHDURZ1.

One goal I am working on with Joachim is to make available the VM/CMS distribution by FTP and over Bitnet. Hopefully before long, an issue of TUGboat will describe how to do this.

Maria Code now has TEx 2.991 for distribution. Peter Breitenlohrer (Germany) is working diligently on TEx 3.0, and as soon as all of the conversion is done, Maria will be given a new copy for distribution.

◊ Dean Guenther
Computer Service Center
Washington State University
Computer Science Building
Pullman, WA 99164
Bitnet: GUENTHER@WSUVM1

Typesetting on Personal Computers

Une version complète de **TEX du domaine public pour compatibles PC : les "deux disquettes GUT"**

NICOLAS BROUARD *

Abstract

*A complete Public Domain version of **TEX** for PCs
on two diskettes from GUTenberg*

The GUTenberg association of frenchspeaking **TEX** users has recently gathered on two 1.2Mbyte diskettes a collection of Public Domain **TEX** compiler and **TEX**tools running on DOS. Complete collection means here that it includes compiler, text editor, previewers and drivers for printers. Three additional diskettes are available for the 300dpi fonts (laser printer) and two for the 240dpi resolution (matrix printers). The compiler comes from Wayne Sullivan. It is now able to read directly 8-bit characters (accented characters) and is named **SB26TEX**. A complete set of **LATEX** macros is included. The **BIBTEX** compiler (version 0.99c) comes from Niel Kempson and has been slightly changed to read 8-bit characters too. The text editor is Jonathan's Own Version of Emacs which has been recently ported to PCs. The previewer is the very fast CDVI 1.0 from Wayne Sullivan. The DVIXXX driver family of N. Beebe has also been included. Available documentation has been included (mainly in english). French styles for **LATEX** and **BIBTEX** have been added. Some documents like the installation files and an introduction to **LATEX** are in french. If you add some other tools like the compressor (PKZIP, which was used to compress all the files) and tools for accessing and decoding messages from **TEX**-servers you can discover and enjoy the use of **TEX** and **LATEX** on either a PC/XT or 486 with those two diskettes.

Lors du dernier congrès GUTenberg de mai dernier, nous avions, le président de GUTenberg et moi-même, décidé d'annoncer que l'association GUTenberg allait distribuer une version de **TEX** du *domaine public* pour compatibles PC. L'opération était assez hasardeuse car il n'y avait pas réellement de compilateur **TEX** du domaine public opérationnel

en français ; mais grâce au réseau EARN et aux serveurs **TEX** il m'apparaissait possible de joindre différentes personnes à l'étranger et d'obtenir des compilateurs corrects. L'ambition ne s'arrêtait pas au seul compilateur **TEX**, il fallait en effet offrir aux utilisateurs une version complète de ce traitement de texte, c'est à dire aussi, une version **LATEX** avec ses deux acolytes **SLI^TEX** et surtout le compilateur **BIBTEX**, un éditeur de texte très performant du type Emacs, une visionneuse à l'écran pour de multiples écrans, un pilote laser pour de multiples lasers, un pilote d'imprimante matricielle. Comme le monde **TEX** évolue, il était important aussi de fournir les logiciels de base pour accéder aux réseaux et décoder les messages contenant les nouvelles versions des outils précédemment décrits. La forme de distribution la plus évoluée est en effet l'accès à un réseau EARN, mais comme l'accès à un tel réseau nécessite une structure importante, et qu'au contraire, l'écriture d'un texte avec **TEX** ne nécessite qu'un PC, nous avons regroupé l'ensemble des logiciels décrits plus haut sous deux disquettes 1.2Mo qui ont été appelées *les disquettes GUT* et que nous avons le plaisir de mettre à la disposition du public¹.

Les disquettes GUT sont donc l'œuvre collective d'un nombre impressionnant d'auteurs qui ont mis le fruit de leur travail dans le domaine public. L'association GUTenberg n'a fait que regrouper ou susciter des modifications des programmes les plus récents du domaine public pour faire un ensemble le plus cohérent possible. L'utilisateur ancien de **TEX** ne trouvera sans doute rien qu'il n'ait pu acheter déjà chez des distributeurs privés, mais les compilateurs des disquettes GUT sont souvent plus rapides. En effet, notre souci a été de faire une sélection parmi l'ensemble des compilateurs du domaine public et de mettre dans les disquettes le meilleur du moment. Récemment, le source d'un compilateur **TEX** sur PC, à savoir **PUB-TEX** a été mis sur un réseau public allemand ; il fait l'objet de nombreuses modifications actuellement et si ses performances, assez déplorables dans sa toute première version, venaient à concurrencer le compilateur choisi actuellement, il serait sélectionné l'année suivante.

Venons-en maintenant à une description plus détaillée du contenu des disquettes.

¹ Les polices de caractères sont distribués à part : 3 disquettes 1.2Mo pour une laser 300 points par pouce, et 2 disquettes 1.2Mo pour une imprimante compatible Epson 240 points par pouce.

* Chargé de recherche à l'Institut national d'études démographiques (INED-PARIS).

1 Les principaux logiciels inclus dans les 2 disquettes GUT

L'idée d'une distribution *T_EX* du domaine public sur PC n'est pas nouvelle puisqu'il existe déjà une version DOSTEX distribuée sur certains réseaux. Outre que le compilateur DOSTEX est nettement moins rapide que SB26TEX, finalement sélectionné, cette distribution ne comporte que *T_EX*, sans éditeur ni pilote, et surtout n'est pas modulable comme le prétend l'être la distribution GUT. En effet, voici le contenu des deux disquettes :

```
Directory of a:GUT1/2\
alire.txt      12254 7-26-89  5:35a
pkz090.exe     91520 7-16-89 10:38p
sb26tex.zip   222541 10-1-89  6:05p
fonttfms.zip  97938 7-23-89  8:11a
texinput.zip   323032 7-23-89  7:40a
jove.zip       156522 8-17-89  3:54p
latexdoc.zip  143257 7-20-89 11:43a
doc.zip        39366 8-17-89  8:55p
dviman.zip    67005 8-17-89  5:09p
instal.bat     726   8-17-89  8:40p
instex1.bat   2470   8-17-89  5:46p
instex2.bat   1256   8-17-89  9:07p
instex3.bat   2246   8-17-89  6:15p
instex4.bat   1365   8-17-89  6:28p
instex5.bat   2054   8-17-89  6:33p
instex6.bat   4366   8-17-89  9:24p
etiq.dvi      2016   8-15-89  6:52p

Directory of a:GUT2/2\
pilotes.zip   382136 7-23-89  9:45a
bibtex99.zip  243145 7-21-89  6:38a
jovedoc.zip  78636 7-25-89  2:49p
outils.zip   261030 7-25-89  2:52p
pdcdvi.zip  221141 7-20-89 11:40p
```

Outre des fichiers d'installations d'extension .bat, les disquettes contiennent des fichiers compressés d'extension .zip qui correspondent chacun à une tâche précise et proviennent généralement d'une même origine. On trouve en particulier le compilateur *T_EX* dans le fichier sb26tex.zip.

Le fichier pkz090.exe est un compresseur très puissant et très rapide avec lequel l'ensemble des logiciels a été compressé. pkzip est en effet le successeur des célèbres pkarc et arc, et autre que son pouvoir de compression leur est supérieur, le compresseur permet maintenant d'archiver des arborescences².

² Notons ici, comme l'indique la notice des auteurs, que les personnes qui trouvent ce compresseur rapide et facile à utiliser peuvent envoyer une somme

La distribution ne comporte que les fichiers exécutables ; en effet, parfois (et on peut le regretter tout en comprenant les raisons) seuls les exécutables sont dans le domaine public. Parfois, le source est public et peut généralement être obtenu sur un serveur.

1.1 Le compilateur SBTEX, version 26

Ce compilateur, dont seul l'exécutable est du domaine public est dû à l'irlandais Wayne Sullivan. Il est écrit en Turbo-Pascal et est parmi les compilateurs les plus rapides du moment. Ayant personnellement fourni à Wayne un fichier de modification en Pascal (change-file)³ pour pouvoir lire les caractères accentués 8 bits, celui-ci a accepté de l'adapter à son compilateur. Plus généralement, Wayne a eu l'idée de mettre dans les 128 premières lignes du fichier tex.poo la définition des caractères 8 bits ; ainsi la traduction n'est pas figée dans un exécutable. Voici un extrait de quelques lignes de ce fichier :

```
05\c{C} ; 128\c{C} DOS
05\"u} ; 129\"u DOS
03\'e ; 130\'e DOS
03\^a ; 131\^a DOS
..
06\"{\i}; 139\"{\i}DOS
..
03\<< ; 171 Guillemet ISO
..
03\'e ; 233\'e ISO
03\^e ; 234\^e ISO
03\"e ; 235\"e ISO
06\`{\i}; 236\`{\i}ISO
```

En effet, on peut souhaiter une définition particulière pour certains caractères, comme les guillemets à la française qui n'existent pas, hélas, dans le jeu des polices standards de *T_EX*. De plus, il est possible de faire cohabiter des jeux de caractères comme celui de IBM pour PC, avec celui de DEC (ISO) car les jeux ne se recouvrent pas⁴. Mais on peut par exemple souhaiter utiliser directement le jeu de macros des caractères graphiques IBM, mis au point par Maurice Laugier (cahier GUTenberg, n° 2, mai 1989) sans utiliser de traducteur spécialisé. La modification du compilateur ne concerne que la procédure modique à l'adresse de PKWARE Incorporation, et obtenir aussi une version plus à jour.

³ Ce fichier avait d'abord été testé sur PUB-T_EX.

⁴ Il est très pratique de pouvoir compiler un texte provenant d'un VAX sans utiliser de transcodeur.

`input_ln` et revient à remplacer chaque caractère supérieur à 128 en plusieurs caractères qui correspondent à sa définition.

Le compilateur utilise la version ascii (7 bits) du fichier de coupures de mots français mise au point par Jacques Désarménien, mais n'est pas capable de changer dynamiquement de fichier de coupures de mots comme le prévoit la version n° 3 de `TeX`. On a donc hélas, une commande `flatex` ou `f tex` pour le français. De plus, il s'agit d'un `TeX` standard qui ne sait pas couper des mots après un caractère accentué ; autrement dit pour les textes de largeur faible écrits en français, on doit faire appel au `TeX` multilingue de Michael Ferguson distribué par `PCTeX` pour les PCs ou à la dernière version de μTeX qui, elle aussi résoud le problème.

Il est clair pour tous les francophones que le problème primordial que pose l'utilisation d'un `TeX` standard en français est son impossibilité à couper des mots qui contiennent des caractères accentués⁵.

La possibilité offerte par la version 3 de multiples coupures de mot apparaîtrait comme un luxe si ce problème de simple coupure n'y était pas résolu.

La question est si primordiale que Michaël Ferguson et μTeX l'ont résolu ; leurs compilateurs sont les seuls compilateurs utilisés professionnellement. Aujourd'hui un même texte français ou plus généralement comprenant des caractères accentués, compilé avec différents compilateurs `TeX` fournit des fichiers DVI différents ! A notre avis, le fichier `TRIP.tex` de la version 3 devrait contenir des caractères accentués. Jacques Vallin de l'INED a proposé la phrase : «Éberluée, décontenancée, hébétée, défaite de toute sérénité devant cet abécédaire décapité, étêté, évidé, émasculé, désagrégé, débilité, dégénéré, parce que privé de lettres accentuées et correctement orthographiées, Thérèse-Éléonore a décreté avec sincérité et opiniâtré l'immortalité des çàçâçèçéçéççöç ûçûçü».

Après cette parenthèse qui n'est qu'un appel déguisé à D. Knuth, reprenons la description du compilateur *standard* `sb26tex`.

Le compilateur fonctionne sur XT (8086) mais est évidemment beaucoup plus rapide sur 386 ! Il

⁵ Le présent article, s'il a été compilé par un compilateur standard avec des coupures pourtant à la française, a dû faire l'objet de coupures manuelles pour éviter les "overfull boxes".

faut néanmoins 640K pour utiliser `LATeX`, et un disque dur assez rapide est nécessaire si on utilise beaucoup de polices car le compilateur décharge une partie de sa mémoire sur disque au besoin. Une distribution complète de `TeX`, `AMSTeX` et `LATeX` est fournie. Pour utiliser le minimum de place sur disque les fichiers de style (`.doc`) ne sont pas chargés sur disque au moment de l'installation standard mais se trouvent compressés sous le nom `latexdoc.zip`.

1.2 BIBTeX 0.99

L'anglais Niel Kempson a récemment proposé la dernière version de `BIBTeX` (0.99c) en source (Turbo-C) sur le serveur anglais d'Aston. Comme cette version ne pouvait pas lire les caractères 8 bits, nous avons modifié le source, d'une manière analogue à celle proposée par Wayne Sullivan, pour que le compilateur puisse lire un fichier optionnel `bibtex.poo` constitué des 128 premières lignes du `tex.poo` du compilateur `sb26` et où se trouvent les définitions des caractères 8 bits. Par défaut et en l'absence d'un tel fichier, une conversion standard, IBM+ISO, est utilisée. Le source sera mis sur le réseau GUTenberg, dès que celui-ci fonctionnera.

Il était nécessaire d'avoir des styles de bibliographie à la française ; c'est chose faite avec l'aide de Jacques Vallin (INED) qui s'est inspiré des recommandations de la Bibliothèque Nationale. Le fichier `fbtxbst.doc` permet de générer tant les styles anglais (`plain.bst`) que français (`fplain.bst`). Notons ici qu'il ne s'agit pas d'une simple traduction comme dans les styles `LATeX` mais d'une structure très différente propre au français et nettement plus standard qu'en anglais.

A propos de francisation de style et en l'absence de norme reconnue, nous avons inclus des petits fichiers d'option, de nom `arfrench.sty`, `refrench.sty`, `bofrench.sty`, respectivement pour les styles "article", "report" et "book" de `LATeX`.

1.3 L'éditeur de texte JOVE

Quelques versions du domaine public de l'éditeur `Emacs`⁶ existaient déjà sous DOS mais c'est seulement très récemment que l'excellente version `JOVE` (ou Jonathan's Own Version of `Emacs`) a été portée sur PC avec la plupart de ses potentialités sous Unix et de plus la capacité à gérer les caractères accentués (encore eux !). Sans rappeler toutes les qualités de cet éditeur, mentionnons qu'un embryon de mode `TeX` a été mis au point avec l'aide de Jean-

⁶ La "norme" `Emacs` a été mise au point par le MIT il y a quelques années.

François Vibert (CHU St Antoine-Paris), pour au moins vérifier l'appariement des parenthèses ou accolades et créer des masques vides à remplir pour les tableaux et dessins L^AT_EX mais aussi pour les bases bibliographique BIBL^AT_EX.

On considère ici qu'un éditeur a un véritable mode T_EX si une fonction, analogue à la commande `next-error` pour les autres langages, permet de lire le fichier `.log` de la compilation et de se positionner directement à la ligne du fichier source où se trouve une erreur. De même, il est souvent souhaitable avec des textes longs de ne sélectionner qu'une partie d'un texte, comme des équations particulièrement lourdes, puis de la compiler et de la visionner sans sortir de l'éditeur. Ceci est prévu en standard avec GNU-Emacs sur station de travail, mais est également possible sur PC avec une version commerciale de Emacs sur PC, à savoir Epsilon (distribué Lugaru Software aux États Unis et Santa Klaus en France). Pour ceux qui disposent d'Epsilon sur PC, le source de ce mode (langage C) est inclus dans les disquettes.

La documentation de JOVE a été traduite depuis "troff" en L^AT_EX et constitue près de 80 pages (fichier `jove.zip`).

1.4 La visionneuse à l'écran CDVI 1.0

Wayne Sullivan, encore lui, propose une visionneuse à l'écran extrêmement rapide de nom CDVI 1.0 du domaine public. La rapidité qui est sans comparaison avec les visionneuses connues à ce jour, provient en partie du fait que les polices de caractères, limitées en nombre sont "codées en dur" dans l'exécutable. Évidemment l'image à l'écran n'est pas une homothétie de l'image sur papier, mais c'est justement ce qu'on peut rechercher sur un écran qui a une définition moindre qu'une laser. Les notes en bas de pages apparaîtront en caractères plus gros donc lisibles à l'écran et réciproquement les gros caractères longs à afficher apparaîtront rapidement, plus petits, mais à des positions correctes (espacées). Même sur un PC/XT l'affichage d'un fichier de 100 pages est extrêmement rapide (la compilation par contre est fastidieuse). Pour un travail plus fin à l'écran, avec des grossissements interactifs, il faudra se procurer une visionneuse commerciale. L'inconvénient majeur de la version du domaine public de cette visionneuse est qu'elle ne peut pas visionner les fichiers qui utilisent d'autres polices que celles de Plain-T_EX. Il y alors deux solutions : soit on achète pour une somme très modique la version commerciale 2.0 de CDVI auprès de Wayne Sullivan (il y a alors beaucoup plus de polices et une possibilité de substitution des polices),

soit on bride les "formats" en les forçant à n'utiliser que les polices de Plain. Cette dernière solution peut être envisagée pour L^AT_EX sans grande perte de qualité (on s'aperçoit que L^AT_EX utilise beaucoup de polices). On remplace alors le fichier `lfonts.tex` par un fichier `plfonts.tex` (Plain-L^AT_EX-fonTS) ; évidemment les dessins L^AT_EX ne peuvent pas être visualisés correctement. D'autres visionneuses du domaine public semblent exister de part le monde, mais elles ne fonctionnent généralement que pour des cartes graphiques de type Hercules, nécessitant des polices de taille intermédiaire et non pas encore été testées.

1.5 Les pilotes d'imprimantes de Nelson Beebe

Pour l'impression sur papier, nous avons choisi la famille des pilotes DVIXXX (version 2.10) de l'américain Nelson Beebe. Les avantages de ces pilotes sont assez connus : très grande variété d'imprimantes tant de type laser que matriciel, fonctionnement sous de multiples systèmes d'exploitation dont VMS et Unix, disponibilité du source sur des serveurs pour piloter d'autres lasers, très grande rapidité d'exécution pour les pilotes lasers. Généralement, les pilotes et la documentation sont distribués avec les sources ; ici nous n'avons retenu que les exécutables disponibles sous DOS. Seul le manuel "DVIMAN" est inclus sous forme source et non le guide de référence pour la constitution d'autres pilotes. La présence de ce fichier source permet aux non-initiés d'apprendre à se servir de L^AT_EX par l'exemple.

Signalons ici, que l'impression d'un fichier T_EX sur une imprimante matricielle est extrêmement longue et bruyante. On ne se servira de ces imprimantes que pour des textes très courts. Par contre, la qualité est excellente. Regrettions ici que le pilote `dvieps` de Nelson Beebe, qualifié il est vrai d'expérimental, soit particulièrement lent comparé à certains produits commerciaux.

1.6 Une petite documentation sur L^AT_EX

Nous avons aussi inclus le source d'une documentation d'une quarantaine de pages intitulée "L^AT_EX en quelques pages" que nous avions faite pour des formations à l'INED. Il s'agit en introduction d'un plaidoyer contre les traitements de texte visuels et pour les systèmes dits structurés comme Scribe ou MINT et donc L^AT_EX, puis d'une introduction au langage L^AT_EX avec quelques exemples. Nous avons aussi ajouté quelques commentaires sur les commandes nécessaires à l'utilisation de L^AT_EX sur trois machines VAX/VMS, DOS et station de tra-

vail sous Unix (Ultronix), avec des résumés des commandes des éditeurs, visionneuses et pilotes sur chacune des machines. Cette documentation est un peu spécifique à l'environnement informatique de l'INED, mais peut fournir quelques renseignements utiles pour l'utilisateur des disquettes GUT.

2 Les outils généraux de communication

Pour les communications entre ordinateurs ou entre réseaux, un certain nombre d'outils sont nécessaires. Tout d'abord, lorsqu'on dispose d'un modem, il est assez judicieux d'utiliser Kermit pour le transfert de fichier. Kermit permet aussi d'émuler un terminal VT100 et un écran graphique Tektronix 4014. Pour le transfert de fichier par le courrier électronique, il est généralement recommandé d'encoder ses programmes exécutables en des caractères ascii imprimables avant de les transférer. Il existe beaucoup (trop) de programmes d'encodages ; vous trouverez sur les disquettes les plus connus : uuencode (Unix to Unix encode), btoa (binary to ascii), boo (for bootstrap). Auparavant, pour réduire la transmission il est d'usage de compresser le fichier à envoyer. On utilise généralement arc ou pkarc. Lorsqu'on désire transférer toute une arborescence et avant l'existence de pkzip, on utilise souvent l'outil tar (Unix tape archiver). Ces outils existent généralement sur tous les systèmes, et se trouvent sur les disquettes.

Nous avons ajouté d'autres logiciels divers. Nelson Beebe fournit par exemple avec ses pilotes, deux logiciels fort utiles pour les lasers PostScript : lptops pour imprimer un fichier ascii sous des formes très variées, et xport pour dialoguer (Xon/Xoff) avec une imprimante PostScript connectée à un port série.

Pour la création d'index (grave lacune de LATEX), Nelson Beebe propose texidx avec les styles qui s'y rapportent. L'utilitaire make (utilitaire Unix porté sous DOS) permet lorsqu'un fichier makefile est correctement agencé d'enchaîner différentes opérations de compilation, indexation, visualisation, impression. Enfin, nous avons aussi ajouté un transcodeur qui permet de transformer des caractères accentués 8 bits en leur équivalent TEX de nom tex8a7 et tex7a8.

2.1 Ce qui n'est pas sur les disquettes

Avec une imprimante de type PostScript, et le pilote dvialw de N. BEEBE il est possible d'utiliser la commande \special et d'inclure des dessins PostScript provenant de toutes sortes d'outils. Mais en l'absence de norme sur l'insertion des graphiques nous n'avons pas détaillé ces points et en partic-

ulier pas fait mention d'outils comme "psfig" ou "dvips" ou même d'outils graphiques du domaine public comme GNUPLOT, qui ont été portés sur PC.

Plus généralement, un certain nombre d'outils ou "texware" comme "tangle", "weave" ou les "pk-topxl" et autres avatars ne figurent pas sur ces disquettes. C'est une lacune, mais il est assez difficile de trouver un ensemble cohérent, du domaine public, fonctionnant sous DOS d'une manière non restrictive (DOS n'a que 640K adressable).

Pour le reste c'est à vous utilisateurs de signaler à l'association ce qui manque pour une version ultérieure.

3 Comment trouver ces disquettes ?

L'association GUTenberg ne dispose pas actuellement des structures nécessaires pour distribuer ces disquettes. Pour l'instant, nous avons fait appel à des personnes membres de GUTenberg qui sont volontaires pour faire des copies sous certaines conditions.

Ainsi, on peut obtenir les deux disquettes programmes 1.2Mo, les 3 disquettes 1.2Mo de polices de caractères pour laser (300 points par pouce) et optionnellement les 2 disquettes de police de caractères pour imprimante matricielle de type Epson (240 points par pouce) à condition d'envoyer au préalable le nombre exact de disquettes (haute densité, 5 pouces $\frac{1}{4}$). Ces disquettes doivent être formattées et munies d'un emballage qui sera retourné, d'une étiquette à l'adresse du demandeur ainsi que des timbres pour les frais de port en France.

L'expérience de ces derniers mois s'est révélée concluante à condition de respecter ces clauses puisque les volontaires n'ont pas changé. Adressez vos disquettes vierges, en respectant si possible la zone géographique, à l'une des adresses suivantes :

- A. Cousquer. Labo. d'informatique fondamentale de Lille. Université de Lille 1, 59655 Villerupt Cedex. FRANCE.
- G. Weil, Centre de calcul CNRS, 23, rue du Loess, 67200 Strasbourg, FRANCE.
- C. de Montcuit, LIMSI / CNRS, BP 133, 91403 Orsay Cedex, FRANCE.

Pour l'étranger francophone, il serait souhaitable d'avoir au moins une adresse au Canada, en Belgique et en Suisse. Le problème n'est qu'un problème de frais de port pour l'étranger. En prévision d'un abandon bien compréhensible de la part d'un volontaire, il serait souhaitable que les éventuels nouveaux volontaires se fassent connaître auprès de l'association GUTenberg dont nous rap-

pelons ici l'adresse : Association GUTenberg,
c/o IRISA Campus de Beaulieu, 35042 Rennes
Cedex.

Évidemment chacun peut et même doit distribuer les disquettes à son entourage en visant particulièrement les universités, centres de recherche, ou même lycées avec classes préparatoires sans grand moyen mais disposant peut-être d'un PC. Un fichier de nom *etiq.dvi* permet d'imprimer des étiquettes qui améliorent grandement la présentation des disquettes.

Plus généralement et même si certains commentaires sont en français les disquettes GUT ont une vocation internationale puisqu'il s'agit des premiers compilateurs *TeX* et *BIBTeX* du domaine public à pouvoir lire les caractères accentués. Les disquettes GUT ne sont qu'un premier pas vers une distribution internationale de *TeX*. Pour les langues européennes, les différentes associations d'utilisateurs peuvent recevoir ces disquettes pour ajouter les fichiers de coupures de mots propres à leur langue, et traduire tant les styles que les commentaires. Il y a néanmoins quelques obstacles à cette internationalisation aujourd'hui dont par exemple (1) la nécessité d'une version n° 3 de *TeX* qui respecte les coupures de mots tant françaises que scandinaves ; (2) la nécessité d'une rédaction des macros *LATEX* plus générale qui, à l'instar des propositions du groupe allemand DANTE, autorise un unique fichier d'option propre à une langue comme le fichier *german.sty* ; (3) enfin la correction de *BIBTeX* qui, même avec la version 0.99 n'autorise pas, comme l'a fait remarquer un membre de DANTE récemment, l'écriture de la commande :*MACRO {mar} {"M{\\"a}rz"}*.

Malgré ces imperfections, cette première distribution complète sur disquette a le mérite d'exister et de répondre à la demande formulée lors du dernier congrès GUTenberg par des utilisateurs de *TeX* qui ne peuvent accéder aux réseaux électroniques internationaux. Souhaitons qu'une telle distribution fasse connaître et reconnaître les qualités de *TeX* et de *LATEX*.

◊ NICOLAS BROUARD
Institut National d'Etudes
Démographiques
27, rue du Commandeur
75675 Paris Cedex 14
Tel 33 (1) 43 20 13 45
Bitnet: *brouard@frined51*

Macros

Unusual Paragraph Shapes

Victor Eijkhout

Introduction

Although *The TeXbook* [1] states that *TeX*'s paragraph mechanism 'can be harnessed to a surprising variety of tasks', the strangest paragraph shapes that I have implemented use no feature of the line-breaking algorithm. Instead, I have found that the control sequences *\everypar* and *\lastbox* are extremely powerful tools. I give three examples of this.

How many *TeX*ers does it take to typeset a lightbulb? [2]

A while back I found somewhere in Netland a largish collection of jokes—several hundreds of them—consisting of a question of the 'above form, and an answer to it. It seemed to me that this document would make a nice gift, if properly typeset and bound. The following layout was suggested to me [3]:

How many evolutionists does it take to screw in a light bulb?

Only one, but it takes eight million years.

How many folk singers does it take to screw in a light bulb?

Two. One to change the bulb, and one to write a song about how good the old light bulb was.

As all jokes¹ had a question mark concluding the first line, and the file had blank lines separating the jokes, the following macro should do the job when placed in front of each joke:

```
\def\lbj#1? #2\par{\{\\it
    \hangfrom{\vrule width 1in height 1ex
              depth 0in \kern .5em}#1\par}
    #2\par}
```

Only I didn't feel like putting some command in front of a couple hundred jokes. Then I remembered about *\everypar*, the token list that is inserted in front of every paragraph. If I would set

```
\everypar={\lbj}
```

¹ I guess the answer to the above question is: they are still waiting for the *dvi2lightbulb* program

the jokes would automatically be scooped up as the arguments of `\lbj`.

Well, *almost*. The snag is that `\lbj` itself produces two paragraphs, before each of which `\everypar` is inserted. With infinite recursion as a result. Repair may take the following form:

```
\everypar={\bgroup\everypar={}\\lbj}
\def\\lbj#1? #2\\par{ ... %% the same
... %% as above
#2\\par\\egroup\\bigskip}
```

Now `\everypar` opens a group, inside which it is empty, and `\lbj` inserts the closing brace of the group, so after the joke `\everypar` is ready for the next joke again.

What I particularly like about this approach is the fact that, with only five lines of macros, I can typeset arbitrarily much text. No TeX commands appear after a short preamble.

A Maserati, a Mack Truck! [4]

In the previous example, the `\everypar` takes the whole paragraph as its argument. This is not a wise approach, however, as it places memory demands on TeX in case of long paragraphs, and worse, it means that the text is scanned twice, which is a disadvantage if you work with `\catcodes`. One may instead define

```
\everypar{\catchthepar\everypar={}}
\def\\par{\endgraf\\egroup}
```

with for instance

```
\def\catchthepar{\vbox\\bgroup}
```

I took this approach when I implemented ‘vario setting’² of text, a paragraph layout where lines are right adjusted, except when they have to stretch beyond some point. In that case they remain ‘ragged right’. This way of setting text has in recent years become very fashionable in advertisements, especially for automobiles. Provided the paragraphs are suitably long, say five lines or more, and the right proportion of lines is left unadjusted, the visual effect of a block with an occasional gap in the right margin can be, well, interesting. Some typographers, however, loathe this concoction [3].

When I first tried to implement this, I fiddled around with the usual parameters, but that led to nothing. Also I didn’t want to take the approach

² This is the literal, and imperfect, translation of the Dutch term. I don’t know the English equivalent.

from [5], as this would preclude hyphenated words. The only other way that I could think of, was to let TeX set the paragraph right adjusted, and afterwards to unset the glue in some lines. Getting the lines from the paragraph would be possible, I thought, using `\lastbox`.

As it turned out there were two difficulties. The first one was that lines that hang on the main vertical list can’t be picked with `\lastbox`. I solved this by a variation of the above construct:

```
\everypar={\vbox\\bgroup
\everypar={}
\def\\par{\endgraf\\pickthelines
\\egroup}}
```

Implementing `\pickthelines` brought to light the second difficulty: successive calls to `\lastbox` yield void boxes after the first call. At least, this was how it looked to me at first.

After a while I realised that the real reason was that a paragraph consists of an alternation of a box, a penalty, and a glue item, and so on. So I arrived at

```
\def\pickthelines{
\setbox\\investigation=\\lastbox
\ifvoid\\investigation \\else
\\unskip \\savepenalty=\\lastpenalty
\\unpenalty
{\\pickthelines}
\\investigatethebox
\\penalty\\savepenalty \\fi}
\def\\investigatethebox
\\setbox\\tester=\\hbox
{\\unhcop\\investigation}
\ifdim\\wd\\tester<.94\\hsize
\\box\\tester
\\else \\box\\investigation \\fi}
```

The recursion in `\pickthelines` uses TeX’s save stack, so this macro is not suited for very long paragraphs. But that’s not what it is intended for.

Note that TeX automatically reinserts the correct baselineskips, but that I have to do the appropriate penalties myself. The value beyond which a line should ‘snap back’ has to be determined experimentally; it highly depends on the `\hsize`, and the average word length of the text. Also, in order to prevent multi-line gaps, a test should be implemented if the previous line was set at natural width. Further fine tuning may involve the `\spaceskip` and the `\tolerance`.

Romeo, wherfore art thou indented? [6]

Browsing through Shakespeare’s complete works, I lighted upon a phenomenon that I thought would

be pretty hard for T_EX. Rather than explaining it at length, I'll just show an example.

Jul. Romeo!

Rom. My dear!

Jul. At what o'clock to-morrow shall I send to thee?

Rom. At the hour of nine.

Jul. I will not fail; 'tis twenty years till then.

The input for this example is:

```
\open
\Jul Romeo!
\Rom[ My dear!
\Jul[ At what o'clock to-morrow
     shall I send to thee?
\Rom[ At the hour of nine.
\Jul I will not fail; %
     'tis twenty years till then.
\close
```

The reason for the strange indentation lies in a formal principle of Shakespeare's plays: all lines are in iambic meter which implies that the number of syllables is somewhat predetermined. As this number is nowhere near reached after an exclamation like 'Romeo!', the addressed party will have to finish the line for Juliet. He falls far short of this ideal, by the way, so the lady has to finish the job herself.

For this paragraph shape I saw no use for \everypar, but \lastbox would certainly be needed, namely to measure the last line of the previous player. In order to be able to use \lastbox all the text would have to be included in \vboxes. Each player should then do the following:

- measure the last line of the previous player;
- close the text of that player and hang it to the page; and
- then indent by the measured amount if a square bracket follows.

With some precautions for the first and last line of a scene, the basic commands look like:

```
\def\player#1 {\expandafter
  \def\csname#1\endcsname
    {\global\def\name{#1}\playerline}}
\player {Rom} \player {Jul}
\newif\ifplaying % action started yet?
\open\obeylines \playingfalse
\close{\closepreviousplayer}
\playerline{\ifplaying
  \closepreviousplayer
  \else \playingtrue \fi
  \futurelet\next\maybeindentedline}
```

The job of 'closing the previous player' consists of taking the last box, measuring it, and hanging it back on the list. I leave the details to the reader.

```
\def\closepreviousplayer
  {\getpromptlength\egroup\unvbox\lines}
```

After the previous player has been closed, a new one can be opened, i.e., the box \line can be opened, with an indentation if necessary.

```
\def\maybeindentedline{\if\next[
  \expandafter\eatbracket
  \else \global\promptlength=0cm
    \nonindentedline \fi}
\def\eatbracket[\{\nonindentedline]
\def\nonindentedline{\setbox\lines=
  \vbox\bgroup\leavevmode\strut
  \setbox\tempbox=\hbox
    {\it\name\enspace}%
  \ifdim\promptlength>\wd\tempbox
    \hbox to \promptlength
      {\it\name\hfil}%
  \else \box\tempbox \fi}}
```

For the complete layout used in the edition of Shakespeare's plays I have, these macros will have to be augmented by provisions for narrow columns: a too long line is split with the remainder set flush right.

Conclusion

Aberrant paragraph shapes can be automated in T_EX to such an extent that no commands need to be inserted in the actual text. The token list \everypar is a handy tool for this, and for applications involving the length of lines \lastbox is a powerful instrument.

References

- [1] Donald Knuth, *The T_EXbook*, 1986.
- [2] Anonymous, The canonical collection of light bulb jokes.
- [3] Inge Eijkhout, conversations, 1988/9.
- [4] The Tubes, What do you want from life, 1975.
- [5] Anne Brüggemann-Klein, First line special handling with T_EX, *TUGboat* 8, no. 2, July 1987, pp. 193–197.
- [6] William Shakespeare, Romeo and Juliet, 1594.

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

TEXT1 Goes Public Domain

Dean Guenther

TEXT1 is a macro package for use with plain that has been available for purchase for several years. Beginning in January 1990, *TEXT1* went into the *TeX* public domain collection of macros.

In 1983, a decision was made at Washington State University to put time into creating a set of macros to be used with *TeX*. This macro set was to provide many of the functions found in *LATEX*: multiple columns, table of contents, indexing*, chapters, lists, boxes, margin notes, etc. Yet there were to be two important differences:

1. The *PLAIN.TEX* commands would be able to be used in *TEXT1*.
2. Modifying the *TEXT1* formats would be considerably easier than modifying *LATEX* styles.

The resulting set of macros met the above criteria. One important exception was that no attempt was made to give *TEXT1* a picture environment such as the one in *LATEX*. The documentation for *TEXT1* consists of a 300 page Reference Manual and a 100 page Users Guide. These must still be purchased; see below for details.

Changing the Formats

As an example of how easy it is to modify the *TEXT1* formats, if you want to increase the bottom margin from 6 to 7 picas, you can simply specify:

```
\pageformat{\bottommargin{7pc}}
```

Or if you wanted your running head to contain the chapter title left-justified and the page number right-justified on odd pages, and the chapter title right-justified and the page number left-justified on even pages, you could say:

```
\runningheadformat{
  \oddpages{\line{\lft{\chapttitle}
    \rt{\pagenumber}}}
  \evenpages{\line{\lft{\pagenumber}
    \rt{\chapttitle}}}
}
```

For one last example, it is often desirable to change the text embedded within macros. For example, by default if you say:

```
\chapter{Introduction}
```

* The three different index styles illustrated in Appendix A in *TUGboat* 1, no. 1 can all be achieved with *TEXT1*'s index macros.

you would get "Chapter 1" at the top of the next page in bold, with "Introduction" centered below that. If you were using *TEXT1* in Germany, you would probably prefer "Kapitel" instead of "Chapter", so you could change the chapter's format to say:

```
\chapterformat{
  \titleformat{\newpage
    \centerline{\%
      \bf Kapitel \chapernumber}}
    \centerline{\bf\chapttitle}
    \vs{1\bl}
  }
  \incontents{yes}}
```

Then in your text you would specify the introduction (first chapter) as:

```
\chapter{Einleitung}
```

Additional Products

Other products that came out of this project and are now going into the public domain are a typesetter driver for the Compugraphic 8600 and additional fonts.

Typesetter Driver. The Compugraphic driver is written in *WEB*. It has the ability to add line numbering to the DVI output as it is printed on the 8600. The program does include some Pascal statements unique to IBM's Pascal/VS, so a change file would be necessary for any operating system other than IBM's VM/CMS.

Additional Fonts. Extra fonts were created from the base Computer Modern. These were Computer Modern at 11, 12, 14, 18, 24 and 36 point sizes. I do not know yet if I can call these Computer Modern, but I have asked Professor Knuth for permission to do so.

In addition to the larger Computer Modern fonts, we also created an International Phonetic Alphabet in 9, 10, 11 and 12 point sizes. The IPA font is illustrated at the end of this article.

How to get *TEXT1*

You can get any of the above *TEXT1* products on diskette or by anonymous FTP. Also, I will be distributing *TEXT1* to the various site coordinators who can determine whether to place it on their distribution.

To obtain *TEXT1* on diskette from Jon Radel, send a note to

Jon Radel
 P.O. Box 2276
 Reston, VA
 22090-0276

and ask for *TEXT1* (4 disks). To cover postage and other expenses, if you are in North America, please enclose \$1.50/disk if you send blank disks or \$5.00/disk if you don't; if you are anywhere else in the world, enclose \$2.00/disk if you send blank disks or \$6.00/disk if you don't. *μTEX* and *PC TEX* use different directories, so specify which of these two you are using. If you are using neither of these, then just specify *PC TEX*. You should also specify whether you want pk (3 disks) or gf (5 disks) font files for the additional fonts (these do not include the IPA fonts).

