Machines exist; let us then exploit them to create beauty — a modern beauty, while we are about it. For we live in the twentieth century; let us frankly admit it and not pretend that we live in the fifteenth.

Aldous Huxley
Printing of Today (1928)

TUGBOAT

COMMUNICATIONS OF THE TEX USERS GROUP
EDITOR BARBARA BEETON

VOLUME 9, NUMBER 3 • DECEMBER 1988
PROVIDENCE • RHODE ISLAND • U.S.A.

TUGboat

During 1989, 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. 10, No. 1, is January 17, 1989; the issue will be mailed in April.

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 Production Editor, Alan Wittbecker, at 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. For instructions, write or call Alan Wittbecker at the TUG office.

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

TUGboat Advertising and Mailing Lists

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

TUGboat Editorial Committee

See page 229 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 Alan Wittbecker at the TUG office.

Addresses

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

TEX Users Group Office
P. O. Box 9506
Providence, RI 02940-9506
or
653 North Main Street
Providence, RI 02904
401-751-7760
TUG@Math.AMS.com

Peter Abbott
Computing Service
Aston University
Aston Triangle
Birmingham B4 7ET, England
21 359 5492
pabbott@nss.cs.ucl.ac.uk
Janet: abbottp@uk.ac.aston

Wolfgang Appelt
Gesellschaft für Mathematik und Datenverarbeitung
Postfach 1240, Schloß Birlinghoven
D-5205 Sankt Augustin 1
Federal Republic Germany
uucp: unido!gmdzi!appelt

Mary Armstrong
TEX Users Group
P. O. Box 9506
Providence, RI 02940-9506
401-751-7760
TUG@Math.AMS.com

Elizabeth Barnhart
National EDP Dept
TV Guide
100 Matsonford Road
Radnor, PA 19088
215-293-8890

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

Lawrence A. Beck
Grumman Data Systems
R & D, MS D12-237
Woodbury, NY 11797
516-682-8478

Barbara Beeton
American Mathematical Society
P. O. Box 6248
Providence, RI 02940
401-272-9500
bnb@Math.AMS.com,
bnb@xx.lcs.MIT.Edu,
Beeton@Score.Stanford.Edu

Karen Butler
TEX Users Group
P. O. Box 9506
Providence, RI 02940-9506
401-751-7760
TUG@Math.AMS.com

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@Ohio-State
Bitnet: TS0135@OHSTVMA

Jackie Damrau
Department of Math & Statistics
University of New Mexico
Albuquerque, NM 87131
505-277-4623
damrau@dbitch.unm.edu
Bitnet: damrau@bootes

Michael DeCorte
P. O. Box 652
Potsdam, NY 13676
315-268-2292
mrd@sun.soe.Clarkson.edu

Allen R. Dyer
13320 Tridelphia Road
Ellicott City, MD 21043
301-243-0008 or 243-7283

Maureen Eppstein
Administrative Publications
Stanford University
Encina Hall, Room 200
Stanford, CA 94305
415-725-1717
as.mve@Forsythe.Stanford.Edu

Shawn Farrell
Computing Centre
McGill University
805 Sherbrooke St W
Montréal H3A 2K6, Québec Canada
514-398-3676
Bitnet: CCSF@MCGILLA

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@uwacdc

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

Regina Girouard
American Mathematical Society
P. O. Box 6248
Providence, RI 02940
401-272-9500 x224
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

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

Klaus Guntermann
Technische Hochschule Darmstadt
Fachbereich Informatik
Institut für Theoretische Informatik
Alexanderstraße 24
D-6100 Darmstadt
Federal Republic of Germany
Bitnet: xitikgun@ddathd21

Klaus Heidrich
Inst. u. Abteilung für Forstl.
Biometrie und Informatik
Universität Göttingen
D-3400 Göttingen
Federal Republic of Germany
Bitnet: U0275@DGOGWDG5

Doug Henderson
Division of Library Automation
Office of the President
University of California, Berkeley
300 Lakeside Drive, Floor 8
Oakland, CA 94612-3550
415-987-0561
Bitnet: dlatex@ucbcmsa

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

Don Hosek
Platt Campus Center
Harvey Mudd College
Claremont, CA 91711
Bitnet: dhosek@hmcvax

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

Helmut Jürgensen
Deparment of Computer Science
University of Western Ontario
London N6A 5B7, Ontario, Canada
519-661-3560
Bitnet: helmut@uwovax
UUCP: helmut@julian

David Kellerman
Northlake Software
812 SW Washington
Portland, OR 97205
503-228-3383
uucp: imagen!negami!davek

Thomas Kneser
GWDG
Am Fassberg
34 Göttingen
Federal Republic of Germany
Bitnet: TKNESER@DGOGWDG1

Donald E. Knuth
Department of Computer Science
Stanford University
Stanford, CA 94305
DEK@Sail.Stanford.Edu

J. R. Luyten
Rekencentrum RUG
Landleven 1
9700 AV Groningen, The Netherlands

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

Laurie Mann
Stratus Computer
55 Fairbanks Boulevard
Marlboro, MA 01752
617-460-2610
uucp: harvard!anvil!es!Mann

John S. McCaskill
MPI für biophysikalische Chemie
34 Göttingen
Federal Republic of Germany

Graeme McKinstry
Computing Services Centre
University of Otago
P. O. Box 56
Dunedin, New Zealand
graeme%otago.ac.nz@relay.cs.net

Frank Mittelbach
Fachbereich Mathematik
Universität Mainz
Staudinger Weg 9 D-6500 Mainz
Federal Republic of Germany
Bitnet: SCHOEPF@DMZNAT51

David Ness
TV Guide
Radnor, PA 19088
215-293-8860

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

Arnold Pizer
Department of Mathematics
University of Rochester
Rochester, NY 14627
716-275-4428

Craig Platt
Department of Math & Astronomy
Machray Hall
University of Manitoba
Winnipeg R3T 2N2, Manitoba, Canada
204-474-9832
CSnet: platt@uofm.cc.cdn
Bitnet: platt@uofmcc

David F. Rogers
817 Holly Drive E. Rt. 10
Annapolis, Maryland 21401
dfr@usna.mil

Joachim Schrod
Technische Hochschule Darmstadt
Fachbereich Informatik
Institut für Theoretische Informatik
Alexanderstraße 24
D-6100 Darmstadt
Federal Republic of Germany
Bitnet: XITIJSCH@DDATHD21

Christina Thiele
Canadian Journal of Linguistics
Carleton University
Ottawa K1S 5B6, Ontario Canada
Bitnet: WSSCAT@Carleton

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

Andrew Trevorrow
c/o Computing Service
Aston Triangle
Aston University
Birmingham B4 7ET, England

C. G. van der Laan
Rekencentrum RUG
Landleven 1
9700 AV Groningen, The Netherlands
+31/50 633374 or +31/50 633440
Bitnet: cgl@hgrrug5
DECnet: rugr86::cgl

Samuel B. Whidden
American Mathematical Society
P. O. Box 6248
Providence, RI 02940
401-272-9500
sbw@Math.AMS.com

Alan Wittbecker
TEX Users Group
P. O. Box 9506
Providence, RI 02940-9506
401-751-7760
aew@Math.AMS.com

Dominik Wujastyk
Wellcome Institute for the History
of Medicine
183 Euston Road
London NW1 2BP, England
(01) 387-4477
dow@wjh12.harvard.edu

Ken Yap
Dept of Computer Science
University of Rochester
Rochester, NY 14627
Ken@cs.Rochester.edu
Usenet: ..!rochester!ken

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

General Delivery

From the President

Bart Childs

The TUG meeting in Montréal was a big success. Dean Guenther again put together a great program. He had able assistance from Christina Thiele and Shawn Farrell. Shawn was an entertaining, gracious, and generous host. We are particularly appreciative of his help in understanding drivers, subways, and Olympic taxes. We just wish he had possessed the power to stop construction projects for a few days. Our annual meetings just seem to get better and better every year. We owe a great big thank you to these individuals and to our staff in Providence.

The Sunday evening kickoff events have really been great. My family will long remember my being Gouverneur for an evening and hosting a dinner with my "mistress" at my side. (No gentleman would bring his wife to New France!)

We introduced another new TUG employee at the meeting, Mary Armstrong. Mary's main functions will be in the coordination of recruiting members and selling TUG and its products (especially courses).

I wish to thank Mike Ferguson, Cal Jackson, and Patrick Ion for serving as the Nominations Committee for the past two years. Last year they convinced David Ness to be a candidate for Treasurer and this year they got a slate of four for the office of Vice President. I don't remember ever having a real election before. We elected **Rick Furuta**. Dean Guenther was selected to replace Rick as an at-large member of the Finance Committee.

The Steering Committee changed its name to Board of Directors to agree with the legal terms used in our official Bylaws. Some of the actions of note taken by the Board are:

1. Adding Malcolm Clark, Shawn Farrell, Regina Girouard, and Christina Thiele to our ranks. Malcolm will be our European Coordinator.
2. Changing the VMS Coordinator from Barry Smith to David Kellerman.
3. Selecting next year's Nominations Committee: Liz Barnhart (Chair), Pierre MacKay, and Norman Naugle.

The offices of President and Secretary will expire next year. There is nothing about the qualifications of officers in our Bylaws, but it seems obvious that they should be *active TEX users*. We should consider the matter carefully and help the Committee select the best possible slate of nominees. Our office will make lists of attendees at the last several meetings available for your perusal. Feel free to armtwist and encourage in the best way you can.

Most of the rest of the items in this report were discussed in the meetings of the Board of Directors, the business meeting, or in personal conversations. These are topics that you should know about or we will have to address in the near future.

The Bylaws were generally created with some attention paid to the previous ones from when we were a less official organization. Several items underwent editorial changes to agree more closely with some generic ones used by our lawyer. Several of us would like a few changes, such as:

- Rewrite them in plainer but legal English.
- Give the power to change the Bylaws to the membership (it ended up in the hands of the Board).
- Give some more guidance or rules regarding the membership of the Board.

I have appointed a committee chaired by Allen Dyer (a lawyer himself) to bring a new set to the next meeting. He will be assisted by Barbara Beeton, Lynne Price, Sam Whidden, and Ray Goucher.

We project our budget to show a small loss in this year. The loss is due to the change of accounting methods, purchase of equipment, and a significant—and much needed—increase of staff at TUG headquarters. We expect that these actions will pay dividends, and next year we should return to building the desired cash reserves. Your Finance Committee is closely monitoring these actions.

Cathy Booth pointed out that the attendance at the Exeter meeting was nearly the same as ours. We have several times as many TUG members in the U.S. and Canada as there are in Europe. Can we do something better? Maybe we should send all the officers to the next European meeting to be observers? I have appointed a membership recruitment committee that is chaired by Regina Girouard with volunteers Malcolm Clark and Mary Armstrong (ex officio). Are one or two more of you willing to carry on an E-mail dialogue in making plans? The three main items are: reaching more users; serving European, Asian, and southern hemisphere TEXers; and getting members to participate in local, regional, and national meetings.

By the time you read this, TUG will likely have assumed the responsibility of subsidizing the maintenance of the core software and the TEXhax moderator. We expect this to be carried out by

one graduate student at a university (probably U. of Washington).

You will also notice from a new byline in this issue that maintenance of the LaTeX style files repository has been moved to Clarkson University under the management of a new volunteer, Mike DeCorte. Our warmest thanks go to Ken Yap for taking such good care of it for the past few years.

It has been strongly suggested that TUG be in the business of making distributions of style files, TeXhax, etc., available on diskettes, for users who have no access to any of the electronic networks. We are working on plans for that.

Happy TeXing.

Extra! Extra! TUGboat Becomes a Quarterly

Beginning in 1989, TUGboat will be a quarterly. The fourth issue will be dedicated to the Proceedings of the Annual Meeting. This will ensure that the contents of these papers will be presented to the entire membership.

Another Honorary Degree for Donald Knuth: Doctor of Science, Oxford University 22 June 1988

Presentation by the Public Orator (Mr. Godfrey Bond)

Illvstrissime atqve honaratissime domine cancellarie, vosqve egregii procvratores:

De machinis computatricibus quibus studiorum causa utimur saepissime et in officinis et inter vina disputamus academici. *nota magis nulli domus est sua* quam nobis apparatus illi molliores, ut vocantur, quos in machinis illis dirigendis adhibemus,

visceribus mandata extrinsecus insinuantes.

Levia quidem sunt haec colloquia, χελιδόνων tantum μουσεία. subest tamen scientia subtilissima computandi qua imbuti mandata illa docte machinis iniungimus, ut ordine praescripto data digerant. 'scientiam' dixi, 'artem' tamen hic mavult appellare. artem enim exercentis est eleganter rationem ingeniosam computandi excogitare ex qua non solum artifex ipse sed etiam qui existimatores accuratius intuentur magna concitatione mentis commoventur, magnam capiunt voluptatem.

Facultati igitur Artium debet hic quem produco adscribi, quamvis ad gradum Doctoris in Scientia admittatur. quin etiam φιλόλογον debemus salutare qui de linguis quibus viri docti mandata exprimunt luculenter scripsit nec non de verborum computatoriorum explicatione. etenim opus magnum comparat, voluminibus iam tribus vulgatis, cui titulus est *Ars machinis computatricibus mandata iniungendi.* quibus in libris genera omnia mandatorum percensuit atque quo modo celerrime machina iussa quaelibet exsequatur praescripsit. momentum ita ingens dedit scientiae toti computandi.

Partes vero nonnullas scientiae illius primus hic invenit. nam hoc primo monente collegaque adiuvante didicerunt machinae quo modo formulam quamque mathematicam optime resolvant atque vertant. mathematicorum etiam in penetralia ingressus est, rationem numerandi occultam licet in libro ἐξωτερικῷ perscrutatus qui multum de dialogis Platonis vel Georgi Berkeley refert. quam lepide puellam istam iuvenemque depinxit de legibus mathematicis rebusque infinitis colloquentes! mysteria sunt haec studia quae summa reverentia intueor, πρόσωθεν tamen ἀσπαζομαι.

Multa etiam arti typographicae contulit. rationem enim librorum mathematicorum machinis computatricibus faciendorum cui nomen est τεχ inventam retexit: o τέχνην egregiam!

In civitate California educatus Professor est in Universitatis praeclara Stanfordensi. fidem Lutheranam profitetur, organum pneumaticum feriatus modulatur. elegantiam summam in libris scribendis praestat nec non diligentiam, quippe qui historicorum modo rerum origines soleat attente inquirere. nonne hic aptus est qui origines rerum antiquas indaget cui nomen est antiquum regis nostri modestissimi?

Magnopere vero decet hoc anno quo scholares primi Oxonienses qui mathematicam scientiamque simul computatoriam feliciter excoluerunt ad gradum admittentur computatorum hunc maximum honorari.

Praesento vobis Donaldum Ervin Knuth, ut admittatur honoris causa ad gradum Doctoris in Scientia.

Reprinted, with permission, from the *Oxford University Gazette*, Friday, 24 June 1988.

Admission by the Chancellor

Computandi magister eminentissime, qui in arte numerandi mathematicos multa docuisti, ceteris beneficia innumerabilia contulisti, ego auctoritate mea et totius Universitatis admitto te ad gradum Doctoris in Scientia honoris causa.

For any who may prefer it, a paraphrase has been provided.

Presentation by the Public Orator to the Chancellor and Proctors

There is much academic talk in our laboratories and Senior Common Rooms about the computers we use for research. As Juvenal put it, *nobody knows his own house better* than we know the software we use to programme these machines,

Inserting orders in their entrails from without.

Such conversations are only gossip, mere swallows twittering in a concert hall. But they have as their foundation the elaborate science of computation which we must master to programme our computers correctly. I said a 'science' of computation, but our honorand would rather call it an art. For it is characteristic of an art to work out elegantly an ingenious program which inspires a pleasurable intellectual excitement in the programmer himself and in the critics who review his work.

So the honorand I now present should really be enrolled in the Faculty of Arts, even though he is admitted to the Degree of Doctor of Science. We should also welcome him as a philologist who has written with great clarity about the languages used by scholars in writing programs and about the analysis of the words used in computing. The *magnum opus* on which he is engaged is entitled *The Art of Computer Programming.* In the three volumes already published he reviews all kinds of programming and determines which programs will be most rapid in getting the right results from the computer. This book has had an enormous influence on computer science in general.

He has done pioneer work in parts of his subject. He and his collaborator Bendix were the first to devise term-rewriting systems for algebraic computation. He has even ventured into the inner shrine of mathematics, examining its hidden laws in a popular book, *Surreal Numbers*, which is reminiscent of the dialogues of Plato or Berkeley. How charmingly he depicts the young man and his girl discussing the laws of mathematics and the infinite! These studies are mysteries which I gaze on with respect but salute from afar.

He has also contributed to the art of typography, having invented a system for typesetting mathematical books by computer called 'TEX', an ingenious piece of technical detection.

He was educated in California and is Professor of Computer Science at Stanford University. He is a Lutheran and plays the organ in his spare time. His style of writing is elegant and precise, and he has a historian's knack of tracking down the origins of the things he describes. This is appropriate for one who bears the ancient name of Canute, most modest of our Kings.

It is particularly timely for us to honour a scholar who is eminent in computing this year when the first Oxford degrees in the Honour School of Mathematics and Computation will be awarded.

I present Professor Donald Knuth for the Honorary Degree of Doctor of Science.

Admission by the Chancellor

Leader of computing scientists, who have taught theory to the mathematicians and conferred widespread practical benefits, I, acting on my own authority and that of the whole University, admit you to the Honorary Degree of Doctor of Science.

Editor's note: Thanks to Joe Stoy of Balliol College, Oxford, for supplying the text of this citation in a TEX file, and to Charles Curran, of the Oxford University Computing Service, for providing a copy of the *Oxford University Gazette* in which it appeared.

TEX and TUG Go International — A Trip Report

Barbara Beeton

This summer, I had the pleasure of attending two major TEX meetings — "TEXeter" and the annual TUG meeting, in Montréal. Both (particularly Exeter) were notable for the number of new faces that could be attached to names, and the quality and content of the technical programs made it abundantly clear that TEX is no longer just a computer hobbyist's playground (if it ever was).

TEX88, Exeter

Exeter is an old town in the English midlands, on the Exe river. The university was founded in this century, on the grounds of an old estate that is also a botanical garden and arboretum. The conference was most competently arranged by Malcolm Clark and Cathy Booth, with help from Ewart North. A three-day program was surrounded by short courses and workshops.

Before the conference proper, I was kindly permitted to sit in on the second day of the session on document design. This was led by Paul Stiff, of the University of Reading. As we all have heard many times before, it was stressed that the real purpose of technical (and other) documents is communication, to provide a means by which an author's ideas can be communicated to a reader. Anything that gets in the way of that goal is thus poor design, whether or not the appearance of the document is attractive. (A pleasing appearance is desirable, but secondary.) Though there seems to be no "cookbook" that one can refer to, keeping in mind how a document is to be used should prompt its creator to do the "right" thing. And looking at many instances of similar documents, deciding which are most effective at their task of communication and why, is one of the best ways to develop a sense of appropriate design.

The conference program consisted of talks on various topics related to TEX and METAFONT, with a break on the second afternoon for an excursion on an old steam train and a cruise up the Dart River. Malcolm Clark presented a memorable harangue on how TEX users should make their presence better known in the composition world.

Several speakers presented their experiences providing TEX production services. (My favorite quote: "Academic publishers have to live day by day with the lunatic fringe — they are our authors!") Publishers are interested in lower costs, but without sacrificing quality. With some adustments (e.g., more traditional fonts), TEX is becoming accepted in this environment. One speaker offered this warning about working directly with authors — authors are often willing to accept the limitations of WYSIWYG word processors, but if they know that the back end of a system is TEX, they can and will choose to subvert any style filter provided for them.

Several papers were presented on experiments with METAFONT. Two authors spoke on extracting METAFONT's spline information for use with other graphic processors. Victor Ostromoukhov has developed a method for delivering the splines to PostScript, and his demonstration (on a Mac, in the evening) of letters wrapped around spheres and other "solid" objects was quite captivating.

Other topics covered by the talks included support for authors (usually, but not always, in academic environments), language-specific processing (including the use of non-latin scripts), graphics inclusion in TEX documents (including two papers on chemistry), TEX and databases, and a description of the Aston TEX archive. Space prevents inclusion of the full program in this issue, but the Proceedings will be published early next year — I am looking forward to reading them.

A topic of particular interest, though nowhere was it listed formally on the program, was how to deal effectively with A4 paper. TEX, and even more explicitly, LATEX, assume the use of $8\frac{1}{2} \times 11''$ paper; and output drivers assume that the reference point of a page (the top left corner) is one inch from the top and one inch from the left edge of the paper. These assumptions are not ideal for A4 paper (297mm × 210mm), and much discussion was devoted to how best to adjust both the dimensions specified in TEX macro files and LATEX style files and/or the output drivers' assumed reference point to compensate for the different dimension systems. However, one of the philosophical underpinnings of TEX is the ability to move documents from place to place with the assumption that they will get the same treatment and presentation. No good answer was found, but it seems clear that this is an area that could benefit from rethinking, as TEX is accepted in Europe and other areas of the world even more readily (if possible) than in North America.

In the evenings, there was plenty of time to discuss the day's events and other topics of mutual interest. Several personal computers were set up in the lounge of the residence hall, and experiments were encouraged. Chris Rowley and I were "fingered" to lead a clinic one evening; apparently, most of the attendees didn't have many problems, since only a few came to visit. There was, however, a request for an open problems session that couldn't be accommodated at Exeter, but should be seriously considered for inclusion at the next EuroTEX conference.

After the close of TEX88 proper, I attended another workshop, on the hackery of LATEX style files, led by Sue Brooks. Once again, the A4 controversy surfaced. When someone asked what was the reason for the "one inch" reference-point, I said that, to the best of my knowledge, it was

arbitrary, to define some standard to which output device drivers could be written.

TUG annual meeting, Montréal

Montréal is a beautiful city, with a cosmopolitan French flavor unique in North America. The meeting was held at McGill University, in a new high-rise building at the edge of the campus. The city surrounding was evident in many ways, not the least of which was the ubiquitous construction that seemed at times to be tunneling under the very foundations of the building where we were meeting. The program was put together by Dean Guenther (again), Christina Thiele and Shawn Farrell; Shawn also coordinated the local arrangements. As at Exeter, the main program was preceded and followed by short courses and workshops.

The evening before the meeting, almost everyone gathered at Le Festin du Gouverneur, an eating place set up in Montréal's old fort, where a feast and entertainment in the style of the 17th century French settlers were provided. (Picture yourself eating a several-course meal with only a knife between you and bad table manners.) As the TUG contingent was the largest of several groups present, the erstwhile Gouverneur was chosen from our ranks — none other than Bart Childs. The Master of the Feast saw to it that the serving wenches were most attentive. (It should be noted, though, that Bart was on hand the next morning in time to present the annual introduction for new members. A worthy performance.)

The general theme of the meeting was TEX in production environments. The variety of publications "produced" by TEX is truly astounding. (When I was first introduced to TEX it was still the preserve of computer science students and a few visionary mathematicians and physicists.) NASA technical reports, textbooks and computer reference manuals are natural applications for TEX; more surprising are the kennel club yearbooks and TV Guide (for which the first copies with feature pages prepared by TEX rolled off the press in May).