You may also request *TEXT1* for a Macintosh from Jon (1 disk). It is set up to run with *Textures*.

To order the IPA fonts from Jon, request the WSU IPA font. This consists of two diskettes: 1) the IPA source files and Users Guide, and 2) pk or gf files.

To get *TEXT1* by anonymous FTP, you can connect to the machine BOBCAT.CSC.WSU.EDU. If that does not work, use 134.121.1.1. Bobcat is a VAX 11/785, and is prone to being a bit slow at times. Log on as ANONYMOUS, (the password *must* be GUEST), and change to the *TEXT1* directory. You will see ten subdirectories from which you can GET files:

1. **BLOCKS:** this directory contains all of the default formats (or building "blocks" such as the *chapterformat*, *runningheadformat*, etc.)
2. **CMS_HELP_FILES:** these are some of the help files used on the CMS system at WSU. Some sites have taken these files and used them in creating help files on a PC or on a VAX.
3. **COMPUTERGRAPHICS_8600:** this is the Compugraphic 8600 phototypesetter driver program, and its various utilities. Not very much documentation here, so venture further on your own.
4. **FONTS:** This is the source for the additional Computer Modern fonts generated, such as 11, 14, 18, 24 and 36pt sizes.
5. **MACINTOSH:** this contains the index sorting facility used with *TEXT1*. It is written for Turbo Pascal. (The executable binary can be obtained from Jon Radel if you do not have Turbo Pascal yourself.)
6. **MACROS:** this contains the *TEXT1* macros. The file *TEXT1.TEX* is used with *InitEX* to create a new fmt.

7. **MODELS:** contains various simple models such as letter, resume, etc. It also contains a model (*MERGETXT*) which can be used as a model for merge letters. For example, you could tell Dbase III to create an address list, merge it with a standard letter, and using the model print a copy for each person.
8. **TEXIX:** the sorted index program used with *TEXT1*. Presently the only systems this is running on are CMS and Macintosh. *TEXIX* for VAX and PC systems is in the works.
9. **VAX_VMS:** when ready, this will have the sorted index for VMS users.
10. **WSUIPA:** This contains the source for the International Phonetic Alphabet, as well as the tfm and 300pk files.

TEXT1 Documentation

The documentation for *TEXT1* is not free. At present, the documentation must be ordered through:

TEXT1 Distribution
 Computing Service Center
 Washington State University
 Pullman, WA 99164-1220

The cost is \$35 for the Reference Manual and \$25 for the Users Guide. (Add \$5 postage if outside the U.S.) The manuals are not yet available through the *TEX* Users Group, but I hope to have them there as soon as I find a publisher.

If you have any questions, feel free to send me an email note on Bitnet at *GUENTHER@WSUVM1* or to *GUENTHER@WSUVM1.WSU.EDU* on Internet.

◊ Dean Guenther
 Computer Service Center
 Washington State University
 Computer Science Building
 Pullman, WA 99164
 Bitnet: *GUENTHER@WSUVM1*

See following page for display of IPA font.

WSUIPA10

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	æ	ɑ	ɑ	ɔ	ʌ	ɒ	ə	ø	
'01x	ɛ	β	ɸ	ɛ	ɔ	ð	ð	ð	"0x
'02x	d	ð	ðʒ	ð	D	ə	ð	θ	
'03x	ɛ	ɜ	ɔ̄	ɔ̄	g	ḡ	G	γ	"1x
'04x	ɣ	χ	hv	ħ	ħ	ħ	q	ī	"2x
'05x	‡	ı̄	I	‡	j	‡	‡	‡	
'06x	l̄	ł̄	λ̄	χ̄	m̄	w̄	w̄	n̄	"3x
'07x	ŋ̄	ɳ̄	N̄	∅̄	θ̄	ɔ̄	ω̄	ω̄	
'10x	ꝝ	ꝑ	ꝑ	ꝑ	r̄	r̄	ꝑ	ꝑ	"4x
'11x	ꝑ	l̄	R̄	ꝑ	ꝑ	ſ̄	ſ̄	σ̄	
'12x	t̄	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	"5x
'13x	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	
'14x	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	"6x
'15x	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	
'16x	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	"7x
'17x	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	ꝑ	
	"8	"9	"A	"B	"C	"D	"E	"F	

IPA font available from Washington State (see previous article).

Solution to crossword of *TUGboat* 10, no. 3:



Showing-off math macros

Michael J. Wichura

Introduction

The chore of typing math formulas is made less onerous by using macros to \define simple abbreviations (such as \xvec) for complicated constructions (such as (x_1, \dots, x_n)) that occur repeatedly in a document. This saves keystrokes and cuts down on typographical errors and inconsistencies. The only difficulty is remembering the abbreviations and what they stand for. This article presents a macro-listing command that is helpful in this regard. The

Macros on file: `macros.math`

Name and Parameter Text	Value
\xvec	(x_1, \dots, x_n)
\row#1	$(\#1_1, \dots, \#1_n)$
\Row#1#2	$(\#1_1, \dots, \#1_{\#2})$
\ex	$e^{-x^2/2}$
\e#1	$e^{-\#1^2/2}$
\tenth	$\frac{1}{10}$
\fr#1/#2	$\frac{\#1}{\#2}$
\eunm	$\langle \frac{n}{m} \rangle$
\euler	$\langle \rangle$
\tnk	$_n C_k$
\take#1\of#2	$\#2 C_{\#1}$

(`macros.math` is the name of the file containing the \definitions used in this example.) The table is more informative than a simple print-out because it shows what each macro does. You can easily locate the macro that produces a certain construction by scanning down the middle column.

The macro-listing command is

`\ListMacrosOnFile file_name`

where `file_name` is the name of a file containing \definitions of math-mode macros, and only such \defs. This file would be one that you'd ordinarily \input as part of a document: no special organization is required. Several definitions can be given on a single line. A single definition can extend over several lines. Blank lines are allowed, as are \input commands referring to files containing more math-mode macros.

command typesets a file containing math macro \definitions, such as

```
\def\xvec{\row x}
\def\row#1{\Row{#1}n}
\def\Row#1#2{(#1_1,\ldots,#1_{#2})}
\def\ex{\e x}
\def\e#1{e^{-\#1^2/2}}
\def\tenth{\fr 1/10 }
\def\fr#1/#2 {\textstyle{\#1\over #2}}
\def\eunm{\{n \euler m\}}
\def\euler{\atopwithdelims <>}
\def\tnk{\take k \of n}
\def\take#1\of#2{\{#2\}C_{\#1}}
```

in the form of a table, like this:

Replacement Text
\row x
\Row {#1}n
(#1_1,\ldots,#1_{#2})
\e x
e^{-\#1^2/2}
\fr 1/10
\textstyle{\#1\over #2}
{n \euler m}
\atopwithdelims <>
\take k \of n
\{#2\}C_{\#1}

The table that \ListMacrosOnFile produces contains a row for each \def on `file_name`. Recall that the syntax for a macro definition is

```
\def<control sequence name>(<parameter text>
  <replacement text>)
```

(see page 203 of *The TeXbook*). \ListMacrosOnFile places the macro's name and parameter text in the first column of the table, with spaces in the parameter text being shown as '`\`'. It's important to take note of those spaces, because they can play a role in delimiting arguments to the macro; see pages 203–204 of *The TeXbook*. The second column shows the effect the macro has when it is typeset between a pair of \$ signs and invoked with arguments `\rm \#1`, `\rm \#2`, ... corresponding to the parameters `#1`, `#2`, ... in an obvious manner. For example, the "value" of the \take macro in the

opening example was created by the construction '\$\take{\rm #1}\of{\rm #2}\$. Finally, the macro's replacement text is given in the third column. An overwide entry in the first or second column is allowed to stick out to the right and causes its row to be continued on the next line. A long entry in the third column is carried over to subsequent lines.

There are two significant limitations on the use of \ListMacrosOnFile. The first has already been mentioned but is worth repeating: *file_name* and its \input extensions must contain only \definitions of macros that can be used in math mode.* For example, '\def\AB{\alpha + \beta}' is allowed, but '\def\AB{\$\alpha\$ and \$\beta\$}' is not, since in this case the construction '\$\AB\$' would elicit an error message from TeX. The second limitation is that the expansion of the replacement text for each macro must be closed with respect to groups. For example, '\def\BeginBox{\setbox0 = \hbox\bgroup}' is not allowed because the group opened by \bgroup isn't closed by a matching \egroup.

\ListMacrosOnFile will correctly handle any parameter text TeX allows, with two minor exceptions: that text must not contain the construction ' \rightarrow ', nor end with the character '#'.

*It would be nice if there were a simple, automatic way to show what a formatting macro like plain TeX's \begin{section} command does. Unfortunately, such macros typically have a wide-ranging effect that can't be encapsulated in a table entry.

```
\def\BeginAlignRow{%
  \xdef\AmpsSeenSoFar{}%
  \ialign \bgroup
    \BeginColumn ##\EndColumnOfWidth\NameColumnWidth
    &\BeginColumn ##\EndColumnOfWidth\ValueColumnWidth
    &\BeginColumn ##\EndColumnOfWidth\ReplacementColumnWidth
    \cr}
\def\EndAlignRow{\egroup}
```

and their subsidiaries \BeginColumn and \EndColumnOfWidth:

```
\def\BeginColumn{\setbox0 = \hbox \bgroup}
\def\EndColumnOfWidth#1{%
  \ifLastColumn \egroup % now box0 holds the entry
  \box0
```

The macros

\ListMacrosOnFile is implemented through a collection of interrelated macros. I'll first discuss the ones that deal with alignment. \ListMacrosOnFile doesn't use an \halign to construct its table because extremely wide entries in the first or second column could push the entire third column off the page, and because a long list of \defs on *file_name* could produce a multipage table exceeding TeX's memory. An alignment mechanism that uses preset column widths avoids both of these problems because it allows each row to be typeset independently of the other rows. Plain TeX's \+ command functions in this way, but lets an overwide entry in one column overlap the following one, producing illegible effects like this. \ListMacrosOnFile employs an "in house" alignment in which each row is individually set using an \halign, the templates of which position entries flush left in columns of preset widths and avoid overlaps by inserting a \cr and an appropriate number of &'s after an overwide entry.

The \SetColumnWidths command defined by

```
\def\SetColumnWidths#1#2#3{%
  % #1, #2, and #3 are dimensions
  \def\NameColumnWidth[#1]%
  \def\ValueColumnWidth[#2]%
  \def\ReplacementColumnWidth[#3]}
```

is used to set the three column widths to user-defined values. The macros specify

```
\SetColumnWidths
  {.25\hsize}{.20\hsize}{.55\hsize}
```

to establish defaults that experience has shown to work out fairly well. The alignment macros are \BeginAlignRow and \EndAlignRow:

```

\else
  \egroup % now box0 holds the entry
  \setbox2 = \hbox to #1{\unhcopy0 \hss}%
  \copy2
  \xdef\AmpsSeenSoFar{\AmpsSeenSoFar &}%
  \ifdim\wd0 > \wd2
    \xdef\DropDownToNextLine{%
      \noexpand\LastColumntrue\cr
      \noalign{\noexpand\nobreak}%
      \AmpsSeenSoFar}%
    \xdef\AmpsSeenSoFar{}%
    \aftergroup\DropDownToNextLine
  \fi
\fi}
\newif\ifLastColumn % false by default

```

The `\ifdim` clause at the end of `\EndColumn...` is what prevents overlaps. To see what's involved, suppose the `\ifdim` test has just discovered that an entry in the second column is overwide (`\wd0 > \wd2`). Then the `\aftergroup` command will effectively insert the tokens `\LastColumntrue \cr \noalign{\nobreak} &&` in front of the input text for the third column, causing TeX to finish off the current line with an empty third column, issue a penalty preventing a page break, and start a new line with two empty columns, before going on to

set the next entry. `\ListMacrosOnFile` specifies `\LastColumntrue` when it's working on the third column, so that that column is never considered to be overwide.

`\ListMacrosOnFile` itself comes next. It first `\inputs file_name`, so that all the macros there will be defined, and creates the header lines for the table. It then `\inputs file_name` once again, but with the meaning of `\def` changed to `\BeginExhibitMacro`. `\def`'s normal meaning is restored after all the macros on `file_name` have been exhibited:

```

\def\ListMacrosOnFile #1 {%
  \par \rm
  \input #1
  \leftline {Macros on file: {\tt #1}}
  \medskip
  \leftline {\it Name and}
  \BeginAlignRow
  \it Parameter Text\quad \it Value\quad \it Replacement Text\cr
  \EndAlignRow
  \ListMacros
  \input #1
  \DontListMacros}

\def\ListMacros{\let\def = \BeginExhibitMacro}
\def\DontListMacros{\let\def = \PrimitiveDef}
\let\PrimitiveDef = \def

```

`\ExhibitMacro`'s job is to construct a row displaying a macro's name and parameter text (e.g., `\take#1\of_#2`), value (e.g., `#2C_#1`), and replacement text (e.g., `{}_-{#2}C_{#1}`). Since the name and texts immediately follow the `\def` on `file_name`, there would at first sight seem to be no difficulty in fabricating the entries for the first

and third columns. It turns out, though, that the parameter text on `file_name` has to be used to create the value entry in the second column, and that (due to category code considerations) this precludes the use of that same text in the first column. The way out of this bind is to ask TeX for the `\meaning`

of $\langle name \rangle$ and to extract the relevant texts from TeX's response

$\text{macro} : \langle \text{parameter text} \rangle \rightarrow \langle \text{replacement text} \rangle$

```
\def\BeginExhibitMacro#1{%
  #1 = macro's name
  \DontListMacros
  \smallskip
  \BeginAlignRow
    \gdef\Name{\#1}% save name for later use
    \expandafter\ExtractTexts\meaning\#1\EndExtractTexts
    \tt \frenchspacing
    \expandafter\string\Name
    \expandafter\MakeSpacesVisibleA \ParameterText\EndMakeSpacesVisible
    \quad
  &
    \vtop\bgroup\Argument={}\BuildArgumentA
  }

\def\EndExhibitMacro#1{%
  #1 = macro's replacement text
  \ialign {\span\the\ValueTemplate\cr
    \expandafter\Name\the\Argument\cr}
  \egroup
  \quad
  &
  \vtop {%
    \hsize = \ReplacementColumnWidth \tt \frenchspacing
    \noindent \hangindent=1em \rightskip=0pt plus 4em \relax
    \ReplacementText}%
  \LastColumntrue
  \cr
  \EndAlignRow
  \ListMacros}

\def\ExtractTexts #1:#2->#3\EndExtractTexts{%
  \gdef\ParameterText{\#2}%
  \gdef\ReplacementText{\#3}}
```

The replacement text is set ragged right in a paragraph having the width of the third column. The following macros are used in converting the

All the non-space characters in TeX's response have category code 12, so the extracted texts can be typeset without using a verbatim command:

```
\def\\{\let\SpaceToken= }\\
\def\SpaceChar{\char`\\ }

\def\MakeSpacesVisibleA{\futurelet\NextToken\MakeSpacesVisibleB}
\def\MakeSpacesVisibleB{\ifx \NextToken\SpaceToken \SpaceChar \fi
  \MakeSpacesVisibleC}
\def\MakeSpacesVisibleC#1{%
  \ifx #1\EndMakeSpacesVisible \else #1\expandafter\MakeSpacesVisibleA \fi}
\def\EndMakeSpacesVisible{}
```

(invisible) spaces in the parameter text to visible characters («'s) before that text is placed in the first column:

\ExhibitMacro's code for the second column starts out by fabricating an "argument text" (e.g., `\{\rm \#1\}\of \{\rm \#2\}`) out the parameter text on *file_name*. This operation is a little delicate, because category codes have to be assigned

```
\def\BuildArgumentA{\futurelet\NextToken\BuildArgumentB}
\def\BuildArgumentB{%
  \ifx \NextToken\bgroup
    \def\NextCmd{\EndExhibitMacro}%
  \else \ifx ##\NextToken
    \def\NextCmd{\AddParmFieldToArgument}%
  \else \ifx \NextToken\SpaceToken
    \def\NextCmd{\AddSpaceToArgument}%
  \else \def\NextCmd{\AddOtherToArgument}%
  \fi \fi \fi
  \NextCmd}
\newtoks\Argument
\def\AddToArgument#1{\edef\temp{\the\Argument#1}%
  \Argument=\expandafter{\temp}%
  \BuildArgumentA}
\def\AddParmFieldToArgument#1#2{%
  \AddToArgument{\{\noexpand\rm \noexpand\##2\}}}
\def\AddSpaceToArgument#1 {\AddToArgument{space}}
\def\AddOtherToArgument#1{\AddToArgument{\noexpand #1}}
```

The value entry for the second column is created by typesetting the construction '*macro name* *(argument text)*' (e.g.,

`\take \{\rm \#1\}\of \{\rm \#2\}`) in a one row, one column \halign whose template is `$$` by default:

```
\newtoks\ValueTemplate
\def>ShowOffMathMacros{%
  \ValueTemplate={$##$}}
>ShowOffMathMacros
```

There are, however, two other possibilities

```
\def>ShowOffMathMacrosInDisplayStyle{%
  \ValueTemplate={$\displaystyle{##}$}}
\def>ShowOffOrdinaryMacros{%
  \ValueTemplate={##}}
```

that weren't mentioned in the introduction. Specifying \ShowOffMathMacrosInDisplayStyle causes math macros to be exhibited in display style so that, for example,

```
\ListMacros
\def\SumA{\Sum i 1 n}
\def\Sum#1#2#3{\sum_{#1=#2}^{#3}}
\DonListMacros
```

correctly, T_EX's rules regarding spaces have to be obeyed, and active characters and control sequences (like the \of in the \take macro) mustn't be expanded. The following macros are used in building up the argument text token by token.

will produce

\SumA	$\sum_{i=1}^n$	\Sum i 1 n
\Sum#1#2#3	$\sum_{#1=#2}^{#3}$	\sum_{#1=#2}^{#3}

instead of the default

\SumA	$\sum_{i=1}^n$	\Sum i 1 n
\Sum#1#2#3	$\sum_{#1=#2}^{#3}$	\sum_{#1=#2}^{#3}

Non math-mode macros (e.g., abbreviations for long words) can be exhibited by specifying \ShowOffOrdinaryMacros.

- Michael J. Wichura
Department of Statistics
Computation Center
University of Chicago
5734 University Avenue
Chicago, IL 60637
wichura@galton.uchicago.edu

How to Avoid Writing Long Records to TEX's \write Streams

Peter Breitenlohner

In his article 'Macros for Indexing and Table-of-Contents Preparation' in *TUGboat* 10, no. 3 (1989), pp. 394–400, David Salomon mentions a problem with very long records written to one of *TEX*'s *\write* streams (p. 399): "... Since each line of the table of contents goes on the file as a record, its size is limited and, as a result we cannot have chapter or section names which are too long. ..."

There seems to be a similar problem in *LATEX*, since *LATEX* users frequently request that *TEX* implementations be able to write very long records (for CMS up to 1024 characters).

There is, in fact, a very simple method to avoid this problem: split the output into several records at all those points where a new line starts in the user's input. The macro *\chap* defined below (to be used with *plain.tex*) demonstrates how this can be done. The argument of *\chap* is typeset in bold face (with line end characters converted to spaces) and written (*\immediately*) to a file.

```
\newwrite\ind
\def\chap{\bgroup
  \catcode`\^^M\active \chapx}
\begin{group}
  \catcode`\^^M\active
  \gdef\chapx#1{%
    \let^^M=\relax%
    \newlinechar`\^^M%
    \immediate\write\ind{%
      \string\ch:#1\string\\}%
    \let^^M=\space%
    \par{\bf#1}\par\egroup}
\end{group}

\immediate\openout\ind=\jobname.ind
\chap{This is
  one (immediate)
  potentially
  very long text}
\immediate\closeout\ind \vfill\break
```

The definition of a similar macro *\Chap*:

```
\newwrite\inx
\def\Chap{\bgroup
  \catcode`\^^M\active \Chapx}
\begin{group}
  \catcode`\^^M=\active
  \gdef\Chapx#1{\bgroup%
    \lccode`'*`\^^M%
    \lowercase{\egroup \def^^M{*}}%
```

```
\edef\x{\string\ch:#1\string\page}%
\write\inx\expandafter{\x%
  \number\pageno\string\\}%
\let^^M=\space%
\par{\bf#1}\par\egroup}
\endgroup
\newlinechar='^^M

\openout\inx=\jobname.inx
\Chap{This is
  another (delayed)
  potentially
  very long text}
\closeout\inx \vfill\break
```

demonstrates that things are only slightly more complicated for a delayed *\write*.

◊ Peter Breitenlohner
 Max-Planck-Institut für Physik
 München
 Bitnet: PEB@DMOMPI11

Tutorials

Forward References and the Ultimate Dirty Trick

Lincoln K. Durst

In this tutorial, we pick up where we left off in the first episode: See *TUGboat* 10, no. 3 (November 1989), pages 390–394.

The last page, 401, of Appendix D ("Dirty tricks") of *The T_EXbook* describes what surely deserves to be called the ultimate dirty trick. Under the heading "Syntax checking" the creator—of *TEX*, of course!—tells us how to disable the output routine, abolish all fonts, ignore all line and page breaks, and otherwise have *TEX* race through a file doing nothing more than executing the macros it encounters and can recognize. The original intention for providing this feature, as the name makes clear, was to locate typographical or other errors in the names of macros: Any macro

not recognized will be listed in the transcript of the TeX run. It is a tool that may certainly be used effectively for that purpose. But it has other uses as well.

"Syntax checking" will write what we tell it to write into files that can be read in when the text file is really TeXed. The scheme here is to handle forward references during a "preprocessing" run and postpone typesetting until everything is ready for it to proceed. One may expect that a preprocessing run should take less time than a full TeX run. Just how much difference there may be depends on a number of things; we shall come back to this question later.

Note that preparing indexes, tables of contents or lists of illustrations, figures, or tables does not require two runs, any more than making bibliographic citations does, provided that one is sufficiently indifferent about which sheets may come out of the printer first. Two complete runs are, of course, required if one wants to make forward cross references involving page numbers, such as those found in the telephone companies' *Yellow Pages* ("See our display adv. on page 9,901"), if some of the advertisements do not precede all the listings that refer to them. References to chapters, sections, theorems, figures, tables, or equations ordinarily do not require page numbers: Page numbers in cross references always have been an expensive luxury and, although they might be getting less expensive, they are still a luxury.

We require a file, call it *syntax.chk*, which will do what is described on page 401 of *The TeXbook*. Two realizations of *syntax.chk* have been made by Michael Spivak. One is a subset of *amstex.tex*; the other is in the file *vanilla.sty* distributed by Personal TeX, Inc., as a part of PC TeX. Readers who have access to either of these files should look for *\font\dummy* and copy out everything from its first occurrence through the definition of *\syntax*. Don't forget to put

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

at the top of the file. At the bottom of the file put the following three lines:

```
\syntax
\catcode`@=12
\endinput
```

Next we prepare what we shall refer to as a "driver file" like the following one for FIGURE 1 (which also accompanied the first installment of this tutorial):

```
%%% fermat.tex, a driver %%%
%%% for fermat.src, q.v. %%%
\input prepare.tex
\ShowMacros
\RefsInOrderCited
\preprocess{fermat.src} \bye
\compose{fermat.src} \bye
```

Here *prepare.tex* is a file, to be described in detail, containing code that controls what goes on in what follows. *\ShowMacros* and *\RefsInOrderCited* are options (the latter was discussed in the tutorial cited above). *\ShowMacros* sets a switch that causes the marginal notes to appear; default is no marginal notes, so it suffices to "comment" this line out (insert % to its left) when final copy is run. The line after the options starts the preprocessing run and terminates the TeX run when that has been completed. For the composition run, comment out the *\preprocess*-line with % at its left and TeX the same file again. The file *fermat.src* is the file containing the text of the piece to be typeset. The macros *\preprocess #1* and *\compose #1* are defined in the file *prepare.tex*:

```
%%% prepare.tex, first excerpt %%%
\newcount\sectnum \sectnum=0
\newcount\disppnum \disppnum=0
\newwrite\macrodefs
\newif\ifShowingMacros
\ShowingMacrosfalse
\def\ShowMacros{\ShowingMacrostrue}
\newif\ifOrdCited \OrdCitedfalse
\def\RefsInOrderCited{\OrdCitedtrue}
\newif\ifMakingPages
\MakingPagesfalse
\def\preprocess #1{%
\immediate\write16{%
.....preprocessing.....}%
\input syntax.chk
\PrepareDefs
\immediate\openout
\macrodefs=\jobname.ref
\input #1
\closeout\macrodefs}
\def\compose #1{%
\immediate\write16{%
.....making pages.....}%
\MakingPagestrue
\ifOrdCited
\input citation.prp
\else
\input biblio.prp
\fi
\input compose.tex
\input\jobname.ref
\input #1}
```

We interrupt the listing to describe what's here so far. Parts of this file will be recognized as dealing with bibliographic citations as discussed in the previous tutorial.

First, we attend to some preliminaries: Allocations for counters (to number sections and displays), for a file to hold the macro definitions, and for several \if-switches.

Next \preprocess#1 is defined. First it inputs `syntax.chk` in order to cripple plain TeX; what \PrepareDefs is and does will be discussed later; a file is opened next that is to contain the macro definitions (in the case of FIGURE 1, the name of the file opened is `fermat.ref`). Finally it TeXs `fermat.src` and closes `fermat.ref`. What is put into the new file, as we shall see, are definitions for the macros whose names are printed in the margin of FIGURE 1.

The third thing here is the definition of \compose #1, which begins by resetting the switch for making pages. Next the preliminary work required for handling bibliographic citations is done, as described in the previous tutorial, and finally three files are input. These are: `compose.tex` which contains typesetting information required for setting titles, subtitles, running heads and feet, marginal notes, etc., information not required for the preliminary run; the file `\jobname.ref`, created in the preprocessing run, which contains the definitions used to make the cross references; and finally the file containing the text source is read in again and the typesetting is carried out.

Now for more of `prepare.tex`:

```
%%% prepare.tex, second piece %%%
% The next pair of definitions
% will be written over when
```

1. Fermat numbers

\sect.Fermat.

Fermat considered numbers of the form $2^{2^n} + 1$, which are now known as the *Fermat numbers*, F_n , and he may or may not have asserted [3, pp. 23ff] that he had proved they are primes for all natural numbers n . Subsequently Euler found that the sixth Fermat number,

$$F_5 = 2^{32} + 1 = 42949\,67297,$$

is a multiple of the prime 641. (Early results of this kind will be found in Dickson's history [2, volume i] and more recent results in a book by Brillhart, *et al*, published last year [1].)

Euler's result for F_5 follows from the elementary facts given in displays 1.1 and 1.2:

$$641 = 5 \cdot 2^7 + 1 = 2^4 + 5^4 \quad (1.1) \quad \backslash \text{disp.Powers}.$$

hence

$$5 \cdot 2^7 \equiv -1, \quad 5^4 \cdot 2^{28} \equiv 1, \quad 5^4 \equiv -2^4, \quad 2^4 \cdot 2^{28} \equiv -1 \pmod{641}. \quad (1.2) \quad \backslash \text{disp.Congruences}.$$

I learned this arithmetic trick from Olaf Neumann of Friedrich Schiller Universität, Jena, D.D.R.; he did not tell me who invented it.

LKD

2. References

\sect.Refs.

1 Brillhart, John; Lehmer, D. H.; Selfridge, J. L.; Tuckerman, Bryant; Wagstaff, S. S., Jr. *Factorizations of $b^n \pm 1$, $b = 2, 3, 5, 6, 7, 10, 11, 12$ up to high powers*, American Mathematical Society, Providence (Contemporary mathematics 22, second edition), 1988.

\ref.brillhartFOB.

2 Dickson, Leonard Eugene. *History of the theory of numbers*, three volumes, Carnegie Institution of Washington, Washington, D. C. (Publication number 256), 1918, 1920, 1923. (Reprinted by Hafner and Chelsea.)

\ref.dicksonHTN.

3 Edwards, Harold M. *Fermat's last theorem*, Springer-Verlag, New York, Heidelberg, Berlin (Graduate texts in mathematics 50), 1977.

\ref.edwardsFLT.

FIGURE 1

```
% compose.tex is read in
% during the composition run:
\def\section[#1//#2]{\Section{#2}}
  % #1 is the name of the section
  % #2 is a "tag" in the macro name
\def\eqnum #1{\Eqnum{#1}}% #1: "tag"
  % The next three definitions
  % are used in both runs
\def\Section #1{\advance\sectnum by 1
  \dispnum=0 % reset in each section
  \edef\ItemNum{\the\sectnum}
  \writedef[Sect//#1//\ItemNum]
\def\Eqnum #1{\advance\dispnum by 1
  \edef\Itemnum{\ItemNum .\the\dispnum}
  \writedef[Disp//#1//\Itemnum]
\def\writedef[#1//#2//#3]{%
  \ifMakingPages
    \MakeNote{#1}{#2}%
  \else
    \def\macdef{\expandafter
      \def\csname #1#2\endcsname{#3}}%
    \immediate\write\macrodefs{\macdef}%
  \fi}
\def\sect.#1.{\csname Sect#1\endcsname}
\def\disp.#1.{\csname Disp#1\endcsname}
\def\ref.#1.{\csname\endcsname}
```

First we have the definition of `\section`, whose first argument is the title of the section, the second argument being the "tag" to appear in the name of the macro, for example, "Refs" in the name `\sect.Refs`. The macros `\section` and `\eqnum` have two versions (the second versions are in the file `compose.tex`), one used in the preprocessing run, the other in the composition run. In the first run, all that is done is to write the macro definitions into the file `\jobname.ref`. For `\section` the first argument, which is the title for the section, is discarded on the first pass, since it is not needed for the macro definition. In the composition run, `compose.tex` is read in after `prepare.tex`, so for macros having definitions in both of these files, those in `compose.tex` simply write over the earlier versions given above. The macro `\writedef` on the first run puts the macro definitions into the file `\jobname.ref`, and on the second run creates the notes which may or may not appear in the margin.

The last three definitions listed above in `prepare.tex` deserve some attention. The first two define the macros used for cross references to sections and displays using `\csname`, in such a way that if one of these macros is encountered as a forward reference, it will not cause TeX to stop to report an undefined control sequence. (Recall that `\csname...\endcsname` has the value `\relax` until it has been defined—cf. the previous tutorial.) For the same reason, the third one causes TeX to

discard references to bibliographic items, which are of no interest during preprocessing. All the work involved with bibliographies takes place during the composition run.

The macro `\MakeNote`, which appears in `prepare.tex`, is not needed until the second pass; it is defined in `compose.tex`, excerpts of which are considered next.

First we have headings and displays; this part of `compose.tex` writes over definitions contained in the file `prepare.tex`:

```
%%% compose.tex, first part %%%
%%% headings and displays %%%
  % This file is loaded by \compose
  % after prepare.tex is loaded
% headings
\def\hdingfont{\bf} % plain default (\tenbf)
\newskip\preheadingskip
  \preheadingskip=5pt plus 1pt
\newskip\postheadingskip
  \postheadingskip=5pt minus 1pt
\def\section[#1//#2]{\Section{#2}%
  \nobreak
  \vskip\preheadingskip\centerline
  {\hdingfont\ItemNum.\
    \vadjNote\margtext #1}%
  \vskip\postheadingskip}
% displays
\def\eqnum #1{\Eqnum{#1}\eqno
  (\Itemnum)\eolNote}
```

Here we have the new definition of `\section`; first it retraces the steps made in the earlier run and again calls `\writedef` which, instead of writing to the file, this time calls the macro `\MakeNote` to construct the text for the marginal note. After that it sets the text for the heading. The treatment of `\eqnum` is similar: The source file contains, in display 1.1, `\eqnum{Powers}`. There are at this point three remaining mysteries: `\MakeNote`, `\vadjNote\margtext` and `\eolNote`; these are the control sequences that put the notes (if any) in the margin.

Next, marginal notes:

```
%%% compose.tex, second part %%%
%%% marginal notes %%%
\font\margfont=cmtt8 \newbox\mbox
\newbox\margbox \newbox\Margbox
\def\MakeNote #1#2{%
  \ifShowingMacros
    \gdef\margtext{\char`\\%
      \lowercase{#1#2}}
    \global\setbox\mbox=\hbox{%
      \hskip 1pc\vbox to 0pt{\vss
        \noindent\margfont\margtext}}}
\fi}
```

```
\def\eoNote{%
  \ifShowingMacros
    \hfil\rlap{\box\mbox}
  \else
    \null
  \fi}
\newbox\parenbox \setbox\parenbox=\hbox{}}
\def\parenStrut{\vrule height 1\ht\parenbox
  width Opt depth 1\dp\parenbox}
\def\vadjNote #1{%
  \ifShowingMacros
    \parenStrut\setbox\margbox
      =\hbox{\hspace\hsize\hskip 1pc
        \parenStrut\margfont #1}%
    \setbox\Margbox=\hbox{%
      \raise 1\dp\parenbox\box\margbox}%
    \wd\Margbox=0pt \ht\Margbox=0pt
    \dp\Margbox=0pt \vadjust{\box\Margbox}%
  \else
    \null
  \fi}
```

First a font is chosen and some boxes are allocated to hold material to be printed in the margin. There will be two kinds of boxes in the margin, depending on whether they are called for at the end of a line of printed text or somewhere other than the end. Notice that in \eqnum the marginal box is called at the display number (\eqno), which is at the right end of the line. This is the case in which the control sequence \eoNote is used. It is possible for a section heading to occupy more than one line and it is probable that a bibliographic item will do so. If we want the marginal box in such cases to be even with the number that appears in the first line, we must either know where the first line breaks or have a way to insert the call where the number occurs and arrange for the box to be printed whenever the line does break. In the latter case, we resort to the more complicated control sequence \vadjNote. In any event, the definition of \MakeNote provides for both cases: the simpler \mbox is used in constructing \eoNote, and the text itself, \margtext, is retained for use in the other case.

The big trick used for notes not inserted at the end of the line is to invoke \vadjust, which is described and illustrated on page 105 of *The T_EXbook*; \vadjust is designed to do just what we need, to wait until the end of the line in which it occurs and when the line ends its argument is put into the vertical list. There is a minor complication caused by the fact that the point in the vertical list at which the insertion is made is located at the depth of the box containing the line with the call to \vadjust. That depth is zero if there are

no descenders in the text line, but it may not be zero; also there may be descenders in the name of the macro to be printed in the margin and maybe not. In order to make the two baselines even, we resort to a strut, putting an invisible parenthesis in both lines (the text line and the marginal note) and raise the note by an amount equal to the depth of the parenthesis. [By the way, you could use \vadjNote\margtext with \eqnum (put it before \eqno in the definition), but if you do, the note will appear near the bottom of the display, which may be a good distance below \eqno if large fractions, matrices, etc., are involved.] What's more, we hide the note in \vadjNote by first putting it into one box (\margbox) and then putting that in turn inside another box (\Margbox) along with the upward shift by the depth of the parenthesis. In addition, we make the whole thing invisible to T_EX by lying about its dimensions, thereby preventing the lines on either side of it from being spread further apart than normal.

The next excerpt from compose.tex is pretty dull, by comparison; it contains specifications for the text font and for references:

```
%%% compose.tex, third part %%%
% Text font
\def\TextFont{} % plain default (\tenrm)
\TextFont

% References
\newdimen\thehang \thehang=1.5em
\def\bibfont{} % plain default (\tenrm)
\def\bibstrut{\vrule height Opt width Opt
  depth .4\baselineskip}
\def\bibl#1#2\endbib{\everypar{}\noindent
  \hangindent=\thehang%
  \bibfont\unskip\vadjNote\margtext
  \hbox to\thehang{%
    \hfil#1\ }#2\bibstrut\par}}
\def\Bibl#1//#2//#3\endBib
  {\frenchspacing #1 {\it #2\}/}, #3}
\Bibl... \endBib is used in bibliog.fil; for
example:
\def\brillhartFOB{\Bibl...\endBib}
```

We leave the task of filling in the dots as an exercise for the reader.

In later tutorials in this series, other material will be added to prepare.tex and compose.tex; for example, code dealing with exercises and their solutions, discursive endnotes, and other things. In any case, one line is essential: both these files should end with \endinput.

Some loose ends: biblio.set (v. 0.9) and \PrepareDefs. Here is the third version of

`biblio.set`, which includes provision for printing the names of the macros in the margin:

```
%%% biblio.set (v. 0.9) %%%
\bib=0
\def\bibmac#1{\advance\bib by 1
  \ifShowingMacros
    \xdef\margtext{\char'\\#1}%
  \else
    \let\margtext=\null
  \fi
  \bibl{\bf\the\bib}%
  {\csname #1\endcsname}\endbib}
\input bibliog.fil
\ifOrdCited
  \immediate\write\bibliolist
  {\string\endinput}
  \immediate\closeout\bibliolist
  \input citation.ord
\else
  \input bibliog.ord
\fi
\endinput
```

There is another version (1.0) of `biblio.set` on the disk mentioned below; it contains code for trapping spelling errors in macro names for bibliographic items, mentioned in the first tutorial.

Finally, how much faster is preprocessing and what about the macro `\PrepareDefs`, which appears in the file `prepare.tex`?

Every sufficiently interesting `TEX` project will, by definition, use some specially constructed macro definitions devised just for it. Some users may wish to keep these in one place for ease of reference; suppose we call that file `\jobname.def`. For example, if you are working with collections of n -dimensional vectors, you will surely have a number of definitions of the following sort:

```
\def\vect#1{(#1_1,\ldots,#1_n)}
```

in your collection of special definitions. You may have only a handful of such definitions, or you may have hundreds of them, and even if you don't have many, those few may occur hundreds of times in your source text. Here is an example of a judgment call: Should these definitions go into `prepare.tex` or into `compose.tex`, or should different versions go into the two files? If there are enough occurrences of such macros in the source text, it might save time during preprocessing if they were all set to be `\relax` at that stage, with the real definitions postponed until the composition run, which is where they are actually needed. But if there are not many of them and they don't occur very often, you might as well input `\jobname.def` during the preparation stage just prior to `\input #1`.