There seemed to be no common hardware or operating environment among the installations reported on, or even a common approach. What was common, however, were the reasons that TEX was selected, and the fact that most production sites have tried to integrate TEX into an existing operation. One speaker described her role as "managing a system of hardware, software and people". These features — an existing operation, comprising both skilled people and good resources — are characteristic of a production system.

Two areas in which it was perceived that TEX could be stronger are fonts and graphics. Ordinarily, only Computer Modern fonts are delivered with TEX. To install other fonts requires, at the very least, some effort; however, production users of composition services are simply accustomed to having a richer selection of fonts. With respect to graphics, the most available technique is still pasteup, whether physical or electronic (through the output driver). This was a design decision by Knuth, and an extension to TEX would be required to overcome the limitation.

Some sensible recommendations were made by the speakers:

- Management *must* coordinate all areas involved, and make them work together.
- To be successful, don't scrimp — make sure the hardware and software resources are adequate.
- User support is important. It isn't sufficient to hand *The TEXbook* to a prospective user. Training time is an investment that pays off.
- User training is best done in a language the users understand. When training a design staff, use "typesetter's terms".
- Users will be at different levels; a reasonable support level might be 1 guru : 5 macro hackers : n ordinary users.
- Use or build tools when appropriate. If something happens more than 5 times, automate it; if you build a tool more than 5 times, build a tool-builder.
- Macros should be designed for optimum data-entry use, as well as to produce the correct format.
- Remember that even TEX has limitations. Instead of simply trying to implement an old, unsuitable format, consider how a new approach might be better not only for TEX, but also for the product.
- For a first project, avoid one with a "drop-dead" deadline, if possible.

And several challenges were raised:

- To TEX developers, make TEX part of a *complete* publishing system, including graphics.
- TEX should be more cooperative about fine-tuning; a small change shouldn't lead to possible changes several pages later.
- Translation between other competent systems (*nroff*, etc.) and TEX should be investigated and implemented.

Proceedings of both TEX88 and the TUG meeting will be published. Both will be available from TUG early in 1989.

Some Typesetting Conventions

Graeme McKinstry,
University of Otago,
New Zealand.

One of the major advantages of TEX is that it makes it possible for authors to typeset their own work. However, this new found power has not been automatically associated with a knowledge of typesetting and typographic design and so some very *unreadable* documents have ensued. This is further exacerbated by authors believing they do know something about typesetting ("Doesn't everyone?") and ignoring all attempts to lead them in the right direction, e.g., LATEX.

Although TEX users are less prone to fall into this trap as compared to your average WYSIWYG user there are still some fundamental typographic lessons to be learned. These principles are so fundamental that even a computing consultant, such as myself, is able to learn, and possibly even more importantly, understand *why* we have them.

Readability not legibility

Legibility refers to whether it is *possible* to read a document. With the advent of cheap laser printers this is almost always attained. Readability, on the other hand, refers to how *comfortable* a document is to read. A document may therefore be legible (even very legible) but difficult to read. Typesetting aims to make a document more readable both by laying out the text so it is less wearisome on the eye and in providing clues as to how the document should be read (and therefore understood).

Also, there is much emphasis on *visual* design in our world of desk-top publishing:

> Most authors mistakenly believe that typographic design is primarily a question of aesthetics — if the document looks good from an artistic viewpoint, then it is well designed. However, documents are meant to be read, not hung in museums, so the primary function of design is to make the document easier to read, not prettier. *Leslie Lamport* [1]

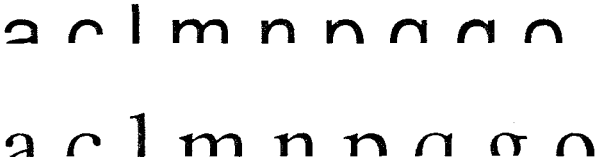
So it is apparent that the emphasis needs to be shifted away from making a document "prettier" to making it easier to read. Marshall Lee in *Bookmaking* [2] lists nine factors affecting the readability of the page:

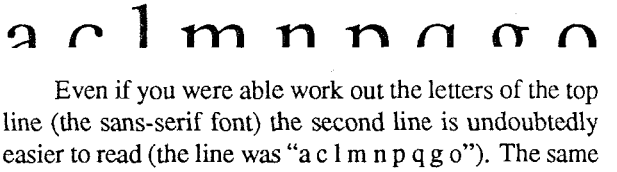
- typeface,
- size of type,
- length of line,
- leading,
- page pattern (which includes "margins"),
- contrast of type and paper (which includes colour),
- texture of paper,
- typographic relationships (heads, folios, etc.), and
- suitability to contents

Not all these factors are equal in their effect on readability nor are all the factors within your control but it is possible to use some of the above factors to make your documents more readable.

Typeface and size of type

There are two broad classes of fonts: serif ("serifs" are the finishing strokes at the end of letters) and sans-serif (without serifs, e.g., fonts such as Helvetica). Of the two, serif fonts (such as Computer Modern, and Times-Roman) are easier to read for large quantities of text, "because it has been shown that we read our own language not letter by letter but by recognizing the shapes of words ..." [3]. The serifs tend to help in this "shape recognition". For example try to decipher the following two lines (they don't form words):





Even if you were able work out the letters of the top line (the sans-serif font) the second line is undoubtedly easier to read (the line was "a c l m n p q g o"). The same test can be applied for upper- and lower-case letters — lower-case letters are found to be easier to read.

From this it is possible to establish two rules of typographic legibility for *continuous* reading:

- Sans-serif type is intrinsically less legible than seriffed type.
- Well designed roman upper- and lower-case type is easier to read than its variants, e.g., italic, bold, caps, expanded or condensed versions.

These rules are from *The Thames and Hudson Manual of Typography* [3]. Small doses of the variant fonts are used for *emphasis*.

For normal documents the body of the text should be set in a 10pt serif font. If your reader is particularly young (i.e., just learning how to read) or suffers from poor eyesight then the size of the type should be increased but probably to no greater than 12 pt. The kind of document also affects the type face and type size used but most of the documents can be handled well by a 10 pt serif font.

The length of the line

One of the fundamental errors is to make the the length of line far too long. This is done more out of habit rather than because of any fore-thought.

> Tests have shown many disadvantages in long lines: (a) the eye must blink at intervals during reading. Af-

One of the most discernible differences of type is their degree of masculinity or femininity. Some are definitely strong and rugged, some are definitely light and delicate, some are, of course, in between. Here, as in other areas of classification by character, there will be differences of opinion due to varying subjective reaction.

It is reasonably safe to say that almost everyone would find Caledonia, Times Roman, and Monticello masculine; Granjon, Weiss and Bodoni Book feminine; but even with borderline faces, a certain amount of the feeling conveyed depends on the way the type is used. [2]

Figure 1: Comparative text widths

> ter each blink, an optical adjustment and refocus of vision takes place. The longer the line, the more frequently blinks occur within, rather than at the end of the line; (b) there is the time and visual effort lost in travelling back to the beginning of the line; (c) when the measure is too wide, there is momentary difficulty in determining which *is* the next line (sometimes the wrong one is selected). Each interruption — the blink, the trip back, and the search for the right line — causes loss of reading efficiency, or poor readability. [2, page 92]

At the normal book-reading distance — about 40 cm — the maximum comfortable span of vision is about 12.7 cm. This suggests a *maximum* of 70 characters[1] per line in a page of average size. Fewer characters is better but any less than 50 tends to make it hard to set justified lines without excessive hyphenation of words and irregular word-spacing — both of which reduce readability. See figure 1 for a comparison of text widths.

Leading and space between words

The term *leading* is derived from the practice of inserting thin strips of lead between lines of type (and hence is pronounced "led-ing") to introduce "white-space" between the lines. For example, many books are set using a 10 pt font with 2 pt of leading, i.e., the baselines of two adjacent text lines are 12 pt apart. In text setting:

> Words should be set close to each other (about as far apart as the width of the letter "i"); and there should be more space between the lines than the words. [3]

If the gap between the words becomes too large it may be larger than the space between the lines thus tempting the eye to jump to the next line rather than the next word. For this reason, if the system you are using does not allow for easy hyphenation then it is best to set the text "ragged-right", i.e., without attempting to justify the text at the right-hand-side. This is not a problem with TEX.

In general, the larger the type size or the longer the length of the line the more leading is required. This is true up to a point:

> When there is too much space between the lines, there is a loss of efficiency (readability) because the reader expects to find the next line at the customary distance. His eye goes first to this point and then makes the adjustment. When the adjustment is small, the loss of efficiency is probably not significant. Where the leading is very large — say 8 pts. — the disturbance is probably considerable and may persist throughout the reading of the book. [2]

One-and-a-half or double spacing is therefore not desirable.

Small sizes, such as 8 and 9 pt, require proportionally more leading to compensate for their lower readability. If the line is short, however, then very little leading, if any, is required as the line becomes easier to read.

And there is more

The best idea is to talk to someone who knows about typesetting, or devour large quantities of books on the subject — or both. Until you have gained the requisite experience why not use LATEX — the styles are designed to take care of typesetting and typographic design for you. Don't let your document design degenerate into mere whims or "what looks good" but use design in a logical and consistent way to help your reader understand your document. Just as the *text* of your document should be purposeful so should the *design* and typesetting promote understanding in the reader.

There are obviously a lot more subjects to cover than I have addressed in this article. Topics such as treatment of headings, running headlines, hyphenation and justification, etc. are just begging to be addressed but unfortunately there is only a limited amount of space available in TUGBOAT. Maybe a column on typesetting and design should become a regular feature where experts can discuss the "dos" and "don'ts". I for one would be very interested in such a column.

Finally, there seems to be an eternal battle raging in the pages of TEXhax over whether paragraphs should be indented after headings. The Chicago Manual of Style would seem to support LATEX's suppression of indentation after headings [4, page 575]. This is because such

[1] Spaces are not counted as characters. Combinations of letters, e.g., the fi ligature count as one character.

suppression draws attention to the first paragraph after the heading which is supposed to be an important paragraph. The only argument I have read against indentation suppression is that it looks "ugly" (or even worse "UGLY") which only proves the point that, on the whole, people are woefully ignorant of the purpose of typesetting.

References

[1] Leslie Lamport. *LaTeX: A Document Preparation System.* Addison-Wesley, Reading, Massachusetts, 1986.

[2] Marshall Lee. *Bookmaking: The illustrated guide to design/production/editing.* R. R. Bowker Company, New York, second edition, 1979.

[3] Ruari McLean. *The Thames and Hudson Manual of Typography.* Thames and Hudson Ltd, London, 1980.

[4] University of Chicago Press. *The Chicago Manual of Style.* The University of Chicago Press, Chicago, thirteenth edition, 1982.

Software

Software-Ergonomics on the ST

Klaus Heidrich
Universität Göttingen

The Atari-ST is a typically mouse-directed machine. Most of the available programs are embedded into GEM (Graphics Environment Manager), which supports an easy and quick data-access. Pull-down-menus and interactive dialogue-boxes enable a self-evident software-handling, which often makes manuals superfluous. The spoiled user — confronted with the gigantic TEX-System — misses this comfort. Nevertheless, in my opinion there wouldn't be much sense in an interactive solution (see Leslie LAMPORT, *TUGboat* Vol. 9, No. 1, 1988). But as a good compromise, an interactive TEX-Shell was developed, which reduces mouse- and key-hacking to a minimum. The concentration of the user can be fixed on the important parts of the hacking-session.

How it works: The very special effect is to link the three-step-system `edit-TeX-DVI` together (`develop`), so that a two-step-system results. Upon leaving the editor, `TEX.TTP` and then `DVI.PRG` are called automatically. There is no need to wait until `TEX.TTP` is loaded (2 sec.) and to react at the prompt of the two asterisks, where I often made typing errors in former times. Naturally, all parts of the system can be called separately with only one click. The (mouse-)selected source-file and the respective format are saved in a current storage. The name of your own format-files (generated with `INITEX`) can be fed into a dialogue-box. Additionally the default values may be set in the environment-file. For that purpose there are three additional variables: `mytext`, `myformat`, and `myeditor`. Last but not least, the input-files are rarely immediately error-free (...). The error-menu of TEX offers the possibilities '`e=edit`' and '`x=exit`'. So long as this feature is not implemented in ST-TEX, you will be sent back to the editor by the `Shell`. This loop can be interrupted by an error-free *TEX-Run* or by pressing ⟨`CTRL-C`⟩.

Sample: The turn-around-time for a complete cycle `edit-TeX-DVI-edit` for a short text "`\nopagenumbers` This is TEX. `\bye`" is only 20 seconds, including the preview or the output on the printer and the recall of the editor with the source-file for further modifications.

Outlook: I have a running TEX-METAFONT-`Shell`, but up to now it is only adequate for my individual configuration. METAFONT is useful for avoiding the problems with missing fonts or with *overfull* hard disks. Recalling Don Knuth's words: METAFONT and TEX are designed to be "good friends and to live together for a long time" (*The METAFONTbook*).

Hyphenation Exception Log

Barbara Beeton

Below is a list of words that TEX fails to hyphenate properly. This is the annual update; the list last appeared in Volume 8, No. 3, starting on page 266. Everything listed there is repeated here.

The first column gives results from TEX's `\showhyphens{...}`; entries in the second column are suitable for inclusion in a `\hyphenation{...}` list.

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

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

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

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

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

English Hyphenation

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

In order to make TEX defer hyphenation until after three initial letters, some new patterns can be added, as communicated to me my Donald Knuth:

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

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

The List

`academy`	acad-e-my
`al-ge-brais-che`	al-ge-brai-sche
`anal-yse`	an-a-lyse
`anal-y-ses`	analy-ses *
`anomaly(ies)`	anom-aly(ies)
`an-tideriva-tive`	an-ti-deriv-a-tive
`anti-nomy(ies)`	an-tin-o-my(ies)
`an-tirev-o-lu-tion-ary`	an-ti-rev-o-lu-tion-ary
`ap-pendix`	ap-pen-dix
`asymp-totic`	as-ymp-tot-ic
`at-mo-sphere`	at-mos-phere
`at-tributed`	at-trib-uted
`au-toma-tisierter`	auto-mati-sier-ter
`ban-dleader`	band-leader
`base-li-neskip`	`\base-line-skip`
`Be-di-enung`	Be-die-nung
`be-haviour`	be-hav-iour
`bib-li-ographis-che`	bib-li-o-gra-phi-sche
`bid-if-fer-en-tial`	bi-dif-fer-en-tial
`biomath-e-mat-ics`	bio-math-e-mat-ics
`bornolog-i-cal`	bor-no-log-i-cal
`Brow-n-ian`	Brown-ian
`buz-zword`	buzz-word
`cartwheel`	cart-wheel
`cholesteric`	cho-les-teric
`Columbia`	Co-lum-bia
`congress`	con-gress
`Czechoslo-vakia`	Czecho-slo-va-kia
`database`	data-base
`dat-a-p-ath`	data-path

defini-tive	de-fin-i-tive
democratism	de-moc-ra-tism
de-mos	demos
dis-tribute	dis-trib-ute
Di-jk-stra	Dijk-stra
duopolist	du-op-o-list
duopoly	du-op-oly
dy-namis-che	dy-na-mi-sche
eco-nomics	eco-nom-ics
economist	econ-o-mist
elec-trome-chan-i-cal	electro-mechan-i-cal
elec-tromechanoa-cous-tic	electro-mechano-acoustic
En-glish	Eng-lish
equiv-ari-ant	equi-vari-ant
Eu-le-rian	Euler-ian
ex-traor-di-nary	ex-tra-or-di-nary
Febru-ary	Feb-ru-ary
fermions	fermi-ons
flowchart	flow-chart
Forschungsin-sti-tut	For-schungs-in-sti-tut
funk-t-sional	funk-tsional
Gaus-sian	Gauss-ian
ge-o-met-ric	geo-met-ric
gnomon	gno-mon
Greif-swald	Greifs-wald
Grothendieck	Grothen-dieck
Grundlehren	Grund-leh-ren
Hamil-to-nian	Hamil-ton-ian
heroes	he-roes
Her-mi-tian	Her-mit-ian
hex-adec-i-mal	hexa-dec-i-mal
holon-omy	ho-lo-no-my
ho-mo-th-etic	ho-mo-thetic
ide-als	ideals
id-i-o-syn-crasy	idio-syn-crasy
ig-nores-paces	ignore-spaces
in-finitely	in-fin-ite-ly
in-finites-i-mal	in-fin-i-tes-i-mal
in-fras-truc-ture	in-fra-struc-ture
in-ter-dis-ci-plinary	in-ter-dis-ci-pli-nary
Japanese	Japan-ese
jeremi-ads	je-re-mi-ads
Kadomt-sev	Kad-om-tsev
Karl-sruhe	Karls-ruhe
Ko-rteweg	Kor-te-weg
Leg-en-dre	Le-gendre
Le-ices-ter	Leices-ter
Lip-s-chitz(ian)	Lip-schitz(-ian)
macroe-co-nomics	macro-eco-nomics
Manch-ester	Man-ches-ter
manuscript	man-u-script
marginal	mar-gin-al
Marko-vian	Mar-kov-ian
Mas-sachusetts	Mass-a-chu-setts
met-a-lan-guage	meta-lan-guage
mi-croe-co-nomics	micro-eco-nomics
mi-crofiche	mi-cro-fiche
mis-ogamy	mi-sog-a-my
mod-elling	mod-el-ling
molecule	mol-e-cule
mo-noen-er-getic	mono-en-er-getic
monopole	mono-pole
monopoly	mo-nop-oly
monos-pline	mono-spline
monos-trofic	mono-strofic
mul-ti-pli-ca-ble	mul-ti-plic-able
mul-tiuser	multi-user (better with explicit hyphen)
ne-ofields	neo-fields
Noethe-rian	Noe-ther-ian
none-mer-gency	non-emer-gency
nonequiv-ari-ance	non-equi-vari-ance
noneu-clidean	non-euclid-ean
non-i-so-mor-phic	non-iso-mor-phic
nonpseu-do-com-pact	non-pseudo-com-pact
non-s-mooth	non-smooth
No-ord-wi-jk-er-hout	Noord-wijker-hout
oligopolist	oli-gop-o-list
oligopoly	oli-gop-oly
paradigm	par-a-digm
parabolic	par-a-bol-ic
parametrized	pa-ram-e-trized
paramil-i-tary	para-mil-i-tary
paramount	para-mount
petroleum	pe-tro-le-um
phe-nomenon	phe-nom-e-non
Poincare	Poin-care
polyene	poly-ene
poly-go-niza-tion	polyg-on-i-za-tion
poroe-las-tic	poro-elas-tic
postam-ble	post-am-ble
Po-ten-tial-gle-ichung	Po-ten-tial-glei-chung
pream-ble	pre-am-ble
preloaded	pre-loaded
pre-pro-ces-sor	pre-proces-sor
pre-s-plit-ting	\pre-split-ting
pro-cess	process
pseu-dod-if-fer-en-tial	pseu-do-dif-fer-en-tial
pseud-ofi-nite	pseu-do-fi-nite
pseud-ofinitely	pseu-do-fi-nite-ly
pseud-o-forces	pseu-do-forces
pseu-doword	pseu-do-word
quadrat-ics	qua-drat-ics
quadra-ture	quad-ra-ture
quasiequiv-a-lence	qua-si-equiv-a-lence
quasi-hy-ponor-mal	qua-si-hy-po-nor-mal
quasir-ad-i-cal	qua-si-rad-i-cal

quasiresid-ual	qua-si-resid-ual
qua-sis-mooth	qua-si-smooth
qua-sis-ta-tion-ary	qua-si-sta-tion-ary
qu-a-si-tri-an-gu-lar	qua-si-tri-an-gu-lar
re-ar-range-ment	re-arrange-ment
Rie-man-nian	Rie-mann-ian
righ-teous(ness)	right-eous(-ness)
schedul-ing	sched-ul-ing
schot-tis-che	schot-tische
Schrodinger	Schro-ding-er
Schwarzschild	Schwarz-schild
semidef-i-nite	semi-def-i-nite
semi-ho-mo-th-etic	semi-ho-mo-thet-ic
seroepi-demi-o-log-i-cal	sero-epi-de-mi-o-log-i-cal
ser-vomech-a-nism	ser-vo-mech-anism
setup	set-up
severely	se-vere-ly
solenoid	so-le-noid
spheroid	spher-oid
spinors	spin-ors
stan-dalone	stand-alone
startling	star-tling
statis-tics	sta-tis-tics
stochas-tic	sto-chas-tic
Stokess-che	Stokes-sche
summable	sum-ma-ble
tele-g-ra-pher	te-leg-ra-pher
tech-nis-che	tech-ni-sche
ther-moe-las-tic	ther-mo-elas-tic
times-tamp	time-stamp
ve-r-all-ge-mein-erte	ver-all-ge-mein-erte
Verteilun-gen	Ver-tei-lun-gen
vs-pace	\vspace
Wahrschein-lichkeit-s-the-o-rie	Wahr-schein-lich-keits-the-o-rie
waveg-uide	wave-guide
whitesided	white-sided
whites-pace	white-space
widespread	wide-spread
Winch-ester	Win-ches-ter
workhorse	work-horse
wraparound	wrap-around
Yingy-ong Shuxue Jisuan	Ying-yong Shu-xue Ji-suan

Fonts

LaTeX Fonts and Suggested Magnifications

Joachim Schrod
Technische Hochschule Darmstadt

In [1], BART CHILDS has presented several tables which contain fonts used by TeX and LaTeX. Such tables with all fonts required by a macro package are urgently needed in order to allow (TeX) system administrators or users to customize their fonts. But Bart Childs' tables miss some required fonts of LaTeX. Before I list all missing fonts I will give an overview about the way LaTeX does its font handling.

The data in this article is based on a UNIX tape from PIERRE MACKAY which was written in the beginning of June, 1988.

1. Font Handling of LaTeX

The font handling of LaTeX is described in the file `lfonts.tex` [4]. This file consists of four parts: First the principles of font usage are explained and commands are declared to realize these principles. Then all preloaded fonts are specified and the usage of the fonts is defined. Finally follows the definition of some LaTeX-specific symbols.

1.1. Principles

For LaTeX users several size-changing commands are available. According to the selected document style option they address different type sizes. Table 1 gives an overview about the used type sizes; it is taken from [5]. To realize the switch to the different type sizes, `lfonts.tex` contains a size changing (internal) command for each pt-size that is needed, e.g. `\xpt` for the switch to the 10pt-fonts and `\viiipt` for the 8pt-fonts. After giving one of these commands, the typeface change will be done in this size, e.g. `\xpt\bf` leads to the usage of font `cmbx10` and `\viiipt\bf` addresses font `cmbx8` (sic!).