For whatever reason, if you're impatient (for example) or if a test shows that it will noticeably shorten preprocessing, here is a way to disable the special definitions during that stage. Include another option, `\RelaxDefs`, in the driver file to control a new `\if`-switch in `prepare.tex`:

```
\newif\ifRelDefs \RelDefsfalse
\def\RelaxDefs{\RelDefstrue}
```

Before we go into how this switch is to be used, let us recognize that `\jobname.def` may not even exist, that the text requires *nothing* not already provided in `plain.tex`. (Maybe all we're doing is setting Shakespeare's sonnets, or somebody else's.) So here comes another option, `\DefFileExists`, which controls the following switch:

```
\newif\ifdeffile \deffilefalse
\def\DefFileExists{\deffiletrue}
```

Then put the following code in `prepare.tex`:

```
\def\gobble #1{} % TeXbook, p 308, ex 7.10
\def\dropslash{\expandafter\gobble\string}
\gdef\Relax#1#2{\expandafter
  \let\csname\dropslash#1\endcsname
  =\relax}
\def\PrepareDefs{%
  \def\switchdef{%
    \let\Def=\def \let\Font=\font
    \let\Long=\long \let\long=\relax
    \let\def=\Relax \let\font=\Relax
    \catcode`\#=12 \catcode`\^=12 }
  \def\resetdef{\let\def=\Def
    \let\font=\Font \let\long=\Long
    \catcode`\#=6}
  \ifdeffile
    \ifRelDefs
      \switchdef
      \input\jobname.def
      \resetdef
    \else
      \input\jobname.def
    \fi
  \fi}
```

Readers of the first tutorial in this series will recognize the manoeuvre here as a variation on the program `bibmac.tex` described in the construction of `bibliog.ord` from `bibliog.fil`. We fool `TEX` by telling it that `#` is an 'other' (see *The TeXbook*, page 37) so `TEX` will not worry when it is found in horizontal mode; similarly for `^`, the reason for the latter will appear in a later tutorial (when we consider index construction).

You should also include the following line in `compose.tex`:

```
\ifdeffile\input\jobname.def\fi
```

so that the definitions will be present when needed.

Here is `\jobname.def` for FIGURE 1:

```
%%% fermat.def %%%
% Cf. TeXbook, page 106 "\signed"
\font\smc=cmcsc10
\def\initials #1.{\{\unskip\nobreak
  \hfil\penalty50\hskip2em\hbox{\{}%
  \nobreak\hfil\smc #1\parfillskip=0pt
  \finalhyphendemerits=0\par\}}
\endinput
```

This was used to put the initials “LKD” at the end of Section 1 and the caps-small-caps in the caption “FIGURE 1”. This is clearly a case in which exercising the option to disable these definitions is not worthwhile. As a matter of fact, `\smc` is actually seen by TeX in the file `fermat.src` only when `\ifMakingPages` is true; and, although `\initials` is seen in both runs, it’s used only once. So why bother? `\RelaxDefs` is intended to disable the definitions when (a) that actually makes a difference and (b) the job is too big to be worth doing by hand.

According to Knuth (last sentence on page 401, *The TeXbook*), syntax checking, together with writing to files disabled, “usually make TeX run four times as fast.” Here we certainly do not want to disable writing to files, and even though we go out of our way to avoid showing TeX things it has no need to see during preprocessing (such as section titles, bibliographic citations, and—in the next tutorial—texts of exercises, solutions, or endnotes), we find that we can achieve speeds in the range one-to-two times as fast.

Last Question: How can you tell whether relaxing some of the definitions is worth the bother? You can time the preprocessing run and see if it makes any difference. A pair of simple programs that may be used to do this will be found on the disk referred to in the Note below; they are written in ANSI/ISO Standard C. The disk contains for both programs the source code and executable versions for MS DOS systems. If you don’t want the disk, write the programs yourself: for the first program (`start`), use `difftime()` to write into a file the number of seconds since some reference point just before giving the command `tex \jobname`, and just afterwards, use the second program (`elapse`) to open the file and subtract its contents from the present number of seconds since the same point of reference. You could use a shell script or “.BAT file” containing, say,

```
start
tex <filename>
elapse
```

to do this. If you don’t like C, you can use TeX instead, but if you do, you will have to use minutes instead of seconds. Cf. `\time`, page 273, *The TeXbook*; to keep things simple, you may wish to avoid using the TeX version in the middle of the night since `\time` gives you the number of minutes since midnight. The reference point for `difftime()` used by (at least some) C compilers appears to be New Years 1970.

Note. A disk (5.25in DSDD) containing source text for the figures in these tutorials and the code files used to produce them is available for MS DOS users who are members of the TeX Users Group. The disk includes, in addition to the files mentioned above, source text for this tutorial and some of the others in the pipeline, as well as *TUGboat* style files (described in *TUGboat* 10, no. 3, pages 378–385), that may be used to typeset the tutorials. Send \$6 to the address below, which includes a royalty for the TeX Users Group. Outside North America, add \$2 for air postage.

I had overlooked the existence of `syntax.chk` and `\font\dummy` until Barbara Beeton called them to my attention. Ron Whitney has been a great help when I encountered problems: If I couldn’t solve them, he did; and often when I did solve them, he produced better solutions. I am very grateful to them both.

◊ Lincoln K. Durst
46 Walnut Road
Barrington, RI 02806

Output Routines: Examples and Techniques.

Part I: Introduction and Examples.

David Salomon

A general note: Square brackets are used throughout this article to refer to the *TEXbook*. Thus [400] refers to page 400, and [Ch. 6], [Ex. 15.13] refer to chapter 6 and exercise 15.13, respectively, in the book. Advanced readers are referred to the actual *WEB* code by the notation [§1088]. Also, since the words "output routine" are commonly used in this article, they are replaced by the logo OTR.

"It would be possible to write an entire book about *TEX* output routines; but the present appendix is already too long..." This quotation, from [400], best describes this article. It is not a book, but it tries to do justice to OTRs; justice denied them in the *TEXbook*, because of lack of space.

This article is long and is divided into three parts. Part I is an introduction to OTRs and related concepts, followed by examples. The examples are mostly simple OTRs, useful for common applications. No advanced techniques are used. In the second part, various techniques are developed, for communicating with the OTR by means of marks, special penalty values, kerns, and special boxes inserted in the text. Some methods require several passes, saving information between the passes either on a file or in memory. Some techniques examine the contents of `\box255`, going as far as breaking it (or a copy) up into individual components. Whenever possible, the techniques are applied to practical cases. Part III treats insertions. OTRs with insertions introduce more complexities and deserve detailed treatment. Specifically, the *plain* format OTR is introduced in part III since it supports insertions.

The reader is encouraged to send the author problems related to OTRs, suggestions for new techniques, and errors found in this document.

Certain methods described here are not completely general and may fail in certain situations. Others are more general and seem to work always. However, none of them have been thoroughly tested, and new problems may be discovered at any time.

To make it easy to read and understand the macros described here, they were deliberately kept as simple as possible. As a result, they may not be general and may not handle every possible situation. The point is that the macros should not be copied and used verbatim. Rather, they should be studied and modified for specific problems.

Advanced *TEX* users hardly need be convinced that an understanding of OTRs is important, since they must be used whenever special output is desired. However, the entire topic of OTRs has traditionally been considered complex, and it is! The reasons are: (1) OTRs are asynchronous with the rest of *TEX* (this is explained later) and involve difficult concepts such as splitting boxes and insertions. (2) Certain features which could be very useful in OTRs are not supported by *TEX*. Specifically, there are no commands to identify marks, rules, and whatsits in a box, and to break up a line of text into individual characters. One can only hope that, with more interest in *TEX* and more demand from users, there will be a future version of *TEX* (4.0?) with the missing features included.

Introduction

A few introductory concepts are introduced in this section for the benefit of the inexperienced reader.

Boxes. Definition: A box is an indivisible unit of material, inside which *TEX* switches to one of the internal modes. Thus in an `\hbox`, *TEX* is in restricted horizontal mode, which means that items placed in such a box will be positioned side by side. If an `\hbox` is typeset, it is not broken into lines, but is typeset as one line. Similarly, items placed in a `\vbox` are stacked from top to bottom. When such a box is typeset, the entire box is placed on the same page. More information on the properties of boxes can be found in [Ch. 11-12].

Lists. A *list* is best thought of as a bunch of boxes (and other items) positioned either horizontally or vertically. Thus *TEX* supports horizontal and vertical lists. The specific items that may appear in a horizontal list can be found on [95]; those which may appear in a vertical list can be found on [110]. An example of a horizontal list is the *TEX* logo. Its components are the two letters 'T', 'X', a box with the letter 'E', and two pieces of `\kern`. An example of a vertical list is a page of text. Its components are the individual lines of text and the penalties, glue, and other items between them.

The Main Loop. This is also known as the *Inner Loop*, the *Chief Executive*, or the *Main Control* (see [§1029, §1030, §1035]). This is where *TEX* spends most of its time, preparing pages of text which eventually are sent to the OTR. This loop consists of reading characters from the source file, scanning them and converting them into tokens, using the tokens to build boxes, and combining the boxes into lists.

The Main Vertical List and the Page Builder

This section introduces a simple picture of a structure called the *Main Vertical List* (MVL). The precise way the MVL is organized and used is presented in a later section.

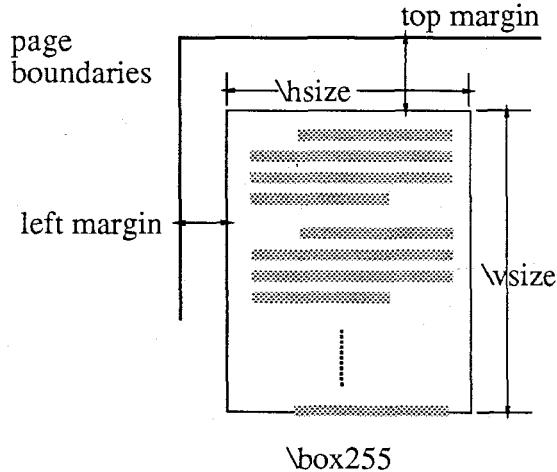
When *TeX* reads the first character of a paragraph, it switches to horizontal mode, where it reads the rest of the paragraph. It then switches back to vertical mode and invokes the line breaking algorithm. The resulting lines are saved in the MVL. Gradually, more and more lines of text are appended to the MVL and, at a certain point, when the MVL contains more than enough material for a page, *TeX* invokes (exercises) the *page builder*. The page builder decides on a good point to break the page, cuts a chunk of material off the MVL, places it in `\box255`, and invokes the OTR. The OTR usually adds things such as a page number, a header, or footnotes, and ships out the page (i.e. writes the page to the `.dvi` file).

The OTR does not have to ship the page out. It can ship out just part of it, and save or discard the rest. It can also return the rest to the MVL. The rule is that any material left over by the OTR (i.e. any material placed on the OTR's vertical list) is returned to the MVL. This is a very useful feature.

An important point, which should be emphasized at this early stage, is the *asynchronicity* of the OTR. `\box255` is *not* filled up gradually with text. It is the MVL that's filled up with lines of text until there is more than enough material for a page. The page builder then cuts a chunk, the size of a page, off the MVL, and places it in `\box255`. At that point, there is normally some material left in the MVL, for the next page. This means that, when the OTR is invoked, *TeX* has already read text from the source file past the end of the current page. Thus the asynchronous nature of the OTR. It is not synchronized with the main loop but is invoked from time to time, as the need arises. Another way to express this idea is to say that the OTR lags behind *TeX*'s main loop.

Note that *TeX* does not know the size of the paper which eventually comes out of the printer. There are only four parameters that relate to the size and position of the page, namely: `\hsize`, `\vsize`, `\hoffset` and `\voffset`. The first two become the width and height of `\box255`. The two 'offset' parameters are the amounts by which the left and top margins differ from the default offsets of the printer which typesets the `.dvi` file. Many drivers default to offsets of 1in, and `\hoffset` and `\voffset` give displacements from that point. Thus

setting `\hoffset=0.5in`, `\voffset=-0.2in` creates a left margin of 1.5in and a top margin of 0.8in. The right and bottom margins are unknown to *TeX*, and are normally only used by the printer driver. In the rare cases where *TeX* needs to know those quantities, they have to be entered by the user or computed. This is demonstrated in one of the examples (for printer registration marks) shown later in this part.



Page Boundaries and the Printed Area
Figure 1

Figure 1 shows how those quantities are related. It also shows that the depth of `\box255` is the depth of the bottom line of text. As a result, the vertical size of the printed area is the height plus depth of `\box255`, and is slightly larger than `\vsize`.

\pagetotal and \pagegoal. Two `\dimen` variables are maintained by the page builder and are used in the page breaking algorithm. They are also used for insertions.

- **\pagetotal** [114], is the vertical height of the MVL. We will denote it by t . This variable starts at zero and is incremented by the page builder each time something with a height (i.e., a box or some glue) is appended to the MVL. Things like marks, penalties, and whatsits do not have any dimensions and do not affect t . Since some vertical glues have flexibility, this variable is generally flexible. Initially, when the MVL is empty, t is set to zero.
- **\pagegoal** [114], which we denote by g , is the desired height of the page. Generally, it is equal to `\vsize` but, when insertions are generated, g is decremented. Initially, when the MVL is empty, g is set to `\maxdimen`. When the first

item is placed in the MVL, g is set to \vsize ("... TeX salts away the value of \vsize ..." [114]). When footnotes or other material are generated, to be inserted in the page, their heights are subtracted from g [123].

These two variables can be 'shown', either in the document itself (using \the) or in the log file (using either \showthe or \message). They can even be modified, but this should be done very carefully. It is even possible to get the two values displayed after each line by setting $\tracingpages=1$ [112]. This option exists mainly to show the feasible points for page breaks, and how the page breaking algorithm works. It is a good idea, however, to try it once, on a short page which also has footnotes, and to follow the changes in the values of the two quantities.

As an example, the two variables will be used to determine how much space is left on the current page. If t is zero, the space left on the page is the entire page (\vsize). Otherwise, it is the difference $g - t$. Macro \pagespace calculates that difference.

```
\newdimen\spaceleft
\def\pagespace{%
  \ifdim\pagetotal=0pt
    \spaceleft=\vsize
  \else
    \spaceleft=\pagegoal
    \advance\spaceleft by -\pagetotal
  \fi}
```

The Current Page and the List of Recent Contributions

The discussion so far has been general, ignoring certain important details. This section presents a more accurate picture of the MVL, and the way the page builder operates.

The MVL consists of two parts, the *current page* and, below it, the *list of recent contributions*. The current page holds the material that will become $\box255$. The recent contributions are used to temporarily hold recently read material. After an entire paragraph has been read, it is typeset, and the lines of text appended to the recent contributions. At that point, the *page builder* is invoked (exercised). Its job is to move lines, one by one, from the recent contributions to the current page. For each line, the page builder calculates the cost of breaking the page after that line. For the first couple of lines the cost is very high because breaking there would result in an extremely stretched page. Thus, for those lines, the badness b

becomes 10000 and the cost c , 100000 (see formula on [111]).

At a certain point, when there are enough lines in the current page for a reasonably looking page, b (and, as a result, c) start getting smaller. A while later, there may be too many lines of text in the current page, and it has to be shrunk, increasing b and c again. The entire process can be seen, in real time, by setting $\tracingpages=1$ [112]. If the page has to be shrunk more than its maximum shrinkability, both b and c become infinite. When c becomes infinite (or when a penalty ≤ -10000 is found, see below) the page builder goes back to the line of text where the cost was lowest, breaks the top of the current page at that point, and places it in $\box255$. The bottom part of the current page is then returned to the recent contributions, and the page builder invokes the OTR.

The page builder is exercised at the end of a paragraph, the end of a display equation within a paragraph, at the end of an \halign , and in a few other cases [122, 286]. The OTR is invoked only by the page builder [§1025], which is why it is never invoked in the middle of a paragraph (unless the paragraph contains display math material).

The advanced reader might want to glance at [§980–1028] for the actual code of the page builder.

Since the page builder is exercised quite often, the list of recent contributions is usually small or empty, and the current page gets larger and larger. When the OTR is invoked, the current page is empty. The \showlists command can always be used to display the two parts of the MVL in the log file.

The quantity t (\pagetotal) mentioned earlier as the height of the MVL is, actually, the height of the current page. It is updated by the page builder each time a line (or glue) is added to the current page.

A better understanding of the page builder and the MVL must include glue and penalties. When a paragraph is typeset, the lines of text are appended to the recent contributions, with glue and penalty items between them. These items are moved to the current page with the lines of text. If the current page is empty, all glues, kerns and penalties moved to it are discarded. This is how discardable items disappear at the top of a page. When the first box is moved to the current page, a special \topskip glue is placed above it, to keep its baseline 10pt (the value in plain format) below the top of the page. Following that, glue, kern, and penalties are moved, with the text, from the recent contributions to the current page.

When a penalty ≤ -10000 is encountered, the page builder breaks a page. If the resulting page does not have enough text lines, it may be underfull. Such penalty values can be used to eject a page (say, by `\vfill\penalty-10000`), or to communicate with the OTR (which knows the penalty associated with the point at which a page was broken).

It should be stressed, however, that a penalty of -10000 does not invoke the OTR *immediately*. If such a penalty is created inside a paragraph, between lines of text, it is saved in the recent contributions with the lines, and is only recognized as special when it is moved, by the page builder, to the current page. As a result, if a paragraph contains:

```
... \dimen0=2pt ... \vadjust{\penalty-10000}
... \dimen0=1pt ... \par
```

the OTR will be invoked after the entire paragraph has been read and broken into lines, and will find `\dimen0` to be `1pt`.

A page can only be broken at (i.e. just above) a glue, kern or penalty. If a page is broken at a glue or kern, the glue stays in the recent contributions (to be discarded when moved to the top of the next page). If the page is broken at a penalty, the penalty is saved in variable `\outputpenalty`. This variable can be used to communicate with the OTR. Also, if the OTR decides to return some material from `\box255` to the current page, it may want to place back the original penalty at the breakpoint, by saying `\penalty\outputpenalty`.

The precise rules of where a page can be broken are listed on [110]. One of them says "A page break may occur at glue, provided that this glue is immediately preceded by a non-discardable item." If a set of successive glues is moved to the current page, a page break can occur either before or after that set, but not inside it. If the page builder decides to break the page before the set, the entire set is returned to the recent contributions, to disappear at the top of the next page. If the page is broken after the set, it becomes glue at the bottom of the page. This information will be used in part II, when we try to communicate with the OTR by means of `\kern`.

The depth of the current page. The discussion so far has been kept simple (even though some readers may disagree) by ignoring certain features that have to do with the depths of the boxes involved. These features are important since, normally in a document, successive pages should have the same (or almost the same) vertical size.

The height of a page is controlled by (actually, it is equal to) `\vsize`. The depth of a page should also be under the user's control since, in certain situations, it may spoil the uniform appearance of the document. This is why it is important to consider TeX features which have to do with the depth of a page, and we start by introducing certain quantities that have to do with the depth of vboxes in general.

Consider a large `\vbox` with lines of text, separated by glue and penalties. The depth of this `\vbox` [80] is the depth of the bottom component. If that component is a glue or penalty, the depth is zero. If it is a box, then its depth becomes the depth of the entire `\vbox`, except that it is limited to the value of parameter `\boxmaxdepth`. If `\boxmaxdepth=1pt` and the depth of the bottom box is `1.94444pt`, then the depth of the entire `\vbox` will be `1pt` and its height will be incremented by `.94444pt`. This is equivalent to lowering the reference point (or, equivalently, the baseline) of the `\vbox` by `.94444pt`. In the plain format, `\boxmaxdepth=\maxdimen` [348], so it has no effect on the depths of boxes. However, `\boxmaxdepth` can always be changed by the user.

The *current page* is that part of the MVL that contains the material for `\box255`. Its current height is t , its goal height is g , but what is its depth? It is, of course, the depth of the bottom line of text — normally a small dimension which may vary a little from page to page. This results in pages with slightly different vertical sizes (i.e. height + depth). However, if the bottom line of text contains a large symbol with a depth of, say, `35pt`, the vertical size of the page will be `\vsize + 35pt`. The page will be much taller than its neighbors, spoiling the uniform appearance of the document. To avoid this, the page builder uses another parameter, `\maxdepth`, when it appends lines to the current page [125]. The plain format sets [348] `\maxdepth=4pt`. In our example, when the line with depth = `35pt` is added to the current page, the depth of the current page is set to `4pt` and the difference of `31pt` is added to its height t . A good way to visualize this situation is to say that the baseline of the current page no longer coincides with the baseline of the bottom line, but is located `31pt` below it.

The internal quantity `\pagedepth` (d) contains the depth of the current page, and is updated each time a line (or glue) is added to the current page. d is a 'relative' of t , the height of the current page. It should be noted that t has a few more 'relatives' [114], the most important of which are `\pagestretch` and `\pageshrink`, the amounts of

stretchability and shrinkability in the current page. t and its relatives are used by the page builder to determine a pagebreak (some of them are also used for insertions).

A simple test can show how those quantities are updated. First, change `\parskip` to some flexible value such as `1pt plus2pt minus1pt`, then typeset text in small pages and place a command such as

```
\message{(total: \the\pagetotal;
depth: \the\pagedepth;
shrink: \the\pageshrink;
stretch: \the\pagestretch)}
```

at the start of each paragraph. It will show the values of the 4 parameters at the end of the preceding paragraph.

The `\message` commands will show the value of t growing from paragraph to paragraph, until a page is shipped out by the OTR. The value of d is usually 1.94444pt (the depth of many letters in cmr10) but is different when anything other than cmr10 is used. If the last line of a paragraph happens to contain letters without any depth, d will be zero at the end of that paragraph. Note that d can only be displayed while in vertical mode, between paragraphs (within paragraphs, `\showthe\pagedepth` only shows 0pt, not the depth of the last line of the previous paragraph). Each time another `\parskip` is inserted, between paragraphs, `\pageshrink` is incremented by 1pt, and `\pagestretch`, by 2pt.

Controlling the depth of the current page. It has already been mentioned that d is limited to `\maxdepth`. If a line of text with a $\text{depth} > \maxdepth$ is moved to the current page, the depth of the current page (`\pagedepth`) is set to `\maxdepth`, and its height t is incremented by the difference (the baseline of the current page is lowered, and is now located below the baseline of the bottom line of text.)

Here is a simple experiment to clear up this point. First, set

```
\hsize=3.5in \vsize=2in
\tracingpages=1
```

and typeset some text in font cmr10. Most lines of text will have a height of $250/36 \approx 6.94444\text{pt}$ and a depth of $70/36 \approx 1.94444\text{pt}$. The `\baselineskip` glue between the lines is thus set to 3.1112pt, to achieve a separation of 12pt between consecutive baselines. The last three % lines of TeX's tracing report should be:

```
% t=130.0 g=144.54 b=10000 p=0 c=100000#
% t=142.0 g=144.54 b=204 p=0 c=204#
% t=154.0 g=144.54 b=* p=0 c=*
```

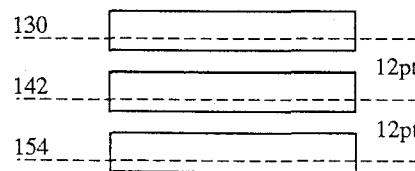
This shows that t is incremented by 12pt between lines of text. The last line results in infinite cost, so the page is broken after the line with $c=204$.

Next, repeat the experiment with the depth of the second line increased to 7pt by placing an `\hbox{\vrule depth7pt}` in it. Typesetting the same material now results in different % lines:

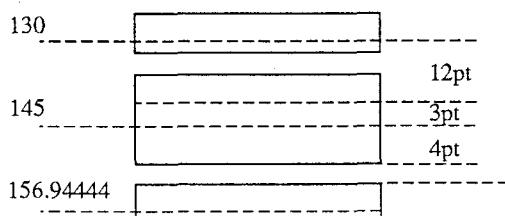
```
% t=130.0 g=144.54 b=10000 p=0 c=100000#
% t=145.0 g=144.54 b=10 p=0 c=10#
% t=156.94444 g=144.54 b=** p=0 c=**
```

Fig. 2 shows the layout of the last three text lines in both cases. In 2a, the lines all have the same height and depth and are separated with the same size glue. In 2b, however, the situation is more complex. A `\baselineskip` of 3.1112pt is inserted between the first and second lines, so their baselines are separated, as usual, by 12pt. Since the second line has a depth of 7pt, the value of `\pagedepth` is set to `\maxdepth` (= 4pt), and the difference of 3pt is added to the height t . The baseline of the entire page is thus lowered 3pt below the baseline of the second line. Normally, t would be set to $130 + 12 = 142$ pt. Instead, its value now is 145pt.

Because of the large depth of the second line, it is separated from the third line by `\lineskip` [78], which has a `plain` format value of 1pt. The baseline of the third line is set $4 + 1 + 6.94444 = 11.94444\text{pt}$ below the baseline of the page, and t is incremented



2a



2b

Figure 2

by that amount, to 156.94444pt. d is reset to the depth of the third line, namely 1.94444pt.

Again, appending the third line to the current page has resulted in infinite cost, and it is eventually returned to the recent contributions.

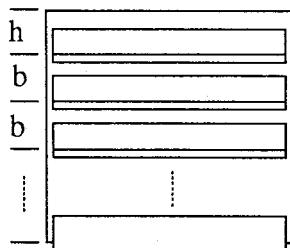
This is part of the overall task of the page builder while constructing the current page.

The height of a box of text. We denote the value of \baselineskip (normally 12pt) by b . A large \vbox with text consists mainly of lines of text, each an \hbox , separated by globs of glue, normally in the (varying) amounts necessary to separate baselines by exactly b , but sometimes just the amount \lineskip . We assume a simple case where no large characters or equations are used. In such a case, all lines of text are separated by b . The height of the box is thus

$$b(n - 1) + \text{the height of the first line}$$

where n is the number of text lines.

The height of $\box255$. In the case of $\box255$, enough glue is placed above the first line of text to reach to \topskip from the first baseline. We denote the value of \topskip by h (10pt in plain). So if the baseline of the first line is now h below the top of the page, the height H of $\box255$ should be $b(n - 1) + h$ (Fig. 3). However, the height of $\box255$ is always set, by the page builder, to \vsize . The difference between the two heights is usually supplied by the flexible glues on the page, the most common of which is \parskip .



The Height of a Page Box
Figure 3

Example: $\vsize=180pt$ creates 15-line pages, since $12(15 - 1) + 10 = 178$. The \parskip glues on the page are stretched by a total of 2pt. Similarly, $\vsize=189pt$ creates 15-line pages, but each page has to be stretched by 11pt.

What if there isn't enough stretchability? In such a case, the bottom of the page remains empty. This is an unusual situation where the height of a box is greater than the sum of the vertical

dimensions of its components. Normally such a case is considered an 'underfull box' but, in the case of the page builder and $\box255$, "underfull and overfull boxes are not reported when $\box255$ is packaged for use by the OTR" [400].

A simple experiment is recommended, to clear up this point (note that this experiment uses \output , which hasn't been introduced yet, nevertheless it is a useful experiment to perform at this point.)

```
\vsize=189pt \parskip=0pt
\output={\setbox0=\vbox{\unvbox255}
\message{[\the\ht0, \the\ht255;]}
\setbox1=\vbox to\vsize{\hrule width3in
\vfil\hrule width3in} \wd1=0pt
\shipout\vbox{\box1 \box255}
\advancepageno}
```

```
\vrule height10pt <text to be typeset...>
```

The \parskip glue is now rigid, so there is no flexibility on the page at all. The messages will be [178pt, 189pt;]. $\box1$ is set to two rules with a 'fil' in between, and is superimposed on $\box255$. A look at the typeset page will show that the top rule is placed exactly 10pt above the baseline of the top line, and the bottom rule is well below the bottom line of text.

A better understanding of $\box255$ is gained by trying

```
\setbox0=\vbox to\vsize{\unvbox255}
```

Even though both $\box255$ and $\box0$ have the same height namely, \vsize , they don't have the same status. $\box255$ may be created underfull without an error message but, when transferred to $\box0$, the destination box becomes underfull, with an error message.

The most natural thing to do, in such a case, is to try to fill up $\box0$ by saying $\setbox0=\vbox to\vsize{\unvbox255 \vfill}$. Surprise! This may, sometimes, cause an 'overfull box'. The explanation has to do with the depth of $\box0$. Without the \vfill , it is the depth of the bottom line. With the \vfill , it is zero, and the depth of the bottom line is added to the *height* of $\box0$, which may cause it to be overfull.

Examples of OTRs

The following sections show how to write an OTR, and illustrate typical OTRs for common applications. It should again be stressed that the examples are kept simple and, therefore, are not completely general. They should be read, understood and

modified for specific needs, rather than copied and used verbatim.

An OTR is simply a sequence of *T_EX* commands assigned to the token-register *\output*. Thus *\output={...}* would cause *T_EX* to execute the commands ... whenever it decides to invoke the OTR.

The simplest OTR is *\output={}*. When *T_EX* sees this OTR, it substitutes the *default* OTR, which is: *\output={\shipout\box255}*. This default OTR is the simplest one which ships out a page.

\shipout is a the *T_EX* primitive which creates a page in the *dvi* file. That page reflects the contents of whatever box follows the call to *\shipout*. An interesting feature is that *\shipout* can be invoked anytime, not just from an OTR. This is demonstrated in one of the examples below. Another interesting point is that *\shipout* can be redefined (which, of course, is true for any control sequence, whether a macro or a primitive.) One of the methods shown later, for double-column pages, does just that.

The OTR itself can be redefined during a *T_EX* run. The new OTR will be used when the page builder next invokes the OTR. It is possible to define a macro such as *\def\newotr{...}* and assign *\output=\newotr* at any time. This will redefine the OTR. It is even possible to write an OTR which redefines itself! Here is an example:

```
\output={  
  \shipout\box255 \advancepageno  
  \global\output={  
    \shipout\vbox{  
      \box255  
      \bigskip  
      \centerline{\folio}  
    }\advancepageno  
  }  
}
```

This OTR typesets the first page without a page number. It then (globally) redefines itself to typeset the rest of the document with the page number centered below the text. The reason for the *\global* is the local nature of the OTR, which is explained below.

It should be noted that the OTR is expected to empty *\box255*; it can ship it out, move it to some other box, or return it to the MVL. The latter is done simply by saying *\box255* or *\unvbox255* inside the OTR. An OTR which does not do anything with *\box255* will cause the error message "unvoid *\box255*."

The following OTR just empties *\box255*. *\output={\setbox0=\box255}*. This does not

cause an immediate error message but is probably not what you want to do. It amounts to tossing away the entire document, page by page.

The next example is *\output={\uvbox255}*. This OTR always returns the page to the MVL, which will cause the page builder to immediately find a new page break and invoke the OTR again. The new page break, by the way, may not be the same as the original one, because of the penalty at the breakpoint. When the breakpoint is chosen, the page builder places the penalty found at the point in variable *\outputpenalty*, not in *\box255*. The OTR can return the penalty to its original place by saying *\unvbox255 \penalty\outputpenalty*. This guarantees that the page builder will find the same breakpoint.

An execution of the OTR which does not ship out anything is called a *dead cycle*. Dead cycles have their uses and are illustrated by some of the examples shown later. However, many consecutive dead cycles normally indicate an error. This is why *T_EX* counts the number of consecutive dead cycles (in register *\deadcycles*) and stops the run if *\deadcycles ≥ \maxdeadcycles*. The plain format value of *\maxdeadcycles* is 25, and it can be changed at any time. Each time *\shipout* is invoked, it clears *\deadcycles*.

The Page Number. The page number can come from any source. Here is an example where the OTR typesets a page number, from a *\count* variable, centered below the printed area:

```
\newcount\pageNum  
\output={  
  \shipout\vbox{  
    \box255\smallskip  
    \centerline{\tenrm\the\pageNum}}  
  \global\advance\pageNum by1  
}
```

Note the *\tenrm* in the preceding example. It is necessary because of the asynchronous nature of the OTR. When the OTR is invoked, *T_EX* can be anywhere on the next page. Specifically, it could be inside a group where a different font is used. Without the *\tenrm*, that font (the current font) would be used in the OTR.

In the plain format, the *\count0* variable serves as the page number, and the following two macros are especially useful.

- *\folio* typesets *\count0* as the page number. However, if *\count0* is negative, *\folio* typesets a roman numeral.

- `\advancepageno` advances the page number by one. This is done by either incrementing or decrementing `\count0`, according to its sign.

Any `\count` variable can be used to typeset the page number. However, the main advantage of using `\count0` is that TeX writes its value on the dvi file, so a page preview program can easily display the page number with the page. Stated more negatively, driver programs may *only* understand the values of `\count0` written into the dvi file as page numbers and may not be able to selectively print pages whose numbers correspond to some other counter.

Actually, the ten variables `\count0...count9` are used as a (composite) page number. Any non-zero variable in this group is written on the dvi file. Typesetting and advancing any of them must be done explicitly by the user. Macros `\folio`, `\advancepageno` only handle `\count0`.

Grouping and the OTR. The OTR, as mentioned earlier, is a list of tokens. It is also implicitly surrounded by a pair of braces, making it into a group. This means that anything done inside the OTR is *local*, unless preceded by `\global`. This is a useful feature since, normally, operations within the OTR should be local (i.e. hidden from TeX's usual operations of making paragraphs and putting together math formulas).

Example: A ‘Boxed’ Page

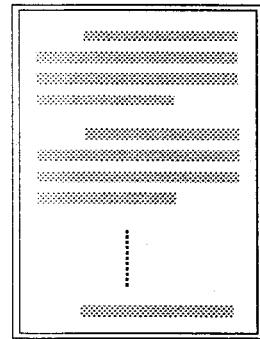
Here is an OTR for a ‘boxed’ page. It surrounds the page with double rules on all sides, and centers the page number below the double box. Note that the page shipped out is wider and taller than `\box255`. The value of `\hsize` in this case is, therefore, not the width of the final page shipped out, but the width of the text lines in `\box255`.

Macro `\boxit` typesets text and surrounds it with 4 rules (see [Ex. 21.3]). Parameter #2 is the space between the rules and the text. #1 is a box containing the text.

```
\def\boxit#1#2{%
  \vbox{\hrule
    \hbox{%
      \vrule \kern#2pt
      \vbox{\kern#2pt #1
            \kern#2pt}
      \kern#2pt\vrule}
    \hrule}}
\output={
```

```
\medskip
\centerline{\tenrm\folio}
\advancepageno}
```

Figure 4 is an example of a small, doubly-boxed page.



A Boxed Page

Figure 4

Example: Header and Footer

This OTR typesets a header and a footer, both token lists supplied by the user. Typically one of them contains the page number.

```
\output={%
  \shipout\vbox{%
    \offinterlineskip
    \vbox to3pc{%
      \line{\the\headline}
      \vss}
    \box255
    \vbox to3pc{%
      \vss
      \line{\the\footline}}}
  \advancepageno}
```

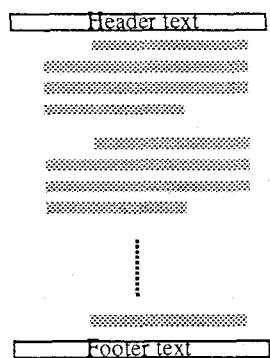
The headline and footline occupy 3pc each, enough for about 4 lines of text. If the user places more than 3pc worth of material in any of them, there will be no error message (because of the `\vss`), and the extra material will be typeset on top of the main body of the page. Both `\headline` and `\footline` are token lists, and may contain `\if` commands. Examples are

```
\headline=%
\ifodd\pageno
  \line{\hfil\bf Header\hfil
    \llap{\tenrm\folio}}%
\else
  \line{\rlap{\tenrm\folio}\hfil
    \bf Header\hfil}%
\fi}
```

```
\footline={\it footer text\hfil}
```

The vertical size of the box shipped out (the printed page) is 6pc plus $\vht255$ (which is \vsize) plus $\dp255$ (which is limited to \maxdepth and thus can be kept small).

Figure 5 is an example of such a page.



Header and Footer
Figure 5

Example: A Title Page for a Chapter

Sometimes, the book designer specifies a separate title page at the start of each chapter. Here is an example of a `\chapter` macro which typesets such a page *outside* the OTR. It starts with an `\eject`, to eject the last page of the previous chapter, then invokes `\shipout` to ship out a page with the chapter number and name. Note that, even though the page number isn't typeset, `\chapter` still has to advance it.

```
\def\chapter#1 #2;{%
  \vfill\eject
  \shipout\vbox to\vsized{
    \linef\bf Chapter\hfil#1}
  \vfil
  \vbox{\raggedcenter\bf#2}
  \advancepageno}
```

There seem to be two problems with our simple macro:

1. If the first chapter starts on the first page of the document, our macro will eject a blank page before any text has been typeset.
2. If a page was ejected just before the new chapter started, our macro will eject a blank page.

It turns out that neither of these is a problem. An `\eject` is essentially a `\penalty-10000` and, if the very first thing in the document is a penalty, it gets discarded. Also, if the first thing on a new page

is such a penalty, it gets discarded. Here is a relevant quote (from [114]) "If you say `\eject\eject`, the second `\eject` is ignored, because it is equivalent to `\penalty-10000` and penalties are discarded after a page break."

Example: Printer Registration Marks

We now turn to an OTR that optionally typesets the registration marks, also known as *crop marks*, which are used to align the pages for photography prior to printing and to indicate the size of the final page. The registration marks should be positioned at the 4 corners of the page, not at the corners of the printed area. The top left mark, e.g., should be located $1in+\voffset$ above the top of the printed area, and $1in+\hoffset$, to the left. Similarly, the top right mark should be placed up and to the right, but by how much?

The problem is that `TEX` has no idea how wide and tall the paper is. All it knows is the left and top offsets, and the dimensions of the printed area (\hsize and \vsize). To place the registration marks properly, the user should specify the dimensions of the paper.

The document should thus start by specifying:

```
\newdimen\paperheight
\newdimen\paperwidth
\paperheight=..in \paperwidth=..in
```

It is also possible, although less desirable, to prompt the user to enter the two dimensions.

```
\newdimen\paperheight
\newdimen\paperwidth
\message{Enter paper height }
\read-1to\tmp \paperheight=\tmp
\message{Enter paper width }
\read-1to\tmp \paperwidth=\tmp
```

The next step is to create a `\vbox` of these dimensions, with the marks at the corners.

```
\newif\iffinalrun \finalruntrue
\newdimen\ruleht \ruleht=.5pt
\newdimen\gap \gap=2pt
\def\verrules{%
  \hbox to\paperwidth{%
    \vrule height1pc width\ruleht depth0pt
    \hfil \vrule width\ruleht depth0pt}}
\def\horrules{%
  \hbox to\paperwidth{%
    \llap{\vrule width1pc height\ruleht
      \kern\gap}
    \hfil
    \rlap{\kern\gap
      \vrule width1pc height\ruleht}}}}
```

```
\newbox\rmarks \setbox\rmarks=
  \vbox to\paperheight{
    \offinterlineskip
    \vbox to0pt{\vss
      \verrules
      \kern\gap
      \horrules}
    \vfil
    \vbox to0pt{
      \horrules
      \kern\gap
      \verrules\vss}}

```

The dimensions of the box are then set to zero, so it will be superimposed on the printed page

```
\ht\rmarks=0pt \wd\rmarks=0pt
```

and the OTR typesets the box (which does not move the reference point), followed by the printed page. This causes the top left corner of the printed page to coincide with the top left registration mark. To center the printed page, it should be lowered and moved to the right.

```
\newdimen\Mdown \Mdown=\paperheight
\advance\Mdown by-\vsize
\advance\Mdown by-6pc \divide\Mdown by 2
\newdimen\Mright \Mrigh=\paperwidth
\advance\Mrigh by-\hsize
\divide\Mrigh by 2
\output={

  \shipout\vbox{
    \offinterlineskip
    \iffinalrun
      \copy\rmarks
      \kern\Mdown
      \moveright\Mrigh
    \fi
    \vbox{
      \vbox to3pc{
        \line{\the\headline}\vss}
      \box255
      \vbox to3pc{
        \vss\line{\the\footline}}}}
\advancepageno}
```

Sometimes the book designer specifies off-center pages. Even-numbered pages should be moved to the right and odd-numbered ones, to the left. This brings facing pages closer together, and leaves extra room on the outside margins.

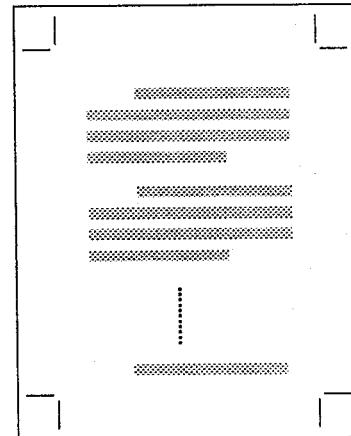
```
...
\divide\Mrigh by 2
\newdimen\Mleft \Mleft=\Mrigh
\advance\Mrigh.5in \advance\Mleft-.5in
```

```
\output={

  ...
  \moveright
  \ifodd\pageno\Mleft
  \else\Mrigh\fi
  ...


```

This off-centering is only done on the final run. Figure 6 shows a small page with registration marks.



Printer Registration Marks

Figure 6

The reader should now realize that three sets of dimensions should be specified when a book is designed and published. (1) The size of the sheet of paper which actually goes into the printer (specified by the human printer). After coming out of the printer, this sheet of paper is trimmed, at the crop marks, to (2) the size of the final page (`\paperheight` and `\paperwidth`). Finally, there is (3) the size of the printed area (`\hsize` and `\vsize`) on the page.

Example: A Border Around the Page

Here is an output routine which takes a 5" × 3" page and creates a border of size 5.5" × 3.5" around it. The border is done with `\leaders` (see [223] and [Ex. 21.8]). This example shows how easy it is to create a border out of a few characters. Ideally, 8 characters should be especially designed (Ref. 1), 4 for the four sides of the border, and 4 for the corner points. Their sizes should be chosen such that, e.g., the width of the border will be an integral multiple of the width of the top character. The reference point of each of the 8 should be placed such that they will align properly at the corners

(see discussion of the ‘quarter circle’ characters on [389–390]).

The border is built in three steps:

1. \box2 is set to a \vbox containing the border.

```
\font\bord=cmsy10
\def\topp{\bord\char'176}
\def\bott{\bord\char'020}
\def\lft{\bord\char'032}
\def\rt{\bord\char'033}
\setbox2=\vbox to3.5in{
    \offinterlineskip
    \hbox to5.5in{\leaders\hbox{\topp}\hfill}
    \hbox to5.5in{%
        \leaders
        \vbox to3.4in{
            \leaders
            \hbox to5.5in{\lft\hfil\rt}
            \vfil}%
        \hfil}
    \hbox to5.5in{\leaders\hbox{\bott}\hfill}
    \vss}
```

2. All the dimensions of \box2 are set to 0 by

```
\wd2=0pt \ht2=0pt \dp2=0pt
```

3. The output routine now ships out \copy2 (which does not move the reference point), followed by \box255.

```
\output={
    \shipout\vbox{
        \copy2 \vskip.25in
        \moveright.25in\box255}
    \advancepageno}
```

Example: Mailing Labels

This example introduces the concepts of logical and physical pages. The material placed in \box255 constitutes a *logical page*. The material actually shipped out by the OTR is a *physical page*. Usually one physical page is shipped for each logical page generated. In this example, since the mailing labels are small, several mailing labels are combined and shipped together as one physical page.

The data for the labels is assumed to reside on an external file, named `labels`, which contains information with format:

```
\name <the name>\\
\address <several lines of address>\\
```

The mailing labels are 3.5in wide and 1.5in tall each. There are 4 labels arranged vertically on a 6in tall page, without any gaps in between.

The data file is \input and macros \name, \address invoked automatically. Three approaches are described:

Approach 1. Macros \name and \address typeset the name and address in the desired format. The value of \vsize is set to the size of 4 labels. The OTR is thus invoked with a logical page consisting of 4 labels, and it simply ships it out, as one physical page.

```
\nopagenumbers \vsize=6in \hsize=3.5in
\def\name#1\{\nointerlineskip
    \vbox to.25in{\#1 \vfil}}
\def\address#1\{\nointerlineskip
    \vbox to1.25in{\kern.1in#1 \vfil}\penalty0}
\output={\shipout\box255}
\obeylines\parindent=0pt
\input labels
\bye
```

The \obeylines guarantees that each line in file `labels` will become a typeset line on the final page. The \nointerlineskip suppresses the normal interline glue that would otherwise be inserted between the boxes in the MVL. The \penalty0 is necessary to supply a valid page breakpoint. The reader will recall that a page can only be broken at a glue, a kern, or a penalty [110]. Without the \penalty0, the page builder would have no place to break the page, and would place the entire document, as one page, in \box255 at the end of the job.

Approach 2. Macro \name typesets the name in a \vbox. Macro \address typesets the address in another \vbox. \vsize is set to 1.5in, the size of one label. The OTR is thus invoked for each label and receives, in \box255, a logical page consisting of one label. It collects 4 such pages and ships them out as one physical page. This approach distinguishes between a logical and a physical page. Note also that, 3 out of 4 times, this OTR goes through a dead cycle.

```
\nopagenumbers \vsize=1.5in \hsize=3.5in
\def\name#1\{\vbox to.25in{\#1 \vfil}}
\def\address#1\{\vbox to1.25in{\kern.1in#1 \vfil}\penalty0}

\newcount\four \newbox\physpage
\output={
    \global\setbox\physpage=
    \vbox{
        \unvbox\physpage
        \box255}
    \global\advance\four by1}
```

```
\ifnum\four=4
  \shipout\box\physpage
  \global\four=0
\fi}

\obeylines\parindent=0pt
\input labels
\bye
```

The serious reader should try to understand what happens at the end of the document. Suppose we have 6 labels. The first 4 will be printed on the first page, and the last 2 will be accumulated, in `\box\physpage`, for the second page. When `\bye` is found, `TEX` finds out that `\deadcycles` $\neq 0$. It goes into its ‘endgame’ [264], where it prepares an empty page and invokes the OTR. This is repeated twice, accumulating two empty logical pages in `\box\physpage`, and then the OTR executes a `\shipout`, which clears `\deadcycles`, thereby stopping the ‘endgame’.

Approach 3. Macro `\name` typesets the name in a `\vbox`. Macro `\address` appends the address to the same box, sets its height to `1.5in`, and invokes the OTR. The OTR again collects 4 such boxes and ships them out as one physical page. This approach is interesting since it makes minimum use of the MVL, the page builder, and `\box255`. It does not use `\vsize` which, consequently, can be set to any value.

```
\nopagenumbers \vsize=0in \hsize=3.5in
\def\name#1{\strut\setbox0=\vbox{#1}}
\def\address#1{%
  \setbox0=\vbox to1.5in{
    \unvbox0
    \kern.1in
    #1 \vfil}
  \eject}
\newcount\four \newbox\physpage \newbox\toss
\output=%
\setbox\toss=\box255
\global\setbox\physpage=\vbox{%
  \unvbox\physpage\box0}
\global\advance\four by1
\ifnum\four=4
  \shipout\box\physpage
  \global\four=0
\fi}
\obeylines\parindent=0pt
\input labels
\bye
```

We want macro `\address` to invoke the OTR. However, the only part of `TEX` which invokes the

OTR is the page builder. It does that when it calculates a page break with infinite cost, or when it moves a penalty ≤ -10000 to the current page. Macro `\address` thus says `\eject`, which is essentially a `\penalty-10000`. The penalty is placed in the recent contributions and the page builder is exercised. The page builder tries to move the penalty to the current page but, since the current page is empty, the penalty is discarded [112]. The result is that the OTR is never invoked, and the job terminates without shipping out any pages.

To avoid this situation, macro `\name` typesets a strut. The strut is moved to the current page and, since the current page is no longer empty, the page builder agrees to move the `\penalty-10000` to the current page. This causes the page builder to break the page, place the current page (just the strut) in `\box255`, and invoke the OTR. The OTR does not need `\box255` and simply empties it.

It is possible, of course, to use any text instead of the strut.

The `\vsplit` Operation. It is important, when working with OTRs, to fully understand the `\vsplit` operation. Its syntax is: `\vsplit<box number>` to `<dimen>`, and the result is a box. Most often it appears in an assignment such as: `\setbox1=\vsplit0 to2.6in`. This sets `\box1` to a height of `2.6in`, moves material from the top of `\box0` to `\box1`, and keeps the remainder in `\box0`.

`TEX` assumes that the new `\box1` may have to be shipped out as part of the page. It therefore places a glue similar to `h` at the top of `\box1`. This glue is called `\splittopskip` and has a plain format value of `10pt` [348].

The most important thing to keep in mind is that a box can only be split *between* lines of text. If we perform a `\vsplit` to an ‘inconvenient’ size, `\box1` will come out underfull.

Example: `\vsize=375pt`. This creates, in `\box255`, a page of 31 lines (since $12(31 - 1) + 10 = 370$). The assignment `\setbox0=\vsplit255 to 184pt` will set `\box0` to a height of `184pt` with 15 lines of text. Since 15 lines of text occupy `178pt`, the remaining `6pt` should be filled with some flexible glue. If there is not enough flexible glue, `\box0` will come out underfull. If there are, e.g., two paragraphs in `\box0`, then its total stretchability is `2pt`. `TEX` will stretch it by `6pt` and will report an underfull box with a glue set ratio of `3 (300%)`. The remaining 16 lines in `\box255` will now occupy $12(16 - 1) + 10 = 190pt$. Any flexible glues in `\box255` will return to their natural size.

Here is an OTR which splits the page, ships out the top part and returns the rest to the MVL (actually, to the recent contributions):

```
output={\setbox0=\vsplit255 to1in
\shipout\box0 \unvbox255}
```

Splitting a Box in Two

Imagine a box with n lines of text, and with no flexible vertical glues. How can it be split in two equal parts? If n is even, this is easy. However, if n is odd, it is impossible. If H is the height of the box then $H = b(n - 1) + h$. Our approach is to split the box such that the top part will have $m = \lceil n/2 \rceil$ lines, and its height will thus be $H' = b(m - 1) + h$. To calculate H' we start with:

$$m = \lceil n/2 \rceil = \begin{cases} n/2, & n \text{ even}; \\ \lfloor n/2 \rfloor + 1, & n \text{ odd}; \end{cases}$$

Since TeX always performs an integer by integer division, the case where n is even is simple. It satisfies $m = n/2$ or $m - 1 = \frac{n-2}{2}$ and, therefore,

$$\begin{aligned} H' &= b(m - 1) + h = \frac{b(n - 2)}{2} + h \\ &= \frac{b(n - 1) + h + h - b}{2} \\ &= \frac{H + h - b}{2}. \end{aligned}$$

We therefore split to $H + h - b$. For even n , this is ideal. For odd n , the top part will contain one less line than the bottom. Since we want the top part, in such a case, to be larger, we loop, increasing the split size slightly, until the top part becomes larger than the bottom one.

```
\halfsize=\ht0
\advance\halfsize by\topskip
\advance\halfsize by-\baselineskip
\divide\halfsize by 2
\splittopskip=\topskip
{\vbadness=10000
\loop
\global\setbox3=\copy0
\global\setbox1=\vsplit3 to\halfsize
\ifdim\ht3>\halfsize
\global\advance\halfsize by1pt
\repeat}
```

This method is used on [417] to implement double-column pages (see below). It can also serve to split a box into k equal parts [397]*. The

principle is to first split the box to size $[n/k]$, then to split the rest recursively.

Next, consider the case where the original box contains flexible vertical glues. This considerably simplifies the problem, since the box can now be split to parts of almost any size and the glues will be flexed to fill up all the individual parts, eliminating any over/underfull boxes.

Example: Double-Column Pages

Multi-column pages are common in newspapers and technical publications. Note that, when using narrow columns, the tolerance usually has to be increased, leading to low quality results (see the experiments in [Ch. 6]).

Three approaches are described. The first one treats the two columns as separate logical pages, which are combined, in the OTR, into one physical page. This approach is described on [257], and is shown here for the sake of completeness. See also [Ex. 23.4] for a generalization of this case to three columns.

The second approach [417] creates one long and narrow column, and breaks it into two equal parts, which are typeset side by side. The problem of splitting a box into two equal parts has been discussed earlier.

The third approach uses the technique of the second approach to make it possible to switch between one- and two-columns at will.

Approach 1. The quantity $\backslash lr$ is a new control sequence, which is defined and redefined by the OTR depending on the current column (left or right). When the OTR is invoked for the first time (or the third, fifth,... times) it saves $\backslash box255$ (containing the left column) in another box. These are dead cycles. When it is invoked for the second time (or the fourth,...times), it ships out both columns.

```
\hsize=3.2in
\newdimen\fullhsize \fullhsize=6.5in
\let\lr=L \newbox\leftcolumn
\output={%
\if L\lr
\global\setbox\leftcolumn=\box255
\global\let\lr=R
\else
\shipout\hbox to\fullhsize{%
\box\leftcolumn\hfil\box255}
\global\let\lr=L
\advancepageno
\fi}
```

* It seems that the inequality shown there is wrong.

Again, it is important to understand what happens with the last page. If the last page ends somewhere in the left column, the \bye invokes the OTR, which saves the left column and returns without shipping out anything (a dead cycle). Since \deadcycles is now non-zero, TeX places an empty page in \box255 and invokes the OTR again. This time it ships a full page, so \deadcycles is reset. The last page comes out with unbalanced columns.

It is easy to add a headline and footnote to this example:

```
\vsize=2in \hsize=3.2in
\newdimen\fullhsize \fullhsize=6.5in
\let\lr=L \newbox\leftcolumn
\def\fulline{\hbox to\fullhsize}
\output={
  \if L\lr
    \global\setbox\leftcolumn=\box255
    \global\let\lr=R
  \else
    \shipout\vbox{
      \vbox to3pc{%
        \fulline{\the\headline}\vss}
      \fulline{\box\leftcolumn\hfil\box255}
      \vbox to3pc{
        \vss\fulline{\the\footline}}}
    \global\let\lr=L
    \advancepageno
  \fi}

\headline{\hfil\bf A Header\hfil}
\footline{\hfil\textrm{folio}\hfil}
```

Approach 2. The OTR is invoked with a long and narrow column. It splits it into 2 equal parts (see earlier discussion) and ships out a page consisting of the two parts, laid side by side. The original value of \hsize is saved in \ohsize. \hsize is then set to the width of a single column. \vsize should be set to twice its original value.

1. \newdimen\ohsize \ohsize=\hsize
2. \hsize=0.5\hsize \advance\hsize -.1in
3. \newdimen\halfsize
4. \output={
5. \setbox0=\vbox{\unvbox255}
6. \halfsize=\ht0
7. \advance\halfsize by\topskip
8. \advance\halfsize by-\baselineskip
9. \divide\halfsize by 2
10. \splittopskip=\topskip
11. {\vbadness=10000
12. \loop
13. \global\setbox3=\copy0

```
14.  \global\setbox1=\vsplit3 to\halfsize
15.  \ifdim\ht3>\halfsize
16.    \global\advance\halfsize by1pt
17.  \repeat}
18.  \shipout\hbox to\ohsize{\box1 \hfil\box3}
19.  \advancepageno}
```

On line 5, \box255 is unveiled. This is necessary since \ht255 equals \vsize but we want to start with a box of height $b(n - 1) + h$. Also, for the last page, \box255 may have a large \vfil at the bottom, which should be removed before the split. The right value for the split is calculated on 6-9. If the number of lines, n , is even, the split produces two equal parts. For odd n , the loop on lines 12-17 keeps incrementing the left column until it becomes one line larger than the right one. Both halves are shipped out, on line 18, side by side.

If the document contains just text, without equations or figures, the following can be used to calculate the best value of \vsize.

```
\newcount\col
\message{Enter number of lines per page: }
\read-1to\ent \col=\ent
\advance\col by-1 \multiply\col by12
\advance\col by10
\vsize=\col pt
```

It is easy to add a header and footer to the final page.

Exercise: Do it!

Approach 3. Switching between single- and double-columns. The principles of this method are described here, along with macros taken from the *manmac* [417]. They are the macros used to typeset the index of the *TeXbook*, which is why they use values such as 14pc and 89pc. The macros are easy to modify for different formats. The reader should also consult reference 2 for two corrections of the macros.

```
\newdimen\pagewidth \pagewidth=\hsize
\output{\shipout\box255}

\newbox\partialpage
\def\begindoublecolumns{\begingroup
  \output={\global\setbox\partialpage=
    \vbox{\unvbox255\bigskip}}
  \eject
  \output={\doublecolumnout}
  \hsize=14pc \vsize=89pc}
\def\enddoublecolumns{%
  \output={\balancecolumns}\eject
  \endgroup \pagegoal=\vsize}
```

```

\def\doublecolumnout
  {\splittopskip=\topskip
   \splitmaxdepth=\maxdepth
   \dimen0=44pc
   \advance\dimen0 by-\ht\partialpage
   \setbox0=\vsplit255 to\dimen0
   \setbox2=\vsplit255 to\dimen0
   \onepageout\pagessofar
   \unvbox255 \penalty\outputpenalty}
\def\pagessofar{\unvbox\partialpage
  \wd0=\hsize \wd2=\hsize
  \hbox to\pagewidth{\box0\hfil\box2}}
\def\balancecolumns
  {\setbox0=\vbox{\unvbox255}
   \dimen0=\ht0
   \advance\dimen0 by\topskip
   \advance\dimen0 by-\baselineskip
   \divide\dimen0 by2 \splittopskip=\topskip
   {\vbadness=10000
    \loop
      \global\setbox3=\copy0
      \global\setbox1=\vsplit3 to\dimen0
      \ifdim\ht3>\dimen0
        \global\advance\dimen0 by1pt
      \repeat
    \setbox0=\vbox to\dimen0{\unvbox1}
    \setbox2=\vbox to\dimen0{\unvbox3}
   } \pagessofar}

```

Macro `\begindoublecolumns` starts by defining an OTR which saves the page so far (single-column) in `\box\partialpage`. The `\eject` invokes this OTR. The OTR is then redefined to do double-column by splitting a long, narrow, `\box255`.

Macro `\enddoublecolumns` again redefines the OTR to split the page-so-far in two, and typeset the two halves, side by side, below the single-column in `\box\partialpage`.

Example: Facing figures

When two figures are textually related, the user may want them typeset on facing pages. The first figure should be typeset on top of the next even-numbered page and the second one, on top of the following page.

A macro `\facefig#1#2` is defined, with 2 parameters, the heights of the 2 figures. It saves the two values in `\dimen` variables `\figa` and `\figb`. The OTR checks the two variables. If the current page number is even and `\figa > 0`, room is reserved on top of the current page for the first figure by placing an empty box on the MVL, followed by `\box255`. If the current page number is odd, `\figa = 0` and `\figb > 0`, the OTR reserves

room for the second figure in a similar way. In either case the OTR goes through a deadcycle.

```

\newdimen\figa \newdimen\figb \newif\ifdead
\def\facefig#1#2{\figa=#1 \figb=#2}
\output={%
  \deadfalse
  \ifodd\pageno
    \ifdim\figa=0pt \ifdim\figb>0pt \message{b}%
      \vbox to\figb{}\unvbox255
      \penalty\outputpenalty
      \global\figb=0pt \deadcycles=0 \deadtrue
    \fi\fi
  \else
    \ifdim\figa>0pt \message{a}%
      \vbox to\figa{}\unvbox255
      \penalty\outputpenalty
      \global\figa=0pt \deadcycles=0 \deadtrue
    \fi
  \fi
  \ifdead\else
    \shipout\box255
    \advancenextpage
  \fi}

```

The simple macros above only use one pair of variables to save the heights of the figures. In practice, the user may expand `\facefig` before the OTR has handled the previous pair of figures. This will place new values in our variables before the old ones have been used. Our macros should, therefore, be extended so that any number of pairs of heights can be saved. The macros below save such pairs, as `\kern` values, in a `\vbox`.

```

\newbox\save
\newdimen\figa \newdimen\figb
\newif\ifdone \donetrue \newif\ifdead
\def\facefig#1#2{%
  \setbox\save=\vbox
    {\kern#1 \kern#2 \unvbox\save}}
\output={%
  \deadfalse
  \ifvoid\save\else
    \ifdone
      \global\setbox\save=
        \vbox{\unvbox\save
          \global\figb=\lastkern \unkern
          \global\figa=\lastkern \unkern}
      \global\donefalse
    \fi\fi
  \ifdone\else
    \ifodd\pageno
      \ifdim\figa=0pt \ifdim\figb>0pt
        \message{b=\the\figb;}%
        \vbox to\figb{%

```

```

\unvbox255
\penalty\outputpenalty
\global\figb=Opt \deadcycles=0
\deadtrue \global\donetrue
\fi\fi
\else
\ifdim\figa>Opt \message{a=\the\figa;}
\vbox to\figa{}}
\unvbox255
\penalty\outputpenalty
\global\figa=Opt
\deadcycles=0 \deadtrue
\fi
\fi\fi
\ifdead\else
\shipout\box255
\advancepageno
\fi

```

Another boolean variable, `\ifdone`, is declared. It is set to ‘false’ when two values are extracted from `\box\save`, and to ‘true’, when the two figures have been typeset. As long as it is ‘false’, we are in the process of typesetting two facing figures, and no new values are extracted.

The following extensions are left as an **Exercise**:

1. Macro `\facefig` should check to make sure none of its parameters exceeds `\vsize`.
2. `\facefig` should also accept the captions of the two figures, as additional parameters, and save them. The OTR should later retrieve and typeset the captions below (or above) the reserved areas.

Note! Our macros do not use insertions and are therefore incompatible with `\midinsert` and its relatives. Using both `\midinsert` and `\facefig`, the figures would be inserted in an unpredictable order.

Example: Shipping-Out Pages Selectively

The following code, part of the `manmac` format, can be used to produce only a subset of pages. The numbers of the desired pages should be placed on separate lines in a file called `pages.tex`.

The first line saves `\shipout`, which is a primitive, in macro `\Shipout`. Later, `\shipout` is redefined as either `\Shipout` or `\Tosspage`.

```

\let\Shipout=\shipout
\newread\pages \newcount\nextpage
\openin\pages=pages
\def\getnextpage{%
\ifeof\pages\else
{\endlinechar=-1\read\pages to\next
\ifx\next\empty % we should have eof now

```

```

\else\global\nextpage=\next\fi}%
\fi}
\ifeof\pages\else\message
{OK, I'll ship only the requested pages!}
\getnextpage\fi
\def\shipout{%
\ifeof\pages\let\next=\Shipout
\else\ifnum\pageno=\nextpage
\getnextpage
\let\next=\Shipout
\else\let\next=\Tosspage\fi\fi
\next}
\newbox\garbage
\def\Tosspage{\deadcycles=0\setbox\garbage=}
```

Detecting the End of the Document

How can the OTR find out if the page it has been given is the last one? The easiest way is to detect the `\vfill` at the bottom of that page. This can be done by:

```

\def\vfill{\vskip 1sp plus 1fill}
\output={%
\setbox0=\vbox to\vsized{
\unvcopy255
\ifdim\lastskip>Opt
\message{last page}\fi}
\shipout\box255
\advancepageno}
```

(The `\lastskip` command is explained in part II.) This usually works but may fail in cases where the last page is full, or almost full. An example is `\vsize=1in`, which leaves room for about six lines of text on the page. Let’s assume that the document has text for 6 lines, and the last line contains a deep character, such as a ‘]’, whose depth is 2.5pt. Setting `\tracingpages=1` generates the following in the log file:

```

% goal height=72.26999, max depth=4.0
% t=10.0 g=72.26999 b=10000 p=0 c=100000#
% t=22.0 plus 1.0 g=72.26999 b=10000 p=150 c=100000#
% t=34.0 plus 1.0 g=72.26999 b=10000 p=100 c=100000#
% t=46.0 plus 1.0 g=72.26999 b=10000 p=100 c=100000#
% t=58.0 plus 1.0 g=72.26999 b=10000 p=150 c=100000#
% t=70.0 plus 1.0 g=72.26999 b=1168 p=0 c=1168#
% t=72.5 plus 1.0 plus 1.0fill g=72.26999 b=** p=-20000 c=**
```

The cost of breaking the page after the first six lines is 1168, very low. The page builder, however, waits for a point with infinite cost before it decides on a page break. It continues reading the source and finds the `\bye`. The definition of `\bye` is [357] `\par\vfill\penalty-20000\end`. The page

builder adds the `\vfill` to the current page, causing the depth of the current page to become zero. The depth of the bottom line (2.5pt) is now added to the height of the current page, with the result that it is too high (72.5pt). The `\vfill` is therefore removed, and the last page is shipped out without any fill at the bottom.

References

1. Knuth, D. E., *A Course on METAFONT Programming*, TUGboat 5, no. 2, pp. 105–118, Nov. 1984.
2. Platt, C., *Macros for Two-Column Format*, TUGboat 6, no. 1, pp. 29–30, March, 1985.

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

or vice versa in a document. The only way that I have been able to do this is to have two macro files containing the following information. The first file I call `double.tex`:

```
\renewcommand{\baselinestretch}{2}
\large
\normalsize
```

and the second file I call `single.tex`:

```
\renewcommand{\baselinestretch}{1}
\large
\normalsize
```

An example of how to use these files follows below.

Spacing Example

The first paragraph of this example exhibits normal single spacing. Now we give commands to change this.

This paragraph should be double spaced. Fortunately the columns of *TUGboat* are narrow and I do not need much text. Let's switch again.

Now we proceed with the remainder of our document to demonstrate that the spacing has returned to normal.

Example Commands Shown

```
%  

The first paragraph of this example exhibits  

normal single spacing. Now we give commands  

to change this.
```

```
% blank line must appear
\input double % switch to double spacing
This paragraph should be double spaced.
Fortunately the columns of \TUB{} are
narrow and I do not need much text.
Let's switch again.
```

```
% blank line must appear
\input single % switch to single spacing
Now we proceed with the remainder of our
document to demonstrate that the spacing has
returned to normal.
```

Comments

Comments are always welcome. I will try to take most of the material for this column from the telephone calls or e-mail that I receive. They may appear to be beginner-ish for some people, but there are new users entering the *TeX* world everyday. If anyone feels that they would like a more advanced

The LATEX Column

Jackie Damrau

It has been a while since my last column, but for a very good reason. I have been relocating to a better position and am now stable enough to get back to producing this column for each issue.

With this issue, I will answer the question that is asked most often by those of you who have contacted me by telephone: "How does one doublespace a LATEX document?" To accomplish double-spacing, place the following command in your preamble:

```
\renewcommand{\baselinestretch}{2}
```

The preamble is the area after the `\documentstyle` declaration and before the `\begin{document}` declaration.

Another question that I am repeatedly asked is how to switch from singlespacing to doublespacing

column, I welcome any comments on suggested topics.

I am available through e-mail, surface mail and the telephone at the addresses and numbers listed below.

- ◊ Jackie Damrau
Physics Research Division
MS-2002
SSC Laboratory
2550 Beckleymeade Avenue
Suite 260
Dallas, TX 75237-3946
(214) 708-6048
Bitnet: damrau@sscvx1
Internet: damrau@sscvx1.ssc.gov

Announcing Two Reports from the Rijksuniversiteit Groningen

C. G. van der Laan

Two reports on topics related to L^AT_EX have been issued recently by the Computer Center of the Rijksuniversiteit Groningen. Copies of these reports can be obtained by writing to the publisher at this address:

Rekencentrum RUG
Landleven 1
9700 AV Groningen, The Netherlands

SGML-L^AT_EX

C.G. van der Laan, D.C. Coleman, J.R. Luyten (1989):

*SGML-L^AT_EX 1. Mathematical formulas.
(English version)
RC-RUG report 24
Rekencentrum RUG*

The text of the foreword follows.

Within the Dutch SGML and T_EX user groups the question arose to what extent SGML and L^AT_EX are related. A working group comprised of J. Bleeker, H. Dekker, R. Doornbal, C.G. van der Laan, and D. van Wijnen, was formed in order to consider the question.

It was recognised that extensive copy with complex document elements, not extensive copy with simple structures nor copy of limited size with complex structures, is the issue that has to be addressed.

We gathered qualitative information about how people practise 'electronic publishing' by means of a (mini) inquiry.

Some relevant document element categories were selected such as mathematical formulas, tables, and illustrations. The original aim was to work out some representative examples and bundle these in one report. In the course of the project it appeared more practical to report in parts.

The first reports to see the light of day were the specifications of card distributions in BRIDGE. The L^AT_EX aspects are published in [1].

This report concerns mathematical formulas. The L^AT_EX specifications have been worked out by J.R. Luyten, while the SGML work has been done by D.C. Coleman. After the Dutch version of this report emerged, Grootenhuis [2], has been engaged in coupling the SGML descriptions to the L^AT_EX specifications.

References

- [1] C. G. van der Laan, "Typesetting Bridge via L^AT_EX." *TUGboat* 10, no. 1, pp. 113-116.
- [2] J. Grootenhuis, personal communication.

Journal Style Guidelines: A Report on a New L^AT_EX Style

L. Steemers & C.G. van der Laan

*Journal Style Guidelines
RC-RUG report 26
Rekencentrum RUG*

This report is a worked-out example of how the general article style can be adapted towards a specific journal. Since it is not the intention of the authors that people redo the same work for another journal, we would like to share our experiences with our readers. The next thing we are considering is Generic Journal Style Guidelines, with the aim that targeting a specific journal can be accomplished within hours instead of months.

- ◊ C. G. van der Laan
Rijksuniversiteit Groningen
The Netherlands

International L^AT_EX Is Ready To Use

Joachim Schrod

Abstract

International L^AT_EX (short IIATEX) is a free, supported version of L^AT_EX which allows users to typeset non-English documents with the standard L^AT_EX-layout. Furthermore, a problem with the selection of math delimiter sizes is solved, and the solution may also be inserted directly into other macro packages which are based on plain T_EX.

1 Why International L^AT_EX?

L^AT_EX [7] is a widespread document markup system which allows users to separate the structural markup of a document from its optical markup. It was written by LESLIE LAMPURT to support the creation of *American* documents. This aim is manifested twofold: (i) all markups produce text in American English (e.g., \today results in 3 Mar 1990, \chapter in *Chapter*, etc.); and (ii) the layout uses very American fashioned design principles. But because it was the first (and only) macro package widely available which allowed the logical handling of documents, it is now in use all over the world.

The American nature of L^AT_EX has led very often to modifications of L^AT_EX styles which substitute the fixed American texts with fixed texts in another language (e.g. German or even English). Subsequently some style options have floated around in which the texts are produced by macros; one of these is the *german* style option [10] of HUBERT PARTL in Vienna, which first resulted in an "Austrian L^AT_EX," later extended to a "German L^AT_EX," and now even may result in a "French L^AT_EX" and an "English L^AT_EX." But all these style options have a drawback: the small word *may*. They need a L^AT_EX version where the fixed texts in the style files are substituted by macro calls—they need "IIATEX."

IIATEX has *all* fixed strings within L^AT_EX and its standard styles replaced by macro calls. It takes care that it is fully input compatible with the original L^AT_EX. Only a few new command names have been reserved. It is a supported, but free, framework for those who want to use language specific style options like *german* not only with the *article* style but also with the other L^AT_EX styles and style options. Details about the usage, the realization, the distribution, the support, and about future work follow in the next sections.

IIATEX does not solve the problem of the American layout. It is a shortcut—what we really need are special document styles for other countries with

other typographic habits. It solves the problem of being able to exchange our documents now; but it works against the encouragement: "GO FORTH now and create *masterpieces of the publishing art!*" [6, p. 303]. (Perhaps this is just a personal point, but I am strongly influenced by traditional typography in the spirit of the late JAN TSCHICHOLD and I always get a headache when I look at those large boldface chapter headings...) But I don't know of any European book designer¹ who would be willing to share his knowledge for completely new document styles, besides the fact that new styles are not easy to write with the unspecified and often unmodular internal interface of L^AT_EX.

Besides the variability of fixed texts, IIATEX provides two repairs for the typesetting of single characters. These repairs concern problems which have been addressed very often in T_EXhax. They are done within *lfonts.tex* and are independent of the work of FRANK MITTELBACH and RAINER SCHÖPF who have written a new *lfonts.tex*.

- \pounds now produces £ instead of £. This was an often discussed theme in T_EXhax. (Another point where you can see that L^AT_EX was not written by a person coming from Great Britain.)
- If a writer uses a document design size of 11 or 12 pt, the sizes of math delimiters are wrong if e² follows the hints of DONALD KNUTH about "*Fine Points of Mathematics Typing*." Consider the following example which is typeset as it would be in a 12 pt article:

$$f(n) = n_j \cdot \left(\sum_{i=0}^n m_i \right)$$

The parentheses have been produced with \biggl resp. \biggr according to [6, p. 149]. When we change the size of the \sum-symbol to fit with the larger variables, we have to change the resulting size of \biggl and \biggr, too. Have a look at

$$f(n) = n_j \cdot \left(\sum_{i=0}^n m_i \right)$$

In this example, the sizes fit the rest of the formula. This one is produced by IIATEX.

¹ A recent article in *TUGboat* [1] explores a first step in this direction. But it is not noted if a professional book designer was involved in this effort.

² I borrow the notation of "e" as a substitution for "he or she" from MICHAEL SPIVAK.

I think that this section has given you an overview about the functionality of II^AT_EX. In the following sections I will consider some of the above topics in greater detail. But first I want to emphasize that *the fact that so many discussions about L^AT_EX features and L^AT_EX problems have occurred in the last years is the best sign that L^AT_EX is what people need.* Many people grumbling means: many people are using L^AT_EX... I did not and do not want II^AT_EX as competition to or even substitution for L^AT_EX where all the stuff which LESLIE LAMPORT does not incorporate into L^AT_EX gets fed. It's just a small (well, not in file sizes) supplement to standard L^AT_EX for those like me who live not in God's Own Country.

2 The Name of the Game

This version of L^AT_EX is called II^AT_EX because LESLIE LAMPORT holds a copyright on L^AT_EX and I don't want to confuse authors about what they are using. Both the version message of L^AT_EX and all version messages of style files have been changed, too. They now print out the original date of the style file and the date when the international adaption has taken place (this can be earlier, see section 5).

I would be happy if these changes or other changes based on them (cf. section 7) would be incorporated into the standard version of L^AT_EX. (This is especially true because then someone else will have to do the work of distribution and support.)

3 The User Interface

The author usually does not recognize the difference between L^AT_EX and II^AT_EX. E uses a *language* style option in er \documentstyle markup (there is no restriction about the sequence of this language style option and other style options). Or e uses no language style option at all, then e gets the standard American version.

That's all for the author. But there remain those who want to (or are forced to) write their own language style option. Or even worse, those poor souls who write style files for L^AT_EX and want that their styles can be used by foreign people, too. Well, let's start with the first ones. A language style option must define the macros of Table 1 which have the named default values. Of course, e may define several different notations so that the author can switch between them if e wishes. The names of the macros have been chosen according to the proposal of HUBERT PARTL [9].

The most commonly used language style options (e.g. those where national groups have decided

MACRO	DEFAULT VALUE
\contentsname	Contents
\listfigurename	List of Figures
\listtablename	List of Tables
\abstractname	Abstract
\partname	Part
\chaptername	Chapter
\appendixname	Appendix
\refname	References
\bibname	Bibliography
\indexname	Index
\figurename	Figure
\tablename	Table
\enclname	encl
\ccname	cc
\headtoname	To
\pagename	Page
\seename	see
\alsoname	see also
\today	Date in American format

Table 1: Adaptable macro names and default values

to prefer them) should be sent to me. I will incorporate them into the distribution. An example language style option, deutsch with German names, is part of the distribution.

A macro author who wants that er marvellous new style may be used by foreign people too, must use the macros above. E must provide an American default text and e must take care of redefinitions. I.e., e must not define the default value if the macro is already defined (this should be tested with \ifx\mac\undefined and neither with \@ifundefined\mac nor with \@ifdefinable\mac). If e needs a fixed text which is missing e should use a macro name ending in name. This name and its default value should be sent to me, and I will try to coordinate them. Please remember that no user-defined L^AT_EX macro may start with \end.

4 Documentation

The documentation seems to be the most important problem with L^AT_EX. LESLIE LAMPORT has written installation instructions (*latex.ins*) where he points out the things which have to be done during the installation of L^AT_EX, but there seem not to be many people reading them. I have seen commercial versions of T_EX with an added L^AT_EX where they still deliver the Local Guide of DEC SRC, and local.gid still says

To print file `foo.dvi` on the Imagen printer, which is located on the third floor near Jane's desk, [...]

How many people know Jane?

By now, I have finished a "Generic Local Guide" in German language, and an English version will follow soon. Generic Local Guide means that at all places where system dependencies must be specified, a hint to these system dependencies is typeset in a frame. This is done with the markup `\todo` which can be easily searched for so that the process of producing a Local Guide is (I hope) less tedious. Furthermore, this has the advantage that a reader who gets a printout of the Generic Local Guide realizes that there is some special information missing.