In LaTeX fonts are grouped in three classes: (1) preloaded, (2) loaded-on-demand, and (3) unavailable. Please note that the expression 'preloaded' has a different meaning here than in the article of Bart Childs. There fonts are named 'preloaded' if they are provided in `plain.tex` with the control sequence `\preloaded`, i.e. the fonts of which the font metrics (from the `TFM` files) are loaded during an INITEX run and written to the `FMT` file, but which are not available directly for

SIZE	DEFAULT (10PT)	11PT	12PT
\tiny	5pt	6pt	6pt
\scriptsize	7pt	8pt	8pt
\footnotesize	8pt	9pt	10pt
\small	9pt	10pt	11pt
\normalsize	10pt	11pt	12pt
\large	12pt	12pt	14pt
\Large	14pt	14pt	17pt
\LARGE	17pt	17pt	20pt
\huge	20pt	20pt	25pt
\Huge	25pt	25pt	25pt

Table 1: Type Sizes

the user as control sequences (see [2, p. 350]). In this article 'preloaded' means all fonts of which the font metrics are loaded by INITEX. These are those fonts of which it is assumed that they are used often in many documents and where the TFM file should not be read every time again. The rest of the fonts LATEX uses are loaded at the time of the first usage.

From now on I call fonts of the class preloaded as P, loaded-on-demand as D, and unavailable as X.

All fonts of the class P are loaded in `lfonts.tex` with the command `\font`, with one line for each font. But 70% of the lines are commented out and serve only as indicators which other fonts could be preloaded this way.

The fonts of class D are loaded with the command `\@getfont` which also selects this font. These commands can be found in the third part of `lfonts.tex` where for each type size the fonts for the type faces are specified. E.g. the command

```
\def\pbf{\@getfont\pbf\bffam\@viiipt{cmbx8}}
```

in the definition of `\viiipt` means that the font `cmbx8` is to be loaded at the first usage.[1]

If a font is not available, i.e. is of class X, it will be substituted by another with the command `\@subfont`.

As the true font selection is done with the size changing commands like `\viiipt`, the simple change of fonts from class D to class P is not always successful. If, e.g., the font `\fivbf` is preloaded as `cmbx7 scaled 714` this doesn't prevent LATEX from loading the font `cmbx5` on demand. To achieve that a change of the definition of `\pbf` in `\vpt` would be necessary.

[1] This can happen, e.g., if some text in a footnote is typeset in bold face (for a standard document style in 10pt).

1.2. Actual Contents of `lfonts.tex`

The text fonts that are defined in `lfonts.tex` at the moment are listed in table 2 which is taken from [5].[2] All fonts which belong to class D were not listed by Bart Childs; a complete list can be found in section 1.4.

	\it	*\bf*	*\sl*	*\sf*	*\sc*	*\tt*
5pt	*X*	*D*	*X*	*X*	*X*	*X*
6pt	*X*	*D*	*X*	*X*	*X*	*X*
7pt	*P*	*D*	*X*	*X*	*X*	*X*
8pt	*P*	*D*	*D*	*D*	*D*	*D*
9pt	*P*	*P*	*D*	*D*	*D*	*P*
10pt	*P*	*P*	*P*	*P*	*D*	*P*
11pt	*P*	*P*	*P*	*P*	*D*	*P*
12pt	*P*	*P*	*P*	*P*	*D*	*P*
14pt	*D*	*P*	*D*	*D*	*D*	*D*
17pt	*D*	*P*	*D*	*D*	*D*	*D*
20pt	*D*	*D*	*D*	*D*	*D*	*D*
25pt	*X*	*D*	*X*	*X*	*X*	*X*

Table 2: Font Classes

For every installation a 'Local Guide' should be available (provided as a special version by the site coordinator!?) in which it can be looked up if `lfonts.tex` was changed so that additional fonts are used. (E.g. our Atari STTEX distribution contains no fonts of class X any more.)

Caveat: `lfonts.tex` contains inconsistencies in the definitions of boldface (`\pbf`) in 5pt resp. in 6pt, and in the definition of sans serif (`\psf`) in 17pt. The (outcommented) `\font` specifications are different from those which are loaded-on-demand. And `lfonts.tex` still contains a 'kludge': The font `amcsc10` (sic!) is used.

1.3. Desired Contents

`lfonts.tex` should be changed so that for the scaled `amcsc10` fonts corresponding `cmcsc10` fonts are used. And for all unavailable fonts corresponding scaled fonts could be provided. Of course it would be preferable to use fonts in the correct design size — the work of JOHN SAUTER is a step forward. But then `lfonts.tex` must be customized, too. There was a file on the UNIX tape which claims to be such a customized version, but this is only true for the fonts of class P. In class D they are

[2] Well, almost. In [5] `\it` in 5pt was classed as D which does not match `lfonts.tex`.

still loaded in different magnifications. (But this is described in `lfonts.tex` itself—if all else fails...)

1.4. Required Fonts

In addition to those specified by Bart Childs, LaTeX uses the following fonts (all of class *D*).[3]

unscaled: `cmbx5`, `cmbx6`, `cmbx8`, `cmsl8`, `cmsl9`, `cmss8`, `cmss9`, `cmss17`, and `cmtt8`.

in `\magstep0` up to `\magstep2` (for bold math): `cmbsy10`, `cmmib10`, and `lasyb10`.

in `\magstep2` up to `\magstep4`: `cmsl10`, `cmti10`, and `cmtt10`.

in `\magstep2` and `\magstep4`: `cmss10`.

in `\magstep4` and `\magstep5`: `cmbx10`.

Caps and small caps: `cmcsc10` with scale factors `\magstep0` up to 4, `amcsc10 scaled 800` and 900.

2. Font Groups Revisited

Now I will summarize all changes in the tables of Bart Childs that result from section 1. The table numbers are those of [1].

2.1. Additional Magnifications

The additionally needed magsteps are listed in section 1.4.

2.2. A Missing Font

The font `cmbsy10` is missing in the tables. It belongs to table 2 ('LaTeX Fonts') and is needed in the magnifications `\magstep0` up to `\magstep2`.

2.3. Rearrangements

The three fonts `cmcsc10`, `cmss17`, and `cmtt8`[4] from table 4 ('Fonts for Emphasis') must be moved to table 2 ('LaTeX Fonts').

Eight of the 'definite candidates for saving disk space' from table 5 are urgently needed by LaTeX and belong therefore to table 2: `cmbx6`, `cmbx8`, `cmmib10`, `cmsl8`, `cmsl9`, `cmss8`, `cmss9`, and `lasyb10`.

The fonts `cmtex8` and `cmtex10` can be moved from table 4 to table 5. They are only needed by `WEB` for the presentation of the extended character set (in strings) and are generally not necessary for installations that don't use `WEB`.

2.4. Non-standard Fonts

The fonts `lasyb5`, ..., `lasyb9` are unknown to me. They do not exist on the UNIX tape and are not mentioned in `lfonts.tex`. So they should be removed from table 5 (which will leave 10 of the 21 fonts).

Additionally it must be mentioned that the fonts `flogo` and `sklogo` are rather new and do not yet exist in all installations.

A delivery should never contain fonts named `gray`. As Knuth writes in [3] on page 330, all gray fonts are device dependent. Therefore they should be called `grimagen` or something like that—different `TFM` files are needed, too. During installation the local system administrator can rename his 'default' device dependent gray font to `gray`.

3. Conclusion

This article presents the principles of font usage in LaTeX and describes changes that should be made in the tables of Bart Childs in [1]. But this will still only result in a minimal subset of delivered fonts; additional requirements may come from macro package independent applications. E.g., the fonts scaled `\magstep2` are often used to reduce the resulting document afterwards: thus a resolution of 432 dpi can be achieved on a 300 dpi printer. The scaling factor `\magstep4` is often used for the preparation of slides if SliTeX is not used.

Because of these and other reasons we deliver with our stTeX all fonts in all seven magnification steps from `\magstep0` up to `\magstep5` (except for the fonts of SliTeX). Additionally we have included reduced fonts to discard the class *X*. But they will be replaced by fonts in the correct design sizes soon: I fully agree with the statements of Pierre MacKay and Bart Childs about 'scaled fonts.'

References

[1] Bart Childs. TeXfonts and suggested magnifications. *TUGboat*, 9(2):129–130, 1988.

[2] Donald E. Knuth. *The TeXbook*, volume A of *Computers and Typesetting*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1986.

[3] Donald E. Knuth. *The METAFONTbook*, volume C of *Computers and Typesetting*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1986.

[4] Leslie Lamport. `lfonts.tex`. TeX Macro File, DEC SRC, 11 November 1986.

[5] Leslie Lamport. *Using LaTeX at SRC*. DEC SRC, 17 January 1987.

[3] The font `cmbx7` (class *D*) was already mentioned by Bart Childs in his table 1.

[4] `cmtt8` is listed in table 3, too.

Mode_def's please

Doug Henderson

This article is a follow-up to the plea I made at the 1988 TEX Users Group meeting in Montréal for people to provide me with METAFONT mode definition settings as they develop them. In order to help people who need to create special fonts or just use METAFONT to create the Computer Modern typefaces on new laser printers or typesetters, I am maintaining a list of these settings for the TEX Users Group. I do not, however, have every new laser printer or typesetter at my disposal, and even if I did, I would not have the time to test them all. I am simply acting as a repository for the information sent to me by other members of the TUG community. I also spend a good deal of time (via email) explaining how to create new `mode_def` settings for various devices that I cannot test myself. In order for my efforts to benefit the entire TUG community, I need to have everyone report their findings to me after they have created their fonts.

I will include all `mode_def` settings given to me in the `mode_def` article that is published from time to time. Here is a little background on mode definitions. A mode definition (or `mode_def`) is a body of statements that define various things that the METAFONT program needs to know in order to create fonts for a particular printer or typesetter. Below is a typical example of a `mode_def` for an Epson FX-80 dot matrix printer:

```
mode_def epson =  % Epson
 proofing:=0;
 fontmaking:=1;
 tracingtitles:=0;
 pixels_per_inch:=240;
 blacker:=0;
 fillin:=0;
 o_correction:=.2;
 aspect_ratio:=216/240;
enddef;
```

The Parameters

The first parameter, *proofing*, determines whether a proof sheet is being created. Proof sheets are useful when you are creating or viewing new characters. Each character is generally 4 to 6 inches square, and fits quite nicely on an 8.5 × 11 inch sheet of paper. To create a proof mode character, you should set the parameter *proofing* to a positive number; 1 will do. In the example above, the value of 0 indicates that we do not wish to create character proof sheets.

The next parameter, *fontmaking*, is set to 1 to indicate that we are creating a font for use with an output device. This causes a TFM (TEX Font Metrics) file to be created.

The *tracingtitles* parameter determines whether the character description, such as "The letter A", will be written to the screen or the log file. Higher resolution METAFONT runs sometimes take quite a while, and it's reassuring to see signs of life (like a title coming on the screen) during the process.

Next, we see the *pixels_per_inch* parameter, which sets (surprise) the number of pixels in an inch for this particular printer. This information is generally supplied by the manufacturer of your printer. Since an Epson FX-80 dot matrix printer has 240 dots per inch horizontally, the setting in the example is 240.

The *blacker* parameter determines how dark a METAFONT pen will stroke through a given character (since some pixels are "burned off" during the pixel rendering process). In general, the higher the resolution of the laser printer or typesetter, the less we need to correct for pixels being shaved off and, consequently, the lower the *blacker* value needs to be.

The *fillin* parameter determines how much "ink" needs to be taken out of the corners where two diagonal strokes meet in characters such as the V and M. Some filling in occurs here and makes certain characters appear darker than they should be. A positive *fillin* value will remove pixels where needed.

The *o_correction* determines how much the bottoms of characters such as O will overshoot the baseline. These three parameters, *blacker*, *fillin*, and *o_correction* are the ones that need to be experimented with to create a new mode definition for a laser printer or typesetter, since changing these values is what really tunes a font correctly for a specific printer.