5 Realization

The original `LATEX` files have not been changed. Instead for each `LATEX` file there exists one or more (WEB) changefiles where the necessary changes are specified. These changefiles are merged into the original `LATEX` files with `TIE` [3]. This process makes it easy to update `ILATEX` files when new `LATEX` files arrive: just run `TIE` again and in most cases the work is done.

Besides the mentioned changes, all form feeds are removed from the files. This has been done because they often result in unwanted garbage during the transfer over electronic networks, and the advantage of a more structured printout does not seem to be worth the trouble.

`ILATEX` is distributed with three different `lfonts.tex` files, and two others perhaps will be added in the near future:

1. The (almost) original `lfonts`, renamed to `lfonts_ori`. In this file the comments were adapted to the code which has been changed in the meantime. Furthermore the code has been updated according to a newer version of `plain.tex`.
2. The default `lfonts.tex`. In this file `\pounds` and the `\big` macros are redefined. `\pounds` is typeset in the upright italic typeface `\ui` which expands to `cmu-fnts`.

Every font-size command sets up boxes with the proper height and depth for `\big` delimiters. The reference characters are the opening parentheses in the font `\textfont3` (i. e. `cmex`). This can be used afterwards for the construction with `\left` and `\right`. This is a general and precise solution which can be used in Plain `TEX`, too. JOHN HOBBY's solution [5] of

computing the height and the depth out of the dimensions of a strut is not similarly general. H. KOPKA's solution (presented at the German `TEX` User Meeting 1989 in Eichstätt) of inserting the explicit dimensions in every size specific style file is not a solution but a patch.

The above implies in particular that this `lfonts.tex` uses a few new fonts, namely `cmu10` (loaded on demand) and `cmex10` (preloaded) in scaled and unscaled versions. The tables in [12] have to be changed accordingly.

3. The third `lfonts` is named `lfonts.ful`. It has no unavailable fonts any more and uses reduced fonts for unavailable small fonts like `cmsl7`. Please note that it is not sufficient to remove the comments in the first part of `lfonts.tex`—the font-size commands must be changed, too!
4. JOHN SAUTER's `fonts.truesize` [11] will perhaps be added. It's the `lfonts`-version I like most.
5. The `lfonts`-version of FRANK MITTELBACH and RAINER SCHÖPF [8] may be added. But please note that their work is independent of mine: they modified the font selection scheme—I advanced `lfonts` at the level below, at the availability of fonts and the definition of characters.

The current date of `ILATEX` is December, 1989.

6 Distribution and Support

The support of `ILATEX` is done by me: as soon as new `LATEX` files arrive in Stanford they are checked to see if the changefiles still match and the derived `ILATEX` files are produced.

Distribution is a problem. First the good messages: `ILATEX` is or will be available over the Bitnet Listserver `LISTSERV@DHDURZ1`³, the Clarkson server, the Aston `TEX` archive, and per anonymous ftp from Washington and Utah. It will be distributed by DANTE e.V., the `TEX` Users Group for the German-speaking part of the world. And it will be included in the VM/CMS and the BSD/UNIX-flavoured `TEX` distribution. Of course, it is delivered with our commercial `sTEX` system for the Atari ST as well. I want to thank Nelson Beebe, Michael deCorte, Joachim Lammarsch, Pierre MacKay, and Sebastian Rahtz for providing this service.

But now the bad message: I'm not able to e-mail it out individually. The complete system with original `LATEX` files, changefiles, derived `ILATEX` files,

³ I have been told that there will be a slave list-server in the States soon.

documentation, and TIE is about 2 MB [sic!]. I have it as a compressed archive on a PC floppy disk (still 850 KB) but the free distribution is too expensive for me (especially in terms of my working time). If you, dear reader, are someone who manages a TeX distribution please contact me. I will send it to you for free. But if you are "just" a user, you have to pay for it: the basic handling charge will be about 15 DM (\approx 8 \$), postage and bank fees must be added (the more distant from the FR Germany, the more).

The changefiles and the English Generic Local Guide are free software, i. e. they are available under the conditions of the GNU General Public License. In short this means: if you give them away you may not hinder the receiver from distributing and using them freely. This restriction is transitive. For the sake of all TeX companies out there: "Mere aggregation of another independent work with the program (or its derivative) on a volume of a storage or distribution medium does not bring the other work under the scope of these terms." [2]

The legal state of the derived LATEX files is unclear to me—LESLIE LAMPORT holds a copyright on the original LATEX files.

The German Generic Local Guide may be used freely for individual purposes (within companies, too) but a distribution with other TeX stuff—be it public domain or commercial—must not be done without my prior written permission.

7 Future Work

There still remain a few things to do: the English Generic Local Guide is not finished yet. CHRISTINE DETIG was so kind to start preparing an index for the Generic Local Guide which will enhance the usability in an enormous way.

Someone whose name I do not remember has pointed out in a message to the GUTenberg discussion-list that it would be convenient not only to make the fixed strings within LATEX adaptable but also to be able to rearrange the whole text (e.g. to print "*1st Chapter*" instead of "*Chapter 1*"). Since all places where such changes have to take place are isolated in the changefiles, this would be a straightforward task. Any contributors?

TIE—which is included in the distribution—was written in WEB and later rewritten in CWEB. But this is our CWEB [4], not SILVIO LEVY's, and the two versions are not compatible. It should be re-worked so that it can be used with LEVY's CWEB and the Spider WEB, too.

References

- [1] Johannes Braams, Victor Eijkhout, and Nico Poppelier. The development of national LATEX styles. *TUGboat*, 10(3):401–406, 1989.
- [2] Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA. *GNU General Public License*, February 1989. Version 1.
- [3] Klaus Guntermann and Wolfgang Rülling. Another approach to multiple changefiles. *TUGboat*, 7(3):134, 1986.
- [4] Klaus Guntermann and Joachim Schrod. WEB adapted to C. *TUGboat*, 7(3):134–137, 1986.
- [5] John Hobby. *fixup.sty*. LATEX style option, September 1988.
- [6] Donald E. Knuth. *The TeXbook*, volume A of *Computers and Typesetting*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1986.
- [7] Leslie Lamport. *LATEX: A Document Preparation System*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1986.
- [8] Frank Mittelbach and Rainer Schöpf. A new font selection scheme for TeX macro packages — the basic macros. *TUGboat*, 10(2):222–238, 1989. This article contains an unreferenced hint to "The new font family selection—User Interface to standard LATEX".
- [9] Huber Partl. German TeX. *TUGboat*, 9(1):70–72, 1988.
- [10] Huber Partl. *german.sty*. LATEX style option, TU Wien, December 1989.
- [11] John Sauter. Building computer modern fonts. *TUGboat*, 7(3):151–152, 1986.
- [12] Joachim Schrod. LATEX fonts and suggested magnifications. *TUGboat*, 9(3):241–243, 1988.

◇ Joachim Schrod
Detig · Schrod TeXsys
Kranichweg 1
D-6074 Rödermark-Urberach
FR Germany
Tel. (0 60 74) 16 17
Bitnet: XITIJSCH@DDATHD21

The New Font Family Selection — User Interface to Standard L^AT_EX

Frank Mittelbach
Rainer Schöpf

Contents

1 General remarks	91
1.1 Special math alphabets	
2 Processing older documents	94
3 Setting up a new format	94
3.1 Preparations	
3.1.1 Preloading Fonts	
3.1.2 Making more fonts available	
3.2 Running IniTeX	
4 Remarks on the development of this interface	96
5 Acknowledgements	96
6 List of distributed files	97

Abstract

In this article we describe the use of the new font selection scheme in the standard L^AT_EX environment. The main characteristics are:

- The possibility to change family, series, shape and sizes independently of one another.
- The existence of a style file to process older documents without any changes to their layout and their input files.
- A macro setup which is consistent with existing standard document styles.¹

It is planned to incorporate this font selection scheme into L^AT_EX version 2.10.

1 General remarks

In TUGboat 10, no. 2 we presented a new scheme to select fonts in TeX macro packages. This article describes the use of this new scheme in the L^AT_EX environment. The technical parts of the interface (which are of some interest to readers who plan to use our scheme with other fonts or with other macro packages) will be published in a separate article.

The necessary macros are distributed by the AMS together with the amstex.sty option which was announced in TUGboat 10, no. 3. The availability of the new font selection scheme at the usual

¹ However, small changes in the document styles would make font changes a bit faster.

servers will be announced separately in TeXhax, etc. Please refrain from asking for personal distribution.

To get a better understanding of this L^AT_EX interface, some words on the organisation of font families are in order. Readers of our article about the basic macros will notice that our understanding of these matters increased while working on this interface and the amstex.sty project; in some regards we have changed our point of view rather drastically. Surprisingly, only a few internal details within the basic macros needed adjustment; it seems that even without the real understanding, we instinctively got most of the things right when we designed them. (But probably we are still ignorant of the underlying concepts.)

In his book about "Methods of Book Design" Hugh Williamson writes [1]

[...] To the printer, an alphabet is a set of twenty-six letters of a certain design and body, together with a few additional combinations of letters. A *fount* is usually made up of a set of alphabets of one size and based on one design. It may consist of one alphabet only, if no more alphabets exist in that design and size. Usually however a text fount will comprise five alphabets — roman and italic upper and lower-case, and small capitals. [...] A *series* is a set of founts closely related to each other in design, and usually very similar to each other, but graded in size. If only one alphabet has been made in a certain design, that alphabet alone may be a series. A *family* is a group of series compatible for composition, but loosely related in design. A family may include excerpts from more than one series.

Since TeX's physical fonts (which is the American word for fount) all contain exactly two alphabets, namely the upper and lower-case alphabets of a certain design, we will use the word font for physical TeX fonts, and fount for bundles of TeX fonts consisting for example of roman (upright or normal), italic and small capitals shapes.

The above quotation gives a good clue how to organize fonts in our font selection scheme. Hence we use the \shape command from the basic macros to distinguish between normal (n), italic (it), small caps (sc), sloped or slanted (sl) and upright italic (u) typefaces within one fount. Founts of differ-

ent sizes form a series, so we use `\size` to access these. We think that the weight and the width of a series are good candidates to distinguish between individual series, therefore we combine them in the `\series` command. Again we use one and two letter abbreviations as shown in table 1. One or more of these series form a family which is accessed via the `\family` command.

To give some practical example, we arranged the most important families of the Computer Modern fonts according to this classification in table 2. Please note that some families like ‘computer modern funny roman’ (`cmff`) or ‘computer modern sans serif quotation’ (`cmssq`) are unclassified. These special purpose fonts are not accessible in the standard distribution of the new font selection scheme, although they could be added easily in a style file.

Given this overview about the classification of fonts it should be clear how to select a specific font with the primitive commands `\family`, `\series`, `\shape`, `\size`, and `\selectfont`. As described in [5], the `\size` macro takes two arguments: the size in printer’s points as a numeral (i.e. without the dimension) and the corresponding `\baselineskip` value (with a dimension). `\selectfont` finally selects the font using values of the surrounding environment if some of the commands are missing. For example, statements like “Concrete roman condensed slanted font at 9pt with 11pt leading”² will be translated into command sequences of the form:

```
\family{ccr}\series{c}\shape{s1}%
\size{9}{11pt}\selectfont
```

This will explicitly load the font mentioned above, provided the necessary font shapes are known to the system.³

\LaTeX knows about two versions called ‘normal’ and ‘bold’. As the name indicates, `\mathversion{normal}` is the default. In contrast, the bold version will produce bolder letters and symbols. This might be suitable in certain situations like headings, but recall that changing the version

² This is pronounced “leading” and measures the distance between the baselines of succeeding lines. To \TeX users it is known as `\baselineskip`.

³ Among the AMS distribution an example style option ‘concrete.sty’ is provided which makes the Concrete roman as well as the Euler math fonts available. These fonts were used to typeset [3], [4] and this article.

means changing the appearance (and perhaps the meaning) of the whole formula. If you want to bolden only some symbols or characters within one formula you should not change the `\mathversion`. Instead you should define a special math alphabet for characters (see below) and/or use the command `\boldsymbol` which is provided by the document style option ‘amsbsy’. For historical reasons \LaTeX maintains two abbreviations to switch to its math versions: `\boldmath` and `\unboldmath`.

Other versions could be provided in special style options. For example the ‘concrete’ option mentioned before sets up a version called ‘euler’ to typeset formulas in the same way as it was done in [4].

1.1 Special math alphabets

But simple formulas with one alphabet and a huge number of symbols are not sufficient for mathematicians to expose their thoughts properly. They tend to use every available typeface to denote special things.

To cope with this need for special alphabets in formulas, we introduce the concept of *(math alphabet identifiers)*. These constructs are special commands which switch to a specific typeface. They might correspond to different typefaces in different math versions but within one version they always select the same typeface regardless of surrounding conditions.

A *(math alphabet identifier)* can be defined according to the users’ needs but standard \LaTeX already has a few of them built in. They are described in table 3.

When using such an *(alphabet identifier)* two syntax variants are available: one can understand a command like `\cal` as a switch to a different font, i.e. using a syntax `{\cal ...}` as the old \TeX does, but we prefer to view the *(math alphabet identifier)* as a command with one argument, i.e. to use a syntax of the form `..\cal{A}..` To select the first alternative a style option ‘nomargid’ is provided. This option is automatically selected if the ‘oldlf’ option is used since this option is supposed to produce identical results for older documents.

New *(math alphabet identifiers)* are defined in two steps. First the identifier is made known to the system with the `\newmathalphabet` command. Then specific typefaces in some or all

Weight Class		Width Class		
Ultralight	ul	Ultracondensed	50%	uc
Extralight	el	Extracondensed	62.5%	ec
Light	l	Condensed	75%	c
Semilight	sl	Semicondensed	87.5%	sc
Medium (normal)	m	Medium	100%	m
Semibold	sb	Semiexpanded	112.5%	sx
Bold	b	Expanded	125%	x
Extrabold	eb	Extraexpanded	150%	ex
Ultrabold	ub	Ultraexpanded	200%	ux

Table 1: Weight and width classification for fonts. The percent values are derived from [2]. To combine the abbreviations in the \series command, weight is used first and any instance of medium (m) is dropped except when weight and width are both medium. In this case one single m is used. So bold expanded would be bx whereas medium expanded would be x.

Computer Modern families

family	series	shape(s)	Example of external names
<i>Computer modern roman</i>			
cmr	m	n, it, sl, sc, u	cmr10, cmti10, cmsl10, cmcsc10, cmu10
cmr	bx	n, it, sl	cmbx10, cmbxti10, cmbxsl10
cmr	b	n	cmb10
<i>Computer modern sans serif</i>			
cmss	m	n, sl	cmss10, cmssi10
cmss	bx	n	cmssbx10
cmss	sbc	n	cmssdc10
<i>Computer modern typewriter</i>			
cmtt	m	n, it, sl, sc	cmtt10, cmitt10, cmsltt, cmtcsc10
<i>Computer modern fibonacci</i>			
cmfi	m	n	cmfib8

Table 2: Classification of the Computer modern fonts. You will notice that not all possible combinations of family, series and shape are available. E.g. there is no small capitals shape in the medium series of the computer modern sans serif. However, Philip Taylor announced recently that he has filled some of the holes. It might be a good idea to include such additional parameter files for METAFONT into the general distributions.

LATEX knows about three *(math alphabet identifiers)*. \cal will select calligraphic letters like *ABCD*, \mathrm will select upright roman letters for use in functions like max_i, and finally \mit selects the default math italic alphabet.

Table 3: Predefined *(math alphabet identifiers)* in LATEX

(math versions) are assigned by means of the \addtoversion command.

Let us discuss this process in detail. Suppose that you want to make a sans serif typeface available as a math alphabet. First we choose a new command name (e.g. \sfmath) and tell L^AT_EX about it with the line

```
\newmathalphabet{\sfmath}
```

Then we consult table 2 to find suitable fonts to assign to this alphabet identifier. As you find out, the computer modern sans serif family consists of three series, a medium, semi bold condensed and a bold extended one. The medium and the bold extended series both contain a normal shape typeface. So we add the line:

```
\addtoversion{normal}{\sfmath}{cmss}{m}{n}
\addtoversion{bold}{\sfmath}{cmss}{bx}{n}
```

Now our alphabet identifier is ready for use in these two versions. We demonstrate this with the formula

$$\sum A_i = \tan \alpha$$

which was produced by

```
\mathversion{normal}
\[\sum \sfmath{A}_i = \tan \alpha\]
```

Note that we first switched back to the normal version. This was necessary since this article is typeset with a third version (Euler) in force. If we had tried to use \sfmath in this version we would have gotten an error message stating that this *(math alphabet identifier)* isn't defined for the Euler version.⁴

If we are interested in a slanted shape we have to face a problem: there is no slanted shape in the bold extended series of the Computer Modern sans serif family. So, if we make the identifier known only in the normal version then it would produce an error message when encountered in the bold (or any other) version. Of course we can get by using always the same typeface in all versions. To make this task a bit easier there is also a * variant of the \newmathalphabet command which takes three

⁴ Actually we cheated a bit more in this article: we had to reset the \mathcode of certain characters because they are in different places in the Euler version. A few more details can be found in Don Knuth's article [3]. However, this is not a real problem because such changes can be done in commands similar to \boldmath if such incompatible versions are to coexist in real applications.

more arguments: the default values for family, series and shape for all math versions in which the alphabet identifier is not explicitly defined via an \addtoversion command. So our second example can be set up simply by stating

```
\newmathalphabet*{\sfmath}{cmss}{m}{s1}
```

This would have the additional advantage that this math alphabet identifier is also allowed in math versions which are defined in style files or document styles (like the Euler version mentioned earlier). Any explicit \addtoversion command overwrites the defaults given by \newmathalphabet*; so, it might be a good idea always to specify default values.

Here we show the same formula as above, but this time in the Euler version and with \sfmath instead of \sfmath:

$$\sum A_i = \tan \alpha$$

2 Processing older documents

To typeset documents which are written with the old L^AT_EX (i.e. with a format using the old font selection scheme) only the source line containing the \documentstyle command has to be changed. More exactly the 'oldlf' option must be added to the list of document style options if the new font selection scheme is in force.⁵

3 Setting up a new format

This section is written for people called 'local wizards' by the L^AT_EX manual, which simply refers to the (unfortunate) guys who are always being pestered if things do not work.⁶ If you are using L^AT_EX on your own PC you might have to read this section, too, even if you don't feel like being a wizard.

3.1 Preparations

Before generating a new format it is necessary to rename a few files. This enables you to customize the format to the special needs of your site.

lfonts.tex First of all you should rename the file lfonts.tex (supplied with the standard distribution of L^AT_EX); otherwise you will always end up with an old format. Call it, say, lfonts ori.

⁵ This means that it is the default (see next section).

⁶ YOU might belong to this group!

`hyphen.tex` Another file which should probably be renamed is `hyphen.tex` (the original American \patterns from Don Knuth) because this enables you to insert your favourite \pattern package when `IniTeX` is asking for this file. This might even be useful if you use `TeX` version 3.0 which is multilingual (assuming that your computer has only a limited memory).

3.1.1 Preloading Fonts

Now you have to decide which fonts to preload in your format. Unlike the old font selection scheme of `LATeX`, where only preloaded fonts could be used in math applications (like subscripts etc.), the new font selection scheme poses no restriction at all; documents will always come out the same. So you have to take your pick by weighing the two conflicting principles:

- Preloading often used fonts might make your `TeX` run a bit faster.
- Using more load-on-demand fonts will make your format much more flexible, because you can switch to different families far more easily. After all, there is an upper limit to the number of fonts `TeX` can use in one run and every preloaded font will count even if it is never accessed.

On the PC at home we nowadays always use formats with only 5 fonts preloaded.⁷ We don't think that `TeX` is actually running much more slowly than before.

Together with the new font selection scheme two files `preload.min` and `preload.ori` are distributed. The first one will preload next to nothing while the second will preload the same fonts as the old `lfonts.tex`. You can copy either of these files to `preload.tex` and then change it if you want to preload some other fonts. But please make sure that you don't change one of the original files of the distribution.

3.1.2 Making more fonts available

Besides deciding which fonts to preload, you also have to tell the `TeX` system which external fonts are available and how they are organized in families, series, shapes and sizes. In short you have to set

⁷ This is the absolute minimum. These fonts are accessed by `lplain.tex` and `latex.tex` when the format is generated.

up internal tables giving informations like "family cmr, series b, shape n, size 10 is associated with the external font cmb10 but there is no font with similar characteristics in size 9". This is done with the `\new@fontshape` command, either in a style file (see '`concrete.sty`' as an example) or when dumping a format.

Again two files `fontdef.ori` and `fontdef.max` are distributed. You can copy one of them to `fontdef.tex`. The file `fontdef.ori` defines all fonts which are necessary to run standard `LATeX` documents while `fontdef.max` also defines certain fonts from the AMSFonts collection. To make other font families available you can either append appropriate `\new@fontshape` definitions to `fontdef.tex` (again, leave the originals untouched!) or add them in a style file.⁸ For a detailed description of how to set up new families with the `\new@fontshape` command, see [5] about the basic macros or one of the example files.

3.2 Running `IniTeX`

When setting up a new format one has to start `IniTeX` with `lplain.tex` as the input file. After displaying some progress report on the terminal, `lplain.tex` will try to `\input` the files `hyphen.tex` and `lfonts.tex`.

As we said above, it seems a good idea to rename these files because, when `TeX` complains that it cannot find them and asks you to type in another file name, you get the chance to substitute your favourite hyphenation patterns without changing `lplain.tex` or copying something to `hyphen.tex`. The transcript file will show the name of the file used and this may be very useful to debug weird errors (later).

When the point is reached where `TeX` wants to read in `lfonts.tex`, you now have to specify '`lfonts.new`'. This file will `\input` some other files. After processing them (which will take some time), `IniTeX` stops once more since it cannot find the file `xxxlfont.sty`. This is intentional; in this way you may now specify the desired default by entering one of the following file names:

`oldlfont.sty` If you choose this file, your format

⁸ The latter alternative might be better if you use these fonts very rarely (e.g., at sites with many users) to avoid filling `TeX`'s memory with unnecessary definitions.

will be identical to the standard L^AT_EX version 2.09 except that a few additional commands (like `\normalshape`) are available. Of course, documents or style options which explicitly refer to things like `\tentt` will produce error messages since such internal commands are no longer defined.⁹ Nevertheless it is easy to fix the problem in such a case: if we know that `\tentt` referred to `cmtt10`, i.e. Computer modern typewriter normal at 10pt, we can define it as

```
\newcommand{\tentt}{\family{cmtt}
  \series{m}\shape{n}\size{10}{12pt}
  \selectfont}
```

Since we assume the ‘`oldlfont`’ option as default, where `\tt` resets `series` and `shape`, the definition could be shortened to

```
\newcommand{\tentt}{\size{10}{12pt}\tt}
```

To get the new way of font selection as described in the previous sections (e.g. where `\tt` simply means to switch to another family) you only have to add the ‘`newlfont`’ style option to the `\documentstyle` command in your document.

`newlfont.sty` This is just the counterpart to `oldlfont.sty`: it will make the new mechanism the default and you have to add ‘`oldlfont`’ as a style option if you want to process older documents which depend on the old mechanism.

`basefnt.tex` This file is similar to `newlfont.sty` but does not define the L^AT_EX symbol fonts. These fonts contain only a few characters, and these are also included in the AMS symbol fonts. Therefore we provided the possibility of generating a format which doesn’t unnecessarily occupy one of the (only) sixteen math groups within one math version. Using this file you can easily switch to the old scheme (adding ‘`oldlfont`’ as an option), to the new scheme with L^AT_EX symbol fonts (using ‘`newlfont`’) or to the new scheme with additional AMS fonts

⁹ By the way, such documents were at no time portable since Leslie Lamport stated that it was always permissible to customize `lfonts.tex` according to the local needs. Therefore this is *not* an incompatible change.

by using either the style option ‘`amsfonts`’ (fonts only) or the style option ‘`amstex`’ (defining the whole set of A_MS-T_EX macros in a L^AT_EX like syntax).

We suggest using the `basefnt.tex` file since the new font selection scheme will be incorporated into L^AT_EX version 2.10, but at installations with many users it might be better to switch smoothly to the new font selection scheme by first using ‘`oldlfont`’ as a default.

Anyway, after reading the file chosen, T_EX will continue by processing `latex.tex` and finally displaying the message “Input any local modifications here”. If you don’t dare to do so, use `\dump` to finish the run. This will leave you with a new `.fmt` file (to be put into T_EX’s format area) and the corresponding transcript file. It isn’t a very good idea to delete this one because you might need it later to find out what you did when you dumped the format!

4 Remarks on the development of this interface

We started designing the new font selection scheme around April 1989. A first implementation was available after one month’s work and thereafter the prototype version ran successfully for some months at a few sites in Germany and the UK. Frank’s visit to Stanford as well as our work on the ‘`amstex`’ style option brought new aspects to our view. The result was a more or less complete redefinition of the L^AT_EX interface for this font mechanism. It was a long way from the first sketch (which was about five pages in Frank’s notebook) to the current implementation of the interface which now consists of nearly 2000 lines of code and about 4000 lines of internal documentation. The A_MS-T_EX project itself, which triggered this reimplementation, has about the same dimensions. Surely in such a huge software package one will find typos and bugs. But we hope that most of the bugs in the code are found by now. It is planned that the new font selection scheme will replace the old one in L^AT_EX version 2.10. We therefore hope that this pre-release which runs in version 2.09 will help to find all remaining problems so that the switch to version 2.10 will be without discomfort to the user.

5 Acknowledgements

During this project we got help from many people. A big ‘thank you’ to all of them, especially

to Michael Downes from the AMS for his cooperation and help, to Stefan Lindner for his help with the Atari \TeX and to Sebastian Rahtz for playing a willing guinea-pig. Finally we also want to thank Ron Whitney who did a marvelous job on all our articles so far. This time we posed some extra problems because he had to first make a new format in order to read how to make a new format.

6 List of distributed files

lfonts.new The new version of *lfonts.tex*, to be copied to a file of this name after the old *lfonts.tex* has been renamed.

fontdef.ori The font definitions for the computer modern fonts in the distribution by Donald E. Knuth. To be copied to *fontdef.tex* if this selection is to be used.

fontdef.max Complete font definitions for the computer modern fonts and the AMSFonts collections. To be copied to *fontdef.tex* if this selection is to be used.

preload.ori Preloads the same fonts as the old *lfonts.tex* does. To be copied to *preload.tex* if this is desired.

preload.min Preloads only the absolute minimum of fonts. To be copied to *preload.tex* if this is desired.

fam.tex The basic macros for the new font selection scheme. Automatically read by *lfonts.new*.

latintex \TeX interface for the new font selection scheme. Automatically read by *lfonts.new*.

setsizes.tex Size definitions for the \TeX interface. Automatically read by *lfonts.new*.

newlfont.sty Selects new version of font selection for \TeX .

oldlfont.sty Selects old version of font selection for \TeX .

basefnt.tex Like *newlfont.sty*, but does not define the \TeX symbol fonts.

margid.sty Style file that defines all (*math alphabet identifiers*) to have one argument. This is the default that is built into the new font selection scheme. Therefore this style file is only necessary if 'nomargid.sty' was loaded at dump time.

nomargid.sty In contrast to *margid.sty*, defines all (*math alphabet identifiers*) to switch to the

alphabet. This style option is necessary if you want to be compatible to the old \TeX syntax *in math mode only*.

tracefnt.sty Style file that allows the tracing of font usage. Use \tracingfonts with values 1 to 3 and watch what happens.

syntonly.sty Defines the \syntaxonly declaration. This can be used in the preamble of a document to suppress all output.

amsfonts.sty Defines the commands to select symbols from the AMSFonts collection.

amsbsy.sty Defines the \boldsymbol command.

amssymb.sty Defines additional $\text{AMS-}\text{\TeX}$ symbols.

amstex.sty Defines special $\text{AMS-}\text{\TeX}$ structures (like alignments in math mode) with \TeX syntax.

amstext.sty Defines the $\text{AMS-}\text{\TeX}$ \text command.

euscript.sty Contains the definitions to use the Euler script fonts.

References

- [1] Hugh Williamson, *Methods of Book Design*, Yale University Press, New Haven, London, Third Edition, 1985.
 - [2] International Business Machines Corporation, *Font Object Content Architecture Reference*, First Edition, December 1988.
 - [3] Donald E. Knuth, "Typesetting Concrete Mathematics," *TUGboat* 10, no. 1, 1989, pp. 31–36.
 - [4] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik, *Concrete Mathematics*. Addison-Wesley, 1989.
 - [5] Frank Mittelbach and Rainer Schöpf, "A New Font Selection Scheme for \TeX Macro Packages," *TUGboat* 10, no. 2, 1989, pp. 222–238.
- | | |
|---|--|
| ◊ Frank Mittelbach
Electronic Data Systems (Deutschland) GmbH
Eisenstraße 56
D-6090 Rüsselsheim
Federal Republic of Germany
Bitnet: pzf5hz@drueds2 | ◊ Rainer Schöpf
Institut für Theoretische Physik der Universität Heidelberg
Philosophenweg 16
D-6900 Heidelberg
Federal Republic of Germany
Bitnet: BK4@DHDURZ1 |
|---|--|

A style option to adapt the standard L^AT_EX document styles to A4 paper*

Nico Poppelier and Johannes Braams

Abstract

This article describes a new style option that can be used with the document styles that are distributed with the L^AT_EX distributions. It modifies the page layout to conform to the paper format most commonly used in Europe, portrait A4.

1 Introduction

This file is based on the document style options **A4.sty** and **A4wide.sty**, which can be found in the Rochester style archive. The original style option **A4.sty** we started from was written by John Pavel, and is dated May 1987. This option only changes the vertical size of the text somewhat, by increasing the number of lines on a page. The style option **A4wide.sty** was written by Jean-Francois Lamy, and is dated July 1986. This option only increases the width of the text.

2 Goals and design decisions

As many people before us, we found the page layout as implemented in the standard L^AT_EX document styles too much geared towards the American-sized paper, which is somewhat wider than A4 paper, but also noticeably less high.

Our goal was to get a page layout that was suitable for A4 paper, and produced legible texts. There are a number of layout parameters that influence the legibility of a text. A parameter of major importance is the number of words (or characters) on a line. The maximum number of words per line is ten to twelve for optimal legibility, a rule-of-thumb that can be found in typographic literature (we used [1]). This results in a number of characters per line which lies somewhere between sixty and seventy.

Another important parameter is the amount of white space surrounding the text. Here we have to distinguish between texts that are printed one-sided and texts that are printed two-sided (back to back). In the first case the margins on odd and even pages should be equal; in the latter case care should be taken that the texts on both sides of the paper overlap. Also a printed document is likely to be bound some way or another, so there should be enough white space in the ‘inner’ margin¹ of the text to allow this.

There is yet one more thing to take into account when designing a page layout. L^AT_EX offers the possibility of using marginal notes and if someone wants to use marginal notes, they should of course fit on the paper.

So, we have the following goals:

1. Choose the text width such that there will be sixty to seventy characters on a line;
2. See to it that in documents that are printed two-sided, the texts which end up on two sides of one sheet of paper overlap;
3. Leave enough white space in the ‘inner’ margin to allow for the binding of the document;
4. Leave enough white space in the ‘outer’ margin for marginal notes if they are going to be used.

*This file has version number 1.2- last revision 26 Feb 90, documentation dated 26 Feb 90.

¹ For two-sided printing, this is the left margin on odd-numbered pages and the right margin on even-numbered ones; for one-sided printing, this is always the left margin.

3 The implementation

3.1 The starting point

Thus we set out to modify some of the design decisions in the standard document styles. Because we knew that we were not the first to tackle the problem, we started by having a look at what was already available. We came up with the two options mentioned earlier, which are publicly available. Undoubtedly there will exist many more such files, some of them maybe modifications of those two files.

We had a look at the layout produced by both options and were not satisfied with it. For one thing, both of the original options `A4` and `A4wide` modify only one aspect of the page layout. The first thing to do was to put these two files together. This resulted in a layout which was still unsatisfactory, since for the 10-point and 11-point options lines in the text contained on the average eighty characters or more.

3.2 What else?

- `\textwidth` Because the result so far gave us lines that contained too many characters, we decreased the `\textwidth` to get lines that contain about sixty to seventy characters for all three size options. Still more work had to be done. As it turned out, using our new `A4.sty` together with the option `twosided` had a drawback: when the document was printed two-sided the texts on both side of one piece of paper overlapped only partly, which does not look good. We solved this by modifying the width of the margins for two-sided printing. At the same time we modified the `\marginparwidth` so that if someone uses a marginal note it would completely fit on the paper instead of falling off the page, which obviously would render the note unreadable.
- `\WideMargins` The decisions described above allow for marginal notes to be printed along with the normal text, but if someone makes heavy use of marginal notes, the resultant layout will not be very satisfactory, because if the full width of the marginal notes is used, they will take up too much space in the ‘outer’ margin. For this case we provide the macro `\WideMargins`. This macro modifies the page-layout parameters in such a way that the width reserved for marginal notes becomes 1.5 inches. To achieve this the width of the main body of the text is decreased. This macro is meant to be used only in the preamble of the document.

3.3 The code

We begin by identifying the version of this file on the terminal and in the transcript file.

```
\typeout{Style option 'A4' \fileversion\space<\filedate> (NP and JLB)}
\typeout{English documentation\space\space\space<\docdate> (JLB)}
```

- `\topmargin` First, we redefine the `\textheight` and `\topmargin`. The `\topmargin` is the distance from the reference point on the page to the top of the page of text. In most cases extra white space is not necessary since one inch of white space at the top of the page suffices.

```
\topmargin Opt
```

- `\textheight` The dimension parameter `\textheight` gives the total height of the text, including footnotes and figures, excluding the running head and foot. This height is given as an integral number times the `\baselineskip`, which results in an integral number of lines on a page.

We have to include definitions of all relevant dimension parameters for each of the cases 10-point, 11-point and 12-point. We do this with a case statement:

```
\ifcase \@ptsize
  \textheight 53\baselineskip
```

which modifies the height of the text for texts to be produced with the ten-point typeface:

```
\or
\textheight 46\baselineskip
```

the same for eleven point:

```
\or
\textheight 42\baselineskip
```

and for twelve point. Finally we close the \ifcase statement:

```
\fi
```

The only thing left to be done is to add the \topskip to the \textheight. The value of \topskip appears always to be 10pt.

```
\advance\textheight by \topskip
```

\textwidth
\oddsidemargin
\evensidemargin

That was the ‘vertical part’ of the work. Now we have some work to do to get things right horizontally. Again we have to distinguish between the various character sizes because sixty eleven-point characters take up more space than sixty ten-point characters. But there’s more to take into account. If documents are printed two-sided, the texts on both sides of the paper should overlap completely. This can be done by assigning appropriate values to \oddsidemargin and \evensidemargin, the parameters that define the left margins on odd and even pages respectively.

First we start a case statement to distinguish between the various typeface sizes.

```
\ifcase \@ptsize
```

Then we specify the width of the text.

```
\textwidth 5.00in
```

Also specify the width of marginal notes. They must have a reasonable width to be of any use, and this should be the same for either one-sided or two-sided printing.

```
\marginparwidth 1.00in
```

Here we need an if statement to test whether the option `twosided` has been specified.

```
\if@twoside
```

If it was, assign appropriate values to the margin parameters

```
\oddsidemargin 0.55in
\evensidemargin 0.75in
\else
```

If the option `twosided` was not used, both margin parameters must have the same value, for texts on consecutive pages have to be put in the same place on the paper.

```
\oddsidemargin 0.55in
\evensidemargin 0.55in
```

Now we close the if statement.

```
\fi
```

We are ready with the modifications for the ten-point typeface size, so now we do something similar for the eleven-point typeface.

```
\or
\textwidth 5.20in
\marginparwidth 1.00in
\if@twoside
\oddsidemargin 0.45in
\evensidemargin 0.65in
```

```

\else
  \oddsidemargin 0.45in
  \evensidemargin 0.45in
\fi

```

One more time, now for the twelve-point typeface.

```

\or
  \textwidth      5.70in
  \marginparwidth 0.80in
\if@twoside
  \oddsidemargin 0.20in
  \evensidemargin 0.40in
\else
  \oddsidemargin 0.20in
  \evensidemargin 0.20in
\fi

```

Finally we close the case statement.

```
\fi
```

\WideMargins This macro is somewhat tricky: it has to find out which typeface size is used, whether the document should be printed two-sided, and whether the `\reversemarginpar` is in effect. `\reversemarginpar` makes the marginal notes appear in the margin on the opposite side of the normal placement.

```
\def\WideMargins{%
```

Because for each typeface size the changes to the parameters that need to be made are similar, the macro `\WideMargins` uses an internal macro `\@widemargins`.

\ExtraWidth In order to store the amount of extra width needed for the marginal notes an extra dimension parameter is defined.

```
\newdimen\ExtraWidth
```

First find out about the point size, then call `\@widemargins` to modify the margin widths by the amount given in `\ExtraWidth`.

```
\ifcase \ptsize
```

For both 10-point and 11-point texts the width for marginal notes is already 1 inch, so we increase it by half an inch. We subtract half an inch from the text width and modify the margins appropriately.

```

\ExtraWidth = 0.5in
\@widemargins
\or
\ExtraWidth = 0.5in
\@widemargins
\or

```

For 12-point texts the marginal notes are only 0.8 inch wide, so now we have to add 0.7 inch to get them 1.5 inch wide.

```

\ExtraWidth = 0.7in
\@widemargins

```

This macro should only be called once, during the preamble of a document, so we `\let` it be equal to `\relax` as soon as the work is done.

```
\fi\let\WideMargins\relax\let\@widemargins\relax}
```

\@preamblecmds We add \WideMargins to \@preamblecmds, which is a list of commands to be used only in the preamble of a document.

```
{\def\do{\noexpand\do\noexpand}
 \xdef\@preamblecmds{\@preamblecmds \do\WideMargins}
}
```

\@widemargins This macro modifies the margin parameters. To do this it uses the dimension variable \ExtraWidth, which was defined by \WideMargins.