The final parameter, *aspect_ratio*, corrects for nonsquare pixel ratios. The Epson FX-80, for instance, has 240 dots horizontally and 216 dots vertically. To let METAFONT know this, we supply the *aspect_ratio* setting of 216/240 (vertical/horizontal).

Font-Tuning Advice

Before I give my two cents worth of advice on font tuning, I would like to mention some other people who have done nice work of testing fonts with various engine types. In TUGboat Volume 8, Number 1 (pages 29–33), there is an excellent article entitled "Write-white printing engines and tuning fonts with

METAFONT" by Neenie Billawala. In the following TUGboat issue, Volume 8, Number 2 (pages 128–129) "Blacker Thoughts" by John Gourlay also has excellent advice on the specifics involved with testing fonts for a Xerox 2700 laser printer. John also has some interesting discoveries as to how the *blacker* parameter really works.

For my own first piece of advice, I strongly suggest you have at least version 1.3 of METAFONT, the most recent version being 1.5a. This is because Professor Knuth made some changes to the base files that specifically help write-white engine tuning. This is a good start.

To correctly tune a set of fonts for your printer, you should be altering the values to determine the best settings. Also, you should test your *blacker*, *fillin*, and *o_correction* settings on a number of different-sized fonts. I test with at least the fonts cmr5, cmr10, and cmr17 so I can get a feel for what these values do at a very small point size (cmr5=5pt), at the largest point size (cmr17=17.28pt) and, of course, at the point size that Computer Modern was patterned after, cmr10.

The model for what Computer Modern Roman fonts should look like (and what we should be striving for as font tuners) can be found in any one of the five-volume set, *Computers and Typesetting*, by Donald E. Knuth. I highly recommend Volume E, "Computer Modern Typefaces", for viewing the complete set of all 75 Computer Modern fonts. These fonts were all tuned for typesetter quality, and we now have high-resolution character sets against which to compare our samples and tests.

Some generalities

My experience with dot matrix printers has been that they usually don't need any extra black placed in the characters, since they come out nice and black anyway. The *fillin* value works well at 0, and the *o_correction* need not be set very high since the print is fairly dark and overshoots some by bleeding on the paper.

With Canon LBP-CX-based engines, which include the Corona LP 300 and the Apple LaserWriter Plus, I have found that a little black(er) was needed (corona = .3; laserwriter = .5;) to make good looking character sets. Some people argue that only one value should be used per printer engine type, but I disagree. Small differences are introduced by the companies, which tune these engines, and I believe they are not all the same. Interestingly, I have noticed in working with these similar type print engines that even from engine to engine and from toner cartridge to toner cartridge there is enough of a difference to warrant slightly different *blacker* settings.

On to new printers

Recently, I have had a rash of requests from people who have purchased new-generation laser printers, among them the HP LaserJet II, the DEC LPS 40, and other Ricoh 408X series or Canon LBP-SX-based engines. Unfortunately, I do not have access to any of these machines for testing.

So, to repeat myself, I would like to ask everyone to please turn in your findings for mode definitions for any new laser printers or phototypesetters you may be testing, and I will try and keep them in order for us in future issues of TUGboat.

Here are the various ways to send me these mode definitions:

bitnet address
DLATEX@CMSA.BERKELEY.EDU
physical mail address
Doug Henderson
Office of the President
Division of Library Automation
300 Lakeside Drive, Floor 8
Oakland, CA 94612-3550
direct hot line
(415) 987-0561

Further Faces

Dominik Wujastyk

Since 'The Many Faces of TEX' appeared in issue **9.2** of *TUGboat*,[1] information about METAfonts has continued to flow in, thankfully in somewhat diminished quantities. Since I wrote the last article I have moved from the USA to England, and although all the network services are available here too (at a price), in practice Janet is not as open a medium as the Internet, and I feel it is more likely now that I might miss news about fonts, especially if it appears in UseNet, to which I currently have no access. I particularly miss the astonishing power of the American brand of FTP. However, the vast TEX archive at Aston, maintained by Peter Abbott, is more comprehensive than any other I know of, and NIFTP and mail server access to this archive greatly compensates for the sense of network isolation. All the same, if you know of any METAfonts that I have not mentioned, I would be glad of the news.

For new subscribers to *TUGboat*, issue **9.2** contained a survey of the existing fonts known to be available for use with TEX, including non-roman scripts, other styles, and much besides. That issue also included METAFONT code and examples for a new punk METAfont called just PUNK, by Don Knuth.[2] Perhaps the most extraordinary part of that article is Don's description of how he coded the font, extremely rapidly, and with no reference at all to drawings for the letters from V to Z, producing the METAFONT code as fast as he could type! This demonstrates a completely new paradigm of typeface design and creation, which many graphic designers will find alien, but which evidently works for some people, at least for smallish projects such as the punk typeface.

People interested in METAFONT should be aware that in addition to past issues of *TUGboat*, TEXhax and UKTEX, Don Hosek's network magazine TEXMaG has regularly included information about fonts, and is required reading for METAFONTers. Volume 2, issue 5, in particular, was dedicated to non-English TEX, and discussed Icelandic, German, French, ancient Irish, and the problems of foreign language hyphenation. Earlier issue of TEXMaG have also contained technical descriptions of TEX font file formats, etc. For information on subscriptions or back issues, contact Don Hosek, network address: `DHosek@HMCVAX.Bitnet`.

Once again, I am beholden to those who provided the information I have merely marshalled below.

1 Music

UKTEX 1988, issues 28 and 29, carried an exchange of information recently, concerning the use of TEX to typeset music. This very challenging application has been receiving some attention, and the work to date was announced in the Music Research Digest. Thence it found its way to UKTEX. Phillip T. Conrad provided summaries of the work of Dunne and Jürgensen, that of Schofer and Steinbach, and of his own.

1.1 Dunne and Jürgensen

Dunne and Jürgensen conducted research at the University of Western Ontario; they defined the concept of i-marks and p-marks. I-marks are invariant marks, the kind of marks that can be put into a font and typeset easily with TEX. P-marks are parameterized marks whose shape and size varies according to certain parameters. The only p-marks TEX is capable of typesetting are the horizontal and vertical rules. Dunne and Jürgensen use PostScript to augment the capability of TEX with a special version of `dvi2ps`.

Shane Dunne wrote to Sebastian Rahtz in August 1988, about his work on music typesetting, and Sebastian reproduced his letter in UKTEX, issue 28. Shane said the following:

> I was working on music printing per se about a year ago, and developed a rudimentary music-setting prototype based on TEX and PostScript. This system knows nothing of the rules of music formatting; the user describes the desired graphic result directly to TEX, using its glue-setting capabilities to handle various problems of spacing. These days I have broadened my research focus to encompass all types of "specialized notations" — those which use a fixed repertoire of symbolic marks — including music and music-like notations, logic diagrams, schematic representations of all kinds (e.g., of automata), flow diagrams, etc. I think the existing "typesetting" paradigm can be extended to "marksetting", where a "mark" is any kind of symbol, not necessarily of fixed form like a text character, but possibly dependent on one or more parameters (e.g., 2 endpoints for a line segment). I

[1] 'The Many Faces of TEX: a Survey of Digital METAfonts', *TUGboat* **9.2** (1988), 131–151.

[2] 'A Punk Meta-font', *TUGboat* **9.2** (1988), 152–168.

> am trying to develop a design for a general-purpose marksetting system, which can be used as a software basis for any number of specialized formatting programs, i.e., a different front-end program for each class of notation.
>
> I am doing this research towards a Master's thesis in Computer Science, which I hope to complete before the end of December. In the meantime there is one technical report which discusses my earlier work on music-setting, including the prototype, and contains some early ideas about general mark- setting.[3] ...
>
> My approach was simpler [than that of Schofer and Steinbach]; I used PostScript's powerful graphic primitives to define parametrized procedures to create things like beams and slurs. Schofer and Steinbach's fonts simply include huge numbers of different beam and slur characters, at different slants, etc., to handle a reasonable set of cases. The results look very good.

Dunne has been somewhat overwhelmed by responses to this report on his work, which was in fact not really intended for general distribution. He followed up with a note which appeared in UKTEX 1988, issue 29, to the effect that he had received a number of e-mail requests for more information from around the world. While pleased at the level of interest in his research, he explained that, just at the moment, he is not in a good position to reply to these queries:

> My problem is that right now, I am at a critical stage of writing my Master's thesis on this topic, and I simply cannot afford the time to reply to all the letters I am receiving. Also, while I would love to send everyone a copy of my recent report, that will take time and money I don't have. I'm going to try and convince my University to take care of it.
>
> On a more positive note, my plans for the near future include finishing my thesis by this December, preparing a paper based on it for one of the computing journals (I'll announce which one when I know), and preparing a distribution version of my mark-setting prototype. (The prototype cannot be distributed as is, because it's written for an experimental programming system that only existed here, and is now obsolete. It won't take much effort to turn it into straight C code, though.)
>
> So while I appreciate the interest in my work, I just wanted to let the ... readers know that I'll be a bit of a hermit for the next few months, and that right now, I don't really have anything in the way of software to distribute.

[3] University of Western Ontario Technical Report 171.

Contact (or not!)

Shane Dunne,
Dept. of Computer Science,
University of Western Ontario,
London, Ontario, N6A 5B7
CANADA
Net: `shane@uwocsd.UWO.CA`,
`shane@UWOVAX.Bitnet`
UseNet from US: `...!{ihnp4!decvax|seismo}!{watmath|utzoo}!julian!uwocsd!shane`
UseNet from Europe: `...!mcvax!seismo!watmath!julian!uwocsd!shane`

1.2 Schofer and Steinbach

There is a thesis written by Angelika Schofer and Andrea Steinbach at the Institut für Angewandte Mathematik at the Rheinische Friedrich-Wilhelms-Universität at Bonn, entitled *Automatisierter Notensatz mit TEX.*

Schofer and Steinbach operated from the assumption that a font of beams and slurs is in fact feasible; they appear to have generated just such a font, and they use plain TEX alone. Their system appears to 'understand' some form of music-description language, and to apply music-setting rules automatically. The music is printed with TEX by means of the special fonts.

Availability

A copy of their work (in German) may be obtained for 25DM by writing to the Institut at:
Wegler Straße 6,
5300 Bonn,
Federal Republic of Germany.

1.3 Conrad

Phillip T. Conrad noted that he is currently (August 1988) finishing a Master's thesis, at West Virginia University in Morgantown WV, which presents a prototype system for typesetting music notation with TEX. He noted that:

> It would seem that the central obstacle to musical typesetting with TEX is the production of slurs (ties, phrase marks) and

slanted beams. In John Gourlay's cover story in the May 1986 cover story of *Communications of the ACM*, he submits that it is a fair assumption that no two beams or slurs are precisely identical, so it would not be feasible to produce a font of all the possible slurs or beams. I have operated from this premise, as have Dunne and Jürgensen. ...

My own approach builds on the previous work at the Ohio State University of Gourlay et al. announced in the *CACM* article mentioned above. I use the TEXtyl program of John Renner (OSU Tech Report OSU-CISRC-4/87-TR9) rather than PostScript to draw the beams and slurs. In theory this provides device independence; in practice, the following restrictions apply:

1) The target system must support METAFONT for generation of the vector fonts necessary to TEXtyl,

2) Three Pascal programs and one C program must be ported to the target system; at this time, the programs are written for only BSD Unix 4.3.

Availability

For copies of Conrad's thesis please contact him at the following address:
Phillip T. Conrad,
401–K E. 3rd Street,
Wilmington DE 19801-3964,
U. S. A.
Phone: (302)-652-3938

2 Hershey Fonts

Just as I had begun wondering about the Hershey fonts, Jim Seidman asked about them in TEXhax 1988, issue 70.