First the \ExtraWidth is subtracted from the \textwidth and added to the \marginparwidth.

```
\def\@widemargins{%
 \global\advance\textwidth by -\ExtraWidth
 \global\advance\marginparwidth by \ExtraWidth}
```

Then we modify the margins, but the value of the switch \if@twoside has to be taken into account. Because we have to test another switch (\if@reversemargin) we add another level of macros to modify the margin parameters

```
\if@twoside
  \tw@sidedwidemargins
\else
  \onesidedwidemargins
\fi}
```

\tw@sidedwidemargins Normally the marginal notes are printed in the ‘outer’ margins, so we have to increase the \evensidemargin to keep the text balanced on both sides of the paper, but if \reversemarginpar is in effect we have to increase the \oddsidemargin and decrease the \evensidemargin accordingly.

```
\def\tw@sidedwidemargins{%
 \if@reversemargin
```

Notice that for documents printed two-sided, the \evensidemargin is wider than the \oddsidemargin; this difference in width is transferred to the other margin.

```
\@tempdima=\evensidemargin
\advance\@tempdima by -\oddsidemargin
\advance\oddsidemargin by \ExtraWidth
\advance\oddsidemargin by \@tempdima
\advance\evensidemargin by -\@tempdima
\else
```

If the marginal notes go on the normal side of the paper, only the \evensidemargin has to be increased.

```
\advance\evensidemargin by \ExtraWidth
\fi}
```

\onesidedwidemargins For documents that are printed one-sided, both margins have the same width. The default placement for the marginal notes is in the right margin, so if \reversemarginpar is *not* in effect the margin parameters need not be modified. If it is in effect, both the \oddsidemargin and the \evensidemargin need to be increased.

```
\def\onesidedwidemargins{%
 \if@reversemargin
  \advance\oddsidemargin by \ExtraWidth
  \advance\evensidemargin by \ExtraWidth
\fi}
```

4 Conclusion

We have presented a new approach to adapt the page layout of the document styles that are part of the standard L^AT_EX distributions to the dimensions of A4 paper. The width of marginal notes has been taken into account and a means to get wider marginal notes at the cost of shorter lines in the main body of the text has been provided.

References

- [1] 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.

<ul style="list-style-type: none"> ◊ Nico Poppelier TeXnique Washingtondreef 153 3564 KD Utrecht Poppelier@hutruu53.bitnet 	<ul style="list-style-type: none"> ◊ Johannes Braams PTT Research Neher Laboratories P.O. Box 421 2260 AK Leidschendam JL_Braams@pttrnl.nl
---	---

Some Macros to Draw Crosswords*

B Hamilton Kelly[†]

Abstract

The crossword environment is intended to be used to typeset crossword puzzles for use in newsletters, etc.

Contents

1 Introduction	103	3 Creating the Grid	108
1.1 How to Specify Clue Numbers	104	3.1 Macros used when Populating the Grid	110
2 Definition of the Macros	106	3.2 The \clue Command	112
2.1 Counters and Lengths	107	3.2.1 Finding the clue number to be set in the light	114
2.2 Reading and Writing the Clues	107	3.3 Populating the Crossword Grid	114
2.3 Tabulating the Clues	108	3.4 Setting the Grid	116

List of Figures

1 A Sample Crossword	105	2 An example of the crossword* environment	106
--------------------------------	-----	--	-----

*This file is v2.7, dated 16 Oct 89

[†]Especial thanks to my colleague Niel Kempson for many helpful suggestions, and to Frank Mittelbach of the Johannes Gutenberg University of Mainz, who saved me two pages of code!

1 Introduction

As a small diversion from the statistics of computer availability, lists of new software, and the like, Computer Centre Newsletters often include a crossword for the amusement of their readers.¹

The macros presented in this document provide a L^AT_EX method of typesetting these, and also assist the composer to ensure that the “grid” all goes together correctly. The grid generated is the more usual form, with black squares separating the “lights” which receive the answers to the clues. Work is in hand to be able to handle the *Mephisto/Azed* type of grid, in which only thicker grid lines separate the lights.

A sample crossword appears as Figure 1; I’ve left the grid blank for those who want some intellectual exercise: those who don’t can cheat by reading the source listing at the end of this article!

The whole crossword, including the \clue commands (*q.v.*), is bracketed within the crossword environment. This requires that the user specifies two parameters:

(gridsize) This is a number which specifies the columns (and rows) in the square grid.

(visible) This controls whether the answers are to be “filled in”; obviously of no use for publication, but useful whilst composing the crossword. If the parameter provided is the letter ‘Y’, then the answers will be typeset; if ‘N’ then the lights will be left blank. Any other value² provided for this parameter will cause L^AT_EX to input a yes/no answer by interaction with the user.

An analogous environment is provided especially for typesetting a smaller version of a grid showing, for example, “Last Month’s Solution”. In this of course, the answers *always* appear, and the clues are not printed. Again it takes two parameters:

(gridsize) As before, this specifies the number of squares in each axis.

(header_text) Some text which will be set (in **bold**) above the completed grid.

Figure 2 shows an example of the crossword* environment.

Within the body of these environments appear a succession of \clue commands; each of these takes a total of seven (!) parameters:

(clue_number) The number of the light on the grid, for example {17}. See 1.1 below for details of how more complex specifications may be given for multiple lights.

(Across/Down) This parameter *must* be either the letter ‘A’ or ‘D’, in upper-case.

(col_number) The *x*-coordinate of the first square of the light. The left-most column of the grid is numbered 1.

(row_number) The *y*-coordinate of the first square of the light. The top-most row is numbered 1.

(answer) The answer to the clue (or that part of it which appears in the light numbered *(clue_number)*). This must be a string of upper-case letters *only*; no spaces, punctuation, hyphens, etc.

(text) The text of the clue itself. If you want to use any L^AT_EX macros in this text, such as \dots, each such macro must be preceded by \noexpand. This includes such macros as \&, to produce an explicit ampersand.

(help) Anything to appear after the text, in parentheses; this will most usually be used for giving the length of the answer, such as “7” or “2,6,3-3”. Also used when the text of the clue is associated with another light when this parameter may say something like “see 14d”.

¹ That at RMCS has a bottle of wine as a prize!

² The lower-case letters ‘y’ and ‘n’ are also recognized

1.1 How to Specify Clue Numbers

Sometimes the solution to one clue is split amongst a number of “lights”. To cover this eventuality, provide a `\clue` for *each* of the lights involved, with the solution to that light *alone* given as `(answer)`. All except the `\clue` corresponding to the first light of the solution should have a null `(text)`, and the `(help)` parameter should be something like “see 7d”.

If this final parameter is totally empty, no corresponding clue number is printed:

this facility would be used when the current clue is the next consecutive light, when it is usual to omit any further reference to the clue number.

The `\clue` for the first light of the solution should provide the *entire* clue as its `(text)`, and the `(help)` should say something like “7,3-3”. The `(clue_number)` field should consist of the number of that light, followed immediately by the text required to describe the other lights, separated from it by some non-digit character, for example, a space.

For example, suppose the clue “Bill’s desired outcome?”, has the solution ‘ACT OF PARLIAMENT’ which is to go into lights 9d and 13a. Then³

```
\clue{13}{A}{5}{1}{PARLIAMENT}{}{see 9d}
\clue{9 {\noexpand\rm\&} 13a}{D}{1}{10}{ACTOF}%
{Bill's desired outcome?}{3,2,10}
```

will produce

13 (see 9d)

amongst the ACROSS clues, and

9 & 13a Bill’s desired outcome? (3,2,10)

amongst the DOWN clues

2 Definition of the Macros

As always, we start by identifying this version of the style file.

```
\typeout{Style option: 'crossword' \fileversion\space\space
<\filedate> (BHK)}
```

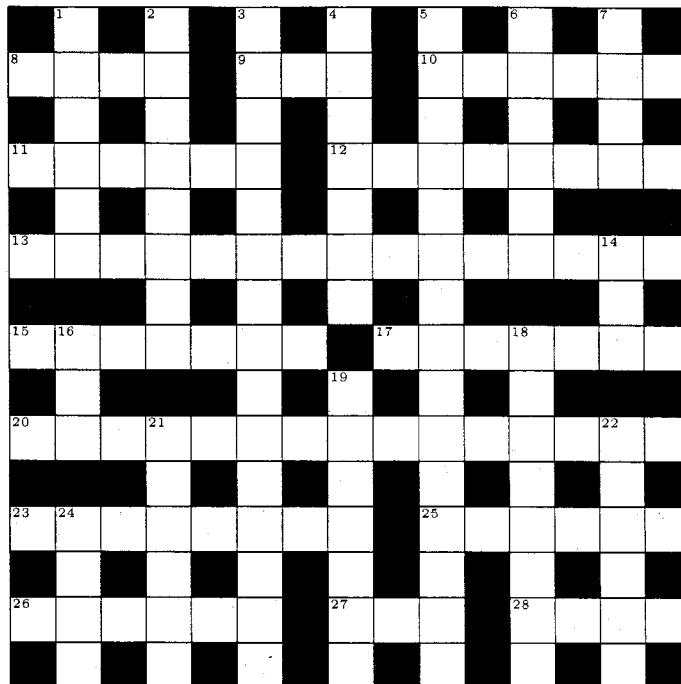
`\ninept` We define a new font size to ensure clues are set at 9pt, no matter what style size option is in effect. This command also defines suitable parameters for list environments set in this size of type.

```
\def\ninept{\@setsize\ninept{11pt}\ixpt\@ixpt
  \abovedisplayskip 8.5pt plus 3pt minus 4pt
  \belowdisplayskip \abovedisplayskip
  \abovedisplayshortskip \z@ plus2pt
  \belowdisplayshortskip 4pt plus2pt minus 2pt
  \def\@listi{\itemsep 0pt
    \parsep \z@ plus 1pt
    \topsep 4pt plus 2pt minus 2pt
  }}
```

2.1 Counters and Lengths

`\ifnumberit`
`\numberittrue` The crossword environment draws a grid (with black and white squares); each “light” into which a clue’s answer is to be written has to be numbered, and this number will be typeset (using `\tiny`) in the top-left corner of the first square of the light.
`\numberitfalse`

³ note the `\noexpand` before the `\rm` for the `\&`

**ACROSS**

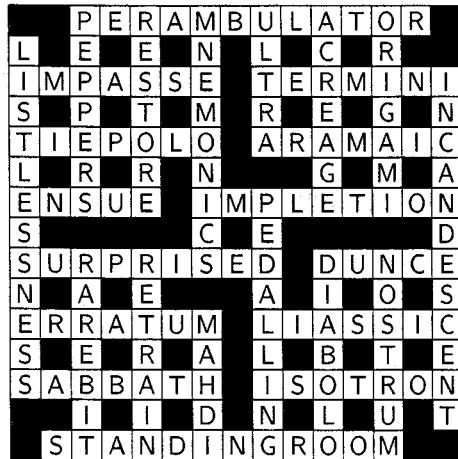
- 8** Points a thousand tested for witchcraft. (4)
- 9** Gourmet's triumphant cry on finding middle-cut Pacific salmon! (3)
- 10** One hundred stride backwards across a Pole. (3-3)
- 11** Fifties' jazz record about Eastern child's play. (2-4)
- 12** Timetable created by editor in synagogue of Spain. (8)
- 13** The dialect a girl mixed up tangle around symbolic diagram used by maritime student! (15)
- 15** Wire fastening bent road narrowly. (7)
- 17** Hammerhead consumes German company and casts a shade. (7)
- 20** Strange cel alien chops to make a figure with odd sides. (7,8)
- 23** Sounds like a hoarse editor came to in total! (8)
- 25** Assert without proof everyone, for example, English. (6)
- 26** Flourished examination of flowers. (6)
- 27** Floor covering discards fuming sulphuric acid and returns to nothing. (3)
- 28** "Latin for a candle" to be silent note about aircraftman? (4)

DOWN

- 1** Water rush noise disturbs show so! (6)
- 2** Mischievous child with cloth measure hesitates to assemble rotor. (8)
- 3** Mercifully inclined to pass round ten? Nay, about short blower! (15)
- 4** Sort ion? An isotron gives it a new twist! (7)
- 5** Wildly and boisterously rearrange Billy May Third, roughly. (15)
- 6** Satirical book or film—give odds about revision of "Dune"? (4-2)
- 7 & 24** Premier took in a Lord Lieutenant and all played an old game in London street. (4-4)
- 14** Work expended in power games? (3)
- 16** A church circle (or part of one). (3)
- 18** Encircle hindrances under hair in long curls. (8)
- 19** Boss over otorhinolaryngology department undergraduate. (7)
- 21** Old dovecote in Parisian museum. (6)
- 22** Like ornamental fabric, for example, that's in bequest. (6)
- 24** (See 7)

Figure 1: A Sample Crossword

Last month's solution

Figure 2: An example of the `crossword*` environment

This style option also provides the `crossword*` environment, which is intended to be used to produce “last month’s solution” in a smaller grid. There is insufficient room for clue numbers to appear on the grid in this mode, so `\ifnumberit` is used to indicate whether the numbers should be set.

```
\newif\ifnumberit
```

`\gr@dsiz` The counter `\gr@dsiz` is used to hold the width of the grid, as the number of squares in each direction.
`\p@csiz`

To prevent too much run-time arithmetic, the counter `\p@csiz` is set to be one count higher than `\gr@dsiz`.

```
\newcount{\gr@dsiz}
\newcount{\p@csiz}
```

`\Down` As we move around the grid, determining whether squares are black or white, we
`\Across` utilize the counters `\Across` and `\Down` to keep track of our location.

```
\newcount{\Down}
\newcount{\Across}
```

2.2 Reading and Writing the Clues

`\tf@acr` Whilst we are determining the appearance of the grid, we copy the text of each of the
`\tf@dwn` clues to an auxiliary file, so that the latter may later be read back to generate the
`\OpenClueFiles` clues themselves after the grid has been printed.

This macro opens a new file, with file extension `.acr`, and puts into it the commands necessary to typeset the Across clues. It also opens a `.dwn` file, which is similarly filled with the Down clues.

These files are created in the same manner as table-of-contents (`.toc`) files, etc; thus L^AT_EX will create file “handles” with names `\tf@acr` and `\tf@dwn`. However, that would ordinarily attempt to *read* the given file first, and also might defer the actual opening; therefore, we start a new group in which we redefine L^AT_EX’s `@input` command and T_EX’s `\openout` primitive.

```
\def\OpenClueFiles{{\let\OpenOut=\openout
\def\openout{\immediate\OpenOut}%
\let\@input=\@gobble
```

```
\@starttoc{acr}
\@starttoc{dwn}
```

Here's the preliminary material that gets inserted into the .acr file.

```
\@writefile{acr}{\string\begin{minipage}[t]{70mm}}
\@writefile{acr}{ \string\centerline{\string\bf\ ACROSS}}
\@writefile{acr}{ \string\sloppy}
\@writefile{acr}{ \string\ninept}
\@writefile{acr}{ \string\begin{ClueList}}
```

Whilst something similar goes into the .dwn file.

```
\@writefile{dwn}{\string\begin{minipage}[t]{70mm}}
\@writefile{dwn}{ \string\centerline{\string\bf\ DOWN}}
\@writefile{dwn}{ \string\sloppy}
\@writefile{dwn}{ \string\ninept}
\@writefile{dwn}{ \string\begin{ClueList}}}
```

- \CloseClueFiles After the grid has been printed, we can close the "clues" files; these will later be read back in (by the \endcrossword command) to set the text of the clues below the grid. Before closing, we insert the material that completes the two ClueList environments; firstly across...

```
\def\CloseClueFiles{%
  \@writefile{acr}{ \string\end{ClueList}}
  \@writefile{acr}{\string\end{minipage}}
```

Then for the down clues.

```
\@writefile{dwn}{ \string\end{ClueList}}
\@writefile{dwn}{\string\end{minipage}}
```

Now we can close those files, and make them "invisible" if someone tries to write to them.

```
\immediate\closeout\tf@acr \let\tf@acr=\relax
\immediate\closeout\tf@dwn \let\tf@dwn=\relax
\endgraf
}
```

2.3 Tabulating the Clues

The auxiliary files contain the texts of the clues, each given as an \item for the ClueList environment. This is similar to a description list, except that overlong labels run on into the text rather than sticking out to the left.

- \ClueList This sets up the ClueList environment, and defines the appearance of the label.
 \ClueListLabel

```
\def\ClueListlabel#1{\hspace\labelsep {\bf #1}\hspace\labelsep}
\def\ClueList{\list{}{\labelwidth\leftmargin
\advance\labelwidth by -\labelsep
\let\makelabel\ClueListlabel}}
\let\endClueList\endlist
```

- \PrintClues The following macro reads in the two files (of Across and Down clues), and sets them alongside each other, separated by a vertical rule. Clues are set in the style of the ClueList environment.

```
\def\PrintClues{%
  \centerline{%
    \begin{tabular}{c|c}
      \input{\jobname.acr}
      &
    \end{tabular}}
```

```

    \@input{\jobname.dwn}
    \end{tabular}
} \endgraf
}

```

3 Creating the Grid

The remaining commands are concerned with creating (and, optionally, populating) the crossword grid

- \crossword** The crossword environment takes two parameters: *viz.* the size of the matrix, and the indication of whether the grid is to include the answers. (If the latter is omitted, L^AT_EX will request it interactively.)

```
\def\crossword#1#2{%
```

We start off with a `\vtop` box and a group to hold everything within the environment, so as to ensure that user-entered text remains with the crossword.

```

\endgraf\leavevmode
\vtop\bgroup

```

The `crossword` environment uses the full-size grid, and has the lights numbered. Furthermore it doesn't have any heading to output (see the `crossword*` environment).

```

\unitlength 6mm\numberittrue
\def\Header{}%
```

We now open the auxiliary files into which the clues are written, and determine (interactively if necessary) whether the answers are to be written into the grid.

```

\OpenClueFiles
\TestAnswers{#2}%
```

Finally, we generate the necessary macros to describe the grid as being entirely filled with black squares; for each square, a macro whose name is of the form `\RiCj`, `\RxiCvii`, *etc.* is created. As the `\clue` commands are read in, these will be redefined to produce the correct appearance when the macros are later expanded.

```
\SetUpGrid{#1}
```

- \endcrossword** When all the clues have been processed, we can invoke `\FinishGrid` to draw the grid. The `\FinishGrid` and `\PrintClues` commands draw the grid and tabulate the clues, respectively. By enclosing them in a vertical mode list, we ensure that they remain stuck together on one page!

The `crossword` environment defines `\Header` to be empty, but the user may give it an explicit definition within the environment; if so, we'll print it just above the grid itself.

```

\def\endcrossword{\endgraf
\centerline{\Header}%
\hbox{\FinishGrid}%
}
```

We can now finish off the auxiliary files and then read them back in to set the text of the clues below the grid.

Finally, we complete the group and the `\vtop` box.

```

\CloseClueFiles
\hbox{\PrintClues}%
\egroup
}
```

- \crossword*** The `crossword*` environment doesn't need a second parameter to control printing of answers, because it **always** populates the grid with the answers. Instead, its second parameter provides the text to appear above the printed grid. Its actions are as for the `crossword` environment except that

- It prints the descriptive text above the grid.
- It **always** outputs the answers, without numbers.
- It draws them in a smaller box.
- It doesn't output the clues (it doesn't even open any auxiliary files!)

```
\expandafter\def\csname crossword*\endcsname#1#2{%
  \unitlength 4mm\numberitfalse
  \endgraf\leavevmode
  \vtop\bgrouptop
    \def\Header{{\bf\strut #2}}%
    \def\answer{Y}%
    \let\tf@dwn=\relax \let\tf@acr=\relax
    \SetUpGrid{#1}

\expandafter\def\csname endcrossword*\endcsname{\endgraf
  \centerline{\Header}}%
  \hbox{\FinishGrid}%
\egroup
}
```

3.1 Macros used when Populating the Grid

\laterletter The following macro calls this to “place” the letters of a solution by defining a macro unique to this square.

```
\def\laterletter#1{\setsquare{\lettersquare{#1}}}
```

\nextletter To determine how much space is required for the light corresponding to an answer, we need to cycle through each of the characters of the answer individually; this macro is called with two parameters — the first indicates the current setting direction (and thus accesses one of the counters `\Across` or `\Down`), whilst the second consists of the characters forming the answer followed by the string `\@nil`. When it is called, this “second” parameter is *not* enclosed in braces, so only the first token in it is accessed. The macro calls itself recursively to process the remaining characters until the `\@nil` has been met.

```
\def\nextletter#1#2{%
```

If the next token is `\@nil`, we've finished; the `\let` ensures that its parameter will be discarded (through the L^AT_EX internal command `\@gobble`) and the recursion will then unravel.

```
\ifx#2\@nil \let\nextlet=\@gobble
```

Otherwise, we have another letter of the *⟨answer⟩* in #2, so we call `\letterput` to define the macro corresponding to this square, and count one more position occupied in the current direction.

```
\else\letterput{#2}\advance#1 by \@ne
```

After we've processed this letter, we want to call this routine recursively to process the remaining letters (if any)...

```
\let\nextlet=\nextletter
```

These letters cannot possibly require a starting square number, so we use a simpler macro for these later letters.

```
\let\letterput=\laterletter
\fi
```

This is where we either exit from the recursion (and `\@gobble` the #1 parameter) or call the macro recursively to process the next character; the direction has to be passed on as the first parameter for `\nextletter` or `\@gobble`.

```
\nextlet{#1}}
```

\blacktest Later we shall want to check that the square into which we are “putting” a letter is either black (and hence should be changed to contain the character) or has already had the *same* letter put into it by an intersecting light.

We don’t want to expand each squares’ macros, so these tests have to use the **\ifx** primitive; therefore, we need a macro which has the same substitution text *at the highest level*.

```
\def\blacktest{\blacksquare}
```

\ifneeded As we create the macros corresponding to each occupied square, we have to decide whether it is necessary to actually perform a (re)definition of the macro; the following permits the code to determine whether such definition takes place (within the **\putsquare** macro).

```
\newif\ifneeded
```

\ifNoerr If an error is detected during the placement of the occupied squares, there will probably be other errors; to save pouring out yards of error messages, we arrange to suppress all but the first such; the following **\newif** provides this facility.

And of course we start without any errors!

```
\newif\ifNoerr
\Noerrtrue
```

\Number To determine whether a square has already been occupied, and if so, by what, we require a couple of new tokens which can be tested for as the result of a macro expansion

```
\newtoks\Number \newtoks\Plain
```

\blank The next two definitions can be temporarily **\let** to be the replacements for **\lettersquare** and **\numbersquare** respectively, for use in conjunction with the aforementioned test.

```
\def\blank#1{\Plain}
\def\numbered#1#2{\Number}
```

\letterinsquare When we first read the clues, we create macros which are unique to each square; later we shall redefine the macros to which the squares’ macros expand to actually perform the setting, but during the first phase we require expansions which correspond to the parameters alone. The following definitions are therefore used during the “filling” phase.

```
\def\letterinsquare#1{#1}
\def\letterinnumbersquare#1#2{#2}
```

\setsquare As we scan the answers for each clue, this macro is called with a parameter which specifies a call of either the **\lettersquare** or **\numbersquare** macros (with appropriate parameters).

It starts by assuming that no redefinition of the square’s macro will be required.

```
\def\setsquare#1{%
\need@dfalse}
```

This macro is also called during the initialization of the grid, when we populate it with black squares; therefore, we definitely want to perform a definition of the square’s macro at this stage.

```
\ifx#1\blacksquare
\need@dtrue
```

Otherwise, let's have a look and see what is already in the square's macro. This test will succeed only if the square contains its initial definition (as `\blacksquare`).

```
\else
\expandafter\ifx\csname\ther@w\thec@l\endcsname\blacktest
```

in which case we *will* want to redefine the macro for this square

```
\need@dtrue
```

If the square has already been redefined, it means we've already "put" a letter (or number+letter) into its definition, so we will have to check that this definition doesn't conflict with the new one (which is in #1). We have to expand the macros to perform this test, but don't want that expansion to return anything except the letter which is being (and has been) placed in the square, so we make a temporary replacement for the two macros which might form our #1.

```
\else
\let\lettersquare=\letterinsquare
\let\numbersquare=\letterinnumbersquare
```

Now we *expand* the original definition and the new one; these should be the same!

```
\expandafter\if\csname\ther@w\thec@l\endcsname#1
\else
```

If they aren't, it means the compiler of the crossword has made a mistake and has solutions which don't correctly intersect: tell him so (well, the first time anyway). We even provide the user with some help!

```
\ifNoerr
\errhelp{Two intersecting lights tried to
put different letters``J in the same square!
You've probably confused their coordinates.``J
Carry on, and examine the printout.}
\errmessage{Illegal redefinition of square \ther@w\thec@l.
Was: \expandafter\meaning\csname\ther@w
\thec@l\endcsname.

Now: \noexpand #1}
\Noerrfalse
\fi
\fi
```

So far, we don't seem to need to change anything, but if the new #1 which has been passed to `\putsquare` specifies that it be numbered, we must subsume any existing definition which *doesn't* have a number.

Again, we make temporary reassessments for the two potential macros, which is where we make use of the two new tokens that we invented.

```
\let\lettersquare=\blank
\let\numbersquare=\numbered
```

If the following test succeeds, we know that the new parameter specifies a number as well as a letter. This code could be expanded to check whether the original definition also specified a number, and if so ensure that they are the same, but would anybody really be silly enough to give different numbers for clues starting from the same square??

```
\expandafter\ifx#1\Number
\need@dtrue
\fi
\fi
\fi
```

Now we know whether we must (re)define the macro which is unique to this square. If we must, we expand our #1, so as to get the actual number and letter passed in through that parameter, but keep the name of the placement macro itself⁴ unexpanded.

```
\ifneed@d
  \expandafter\edef\csname\ther@w\thec@l\endcsname{\noexpand #1}
\fi
}
```

3.2 The \clue Command

- \clue Well, here it is at last. We start off by extracting the parts (if any) which form the *<clue_number>* parameter. We will therefore have the purely numeric first portion of the *<clue_number>* in \cluenumber.

```
\def\clue#1#2#3#4#5#6#7{%
  \findnumber{#1}}
```

We now examine the second (*Across/Down*) parameter of the \clue command to determine whether this is an Across or Down clue. The clue's *(text)* and *(help)* information is then written⁵ to the appropriate auxiliary file, and note taken of the direction in which the *(answer)* should be set into the light of the grid.

Firstly we deal with writes to the .acr file, if the *(Across/Down)* parameter is the letter 'A'. In this case, the counter to be incremented is \Across.

```
\ifx#2A
  \if\@empty#7\relax\else
    \ifx\tf@acr\relax\else
      \writefile{acr}{ \string\item[#1] #6 (#7)%}
    \fi
  \fi
  \let\Direction=\Across
\else
```

If this parameter is the letter 'D', writes go to the .dwn file, and the \Down counter is incremented.

```
\ifx#2D
  \if\@empty#7\relax\else
    \ifx\tf@dwn\relax\else
      \writefile{dwn}{ \string\item[#1] #6 (#7)%}
    \fi
  \fi
  \let\Direction=\Down
\else
```

If this *(Across/Down)* parameter is not one of the two permitted characters, an error message is issued.

```
\errhelp{The second parameter of the \string\clue\space
         command must be 'A' or 'D'}
\errmessage{Illegal direction (#1) specification
            for \string\clue.}
\fi
\fi
```

The *x* and *y* coordinate counters are set from the *(column)* and *(row)* parameters.

```
\Across=#3 \Down=#4
```

⁴ This will be the first token of the parameter itself.

⁵ The writes only take place if the output files exist, and the *(help)* parameter is non-empty.

\letterput The **\letterput** macro is defined anew at the start of each clue to create the correct definition for the *first* square of each light. The **\setsquare** macro redefines this macro internally for the remaining squares.

```
\edef\letterput{\noexpand\setsquare
  {\noexpand\numbersquare
    {\noexpand\cluenumber}{##1}}}%
```

Now we can call **\nextletter** which will cycle through all the letters of the *<answer>* until meeting the token **\@nil**. As each letter is processed, it creates a definition for the current square, initially using the above definition, and then **\laterletter** for subsequent letters of the *<answer>*.

Finally, we ensure that the newlines after **\clue** commands don't lead to unwanted spaces being typeset.

```
\nextletter{\Direction}#5\@nil
\ignorespaces
}
```

3.2.1 Finding the clue number to be set in the light

\findnumber We mentioned earlier that clues with solutions which occupy more than one light require a special format for specifying their *(clue_number)*. If this form is required, the number of the current light is given first, with the remaining text (as it is required to be set) following, separated from the first number by some non-digit character.

The macro **\findnumber** is called with the entire *(clue_number)* parameter passed to **\clue** and sets **\cluenumber** to expand to the first or only number found in that parameter (which should then appear in the first square of the light).

\clueNumber Of course, we require a counter in which to attempt to assemble that number:

```
\newcount\clueNumber
```

\special@gobble This macro is used by **\findnumber** to discard the unwanted portion (if any) of a *(clue_number)*, including the special termination token.

```
\def\special@gobble #1\@nil{}
```

The following mechanism to separate the first (or only) number from the remainder was suggested by Frank Mittelbach of the University of Mainz, and replaced about two pages worth of code.

```
\def\findnumber#1{%
```

We attempt to assign the *(clue_number)* parameter to the **\clueNumber** counter: only that portion consisting purely of digits will actually be assigned. The remainder, if any, including the special terminator sequence **\@nil** is then discarded by the **\special@gobble** command:

```
\afterassignment \special@gobble \clueNumber=0#1 \@nil
```

If the user did not provide a valid *(clue_number)* (i.e. something starting with a digit), then **clueNumber** will have zero assigned to it — seems the user ought to be told about this!

This completes **\findnumber**.

```
\ifnum\clueNumber=0
  \errhelp{The first parameter of the \string\clue\space command
            must commence with a digit}
  \errmessage{Illegal clue number (#1) specified
             for \string\clue.}
\fi
}
```

`\cluenumber` This macro merely produces the number as saved in `\clueNumber`.

```
\def\cluenumber{\the\clueNumber}
```

3.3 Populating the Crossword Grid

`\blackenrow` For each column in a row we create a black square; effectively `\setsquare` will execute the definition

```
% \def\csname ther@w\thec@l\endcsname{\blacksquare}
```

so that for row 3 column 6 of a 15×15 grid, for example, we would end up by defining `\def\RiiiCvi{\blacksquare}`.

Before starting the inner loop, we need to save the definition of `\body` which was created for the outer loop. We cannot do this by creating a new block, since that would require that each square be defined globally, which might give rise to save stack overflow problems.

```
\def\blackenrow{\let\savedbody=\body
\loop\relax\ifnum\Across>\z@
\setsquare{\blacksquare}%
}
```

We then shift ourselves back to the next column to the left and iterate. If we've reached the end of this inner loop, we re-establish the definition of `\body`.

```
\advance\Across by \m@ne
\repeat
\let\body=\savedbody
}
```

`\SetUpGrid` This macro creates an empty grid of the appropriate size.

```
\def\SetUpGrid#1{%
```

We firstly make a note of the `<gridsize>` parameter in the `\gr@dsiz`e counter, from which the width and height of the grid may be computed. We also set the `\p@csiz`e counter to be one greater than `\gr@dsiz`e to save our recomputing this quantity many times over.

```
\gr@dsiz=#
\p@csiz=#1 \advance\p@csiz by \cne
```

Right, this is where we start to generate the grid itself. We start at the bottom edge, because TeX loops are easiest if counting down to zero. Therefore, the `\Down` counter is set equal to the highest row number attainable.

```
\Down=\gr@dsiz
```

We now start a loop, so the following code will be repeated for each row of the grid in turn. As with the rows, we process the columns from highest address to lowest, so the `\Across` counter is also set to the highest column attainable.

```
\loop
\Across=\gr@dsiz
```

Provided we haven't decremented down to the 0th row, we start off the inner loop to process each column: this is done by invoking a separate macro — the alternative to which would be to enclose the inner loop in a group, which would require the use of global definitions for each square.

```
\ifnum\Down>\z@
\blackenrow
```

Afterwards we move ourselves up one row, and iterate for the next row.

```
\advance\Down by \m@ne
\repeat
}
```

And that's the end of \SetUpGrid!

\thec@l \thec@l and \ther@w macros generate a string of letters, starting with 'C' (for column) and 'R' (for row) respectively. The remainder of the string consists of the lower-case Roman numeral equivalent to the current value of the appropriate counter. Such all-letter strings are used to create the names for macros which can be unique for each square of the grid.

```
\def\thec@l{C\romannumeral\Across}
\def\ther@w{R\romannumeral\Down}
```

\TestAnswers This macro interacts with the user, if necessary, to get a yes or no indication of whether the answers shall be written into the grid. No check is made that the user has entered a valid response, but the use of \answer is such that any answer apart from a 'y' (in upper- or lower-case) is treated as if it were 'n'.

\f@rst To determine what parameter has been provided, or the response elicited, we will require a little macro to pass on the first token of a list terminated by a full stop.

```
\def\f@rst#1#2.{#1}
```

We commence by lower-casing the given parameter, setting the lower-cased version into the macro \answer.

```
\def\TestAnswers#1{\edef\next{\def\noexpand\answer{#1}}%
\lowercase\expandafter{\next}}%
```

We can then extract just the first character

```
\edef\answer{\expandafter\f@rst\answer .}%
```

Then we determine whether it's the letter 'y' or 'n'...

```
\if\answer y \else \if\answer n \else
```

If the *visible* parameter isn't either of these, we ask the user to give us an answer!

```
\typein[\answer]{Make answers visible? [Y/N]: }\fi
\fi
```

OK, \answer now contains some response; let's upper-case it and extract just its first character

```
\edef\next{\def\noexpand\answer{\answer}}%
\uppercase\expandafter{\next}%
\edef\answer{\expandafter\f@rst\answer .}%
}
```

3.4 Setting the Grid

\letter This is the expansion which is used (for \letterinsquare, and within the expansion of \numberedsquare) when the grid is actually being *typeset* from the stored squares' macros.

```
\def\letter#1{\put(\Across,-\Down){\makebox(1,1){\tensf #1}}}
```

\numberedsquare This macro is used (for \numbersquare) when the lights are being drawn with light numbering enabled. It puts the number (and the letter of the answer too, depending upon the definition of \letter which is current) at the current coordinate position specified by (\Across, \Down).

```
\def\numberedsquare#1#2{%
\put(\Across,-\Down){%
```

To insert the clue number, we generate it within a sub-picture, of size equal to one square.

```
\begin{picture}(1,1)(0,0)
```

We stick the number in the top-left corner of an (invisible) box which fills the central 81% of the area.

```
\put(0.05,0.05){\makebox(0.9,0.9)[t1]{\tiny #1}}
\end{picture}%
}
```

We also set the letter in the square (depending upon the definition of `\letter` which applies).

```
\letter{#2}}
```

`\unnumberedsquare` Despite its name, this macro is invoked (through `\numbersquare`) for those squares which would ordinarily carry a light's number, were it not for the fact that the numbers have been suppressed by the `crossword*` environment.

It just discards the `<clue_number>` given in the first parameter and sets the current letter by invoking `\letter`, which uses the second parameter...

```
\def\unnumberedsquare#1{\letter}
```

`\FinishGrid` Now we can process all the stored macros which define the appearance of each square in the grid, and thus generate the printed version thereof, using L^AT_EX's picture environment.

This command makes appropriate redefinitions of some macros which produced different effects during the filling of the grid.

```
\def\FinishGrid{%
```

If the customer doesn't want the letters put into the grid, then we need only throw away any parameter to `letter`.

```
\if\answer Y \else \let \letter=\@gobble \fi
```

As stated in the introduction, when typesetting the grid in a smaller version, there is insufficient space to include the numbers for the lights; by testing `\ifnumberit` we can determine whether a macro which expands to `\numbersquare` shall result in a number being printed or not.

```
\ifnumberit
  \let\numbersquare=\numberedsquare
\else
  \let\numbersquare=\unnumberedsquare
\fi
```

Anything that's been stored as a `\lettersquare` is "set" using the `\letter` macro (which we might have just `\let` equal to `\@gobble`).

```
\let\lettersquare=\letter
```

`\blacksquare` Any black squares that are still left in the grid are set by means of this command:

```
\def\blacksquare{%
```

The black square itself is merely a rule of the appropriate dimensions.

```
\put(\Across,-\Down){\rule{\unitlength}{\unitlength}}
```

Now we come to the actual body of `\FinishGrid`.

We start off at the bottom-most row of the grid...

```
\Down=\gr@dsiz
```

The whole grid is created in a centered `\hbox` in a `picture` environment. By offsetting the origin negatively, we can address each row by simply negating the *y* coordinate; thus column *x* in the highest row is (*x*, -1).

```
\centerline{%
\begin{picture}(\p@csiz,\p@csiz)(1,-\p@csiz)
```

We now cycle through each of the rows. The first thing we output is a horizontal rule of the full width of the grid, one such rule being generated for each row of the grid, providing the horizontal lines across vertical lights.

```
\loop\ifnum\Down>\z@  
  \put(1,-\Down){\line(1,0){\the\gr@dsiz}}
```

Now we are about to cycle across all the columns of the current row; again, it's convenient for us to work backwards to the left...

```
\Across=\gr@dsiz
```

To do this we need an inner loop; this has to be inside a group so as to isolate the effects of its `\repeat` command.

```
{\loop \ifnum\Across>\z@
```

To set the appropriate object in this square, we merely invoke the macro which has been associated with the square. Thus we'll end up with a black square, or a numbered or plain square, the latter two of which may also contain a letter.

```
\csname\ther@w\thec@l\endcsname
```

Now we advance to the next column and iterate. That's the end of the inner loop for each of the columns of the current row.

```
  \advance\Across by \m@ne  
  \repeat  
}%
```

Now we can decrement down to the next row and iterate through the rows.

```
  \advance\Down by \m@ne  
  \repeat
```

We've so far drawn a horizontal line *under* each of the rows; the next `\put` draws a final line *above* the top-most row.

```
\put(1,0){\line(1,0){\the\gr@dsiz}}
```

Similarly, a short loop can draw vertical rules at the left-hand edge of each of the columns, starting with a line on the left of an imaginary column to the right of the whole grid, which will therefore form a line to the right of the final column.

```
\Across=\p@csiz  
\loop\ifnum\Across>\z@  
  \put(\Across,0){\line(0,-1){\the\gr@dsiz}}  
  \advance\Across by \m@ne  
  \repeat
```

And that completes the picture.

```
\end{picture}  
}%  
}
```

Finally, here's the input which produced the crossword in figure 1

```
%      \begin{crossword}{15}{N}  
%          \input{grid_1}  
%      \end{crossword}
```

where the file `grid_1.tex` reads as follows:

```
\clue{8}{A}{1}{2}{SWAM}{Points a thousand tested for witchcraft.}{4}  
\clue{9}{A}{6}{2}{OHO}{Gourmet's triumphant cry on finding middle-cut  
Pacific salmon!}{3}  
\clue{10}{A}{10}{2}{ICECAP}{One hundred stride backwards across a
```

```

Pole.}{3-3}
\clue{11}{A}{1}{4}{BOPEEP}{Fifties' jazz record about Eastern child's
play.}{2-4}
\clue{12}{A}{8}{4}{SCHEDULE}{Timetable created by editor in synagogue
of Spain.}{8}
\clue{13}{A}{1}{6}{THALASSOGRAPHER}{The dialect a girl mixed up tangle
around symbolic diagram used by
maritime student!}{15}
\clue{15}{A}{1}{8}{HAIRPIN}{Wire fastening bent road narrowly.}{7}
\clue{17}{A}{9}{8}{UMBRAGE}{Hammerhead consumes German company
and casts a shade.}{7}
\clue{20}{A}{1}{10}{SCALENETRIANGLE}{Strange cel alien chops to make a
figure with odd sides.}{7,8}
\clue{23}{A}{1}{12}{AMOUNTED}{Sounds like a hoarse editor came to in
total!}{8}
\clue{25}{A}{10}{12}{ALLEGE}{Assert without proof everyone, for
example, English.}{6}
\clue{26}{A}{1}{14}{FLORAL}{Flourished examination of flowers.}{6}
\clue{27}{A}{8}{14}{NIL}{Floor covering discards fuming sulphuric acid
and returns to nothing.}{3}
\clue{28}{A}{12}{14}{TACE}{'Latin for a candle' to be silent note
about aircraftman?}{4}
\clue{1}{D}{2}{1}{SWOOSH}{Water rush noise disturbs show so!}{6}
\clue{2}{D}{4}{1}{IMPELLER}{Mischiefous child with cloth measure
hesitates to assemble rotor.}{8}
\clue{3}{D}{6}{1}{COMPASSIONATELY}{Mercifully inclined to pass round
ten? Nay, about short blower!}{15}
\clue{4}{D}{8}{1}{TORSION}{Sort ion? An isotron gives it a new
twist!}{7}
\clue{5}{D}{10}{1}{DITHYRAMBICALLY}{Wildly and boisterously rearrange
Billy May Third, roughly.}{15}
\clue{6}{D}{12}{1}{SENDUP}{Satirical book or film---give odds about
revision of 'Dune'?}{4-2}
\clue{7 {\noexpand\rm\&} 24}{D}{14}{1}{PALL}{Premier took in a Lord
Lieutenant and all played an
old game in London street.}{4-4}
\clue{14}{D}{14}{6}{ERG}{Work expended in power games?}{3}
\clue{16}{D}{2}{8}{ARC}{A church circle (or part of one).}{3}
\clue{18}{D}{12}{8}{RINGLETS}{Encircle hindrances under hair in long
curls.}{8}
\clue{19}{D}{8}{9}{STUDENT}{Boss over oto\rhino\laryng\o\logy
department undergraduate.}{7}
\clue{21}{D}{4}{10}{LOUVRE}{Old dovecote in Parisian museum.}{6}
\clue{22}{D}{14}{10}{LEGACY}{Like ornamental fabric, for example,
that's in bequest.}{6}
\clue{24}{D}{2}{12}{MALL}{}{See {\bf 7}}

```

◇ B Hamilton Kelly
 School of Electrical Engineering &
 Science
 Royal Military College of Science
 Shrivenham
SWINDON
 United Kingdom
 Janet: tex@uk.ac.cranfield.rmcs

Abstracts

**Deutsche Kurzfassungen der
TUGboat-Artikel
[German Abstracts of TUGboat Articles]**

Luzia Dietsche

**Koordination von nicht-englischen
Anwendungen von TeX (M. Ferguson,
S. 8)**

M. Ferguson stellt sich hier als Koordinator für nicht-englische Anwendungen von TeX vor. Zuerst bot er sich als Freiwilliger zum Sammeln von Trenn-Mustern an, erweiterte dies aber dann auf spezielle Makros, Stylefiles, Fonts u.a.m. Außerdem berichtet er in dem Artikel kurz, wie er sich eine Anpassung der nicht-englischen Besonderheiten an TeX 3 vorstellt.

**Der Gebrauch von TeX 3 in einer
mehrsprachigen Umgebung — Einige
Vorschläge (P. Breitenlohner, S. 9)**

Wie soll das neue TeX angewandt werden, so daß es zuerst einmal portabel, dann benutzerfreundlich und zu guter Letzt auch noch effizient bleibt? Dieser Frage geht P. Breitenlohner nach und entwirft ein grobes Schema, durch das seiner Meinung nach die drei genannten Kriterien in der vorgegebenen Reihenfolge verwirklicht werden könnten. Er sieht das Schema nicht als zwingend, sondern als Diskussions-Vorschlag.

**Druckfehler: Die neue Version von TeX
und METAFONT, TUGboat Vol. 10, No. 3
(D. E. Knuth, S. 12)**

Dieser knappe Artikel enthält eine Korrektur von den Autoren, die einen Absatz über das neue TeX und METAFONT auf Seite 326, Spalte 1 ersetzen soll.

**Virtuelle Fonts: Noch mehr Spaß für Genies
(D. E. Knuth, S. 13)**

Das größte Genie der TeX-Welt erklärt allen anderen, sprich den TeX-Implementatoren, wie man fremde Fonts problemlos in TeX einarbeiten kann. Dazu stellt er ein neues Programm, VFtoVP und VPtoVF, vor, das mit TFtoPL und PLtoTF vergleichbar ist. Ein kurzer Ausschnitt des Programms befindet sich am Anschluß an den Artikel.

**Eine vollständige public domain Version von
TeX für IBM PCs und kompatible: die
"zwei GUT Disketten" (N. Brouard, S. 46)**

GUTenberg, der Verein für französischsprachige TeX-Benutzer, hat auf zwei 1.2 MByte-Disketten eine *public domain* Version von TeX und diversen Tools für MSDOS zusammengestellt. Als Tools sind dabei ein Texteditor, Bildschirm- und Druckertreiber. Die TeX-Version (SB26TEX) enthält ein komplettes LATEX sowie BIBTEX 0.99c. Als Bildschirmtreiber ist CDVI 1.0, als Druckertreiber die Treiberfamilie DVIXXX dabei. Außerdem wurde einiges an Dokumentation in französisch und englisch hinzugefügt. Der Artikel selbst ist in französischer Sprache geschrieben.

**Unübliche Formen bei Absätzen
(V. Eijkhout, S. 51)**

Der Autor stellt drei interessante Möglichkeiten des Zeilen- und Absatzumbruchs vor. Er verwendet dazu die TeX-Kommandos \everypar und \lastbox. Die von ihm gezeigten Makrolösungen ermöglichen es unter anderem, Text einzugeben, auch ohne die für die Formatierung nötigen Kommandos dazwischen zu schreiben.

**TEXT1 wird public domain (D. Guenther,
S. 54)**

Die Washington State University hat Anfang Januar verschiedene Makropakete freigegeben, die bislang gekauft werden mußten. Dabei ist TEXT1, ein Paket, das wie LATEX auf plain TeX aufsetzt und ähnliche Möglichkeiten bietet. Laut D. Guenther ist TEXT1 leichter zu modifizieren als LATEX. Weitere freigegebene Produkte sind ein Compugraphics 8600 Fotosatz Dvi-Treiber, zusätzliche Fonts aus der Computer Modern Familie in den Point-Größen 11, 12, 14, 18, 24 und 36 und ein Font für das Internationale Phonetische Alphabet in den Point-Größen 9, 10, 11 und 12.

**Auflisten von mathematischen Makros
(M. J. Wichura, S. 57)**

Oft hat man in einem mit TeX formatierten Dokument Makros für immer wiederkehrende mathematische Ausdrücke definiert und weiß bei Bedarf nicht mehr, wie die Definition aussah. Dafür stellt der Autor eine Möglichkeit vor, solche eigenen Definitionen in einem extra File auflisten zu lassen. Allerdings ist \ListMacrosOnFile nur im mathematischen Modus anwendbar. Definitionen außerhalb werden nicht akzeptiert. Eine ausführliche Beschreibung, wie

\ListMacrosOnFile aufgebaut ist, findet sich im zweiten Teil des Artikels.

**Wie man es vermeidet, lange Sätze in *TEX*'s \write-Ausgabestrom zu schreiben
(P. Breitenlohner, S. 62)**

Verweisend auf einen Artikel von D. Salomon in *TUGboat* Vol. 10, No. 3 stellt P. Breitenlohner ein (*plain*) *TEX* Makro vor, mit dem man es vermeiden kann, daß die Sätze einer Überschrift zu lange werden. Er splittet den Output der Überschrift in mehrere Teile auf und zwar jeweils an der Stelle, wo in der Eingabe eine neue Zeile beginnt.

**Querverweise und der letzte "Dirty Trick"
(L. K. Durst, S. 62)**

Hier führt L. Durst sein Tutorial von *TUGboat*, Vol. 10, No. 3 fort. Er beschreibt, wie man ein "feature" von D. E. Knuth, das dieser zur Syntaxüberprüfung entwickelt hat, auch auf andere Weise nützlich anwenden kann. Er verwendet dieses "Syntax Checking", um verschiedene Informationen, z.B. für Querverweise, in einem Vorlauf in einen externen File schreiben zu lassen. Dieser File wird dann beim tatsächlichen *TEX*-Durchlauf gelesen und verarbeitet. Auf diese Art und Weise ist es möglich, zwei komplette Durchläufe für Verweise, Indexeinträge etc. auf 1 1/2 zu verkürzen.

Beispiele und Anwendungstechniken für Output Routinen. Teil I: Einführung und Beispiele (D. Salomon, S. 69)

Dieser Artikel ist der 1. Teil einer 3-teiligen Serie über Output Routinen von *TEX*. Teil 1 gibt eine anschauliche Einführung mit einfachen Beispielen. Teil 2 wird Markierungen, speziellen *penalty*-Werten, *kerns* und *extra* in den Text eingefügten Boxen gewidmet sein. Teil 3 wird sich schließlich mit Einfügungen beschäftigen. Im vorliegenden Teil definiert D. Salomon einführend Begriffe wie *Box*, *List*, *Main Loop*, *Main Vertical List* und *Page Builder*. Er erklärt, wie, wann, warum und von welchem Makro die Ausgabe einer Seite aufgerufen wird. An konkreten Beispielen (mit Listing) zeigt er, wie \output variiert und an verschiedene Bedürfnisse angepaßt werden kann.

Die *LATEX*-Ecke (J. Damrau, S. 85)

In dieser Ecke vom *TUGboat* will J. Damrau ab jetzt wieder Tips und Tricks für *LATEX*-Neulinge vorstellen. Dieser Beitrag ist dem Problem gewidmet, wie *LATEX* zu Beginn oder auch mitten im Text zu doppeltem Zeilenabstand überredet werden kann.

Richtlinien für das Erstellen eines Journal-Style: Ein Bericht über einen neuen *LATEX*-Style (K. v. d. Laan, S. 86)

Dieser Bericht ist eine Ausarbeitung darüber, wie man den allgemeinen *article style* an spezielle Journale anpassen kann. Das nächste Ziel der Autoren ist es, generische Richtlinien für das Erstellen von Journal-Styles zu entwickeln.

Internationales *LATEX* fertig zum Gebrauch (J. Schrod, S. 87)

Internationales *LATEX* (kurz *ILATEX*) ist eine freie und gewartete Version von *LATEX*, mit der es möglich ist, auch nicht-englische Dokumente mit dem Standard-Layout von *LATEX* zu setzen. Zusätzlich wurde ein bisher vorhandenes Problem mit der Auswahl der Größe von mathematischen Begrenzern gelöst. Diese Lösung kann auch für andere Makropakete verwendet werden, die jedoch auf *plain-TEX* basieren müssen.

Das neue Auswahlschema für Fontfamilien — Eine Benutzerschnittstelle zu Standard-*LATEX* (F. Mittelbach und R. Schöpf, S. 91)

Die beiden Autoren beschreiben in Anlehnung an einen von ihnen veröffentlichten Artikel in *TUGboat* Vol. 10, No. 3, wie das neue Auswahlschema für Fonts in der Standard-*LATEX*-Umgebung verwendet werden kann. Hauptmerkmale dieses Schemas sind die Möglichkeit, Familien, Reihen, Formen und Größen unabhängig voneinander wechseln zu können, ein Style-File, mit dem auch alte Dokumente ohne Veränderung ihres Aussehens und ihrer Eingabe bearbeitet werden können, und ein Makro-Zusatzpaket, das kompatibel zu bereits vorhandenen Standard-Dokument-Stylefiles ist.

Eine Stil-Option, um Standard-*LATEX*-Dokumentstyles an das A4-Maß anzupassen (N. Poppelier und J. Braams, S. 98)

Die meisten *LATEX*-Benutzer aus europäischen Ländern werden sich schon einmal über die ganz und gar amerikanische Seitenaufteilung von L. Lamport geärgert haben, die dem in Europa verbreiteten A4-Maß nicht im geringsten gerecht wird. Die beiden Autoren haben deshalb eine bereits vorhandene Stil-Option weiterentwickelt, so daß jetzt u.a. auch Deckungsgleichheit bei zweiseitigem Drucken, genügend linker Rand zum Binden und genügend rechten Rand für Randnotizen übrigbleibt.

TeX, TUG und Ost-Europa (H. Partl, S. 122)

H. Partl berichtet über eine Tagung in der Tschechoslowakei, bei der erstaunt feststellen konnte, wie weit TeX in Osteuropa verbreitet ist. Die Bereitschaft, Kontakte zu knüpfen, ist in diesen (zu jener Zeit noch unfreien) Staaten enorm hoch, die Entwicklung und Anpassung an die nationalen Bedürfnisse weit gediehen.

◊ Luzia Dietsche
 Rechenzentrum der Universität
 Heidelberg
 Im Neuenheimer Feld 293
 D-6900 Heidelberg 1
 Bitnet: X68@DHDURZ1

News & Announcements

TeX, TUG, and Eastern Europe

Hubert Partl

Having returned from a conference on Man-Machine Interfaces and Scientific Document Processing in Skalský Dvůr (Czechoslovakia) in fall 1989, I was amazed how this conference was dominated by TeX and L^AT_EX. Obviously, since public domain products are of vital importance in Eastern Europe, TeX and L^AT_EX have succeeded in becoming the standard in these countries. Hyphenation patterns and language specific adaptions of TeX and L^AT_EX exist for almost all of the Eastern European languages, and local (nation-wide) TeX users groups are either established or being considered for the very near future. Some of the countries will install national computer networks for electronic mail distribution, and — to come to the point — all of them are eager to get into contact with TUG.

Therefore, I have asked the TUG office to send information material about the TeX Users Group and — if possible — complimentary sets of last year's *TUGboat* issues to several key TeX users in Czechoslovakia, Eastern Germany and

Hungary. In the meantime, one site in Hungary — the Hungarian Academy of Sciences in Budapest — has already become a member of TUG.

I have also suggested that their hyphenation patterns and macro files be obtained and stored in the Aston and Clarkson archives, and that a way be found to bring TeXhax to them.

More about the Skalský Dvůr Conference on "Man Machine Interface in Scientific Environment"

The conference, organized by Dr. J. Nadrlchal of the Institute of Physics of the Czechoslovak Academy of Sciences, took place on September 18–29, 1989.

The conference had about 150 attendees (physicists, informatics scientists, and document processing people) from the following countries: Austria, Bulgaria, Brazil, Switzerland, Czechoslovakia, Eastern and Western Germany, France, United Kingdom, Greece, Hungary, Italy, India, the Netherlands, Libya, Poland, Romania, Sweden, Finland, Soviet Union, USA. The main themes were user interfaces for scientific programs, natural language processing, and document preparation.

Within the conference, the following presentations dealt with TeX or L^AT_EX:

- J. Brandt (Technical University Munich, W. Germany): Computer-aided Production of Scientific Documents
- R. Wonneberger (EDS Rüsselsheim, W. Germany): Structured Document Design — the L^AT_EX approach
- H. Partl (Technical University Vienna, Austria): How to Make TeX and L^AT_EX International
- J. Eickel (Technical University Munich, W. Germany): Logical and Layout Structure of Documents
- G. Krönert (Siemens Munich, W. Germany): Importance of the ISO Standard 8613 for Document Interchange
- D. Miklós, T. Fadgyas (Hungarian Academy of Sciences): Hungarian TeX
- R. Liska, L. Drska (Technical University Prague, Czechoslovakia): REDUCE L^AT_EX Formula Interface

In discussions and coffee breaks, I learned the following about progress on "Eastern European TeX":

- The Hungarians have developed Hungarian hyphenation patterns and extensions to the Computer Modern Fonts that include all the

Hungarian accented characters. Recently, they have also started a national e-mail system and are about to connect it with the international UUCP network.

- The Czechoslovaks have developed hyphenation patterns and font extensions too, and are just establishing a national *TEX* users group. They plan to develop a macro file—similar to *german.sty*—for support of Czechoslovak *TEX*, and a computer network connection of the universities and the Academy of Sciences.
- In Eastern Germany, there is a volunteer to act as contact person for an Eastern German *TEX* Users Group and to build up an electronic mail distribution service within that country. However, with the recent political changes, this idea may become obsolete, and the East German will perhaps just be a part of the already existing DANTE association that serves the *TEX* users of all german speaking countries. [Editor's note: See the articles by Joachim Lammarsch

on page 287 of *TUGboat* 10, no. 3 and by Malcom Clark on page 667 of *TUGboat* 10, no. 4 for more information on DANTE.]

I also heard about busy *TEX* activities in Poland, and a joint effort on Russian *TEX* between the Soviet Union and the USA which uses the Bitnet distribution list RUSTEX-L. [Editor's note: See the article by Dimitri Vulis on page 332 of *TUGboat* 10, no. 3 for information on RUSTEX-L.]

Now, after the fall of the previous political barriers, these activities are likely to increase even more, and I hope that TUG membership will soon cover both halves of Europe equally well.

◊ Hubert Partl
EDP Center
Technical University Vienna
Wiedner Hauptstraße 8-10
A-1040 Wien, Austria
Bitnet: Z3000PA@AWITUW01

Calendar

1990

- | | |
|-------------|--|
| Mar 5 - 9 | Intensive Beginning/Intermed. <i>TEX</i> ,
University of Michigan,
Ann Arbor, Michigan |
| Mar 5 - 9 | Advanced <i>TEX</i> /Macro Writing,
Vanderbilt University,
Nashville, Tennessee |
| Mar 12 - 16 | Intensive Beginning/Intermed. <i>TEX</i> ,
Texas A & M University,
College Station, Texas |
| Mar 19 - 23 | Intensive Beginning/Intermed. <i>TEX</i> ,
University of Illinois,
Chicago, Illinois |
| Mar 22 - 23 | DANTE e.V.: 2 nd meeting,
University of Düsseldorf.
For information, contact
Friedhelm Sowa (Bitnet:
<i>TEX@DDORUD81</i>) or DANTE e.V.
(Bitnet: <i>DANTE@DHDURZ1</i>) |

- | | |
|-------------|--|
| Mar 26 - 30 | Intensive Introduction to
<i>LATEX</i> , Harvard University,
Cambridge, Massachusetts |
| Mar 26 - 30 | Intensive Beginning/Intermed. <i>TEX</i> ,
Northeastern University,
Boston, Massachusetts |
| Apr 10 | <i>TUGboat</i> Volume
11, 2nd regular issue:
Deadline for receipt of manuscripts. |
| May 14 - 15 | GUTenberg'90, Toulouse, France
For information, contact
Pierre Legrand (Bitnet:
<i>LEGRAND@FRICICT81</i>) or
Bernard Gaulle (Bitnet:
<i>UCIRO01@FR0RS31</i>). (See also
page 125.) |

May 15 Papers for TUG Annual Meeting:
Deadline for preprint versions.

**TUG90 Conference
Texas A & M University
College Station, Texas**

Jun 11-15 Intensive Beginning/Intermed. **TEX**
Jun 11-15 Advanced **TEX/Macro Writing**
Jun 11-15 Intensive **L^AT_EX**
Jun 11-15 **L^AT_EX Style Files**
Jun 11-13 **PostScript**
Jun 12-15 **METAFONT**
Jun 13-15 **Output Routines**
Jun 14-16 **SGML**
Jun 18-20 **TUG's 11th Annual Meeting**
Jun 21-22 **Macro Writing**
Jun 21-22 **Document Design**

Jul 15 Papers from TUG Annual Meeting:
Deadline for receipt of camera copy
for **TUGboat** Proceedings issue.

Aug 31 NTG-SGML Holland meeting
Groningen, The Netherlands.
For information, contact
Kees van der Laan (Bitnet:
CGL@RC.RUG.NL)

**TEX90
University College
Cork, Ireland**

Sep 10-12 **TUG's 1st Conference in Europe**

Sep 11 **TUGboat Volume
11, 3rd regular issue:**
Deadline for receipt of manuscripts
(tentative).

Sep 18-20 **EP'90**
National Institute of Standards
and Technology, Gaithersburg,
Maryland. For information,
contact Richard Furuta
(**furuta@brillig.umd.edu**).

Oct 3-5 Seybold Computer Publishing
Conference, San Jose Convention
Center, San Jose, California.
For information, contact Seybold
Publications, West Coast Office
(213-457-5850).
Oct 10-12 9th annual meeting, "Deutsch-
sprachige **TEX-Interessenten**";
DANTE e.V.: 3rd meeting, GWD,
Göttingen. For information, contact
Dr. Peter Scherber (Bitnet:
PSCHERB@DGOGWDG1) or DANTE e.V.
(Bitnet: **DANTE@DHURZ1**).
Dec 6-8 European Publishing Conference,
Netherlands Congress Centre,
The Hague, Holland.
For information, contact Seybold
Publications, U.K. Office
({44} 323 410561).

1991

Jan 15 **TUGboat Volume 12,
1st regular issue:**
Deadline for receipt of manuscripts
(tentative).

Feb 20-22 10th annual meeting, "Deutsch-
sprachige **TEX-Interessenten**";
DANTE e.V.: 4th meeting,
Technical University of Vienna.
For information, contact
Dr. Hubert Partl (Bitnet:
Z3000PA@AWITUW01) or DANTE e.V.
(Bitnet: **DANTE@DHURZ1**)

Apr 9 **TUGboat Volume 12,
2nd regular issue:**
Deadline for receipt of manuscripts
(tentative).

Sep 10 **TUGboat Volume 12,
3rd regular issue:**
Deadline for receipt of manuscripts
(tentative).

For additional information on the events listed
above, contact the TUG office (401-751-7760) unless
otherwise noted.

GUTenberg'90

15 au 17 mai 1990
Université Paul Sabatier

Le Groupe francophone des Utilisateurs de **TEX** organise, comme chaque année, son congrès GUTenberg, regroupant comme l'an passé, des sessions de formation, des conférences et une petite exposition.

Le congrès de cette année a été baptisé « **TEX et les Arts graphiques** » car **TEX** est désormais un environnement logiciel complet et considéré comme indispensable par les professionnels des Arts graphiques. GUTenberg'90 sera l'occasion unique de faire le point ou de s'informer sur le sujet.

Le comité de programme souhaite organiser des conférences techniques de haut niveau en faisant appel aux utilisateurs de **TEX** et aux professionnels des Arts graphiques.

Sujets d'intérêt

Le comité de programme retiendra en premier les sujets traitant de **TEX** dans le domaine des Arts graphiques et s'intéressera ensuite aux sujets ne portant que sur l'un des deux aspects suivants :

Arts graphiques

- Techniques de maquettage
- Graphisme et effets spéciaux
- Utilisation de nouvelles polices
- Chaîne éditoriale informatisée
- Expériences d'utilisation de matériels
- Influences de la P.A.O.
- Normes de balisage.

Environnement **TEX**

- Développements autour de **TEX**, **LATEX**, etc.
- Utilisation/création de polices METAFONT ou non METAFONT
- Pilotes et langages de description de page
- Aspects X-windows
- Digitalisation et restitution
- Éditeurs, correcteurs et pre/post processeurs interactifs
- Spécificités francophones et européennes
- Utilisation des réseaux informatiques
- Accès aux logiciels du domaine public

GUTenberg'90

The French-speaking **TEX** Users Group named GUTenberg, is organizing its yearly congress comprising, as last year, tutorials and conferences, as well as a small exhibition.

This year, the congress will be entitled “**TEX and professional publishing**” because **TEX** has become a full set of software considered by printers, editors, composers ... as an indispensable production tool. GUTenberg'90 will provide a unique opportunity to familiarize yourself with the products and to collect information about the latest experiments and developments. The program committee will organize a series of high level presentations delivered by **TEX** users or professionals of publishing.

Main topics

After giving priority to topics focused on **TEX** and the publishing field, the program committee will select papers in one or the other of the following two areas:

Professional publishing

- Page modeling techniques
- Graphism and special effects
- New fonts usage
- Automated production from editor to printer
- Hardware experiences
- EP/CAP effects
- Markup standards

TEX domain

- Developments around **TEX**, **LATEX**, ...
- Creation/usage of fonts designed with or without METAFONT
- Fonts and page description languages
- X-windows aspects
- Rasterization and output
- Screen editors, spellers and pre/post interactive processors
- Aspects of French and European languages
- Network usage
- Public domain software access

Comité d'organisation / Organisation Committee

P. LEGRAND (C.I.C.T), président

Y. SOULET (U.P.S)

J.C. JOLY (Cepadues éditions)

H. LE TALLEC (Cepadues éditions)

M et MME COLLIN (TRÉMA)

M.H. GELLIS

P. LESGOURGUES (Cepadues éditions)

Contact

Pierre Legrand C.I.C.T. Univ. Paul Sabatier 118, route de Narbonne F31062 TOULOUSE Cedex	Bitnet: LEGRAND@FRCICT81 Tel: (33) 61.36.60.00 Fax: (33) 61.52.14.58
--	---

Comité de programme / Program Committee

BERNARD GAULLE (CNRS-CIRCE)	<i>Chairman</i>
MAURICE LAUGIER (imprim. Louis-Jean)	<i>co-Chairman</i>
JACQUES ANDRÉ (IRISA-INRIA)	
NELSON BEEBE (Univ. Utha, USA)	
ALAIN COUSQUER (Univ. LILLE I)	
NICOLAS BROUARD (INED)	
FRANÇOIS CHAHUNEAU (Berger-Levrault)	
MICHAEL FERGUSON (INRS Tele., Canada)	
LAURENT HEILMANN (Gauthier-Villard)	
CHRISTOPHE DE MONCUIT (CNRS-LIMSI)	

Contacts

Bernard Gaulle CNRS-CIRCE, BP 167 F91403 ORSAY Cedex	Bitnet: UCIR001@FRORS31 Tel: (33-1) 69.82.41.07 Fax: (33-1) 61.28.52.73
Maurice Laugier Imprimerie Louis-Jean Avenue d'Embrun, BP 87 F05002 GAP	Tel: (33) 92.51.35.23 Fax: (33) 92.53.72.27
Nelson Beebe Cent. for Sc. Comput. 220 South Physics Bldg Univ. of Utah Salt Lake City, UT 84112	Internet: Beebe@SCIENCE.UTAH.EDU Tel: (1) 801 581-5254 Fax: (1) 801 581-4801

SGML & TeX Conference Call for papers

Groningen, 31 August 1990

This conference, devoted to SGML, TeX, and their symbiosis, is organized by the NTG (Nederlandstalige TeX Gebruikersgroep, i.e., Dutch TeX Users Group) and the SGML-Holland Users Group. The conference will be held at the University of Groningen.

After some coffee, welcome, etc., the day will start with a survey by an invited speaker about the relationship between SGML and TeX, now and in the future. During the day there will be two parallel streams of presentations (1/2 and 3/4 hour talks) and for those wandering around there will be a vendor booth, a book stand, a selling point for TeX 'gadgets', copying facilities for PD (TeX) programs, and of course an information booth. In the closing session, another invited speaker will talk about Electronic Publishing in the future with a wink to SGML and TeX.

To celebrate the joint happening we will end with a nice cocktail party offered by Elsevier and Samson Publishers.

Authors are invited to present a paper related to the theme—SGML and TeX—and Electronic Publishing in general. We also welcome papers which do not exclusively address the preparation and printing phase of the lifecycle of a document, e.g., papers discussing the interaction between descriptive mark-up and database applications, reusable parts of documents, and don't forget papers dealing with descriptive mark-up and Hypertext.

The conference language is Dutch (this does not hold of course for non-native speakers) but speakers will be urged to have their transparencies in English.

TeX, L^AT_EX, METAFONT (TUG-like) courses as well as SGML courses will be held before the conference and in the week after.

Members of any SGML or TeX Users Group will be charged for f75,-, students for f25,-, and non-members f150,-.

Abstracts should be submitted, before 1 May 1990, to

C.G. van der Laan (CGL@RC.RUG.NL)
RekenCentrum RijksUniversiteit Groningen
Landleven 1, 9700 AV
Groningen, The Netherlands

Please supply: Title of the talk, full name, (e-mail) address, followed by the abstract (1 page at most), preferably in L^AT_EX article style. Notice of acceptance will be sent 1 June 1990.

The Organizing Committee consists of

NTG: C.G. van der Laan
T.A. Jurriens
SGML: J. Maasdam
J. Bleeker

For further information, contact C.G. van der Laan at the address shown above.

Late-Breaking News

Production Notes

Barbara Beeton

Input and input processing

Electronic input for articles in this issue was received by mail and on floppy disk.

Authors who had written articles previously for *TUGboat* typically submitted files that were fully tagged and ready for processing with the *TUGboat* macros—*tugboat.sty* for plain-based files and *ltugboat.sty* for those using *LATEX*. One or two articles were tagged according to the old conventions, but authors who did not have the new macros (see the Authors' Guide, *TUGboat* 10, no. 3, pages 378–385) were immediately sent copies. (The new macros have been installed at *labrea.stanford.edu* and the other archives, and should be retrieved by prospective authors before preparing articles; for authors who do not have network access, the TUG office can provide the macros on diskette.)

Nearly half the articles, and about the same proportion of the pages in this issue are *LATEX*. For convenience in processing, plain or *LATEX* articles were grouped whenever possible. Articles in which no, or limited, *TEX* coding was present were tagged according to the conventions of *tugboat.sty* or *ltugboat.sty* as convenient. Most articles tagged according to the author's own schemes were modified sufficiently to permit them to be merged with the rest of the stream. Especial care was taken to try to identify macro definitions that conflicted with ones already defined for *TUGboat*.

Two articles required extra-special handling. The article by Mittelbach and Schöpf (p. 91) was set using a preliminary version of the new *LATEX* font access technique which it describes. And the articles by Kuiken (p. 24) and Wichura (p. 57) used an experimental enhancement of the plain *TUGboat* macros that permits changing the number of columns in mid-page; this new feature will be described and the macros made available when they are stable.

Test runs of articles were made separately and in groups to determine the arrangement and page numbers (to satisfy any possible cross references). A file containing all starting page numbers, needed in any case for the table of contents, was compiled before the final run. Final processing was done in 7

runs of *TEX* and 5 of *LATEX*, using the page number file for reference.

The following articles were prepared using *LATEX*; the starred items required the *doc*-option.

- Nelson Beebe, Message from the President, page 5.
- Lee S. Pickrell, Combining graphics with *TEX* on IBM PC-compatible systems and LaserJet printers, page 26.
- Barbara Beeton, A proto-TUG bibliography, page 36.
- Nicolas Brouard, Une version complète de *TEX* du domaine public . . . , page 36.
- Victor Eijkhout, Unusual paragraph shapes, page 51.
- Jackie Damrau, The *LATEX* column, page 85.
- C. G. van der Laan, Announcing two reports: *SGML-LATEX* and *Journal style guidelines*, page 86.
- Joachim Schrod, International *LATEX* is ready, page 87.
- Frank Mittelbach and Rainer Schöpf, The new font family selection, page 91.
- * Nico Poppelier and Johannes Braams, A style option to adapt the standard *LATEX* document styles to A4 paper, page 98.
- * B. Hamilton Kelly, Some macros to draw crossword puzzles, page 103.
- Luzia Dietsche, Deutsche Kurzfassungen der *TUGboat*-Artikel, page 120.
- Bernard Gaulle, *GUTenberg'90*, page 125.
- C.G. van der Laan, SGML & *TEX* Conference, page 126.

Output

The bulk of this issue was prepared on an IBM PC-compatible 386 using PC *TEX* and output on an APS-μ5 at the American Mathematical Society using resident CM fonts and additional downloadable fonts for special purposes.

The article by Lee S. Pickrell (cited above) required output to be prepared on an HP LaserJet II.

Only one item (other than advertisements) was received as camera copy: the figures for the *Output routines* tutorial by David Salomon (p. 69), which were prepared on a 300 dpi Apple LaserWriter.

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.

Institutional Members

The Aerospace Corporation,
El Segundo, California

Air Force Institute of Technology,
Wright-Patterson AFB, Ohio

American Mathematical Society,
Providence, Rhode Island

ArborText, Inc.,
Ann Arbor, Michigan

ASCII Corporation,
Tokyo, Japan

Aston University,
Birmingham, England

Belgrade University,
Faculty of Mathematics,
Belgrade, Yugoslavia

Brookhaven National Laboratory,
Upton, New York

Brown University,
Providence, Rhode Island

California Institute of Technology,
Pasadena, California

Calvin College,
Grand Rapids, Michigan

Carnegie Mellon University,
Pittsburgh, Pennsylvania

Centre Inter-Régional de
Calcul Électronique, CNRS,
Orsay, France

College of William & Mary,
Department of Computer Science,
Williamsburg, Virginia

DECUS, L&T Special Interest
Group, *Marlboro, Massachusetts*

Department of National Defence,
Ottawa, Ontario, Canada

Digital Equipment Corporation,
Nashua, New Hampshire

Edinboro University
of Pennsylvania,
Edinboro, Pennsylvania

Emerson Electric Company,
St. Louis, Missouri

Environmental Research
Institute of Michigan,
Ann Arbor, Michigan

European Southern Observatory,
Garching bei München,
Federal Republic of Germany

Fermi National Accelerator
Laboratory, *Batavia, Illinois*

Fordham University,
Bronx, New York

Försvarets Materielverk,
Stockholm, Sweden

General Motors
Research Laboratories,
Warren, Michigan

Geophysical Company
of Norway A/S,
Stavanger, Norway

GKSS, Forschungszentrum
Geesthacht GmbH,
Geesthacht, Federal Republic of
Germany

Grinnell College,
Computer Services,
Grinnell, Iowa

Harvard University,
Computer Services,
Cambridge, Massachusetts

Hatfield Polytechnic,
Computer Centre,
Herts, England

Hewlett-Packard Co.,
Boise, Idaho

Hughes Aircraft Company,
Space Communications Division,
Los Angeles, California

IBM Corporation,
Scientific Center,
Palo Alto, California

Informatika, *Hamburg,*
Federal Republic of Germany

Institute for Advanced Study,
Princeton, New Jersey

Institute for Defense Analyses,
Communications Research
Division, *Princeton, New Jersey*

Intevep S. A., *Caracas, Venezuela*
Iowa State University,
Ames, Iowa

Kuwait Institute for
Scientific Research,
Safat, Kuwait

The Library of Congress,
Washington D.C.

Los Alamos National Laboratory,
University of California,
Los Alamos, New Mexico

Louisiana State University,
Baton Rouge, Louisiana

Marquette University,
Department of Mathematics,
Statistics and Computer Science,
Milwaukee, Wisconsin

Massachusetts Institute
of Technology,
Artificial Intelligence Laboratory,
Cambridge, Massachusetts

Mathematical Reviews,
American Mathematical Society,
Ann Arbor, Michigan

Max Planck Institut
für Mathematik,
Bonn, Federal Republic of Germany

Max Planck Institute Stuttgart,
Stuttgart, Federal Republic of
Germany

McGill University,
Montréal, Québec, Canada

Michigan State University,
Mathematics Department,
East Lansing, Michigan

National Cancer Institute,
Frederick, Maryland

National Research Council
Canada, Computation Centre,
Ottawa, Ontario, Canada

Naval Postgraduate School,
Monterey, California

New Jersey Institute of
Technology, *Newark, New Jersey*

New York University,
Academic Computing Facility,
New York, New York

Nippon Telegraph &
Telephone Corporation,
Software Laboratories,
Tokyo, Japan

- Northeastern University,
Academic Computing Services,
Boston, Massachusetts
- Norwegian Pulp & Paper
Research Institute,
Oslo, Norway
- Pennsylvania State University,
Computation Center,
University Park, Pennsylvania
- Personal TEX, Incorporated,
Mill Valley, California
- Princeton University,
Princeton, New Jersey
- Promis Systems Corporation,
Toronto, Ontario, Canada
- Peter Isaacson Publications,
Victoria, Australia
- Purdue University,
West Lafayette, Indiana
- Queens College,
Flushing, New York
- RE/SPEC, Inc.,
Rapid City, South Dakota
- Rice University,
Department of Computer Science,
Houston, Texas
- Rogaland University,
Stavanger, Norway
- Ruhr Universität Bochum,
Rechenzentrum,
*Bochum, Federal Republic of
Germany*
- Rutgers University, Hill Center,
Piscataway, New Jersey
- St. Albans School,
*Mount St. Alban, Washington,
D.C.*
- Sandia National Laboratories,
Albuquerque, New Mexico
- SAS Institute,
Cary, North Carolina
- I. P. Sharp Associates,
Palo Alto, California
- Smithsonian Astrophysical
Observatory, Computation Facility,
Cambridge, Massachusetts
- Software Research Associates,
Tokyo, Japan
- Sony Corporation,
Atsugi, Japan
- Space Telescope Science Institute,
Baltimore, Maryland
- Springer-Verlag,
*Heidelberg, Federal Republic of
Germany*
- Stanford Linear
Accelerator Center (SLAC),
Stanford, California
- Stanford University,
Computer Science Department,
Stanford, California
- Stefan Ram, Programming and
Trade, *Berlin, Federal Republic of
Germany*
- Stratus Computer, Inc.,
Marlboro, Massachusetts
- Syracuse University,
Syracuse, New York
- Talaris Systems, Inc.,
San Diego, California
- TECOGRAF Software,
Milan, Italy
- Texas A & M University,
Department of Computer Science,
College Station, Texas
- Texcel, *Oslo, Norway*
- TRW, Inc., *Redondo Beach,
California*
- Tufts University,
Medford, Massachusetts
- TV Guide, *Radnor, Pennsylvania*
- TYX Corporation,
Reston, Virginia
- UNI-C, *Aarhus, Denmark*
- Universidad Sevilla,
Sevilla, Spain
- Universidade de Coimbra,
Coimbra, Portugal
- Università degli Studi Milano,
Istituto di Cibernetica,
Milan, Italy
- University College,
Cork, Ireland
- University of Alabama,
Tuscaloosa, Alabama
- University of British Columbia,
Computing Centre,
*Vancouver, British Columbia,
Canada*
- University of British Columbia,
Mathematics Department,
*Vancouver, British Columbia,
Canada*
- University of Calgary,
Calgary, Alberta, Canada
- University of California,
Division of Library Automation,
Oakland, California
- University of California, Berkeley,
Computer Science Division,
Berkeley, California
- University of California, Berkeley,
Space Astrophysics Group,
Berkeley, California
- University of California, Irvine,
Department of Mathematics,
Irvine, California
- University of California, Irvine,
Information & Computer Science,
Irvine, California
- University of California, Los Angeles, Computer
Science Department Archives,
Los Angeles, California
- University of California, San Diego, La Jolla, *California*
- University of Canterbury,
Christchurch, New Zealand
- University of Chicago,
Computing Organizations,
Chicago, Illinois
- University of Chicago,
Chicago, Illinois
- University of Crete,
Institute of Computer Science,
Heraklio, Crete, Greece
- University of Delaware,
Newark, Delaware
- University of Exeter,
Computer Unit,
Exeter, Devon, England
- University of Glasgow,
Department of Computing Science,
Glasgow, Scotland

University of Groningen,
Groningen, The Netherlands

University of Illinois at Chicago,
Computer Center,
Chicago, Illinois

University of Kansas,
Academic Computing Services,
Lawrence, Kansas

University of Maryland,
College Park, Maryland

University of Massachusetts,
Amherst, Massachusetts

Université de Montréal,
Montréal, Québec, Canada

University of Oslo,
Institute of Informatics,
Blindern, Oslo, Norway

University of Ottawa,
Ottawa, Ontario, Canada

University of Salford,
Salford, England

University of Southern California,
Information Sciences Institute,
Marina del Rey, California

University of Stockholm,
Department of Mathematics,
Stockholm, Sweden

University of Texas at Austin,
Austin, Texas

University of Vermont,
Burlington, Vermont

University of Washington,
Department of Computer Science,
Seattle, Washington

University of Western Australia,
Regional Computing Centre,
Nedlands, Australia

University of Wisconsin,
Academic Computing Center,
Madison, Wisconsin

Uppsala University,
Uppsala, Sweden

USDA Forest Service,
Washington, D.C.