2.1 Guthery

Dean Guenther (`Guenther@WSUVM1.Bitnet`) answered in issue 73 that Scott Guthery, (Phone: 512-258-0785) has the Hershey fonts with TFMs for a nominal fee in the region of $20 or $30.

2.2 Kesner

A few weeks later, TEXhax 1988, issue 90, carried a letter from Oliver Kesner (9 October 1988), also in answer to Seidman's query, describing his own work converting Hershey fonts for use with TEX. I reproduce what he said:

> Hershey fonts for the IBM PC are available from SoftCraft, Inc., and from Austin Code Works. The SoftCraft set consists of four separate databases:
>
> - `HERSHEY.CHR`: 1594 characters,
> - `ORIENT.CHR`: 758 characters,
> - `PERSIAN.CHR`: 135 characters,
> - `HEBREW.CHR`: 49 characters.
>
> The `HERSHEY.CHR` database includes, besides several Roman typefaces, Greek, Russian, German Fraktur, and a variety of graphic symbols; the `ORIENT.CHR` database has Hiragana, Katakana, and 623 Kanji characters.
>
> The format of the SoftCraft Hershey databases is given in their *Font Editing: EFONT/CFONT User's Manual* on p. A5–2. Using this description, I wrote ... [a] Turbo Pascal 4.0 program to generate METAFONT source code from the Hershey plotter directives. ...
>
> The characters in the Austin Code Works Hershey database are numbered 1–4326, with gaps, for a total of 1,377 different alphabetic and graphic characters. The format is described in Norman M. Wolcott and Joseph Hilsenrath, *A Contribution to Computer Typesetting Techniques: Tables of Coordinates for Hershey's Repertory of Occidental Type Fonts and Graphic Symbols*, U.S. Department of Commerce, National Bureau of Standards, April 1976.

Oliver has provided the Turbo Pascal 4.0 source code of

`HERSHEY.PAS`: a program that generates METAFONT source code from a Hershey character database in SoftCraft format, and

`ACWtoSC.PAS`: a program that converts the Hershey font tables distributed by The Austin Code Works to the format expected by the Cfont program of SoftCraft, Inc. I.e., it converts the ACW Hershey database to SoftCraft format, from which `HERSHEY.PAS` can generate METAFONT.

He also provides a pair of example files, `ORIENT.LOG` (the output of `HERSHEY.PAS`) and `ORIENT.MF`, which contain a couple of Japanese Hershey characters, and a set of font parameters respectively. I ran these through METAFONT, and was able to print the characters without a hitch.

Terms of Availability

The above information was forwarded to TEXhax by Oliver's son, Jeff Kesner, who has an e-mail address

and is happy to act as postman. Contact him at: jok%gpu.utcs.toronto.edu@RELAY.CS.NET .

The Pascal source code (about 14k) is available by anonymous FTP from Score.Stanford.edu (and from the Aston archive) as file Kesner.txh in the directory <tex.texhax>.

3 Armenian

3.1 Karagueuzian

Emma Pease also informed me that there is a family of Armenian fonts (created with old METAFONT) "wandering around CSLI". Dikran Karagueuzian designed and created these fonts in 1983, modelling them on Knuth's old CMR fonts. This means that they look good in bilingual typesetting with CMR. In fact, there is also an Armenian TEX to go with these fonts. Its hyphenating algorithm, designed by John Hobby and Dikran Karagueuzian, is functional but, according to Dikran, not perfect, so that the user may have to fiddle with the typeset material at the end.

Contact

The Armenian family of fonts, as well as the Armenian TEX, is available to anyone who wishes to use them. Contact Dikran Karagueuzian (dikran@csli.stanford.edu).

4 Logic Diagrams

UKTEX 1988, issue 30, included a letter from David Osborne (cczdao@uk.ac.nott.cian) mentioning a font called milstd.mf created by Rick Simpson for drawing electrical symbols. He included the METAFONT source code for the font, and in the following issue of UKTEX, a small set of TEX macros were published, for making the logical symbols easier to use within plain TEX or LATEX. The font consists of the following 25 symbols: slanting line at 45-degree angle for marking busses in logic diagrams; AND, NAND, OR, NOR gates, facing to the right, down, left and up; buffer and inverter, each facing to the right, down, left and up.

5 Tamil

5.1 Arthanari

There has been no communication from Mr. Arthanari, and it looks very much as though the Ridgeway–Schiffman font holds the greatest promise of a usable Tamil font in the near future.

6 Telugu

The latest news from Mukkavilli Lakshmankumar and his wife Lakshmi about TeluguTEX is as follows:

> We have decided on the grid framework for the font. We also have thought about different global variables that could be used to control various features of the font. We have decided on a set of primitive curves. These are like subroutines that can be used by different letters. Some of these are coded in METAFONT. Only control points are determined. We are yet to determine the stroke thickness and pen angles at the control points. Since we are not imitating any font, it takes a lot of time trying to ensure that various curves are correct. Our approach is to make a rough sketch and then determine the control points. Then we iterate by changing control points until the curve is satisfactory. But the most important thing to keep in mind is that all the curves must be consistent and be able to blend harmoniously. We essentially have the framework for Telugu font ready. But a lot of coding remains. In our spare time we will work on it. Some issues still remain unresolved. We are not sure what we should do to support transliteration of Sanskrit in Telugu. That means adding a lot of subscripts, superscripts. ...
>
> Our file structure is basically like that in CMR. We have parameter, base driver, codes and program (right now empty) files.

7 JTEX

Emma Pease (emma@csli.stanford.edu) mailed me on July 5 1988 with a correction about the availability of JTEX: a Tops-20 version of JTEX no longer exists on Turing since the old Turing machine itself no longer exists.

8 Greek

8.1 Hamilton Kelly

First of all, I apologize heartily for getting Brian's name wrong. The correct surname is 'Hamilton Kelly', not just 'Kelly' (it's been that way since 1638!). I, of all people, should understand about unusual names!

My report about Brian's work on Greek METAFONT characters was written at second hand, and he was astonished (and I hope pleased) when he came across a description of his Greek METAFONT work in *TUGboat*. This has spurred him into polishing up what he has done, and resubmitting it to the TEX archive at Aston. Explaining the genesis of his work, Brian says:

> I wanted access to a Greek font to type my homework, since I am currently learning Modern Greek at an evening class (for general interest, and for holidays). What I did was to take the character definitions already used in the maths italic for the lowercase Greek letters, along with the uppercase ones, and the normal ROMANU which make up the Greek upper-case and put them into a new driver file; I then METAFONTed this with various parameter files such that they now had spacing defined, etc., for use as a normal textual font.

Brian completed this work before hearing of Silvio Levy's work at Princeton, but continues to use his own Greek since it does not require a DVI output program which can read fonts of 256 characters.[4]

Brian has now written a small macro package which eases considerably the selection of the Greek fonts; this is for use under LaTeX, and makes use of the `\@addfontinfo` macro to define the font changing commands such that they scale automatically with LaTeX's size-changing commands. He has also written a short paper, describing what he has generated, and his future directions. He has recently been refining the character programs, to make some of the letters more textual in appearance (for example, the alpha was very wide; fine for maths, but not in text).

Terms of Availability

Brian has sent both the above mentioned files to the Aston archive, together with the revised font files, where all the material will be freely available. An announcement giving details will have appeared in UKTeX by the time you read this.

9 Perso-Arabic

9.1 Goldberg

In early September Jacques Goldberg noted that there has been a lot of progress with the Arabic font. He hoped to have it out by the end of September, all being well. The font consists of:

1. A complete font of 29 characters each at 2 or 4 glyphs (position dependent, isolated, first, middle, last in word).
2. A preprocessor with customizable mapping of the basic 29 chars to a–z, A–Z areas of an ASCII keyboard, that will work out an intermediary file such that unmodified TeX plus a few macros does indeed correctly compose bidirectional texts. The preprocessor reassigns the correct glyph from the isolated form glyph depending on the position in word.

All this will continue to be made available free to the academic community.

No vowel marking scheduled at this time, but Jacques says he could do it if requested. It would be implemented just like the accents in normal TeX.

10 Georgia Tobin

10.1 Century Schoolbook: Liber

In *TUGboat* **9.2**, Georgia gave more information about the process of designing the Century Schoolbook typeface she has been working on.[5] The face is now christened Liber, and the article was printed in the new face, at a resolution of 300dpi.

11 Icelandic

11.1 Pind

In the 'Many Faces' article, I failed to mention that Jorgen Pind had written a full account of his work on Icelandic TeX which appeared in Don Hosek's TeXMaG, volume 2, issue 5.

12 Miscellaneous

12.1 Hosek Pica

Don Hosek (of "Output Devices" and TeXMaG fame) has created a CM Pica, using the METAFONT code of the typewriter style font, CMTT, of Computer Modern as his point of departure. CM Pica is more or less a 10cpi version of CMTT with heightened ascenders and x-height, similar to the Xerox1200 PICA font in appearance.

CM Pica is specially designed as a sop to University authorities, some publishers' editors and the like, who insist on having a typescript manuscript marked up in the traditional manner, i.e., with a squiggly line under bold characters, and underlining under characters which would be italicized in print.

To achieve this, Don created 'bold' and 'italic' fonts (`CMPICAB.MF`, `CMPICATI.MF`) in which each character includes an under-squiggle, or underline respectively. Don also modified the `ligtable` commands controlling begin and end quotes and also hyphens, so that `` ` ` `` and `' '` both become `"`, while

[4] Bill Kaster of Personal TeX Inc., has recently produced a version (1.00t) of their Hewlett Packard LaserJet driver, PTI LASER/HP, which copes beautifully with Silvio's large font. I understand from Bill that their Apple LaserWriter driver, PTI LASER/PS, already has this capability.

[5] 'Designing for Low-Res Devices', *TUGboat* **9.2** (1988), 126–128.

‘ and ’ both become ' (the single straight quote symbol at code '015 in CMTT). Similarly, on output, ‘--’ becomes ‘-’ and ‘---’ becomes ‘--’.

The upshot is that one can code up a document in normal TEX fashion and then, by setting the fonts to be Hosek's Pica, it will print out looking as though it had been typed, with 'italic' text underlined and 'bold' text under-squiggled, etc. Merely reset the fonts to CMR (or whatever) and it will be properly typeset with all the variety of fonts and refinement of punctuation of which TEX is capable.

12.2 Non-standard sizes of CM

John Sauter reported in *TUGboat* 7.3 (1986), 151–152, that he has re-parameterized CM so that any of the existing Computer Modern family may be created with any design size. For example, most of us, when requiring an 11pt CMR will use CMR10 at \magstep half. Apparently this is not satisfactory to the most discerning, and Sauter's algorithms permit one to generate a true CMR11 face. They go further, of course, and permit the generation of any of the CM faces in any (reasonable) point size. This is done by algorithms that interpolate or extrapolate from the values used by Knuth in the METAFONT parameter files for CM. If a standard value, such as 10pt, is chosen, then Sauter's algorithms will produce CM fonts identical to the standard ones. The TFM files for all sizes match exactly.

Don Hosek's TEXMaG, volume 2, number 4 gives further details of Sauter's work, and notes that some of the fonts may start looking bad at larger sizes, lacking inter-character space, and so on. Don has prepared a version of Sauter's work tailored for use on a PC, which is available from him. See the TEXMaG article for details.

Output Devices

TEX Output Devices

Don Hosek

The device tables on the following pages list all the TEX device drivers currently known to TUG. Some of the drivers indicated in the tables are considered proprietary. Most are not on the standard distribution tapes; those drivers which are on the distribution tapes are indicated in the listing of sources below. To obtain information regarding an interface, if it is supposed to be included in a standard distribution, first try the appropriate site coordinator or distributor; otherwise request information directly from the sites listed.