Vereinigte Aluminium-Werke AG,
Bonn, Federal Republic of Germany

Villanova University,
Villanova, Pennsylvania

Vrije Universiteit,
Amsterdam, The Netherlands

Washington State University,
Pullman, Washington

Widener University,
Computing Services,
Chester, Pennsylvania

John Wiley & Sons, Incorporated,
New York, New York

Worcester Polytechnic Institute,
Worcester, Massachusetts

Yale University, Computer Center,
New Haven, Connecticut

Yale University,
Department of Computer Science,
New Haven, Connecticut



\$79
IS ALL IT WILL
TAKE TO ADD A FEW
OF THESE TO YOUR
T_EX DOCUMENTS



With T_EX PIC Graphics language*, you will have the tools to make graphics for your T_EX documents. T_EX PIC is now available from Bob Harris at:



MICRO PROGRAMS INC
251 Jackson Avenue, Syosset NY 11791
Telephone: (516) 921 1351.

*TUG Boat Volume 10, No. 4, Page 627
1989 Stanford Conference Proceedings

Request for Information

The T_EX Users Group maintains a database and publishes a membership list containing information about the equipment on which T_EX is (or will be) installed and about the applications for which T_EX is used. This list is updated periodically and distributed to members with TUGboat, to permit them to identify others with similar interests. Thus, it is important that the information be complete and up-to-date.

Please answer the questions below, in particular those regarding the status of T_EX and the hardware on which it runs. (Operating system information is particularly important in the case of IBM mainframes and VAX.) This hardware information is used to group members in the listings by computer and output device.

If accurate information has already been provided by another TUG member at your site, indicate that member's name and the same information will be repeated automatically under your name. If your current listing is correct, you need not answer these questions again. Your cooperation is appreciated.

- Send completed form with remittance (checks, money orders, UNESCO coupons) to:

T_EX Users Group
P. O. Box 594
Providence, Rhode Island 02901, U.S.A.

- For foreign bank transfers direct payment to the T_EX Users Group, account #002-031375, at:

Rhode Island Hospital Trust National Bank
One Hospital Trust Plaza
Providence, Rhode Island 02903-2449, U.S.A.

- General correspondence about TUG should be addressed to:

T_EX Users Group
P. O. Box 9506
Providence, Rhode Island 02940-9506, U.S.A.

Name: _____
Home [] Address: _____
Bus. [] _____

Qty	1990 Membership/TUGboat Subscription (Jan.-Dec.)											Amount
	New (first-time): [] \$35.00 each											
	Renewal: [] \$45.00; [] \$35.00 - reduced rate if renewed before February 1, 1990											
	Mailing charges per subscription: Canada/Mexico - \$5; Europe - \$10; Other Countries - \$15											
	TUGboat back volumes	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	
	Circle volume(s) desired:	v. 1	v. 2	v. 3	v. 4	v. 5	v. 6	v. 7	v. 8	v. 9	v. 10	
		\$18	\$50	\$35	\$35	\$35	\$50	\$50	\$50	\$50	\$75	

Issues of TUGboat will be shipped via air service outside North America.

Quantity discounts available on request.

TOTAL ENCLOSED: _____
(Prepayment in U.S. dollars required)

Membership List Information

Institution (if not part of address): _____

Date: _____

Title: _____

Status of T_EX: [] Under consideration

Phone: _____

[] Being installed

Network address: _____

[] Up and running since: _____

[] Arpanet [] BITnet
[] CSnet [] uucp
[] JANET [] other _____

Approximate number of users: _____

Specific applications or reason for interest in T_EX:

Version of T_EX:

My installation can offer the following software or technical support to TUG:

[] Pascal

Please list high-level T_EX users at your site who would not mind being contacted for information; give name, address, and telephone.

[] C

[] other (describe)

From whom obtained: _____

Hardware on which T_EX is used:

Computer(s)	Operating system(s)	Output device(s)
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____

T_EX

T
H
E
A
R
B
O
R
T
E
X
T
W
A
Y

DEPTH

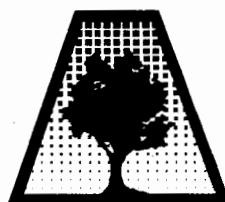
DEDICATION

DIVERSITY

ArborText's T_EX products reflect more than 10 year's experience and the depth of our professional development staff. Our T_EX programs are flexible, fast, and loaded with features. Our work with the forthcoming T_EX 3.0 and virtual fonts is yet another example of how our experience keeps us on the leading edge of T_EX development.

ArborText continues its tradition of dedication to its customers with a knowledgeable and responsive customer support staff, the ArborText T_EX Software Newsletter, and frequent software upgrades. We are also continuing to search for products to complement T_EX, and enhance its usefulness.

ArborText offers more T_EX products for more platforms than anyone. Whether you work on a Sun, Apollo, HP, DEC, Mac or PC, ArborText has the T_EX solution for you. ArborText is also a distributor of T_EX support products including *Mathematica*TM and HiJaak.TM Such diversity makes ArborText the preferred source for all your T_EX needs.



ARBORTEXT INC.

535 W. William St., Ann Arbor, MI 48103, (313) 996-3566, FAX (313) 996-3573

All product names are trademarks or registered trademarks of their respective owners.

VT_EX for the 90's

VECTOR™ VT_EX

Typefaces

Typeface

Typefaces

Typefaces

Retain all the advantages of VT_EX and

- 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 90 points.
- Standard font effects include compression, slant, smallcaps, outline and shading. New: shadow.
- Discover the universe of MicroPress professional typefaces: not available for any other VT_EX.

List price \$399 Introductory offer \$299

Includes the VT_EX typesetter, 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.

Introductory offer expires on April 1, 1990. S/H add \$5. COD add \$5. WordPerfect Interface add \$100. Site licenses available. Dealers' inquiries welcome. Professional typefaces and METAFONT sources available for older implementations of VT_EX. Order by May 1st and receive an additional free font.

MicroPress, Inc.

67-30 Clyde Street, Suite 2N
Forest Hills, NY 11375

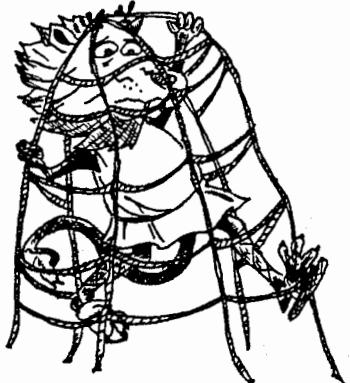
Call: (718) 575 1816 Fax: (718) 575 8038

VT_EX IS A TRADEMARK OF MICROPRESS INC. OTHER PRODUCTS MENTIONED ARE TRADEMARKS OF THEIR RESPECTIVE COMPANIES.



PUT GRAPHICS IN \TeX WITH CAPTURE

CAPTURE is the graphics solution for PC-based \TeX . CAPTURE places graphics in \TeX documents produced on IBM PC systems with Hewlett-Packard LaserJet printers (and compatibles). Doesn't use PostScript.

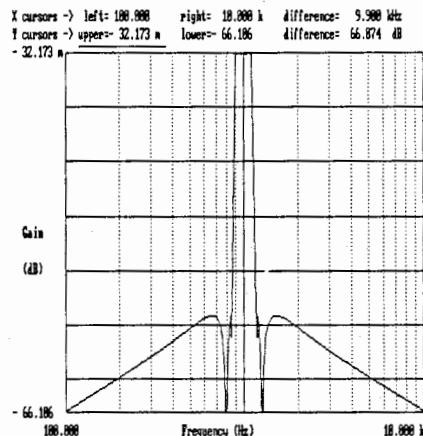
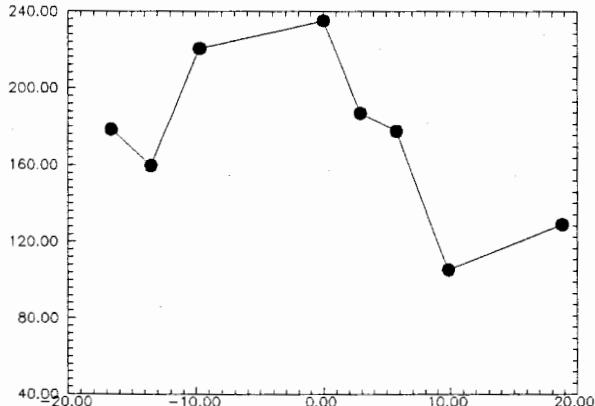


Designed for \TeX . Carefully removes all 28 LaserJet control codes that disrupt \TeX . Tested with PCT \TeX , $\mu\text{\TeX}$, and \TeX^+ . This ad was made using CAPTURE and \TeX^+ .

"Captures" the graphics generated by any application program. Use CAPTURE to include graphics from "paint" programs, circuit design, CAD, scientific data plotters, optics design, terminal emulators, clip art, spreadsheets, databases — any program that supports the LaserJet.

Can position the graphics: flushleft, flushright, centered, etc. Removes leading and trailing blank space. Graphics are treated as an `\hbox{}` in \TeX .

Price: \$137. Special Introductory offer: \$115.



Prime Distributor for Wynne-Manley Software, Inc:
Micro Programs, Inc.
251 Jackson Avenue
Syosset, NY 11791
(516) 921-1351

Additional Distributors:
Oregon House Software

Trademarks: Hewlett-Packard and LaserJet are trademarks of Hewlett-Packard, Inc., IBM is a trademark of IBM, Inc., PostScript is a trademark of Adobe Systems, Inc., PCT \TeX is a trademark of Personal \TeX , Inc., $\mu\text{\TeX}$ is a trademark of ArborText, Inc., \TeX^+ is a trademark of Oregon House Software, Inc. CAPTURE and Wynne-Manley are trademarks of Wynne-Manley Software, Inc.

Acknowledgements: Top graphics produced on PC Paintbrush by Z-Soft, Inc., lower left graphics produced by GRAPHER by Golden Software, Inc., lower right graphics produced by AFD Active Filter Design Program by RLM Research, Inc.

IS YOUR PC STILL TEXLESS?

It doesn't have to be.

*End Texlessness with the Personal TeX, Inc. products below,
at 20% off to TUG members.*

*Coming in spring:
PCTeX 386 and PCTeX Special 5th Year Anniversary catalog.*

PTIJET FOR HP DESKJET. Full featured printer driver for HP DeskJet, PLUS. Laser quality output.
\$119- \$95

PC TeX + PTILASER + PTIVIEW. TeX82, Version 2.9: professional formatting and typesetting results—for amateur prices. Includes INIT_{TeX}, La_{TeX}, AMS-TeX, VANILLA Macro Pak, PC TeX and LaTeX manuals. Plus a PTILaser device driver, to take full advantage of your laser printer. PLUS the PTIView screen previewer for on-screen previewing of your TeX documents and immediate editing. Top performance and low cost make this our most popular package. \$499- \$399

PC TeX + PTILASER. As above, but without the PTIView screen previewer. \$399- \$309

PC TeX + PTIDOT + PTIVIEW. This package gives you all the TeX and PTIView benefits, together with our dot-matrix device driver for reliable, low cost printing. \$399- \$309

PC TeX + PTIJET + PTIVIEW. Same as the above package, but with PTIJet instead of PTIDot. Laser quality output. \$429- \$343

PC TeX + PTIJET. As above, without the PTIView screen previewer. \$329- \$263

PCMF—METAFONT for the PC. Lets you design fonts and create graphics. (Not for the novice.) pcMF Version 1.7. \$195- \$156

PTI LASER HP+, SERIES II. This device driver for the HP LaserJet Plus and Series II laser printers takes full advantage of the 512K resident memory.

\$195- \$156

PTI LASER POSTSCRIPT. Device driver for PostScript printer; allows the resident fonts and graphic images to be used in TeX in documents.

\$195- \$156

PTI FONTWARE Interface Package. Software to generate Bitstream outline fonts at any size. (The Interface is necessary to use Bitstream fonts. Fonts are not included—order below). \$95- \$76

PTI FONTWARE WITH SWISS or DUTCH. Same as above but includes your choice of either Swiss or Dutch at a special bundled price. \$179- \$143

BITSTREAM Font Families. An extensive library of 30 type families, in any size you specify, with true typographic quality. Each family: \$179- \$143



To order, just dial

(415) 388-8853

12 Madrona Avenue Mill Valley, CA 94941 FAX: (415) 388-8865 VISA, MC accepted.

Requires: DOS 2.0 or later, 512K RAM, 10M hard disk. TeX is an American Mathematical Society TM. PC TeX is a Personal TeX, Inc. TM. Manufacturers' names are their TMs. Outside the USA, order through your local PC TeX distributor. Inquire about available distributorships and site licenses. This ad was produced using PC TeX and Bitstream fonts.

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 PCT_EX® 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

ANNOUNCING:

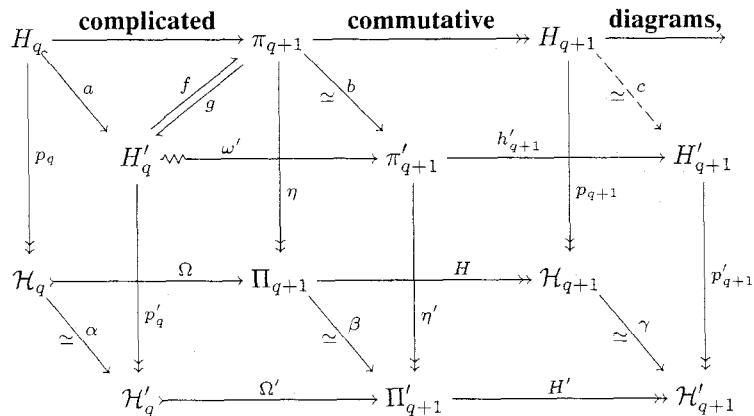
LAMS-TEX

new!

by
Michael Spivak,
author of the
AMS-TEX macro package
and *The Joy of TEX*

The Synthesis

All the functionality of LATEX, with much greater flexibility, and all features of AMS-TEX, PLUS



complicated tables,		Group C							
		Committee A				Committee B	The T <small>E</small> xplorators Corporation 3701 W. Alabama, Suite 450-273 Houston, TX 77027		
		Unit 1		Unit 2					
		Side 1	Side 2 (Left)	Side 1	Side 2 (Left)				

AND MUCH, MUCH, MORE!

- Latest amstex.tex file (version 2.0)
- LAMS-TEX macro packages
- 300 page Manual (assumes some familiarity with AMS-TEX)
- Fonts for commutative diagrams: .tfm files, .pk files at 118, 180, 240, and 300 dpi, and Meta-Font sources; MAC versions have fonts for TEXtires.
- dvipaste program for including tables in files
- index program

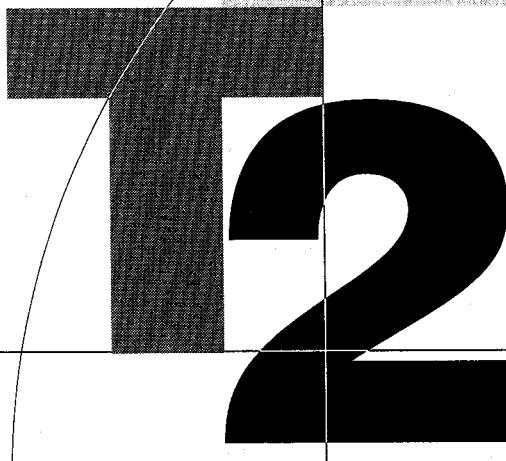
Single user price \$95; Texas orders add appropriate sales tax. Specify MS-DOS (5 1/4" diskettes) or MAC; for other configurations, write for information. Add \$8 shipping/handling in U.S. and Canada (UPS 2nd day air in U.S., first class to Canada), and \$35 for air shipment elsewhere. Send check or money order to the address cleverly hidden in the table. Prices subject to change.

T_EX Device Interfaces for VMS

PostScript

LaserJet

LN03



Northlake Software
812 SW Washington, Suite 1100
Portland, Oregon 97205 USA

503-228-3383 fax 503-228-5662

The VMS T_EX specialists



Updated \TeX Products from the American Mathematical Society



$\mathcal{A}\mathcal{M}\mathcal{S}$ - \TeX Version 2.0

$\mathcal{A}\mathcal{M}\mathcal{S}$ - \TeX , the \TeX macro package that simplifies the typesetting of complex mathematics, has been updated to version 2.0. $\mathcal{A}\mathcal{M}\mathcal{S}$ - \TeX is intended to be used in conjunction with AMSFonts 2.0 (see below). However, $\mathcal{A}\mathcal{M}\mathcal{S}$ - \TeX can also be used without AMSFonts. $\mathcal{A}\mathcal{M}\mathcal{S}$ - \TeX is available on IBM or Macintosh diskettes—either format may be uploaded to many mainframe computers. **Prices:** \$30 list, \$27 AMS member. A free upgrade is available (until September 1, 1990), for those who have purchased a previous version.

AMSFonts Version 2.0

AMSFonts 2.0 are designed for use with either $\mathcal{A}\mathcal{M}\mathcal{S}$ - \TeX 2.0 or Plain \TeX . AMSFonts 2.0 **cannot** be used with previous versions of $\mathcal{A}\mathcal{M}\mathcal{S}$ - \TeX . Two distributions of fonts are available: one for use on PCs and mainframes (with any implementation of \TeX), the other for use on a Macintosh with *Textures*. The fonts included on these distributions are:

Font Name	Description	Point Sizes	Font Name	Description	Point Sizes
CMEX	CM Math Extension	7-9*	EUSM	Euler Script Medium	5-10
CMCSC	CM Caps and Small Caps	8-9*	EUEX	Euler Compatible Extension	7-10
CMMIB	CM Math Italic Boldface	5-9*	MSAM	Symbols	5-10
CMBSY	CM Bold Symbols	5-9*	MSBM	Symbols (w/Blackboard Bold)	5-10
EURB	Euler Cursive Boldface	5-10	WNCYR	Cyrillic Upright	5-10**
EURM	Euler Cursive Medium	5-10	WNCYI	Cyrillic Italic	5-10**
EUFB	Euler Fraktur Boldface	5-10	WNCYB	Cyrillic Boldface	5-10**
EUFM	Euler Fraktur Medium	5-10	WNCYSC	Cyrillic Caps and Small Caps	10**
EUSB	Euler Script Boldface	5-10	WNCYSS	Cyrillic Sans Serif	8-10**

* 10 point is included in the standard \TeX distribution.

** Developed by the University of Washington

AMSFonts for use on a PC or mainframe

- Font Resolution: 118, 180, 240, 300, 400 dpi (one resolution per order).
- Magnification: All the standard \TeX magnifications are included. The standard magnifications are: 100, 109.5, 120, 144, 172.8, 207.4, and 248.8%.
- Format: high-density 5.25" diskettes.
- Prices: \$45 list, \$41 AMS member. A free upgrade is available (until September 1, 1990), for those who have purchased a previous version.

AMSFonts for use on a Macintosh with *Textures*

- Font Resolution: 72, 144, and 300 dpi (all resolutions included in each order).
- Magnification: The standard distribution includes fonts at 100% and 120%. An extended distribution, containing all the standard \TeX magsteps, is also available.
- Format: double-sided double-density 3.5" diskettes.
- Prices: *Standard* (magsteps 0-1): \$30 list, \$27 AMS member. *Extended* (magsteps 0-5): \$45 list, \$41 AMS member. A free upgrade is available (until September 1, 1990), for those who have purchased a previous version.

SHIPPING AND HANDLING CHARGE: \$8 per order in the US and Canada, \$15 elsewhere.

HOW TO ORDER: Prepayment is required. Send orders to: American Mathematical Society, P. O. Box 1571, Annex Station, Providence, RI 02901. When ordering AMSFonts for the PC, specify desired resolution. For more information: Call the AMS at (401) 455-4166, or (800) 321-4AMS in the continental U.S. and Canada, or write to: \TeX Library, American Mathematical Society, P.O. Box 6248, Providence, RI 02940.

T_EX Plus

T_EX Pro

BIG T_EX

Exciting new products for the PC
based on T_EX 3.0

Big T_EX \$200

They said it couldn't be done! Big T_EX puts four times the main memory of existing PC based T_EX implementations onto your PC (256K of T_EX words). Big T_EX also offers the latest version of T_EX (version 3.0), and uses EMS memory if available. If a lack of main memory is keeping you from creating the tables or using the macros you need, Big T_EX is for you! Includes the new AMST_EX style file that lets you use the AMST_EX macros inside LaT_EX.

T_EX Pro \$350

T_EX Pro adds the power of Big T_EX to T_EX Plus! T_EX Pro includes the T_EXWRITE editor, CT_EX, Big T_EX, and printer drivers for the HP LaserJet Plus/Series II and IIP, PostScript laser printers, and Epson FX/IBM Graphics dot matrix printers. Also included are a set of Computer Modern and SliT_EX fonts for the laser printers. The dot matrix fonts can be purchased separately for \$25, or substituted for the laser printer fonts.

T_EX Plus 2.0 \$195

T_EX Plus provides an integrated solution to using T_EX on a PC. New features include support of T_EX 3.0, support for EMS, and faster execution speed. We've also added a dot matrix driver for the Epson FX and compatibles.

T_EX Plus includes the T_EXWRITE editor, C_TE_X, and printer drivers for the HP LaserJet Plus/Series II and IIP, PostScript laser printers, and Epson FX/IBM Graphics dot matrix printers. Also included are a set of Computer Modern (and SliT_EX) fonts for the laser printers. The dot matrix fonts can be purchased separately for \$25, or may be substituted for the laser printer fonts.

C_TE_X 2.0 \$129

C_TE_X is an implementation of T_EX version 3.0 for the PC. Faster than previous versions, C_TE_X version 2.0 provides virtual memory management for handling large macro packages and for environments, such as a network, where the available RAM is reduced by system overhead. Version 2.0 also makes use of EMS memory, where available, for faster processing. Includes LaT_EX, SliT_EX and AMST_EX macro packages.

T_EXPRINT/HP \$129

T_EXPRINT/HP translates T_EX's output for HP LaserJet Plus, Series II and IIP printers. Features include landscape printing, collating, odd or even page selection, reverse order printing, inclusion of graphics (using packages such as CAPTURE), all from an easy-to-use menu interface. A utility for converting HP Soft Fonts for use with T_EX, and a set of Computer Modern and SliT_EX fonts are also included.

T_EXPRINT/PS \$129

T_EXPRINT/PS translates T_EX's output for PostScript printers such as the Apple LaserWriter, the QMS PS-800/810, and the NEC LC-890. Features include support for PostScript fonts, landscape printing, collating, odd or even page selection, reverse order printing, inclusion of graphics, all from an easy-to-use menu interface. A utility for converting Adobe Font Metric files (so that downloaded PostScript fonts can also be used with T_EX), and a set of Computer Modern and SliT_EX fonts are also included.

MAXview \$125

MAXview, produced by Aurion Tecnología, adds easy-to-use screen previewing capabilities to your T_EX system. Maxview can be used with IBM CGA/EGA/VGA, Hercules, and many other monitors.

To order, call
Oregon House Software
(916) 692-1377

Or send your order to:

Oregon House Software, Box 70, 12894 Rices Crossing Rd., Oregon House, CA 95962

In Canada:

Oregon House Software, Box 27057, 1395 Marine Dr., West Vancouver, B.C. V7T 2X8

Publishing Companion translates

WordPerfect

to

T_EX

It doesn't take a T_EXpert to use T_EX.

With **Publishing Companion**, you can publish documents using T_EX with **little or no T_EX knowledge**. Your WordPerfect files are translated into T_EX files, so anyone using this simple word processor can immediately begin typesetting their own documents!

And now, **Publishing Companion** translates **WordPerfect 5.0 and 5.1** files into T_EX.

Retail Price	\$249.00
--------------------	----------

Academic Discount Price	\$199.00
-------------------------------	----------

For the power of T_EX with the ease of a word processor, **Publishing Companion** is your "best friend" for desktop publishing.

For more information to place an order, call or write:



**30 West First Ave., Suite 100
Columbus, Ohio 43201
(614) 294-3535
FAX (614) 294-3704**

**DESKTOP PUBLISHING HAS NEVER BEEN SIMPLER
AND WILL NEVER BE THE SAME**

Public Domain T_EX

The authorized and current versions T_EX software are available from *Maria Code - Data Processing Services* by special arrangement with Stanford University and other contributing universities. The standard distribution tape contains the source of T_EX and METAFONT, the macro libraries for A_MS-T_EX, L_AT_EX, SliT_EX and HP T_EX, sample device drivers for a Versetec and LN03 printers, documentation files, and many useful tools.

Since these are in the public domain, they may be used and copied without royalty concerns. They represent the official versions of T_EX. A portion of your tape cost is used to support development at Stanford University.

If you have a DEC VAX/VMS, IBM CMS, IBM MVS or DEC TOPS operating system, you will want to order a special distribution tape which contains "ready-to-run" T_EX and METAFONT. If you do not have one of these systems, you must perform a more involved installation which includes compiling the source with your Pascal compiler. Ready-to-run versions of T_EX are available for other systems from various sources at various prices. You may want to examine these before ordering a standard distribution tape.

The font tapes contain GF files for the Computer Modern fonts. While it is possible to generate these files yourself, it will save you a lot of CPU time to get them on tape.

All systems are distributed on 9 track, 1600 bpi magnetic tapes. If both a distribution tape and a font tape are ordered, they may be combined on a single 2400' reel, space permitting.

Your order will be filled with the current versions of software and manuals at the time it is received. If you want a specific version, please indicate that on your order.

Please use the form on the next page for your order. Note that postage, except domestic book rate is based on the item weights in pounds. If you want to place your order by telephone, please call (408) 735-8006 between 9:00 am and 2:00 pm West Coast time. Do not call for technical assistance since no one there can help you.

We normally have a good stock of books and tapes, so your order can be filled promptly — usually within 48 hours.

Make checks payable to *Maria Code - Data Processing Services*. Export orders must have a check drawn on a US bank or use an International Money Order. Purchase orders are accepted.

TeX Order Form

TeX Distribution tapes:

- Standard ASCII format
- Standard EBCDIC format
- Special VAX/VMS format Backup
- Special DEC 20/TOPS 20 Dumper format
- Special IBM VM/CMS format
- Special IBM MVS format

Tape prices: \$92.00 for first tape,
\$72.00 for each additional tape.

Font Library Tapes (GF files)

- 300 dpi VAX/VMS format
- 300 dpi generic format
- IBM 3820/3812 MVS format
- IBM 3800 CMS format
- IBM 4250 CMS format
- IBM 3820/3812 CMS format

Total number of tapes _____

Postage: allow 2 lbs. for each tape

Documents:

	Price \$	Weight	Quantity
TeXbook (vol. A) softcover	27.00	2	_____
TeX: The Program (vol. B) hardcover	40.00	4	_____
METAFONT book (vol. C) softcover	22.00	2	_____
METAFONT: The Program (vol. D) hardcover ..	40.00	4	_____
Computer Modern Typefaces (vol. E) hardcover	40.00	4	_____
LaTeX document preparation system	27.00	2	_____
WEB language *	12.00	1	_____
TeXware *	10.00	1	_____
BibTeX *	10.00	1	_____
Torture Test for TeX *	8.00	1	_____
Torture Test for METAFONT *	8.00	1	_____
METAFONTware *	15.00	1	_____
Metamarks *	15.00	1	_____

* published by Stanford University

Payment calculation:

Number of tapes ordered _____ Total price for tapes _____

Number of documents ordered _____ Total price for documents _____

Add the 2 lines above _____

Orders from within California: Add sales tax for your location. _____

Shipping charges: (for domestic book rate, skip this section)

Total weight of tapes and books _____ lbs.

Check One	<input type="checkbox"/> domestic priority mail	rate \$1.50/lb.
	<input type="checkbox"/> air mail to Canada and Mexico:	rate \$2.00/lb.
	<input type="checkbox"/> export surface mail (all countries):	rate \$1.50/lb.
	<input type="checkbox"/> air mail to Europe, South America:	rate \$5.00/lb.
	<input type="checkbox"/> air mail to Far East, Africa, Israel:	rate \$7.00/lb.

Multiply total weight by shipping rate. Enter shipping charges: _____

Total charges: (add charges for materials, tax and shipping) _____

Send to: Maria Code, DP Services, 1371 Sydney Drive, Sunnyvale, CA 94087.
Include your name, organization, address, and telephone number.

Are you or your organization a member of TUG? _____

TEX Users

Take Note . . .

Computer Composition Corporation offers the following services to those who are creating their technical files using TEX:

- Convert your DVI files to fully paginated typeset pages on our APS-5 phototypesetters at 1400 dpi resolution.
- Files can be submitted on magnetic tape or PC diskettes.
- Provide 300 dpi laser-printed page proofs which simulate the typeset page. (Optional service \$1.50 per page)
- Macro writing and keyboarding from traditionally prepared manuscripts **in several typeface families** via the TEX processing system. Send us your manuscript for our review and quotation.
- Full keylining and camera work services, including halftones, line art, screens and full-page negatives or positives for your printer.
- Quick turnaround (**usually less than 48 hours!**) on customer supplied DVI files of 500 typeset pages or less.
- From DVI files: first 100 typeset pages at \$4.75 per page; 100 pages and over at \$3.50 per page. **Lower prices for slower turnaround service.**

*For further information and / or a specific quotation,
call or write Frank Frye or Tim Buckler*



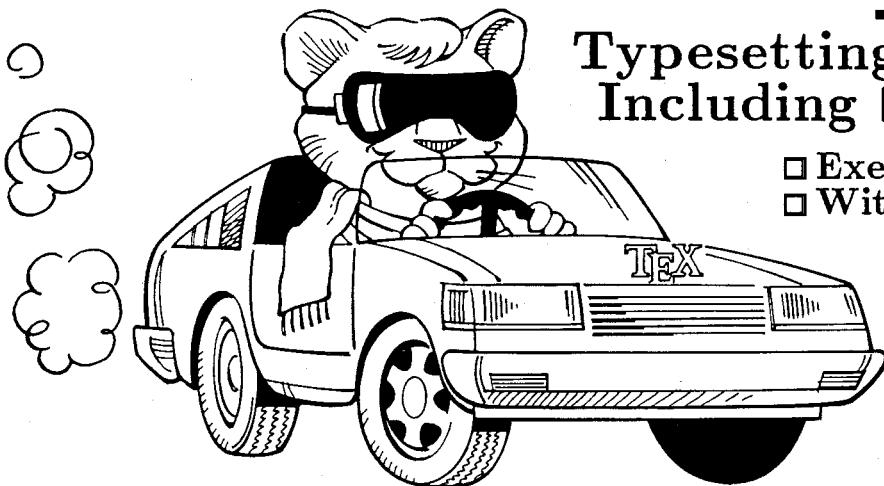
COMPUTER COMPOSITION CORPORATION

1401 West Girard Avenue • Madison Heights, MI 48071

(313) 545-4330 FAX (313) 544-1611

— Since 1970 —

TurboTeX



Typesetting Software Including METAFONT

Executables \$150
 With source \$300

TEX
Update!

TurboTeX Release 3.0 software brings you the latest *TeX* 3.0 and METAFONT 2.0 standards: preloaded plain *TeX*, *L^AT_EX*, *AM_S-TeX* and *AM_S-L^AT_EX*, and plain METAFONT interfaced to CGA/EGA/VGA/Hercules graphics; TRIP and TRAP certification; Computer Modern and *L^AT_EX* fonts, and printer drivers for HP LaserJet Plus/II/IIP, HP DeskJet, PostScript, and Epson LQ and FX dot-matrix printers. This wealth of software runs on your IBM PC (MS-DOS 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, 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.

■ Power Features: TurboTeX breaks the 640K memory barrier under MS-DOS on any IBM-compatible PC with our virtual memory sub-system. Even without expanded memory hardware, you'll

have the same sized *TeX* that runs on multi-megabyte mainframes, with plenty of memory for large documents, complicated formats, and demanding macro packages (like PICTeX and *AM_S-L^AT_EX* 2.0) that break other *TeX* implementations. On larger computers, TurboTeX runs up to 3 times faster in less memory than the Stanford Pascal distribution.

■ Source code: Order the TurboTeX source in portable C, and you will receive more disks with over 85,000 lines of generously commented *TeX*, TurboTeX, METAFONT, and printer driver source code, including: our WEB system in C; PASCHAL, our proprietary Pascal-to-C translator; and preloading, virtual memory, and graphics code. TurboTeX meets C portability standards like ANSI and K&R, and is robustly portable to a growing family of operating systems.

■ Availability & Requirements: TurboTeX executables for IBM PC's include the User's Guide and require 640K and hard disk. Order source code (includes Programmer's Guide) for other machines. Source compiles with Microsoft C 5.0 or later on the PC; other systems need 1 MB memory and a C compiler supporting UNIX standard I/O. Media is 360K 5-1/4" PC floppy disks; other formats at extra cost.

■ Upgrades: If you have TurboTeX Release 2.0, you can upgrade the executables for only \$40. If you have the source distribution, upgrade

both executables and source for \$80. 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 more 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-T_EX Fonts

485 **TEX** fonts identical to
Adobe PostScript Fonts for \$200

Get ready for the quality of Adobe PostScript fonts for your TeX documents and non-PostScript printer! If you use any brand of TeX with an HP LaserJet or DeskJet printer, the APTeX fonts from Kinch add a wealth of attractive typefaces identical to the popular PostScript extended font families.

By de-crypting the Adobe coding, we are able to exactly translate the PostScript fonts into TeX font bit map and metric files. These translated fonts include the renowned Adobe "hints," which render the smaller point sizes of the fonts with remarkable clarity on laser and ink-jet printers. The fonts use the TeX character set encoding and font metrics, including full kerning and ligature programs.

The AP-TEX fonts, supplied on ten 360K 5-1/4" PC floppy disks, contain 35 typefaces in PK format (including TEX font metric (TFM) files) for 300 dots/inch laser and ink-jet printers. The fonts included are identical to the Adobe Post-Script implementations of the trade names and samples shown at right. The point sizes for each typeface included are the TEX sizes 5, 6, 7, 8, 9, 10, 11, 12, 14.4, 17.3, 20.7, and 24.9 points. Headline styles (equal to Times Roman, Helvetica, and Palatino, all in bold) also are included at 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. LaserJet and DeskJet are trademarks of Hewlett-Packard Corporation. PostScript is a registered trademark of Adobe Systems Incorporated. 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 February 8, 1990.

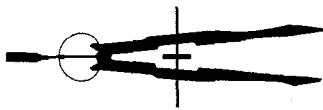
Kinch

DeskJet driver for **TEX**

With The Toolsmith's DVI driver and the Hewlett-Packard DeskJets you get:

- Print Quality - the precision of laser printers (300 dpi) without the price.
- Speed - with downloaded fonts, ≈1 page per minute on a DeskJet, faster on a Plus.
- Quiet - non-impact printing and no fan.

You are only limited by the page size and your **TEXpertise**. \$100 for the DeskJet DVI driver (shipping and any sales tax included). To order or for more information, call or write:



The Toolsmith

P.O. Box 5000
Davis, CA 95617
(916) 753-5040

Requires IBM PC or compatible, hard disk, 512K or more RAM, and MS-DOS 2.11 or later. This ad, including the logo, was created on a DeskJet with **TEX** and METAFONT.

Index of Advertisers

139,148	American Mathematical Society
132	ArborText
cover 3	Blue Sky Research
145	Computer Composition
143,144	DP Services
142	K-Talk Communications
146,147	Kinch Computer Company
130	Micro Programs, Inc.
133	MicroPress, Inc.
138	Northlake Software
140,141	Oregon House Software
135	Personal T<small>E</small>X Inc.
137	T <small>E</small> Xplorators
148	The Toolsmith
136	Type 2000
134	Wynne-Manley Software, Inc.



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.