The codes used in the charts are interpreted below, with a person's name given for a site when that information could be obtained and verified. If a contact's name appears in the current TUG membership list, only a phone number or network address is given. If the contact is not a current TUG member, the full address and its source are shown. When information on the drivers is available, it is included below.

Screen previewers for multi-user computers are listed in the section entitled "Screen Previewers". If a source has been listed previously under "Sources", then a reference is made to that section for names of contacts.

Corrections, updates, and new information for the list are welcome; send them to Don Hosek, Bitnet Dhosek@Hmcvax (postal address, page 229).

Sources

ACC Advanced Computer Communications, Diane Cast, 720 Santa Barbara Street, Santa Barbara, CA 93101, 805-963-9431 (DECUS, May '85)

Adelaide Adelaide University, Australia

The programs listed under Adelaide have been submitted to the standard distributions for the appropriate computers. The PostScript driver permits inclusion of PostScript files in a TEX file. The driver is described in *TUGboat, Vol. 8, No. 1.*

AMS American Mathematical Society, Barbara Beeton, 401-272-9500 Arpanet: BNB@Math.AMS.com

Arbor ArborText, Inc., Bruce Baker, 313-996-3566, Arpanet: Bwb@Arbortext.Com

ArborText's software is proprietary and ranges in price from $150 to $3000. The drivers for PostScript printers, the HP LaserJet Plus, the QMS Lasergrafix, and Imagen printers are part of their DVILASER

OCLC OCLC, Thom Hickey, 6565 Frantz Road, Dublin, OH 43017, 616-764-6075

OSU1 Ohio State University, John M. Crawford, 614-292-1741, Bitnet: Ts0135@Ohstvma, Internet: Crawford-j@Ohio-state.Edu

OSU2 Ohio State University, Ms. Marty Marlatt, Department of Computer and Information Science, 2036 Neil Avenue, Columbus, OH 43210

The drivers are distributed on either ANSI or TOPS-20 DUMPER tapes, with hardcopy documentation. There is a $125 service charge (payable to Ohio State University) to cover postage, handling, photocopying, etc.

Philips Philips Kommunikations Industrie AG, TEKADE Fernmeldeanlagen, Attn. Dr. J. Lenzer, Thurn-und-Taxis-Str., D-8500 Nürnberg, Federal Republic Germany, +49 911 5262019

PPC Princeton Plasma Physics Lab, Charles Karney, Arpanet: Karney%PPC.MFENET@NMFECC.ARPA

Versatec output from TEXspool is produced via the NETPLOT program. TEXspool also produces output for the FR80 camera. Color and graphics primitives are supported through specials.

Procyon Procyon Informatics, Dublin, Ireland, John Roden, 353-1-791323

PTI Personal TEX, Inc., Lance Carnes, 415-388-8853

Graphics output is supported on Imagen, PostScript, and QMS printers.

Rad Eye Radical Eye Software, Tom Rokicki, Box 2081, Stanford, CA 94309, 415-326-5312

RTI Research Triangle Institute, Randy Buckland, Arpanet: rcb@rti.rti.org

The program is available in the comp.sources.misc archives on Arpanet and Usenet.

Saar Universität des Saarlandes, Saarbrücken, Federal Republic of Germany, Prof. Dr. Reinhard Wilhelm, uucp: wilhelm@sbsvax.UUCP

SARA Stichting Acad Rechenzentrum Amsterdam, Han Noot, Stichting Math Centrum, Tweede Boerhaavestraat 49, 1091 AL Amsterdam (see *TUGboat, Vol. 5, No. 1*)

Scan Scan Laser, England, John Escott, +1 638 0536

Sci Ap Science Applications, San Diego, CA, 619-458-2616

SEP Systemhaus für Elektronisches Publizieren, Robert Schöninger, Arndtstrasse 12, 5000 Köln, Federal Republic of Germany

DVIP400 uses PXL files. Landscape printing is supported in all versions and graphics inclusion in all but the IBM PC version. Source is available on request. Cost varies from 300–1848DM.

Stanford Stanford University

The Imagen driver from Stanford is present on most distributions as the file DVIIMP.WEB. It provides limited graphics ability.

Sun Sun, Inc.

Sydney University of Sydney, Alec Dunn, (02) 692 2014, ACSnet: alecd@facet.ee.su.oz

Talaris Talaris, Sam Hassabo, Talaris Systems, Inc., 6059 Cornerstone Court West, San Diego, CA 92121, 619-587-0787

All of the Talaris drivers support Tektronix graphics. Device-dependent special fonts are used for each device.

T A&M1 Texas A&M, Bart Childs, 409-845-5470, CSnet: Childs@TAMU

Graphics is supported on the Data General drivers for the Printronix, Toshiba, and Versatec on the Data General MV. On the TI PC, graphics is supported on the Printronix and Texas Instruments 855 printers. There are also previewers available for both the Data General and the TI.

T A&M2 Texas A&M, Ken Marsh, 409-845-4940, Bitnet: KMarsh@TAMNIL

T A&M3 Texas A&M, Norman Naugle, 409-845-3104

The QMS driver supports inclusion of QUIC graphics commands via specials as well as landscape printing.

T A&M4 Texas A&M, Thomas Reid, 409-845-8459, Bitnet: X066TR@TAMVM1

The TEXrox package includes a GF/PK/PXL to Xerox font converter (PXLrox2), and utility to build TFM files from licensed Xerox fonts (Xetrix). The programs are all written in C. Fonts not present on the Xerox printers can be printed as bitmaps on printers with the graphics handling option (GHO).

At present the TEXrox package is being distributed on a twelve-month trial basis; the trial is free for U.S. educational and government institutions, $100 for foreign or commercial institutions. Licensing agreements will be available when the trial offer expires.

TEXsys TEXsys, Joachim Schrod, Kranichweg 1, D-6074 Rödermark, Federal Republic Germany, +49 6074 1617

The LaserJet driver supports graphics inclusion in device dependent format. PK font files are used. This program is proprietary. Contact TEXsys for further information.

THD Technische Hochschule Darmstadt, Klaus Guntermann, Bitnet: XITIKGUN@DDATHD21

The program uses PK fonts. The Philips Elpho driver is not public domain. Contact Klaus Guntermann for information on obtaining the program.

Tools Tools GmbH Bonn, Edgar Fuß, Kessenicher Straße 108, D-5300 Bonn 1, Federal Republic of Germany

The Tools implementation of TEX and the drivers listed are described in *TUGboat, Vol. 8, No. 1.*

TRC Finl'd Technical Research Centre of Finland, Tor Lillqvist, +358 0 4566132, Bitnet: tml@fingate

UBC University of British Columbia, Afton Cayford, 604-228-3045

UCB University of California, Berkeley, Michael Harrison, Arpanet: vortex@berkeley.arpa

UCIrv1 University of California, Irvine, David Benjamin

UCIrv2 University of California, Irvine, Tim Morgan, Arpanet: Morgan@UCI.ARPA

U Del University of Delaware, Daniel Grim, 302-451-1990, Arpanet: grim@huey.udel.edu

The distribution includes a program to convert font files generated by METAFONT to Xerox font format.

U Ill University of Illinois, Dirk Grunwald, Arpanet: Grunwald@M.Cs.Uiuc.Edu

The previewers are available via anonymous FTP in the directory pub/iptex.tar.Z on a.cs.uiuc.edu.

U Köln Univ of Köln, Federal Republic of Germany, Jochen Roderburg, 0221-/478-5372, Bitnet: AO045@DkOrrzkO

U Mass University of Massachusetts, Amherst, Gary Wallace, 413-545-4296

U MD University of Maryland, Chris Torek, 301-454-7690, Arpanet: chris@mimsy.umd.edu

The UNIX Imagen driver is on the UNIX distribution tape. The drivers may be obtained via anonymous FTP from a.cs.uiuc.edu in the directory pub/iptex.tar.Z or from mimsy.umd.edu in the directory tex.

U Mich University of Michigan, Kari Gluski, 313-763-6069

UNI.C Aarhus University, Regional Computer Center, Denmark

URZ University of Heidelberg, Federal Republic of Germany, Joachim Lammarsch, Bitnet: Rz92@Dhdurdz1

U Shef University of Sheffield, England, Ewart North, (0742)-78555, ext. 4307

Utah University of Utah, Nelson H. F. Beebe, 801-581-5254, Arpanet: Beebe@Science.Utah.edu

All of the Beebe drivers are distributed together. They are available on IBM PC-DOS floppy disks (about 6), or 1600bpi 9-track tape in TOPS-10/20 BACKUP/DUMPER format, VAX/VMS BACKUP format, Unix tar format, and ANSI D-format. Send tape or disks for a copy; there is a $100 fee for this service.

The programs are available for anonymous FTP from SCIENCE.UTAH.EDU on the Internet; information is in the file PS:<ANONYMOUS>OOREADME.TXT. A VAX/VMS binary distribution is available for anonymous FTP (password guest) from CTRSCI.UTAH.EDU. OOREADME.TXT in the login directory gives details.

On JANET, the programs may be obtained from the directory aston.kirk::[public.texdvi210]. The drivers are available from Listserv on EARN to European Bitnet users. Send the command GET DRIVER FILELIST (in an interactive message, or as the first line of a mail message) to LISTSERV@DHDURZ1. Files are obtained with the command GET *filename filetype*. Graphics is supported only in the DVIALW (PostScript) driver.

U Wash1 University of Washington, Pierre MacKay, 206-543-6259, Arpanet: MacKay@June.CS.Washington.edu

The programs listed under U Wash1 are all on the standard UNIX distribution tape.

U Wash2 University of Washington, Jim Fox, 206-543-4320, Bitnet: fox7632@uwacdc

The QMS driver for the CDC Cyber was written under NOS 2.2 and supports graphics.

Vander Vanderbilt University, H. Denson Burnum, 615-322-2357

Wash St Washington State University, Dean Guenther, 509-335-0411, Bitnet: Guenther@Wsuvm1

Wash U Washington University, Stanley Sawyer, 314-889-6703

The IBM PC LN03 driver is a modified version of Flavio Rose's DVI2LN3. Graphics support is provided through inclusion of LN03 plotfiles and line drawing specials. All three PXL formats on the PC are supported. The program is available free of charge with the receipt of a blank disk and return mailer.

W'mann Weizmann Institute, Rehovot, Israel, Malka Cymbalista, 08-482443, Bitnet: Vumalki@Weizmann

Xercx Xerox, Margaret Nelligan, Xerox Printing Systems Division, 880 Apollo Street, El Segundo, CA 90245, 213-333-6058

XOrbit XOrbit, P.O. Box 1345, D-8172 Lenggries, Federal Republic Germany, +49 8042 8081

This driver supports graphics inclusion in device dependent format. PK font files are used. This program is proprietary. Contact XOrbit for further information.

Yale Yale University, Jerry Leichter, Arpanet: Leichter-jerry@Cs.Yale.Edu, Bitnet: Leichter@Yalevms

DVIDIS is available for anonymous FTP from Venus.Ycc.Yale.Edu. Log in as anonymous and do a CD [.DVIDIS]. That directory contains the three required files needed to run the previewer. The image *must* be transferred using BINARY mode.

Screen Previewers — Multi User Systems

- **Data General MV**

T A&M1

- **DEC-20**

OSU2 ASCII Output

Utah BBN Bitgraph terminal

- **HP9000/500**

Utah BBN Bitgraph terminal

▪ **IBM MVS**

GMD GDDM supported devices: IBM 3179, 3192, 3193, and 3279

Milan1 Tektronix 4014

▪ **IBM VM/CMS**

HMC Terminals connected through 7171 Protocol converters: Tektronix compatible, VT-640 compatible, GDDM driven IBM 3179 and 3279 terminals, GDDM driven Tektronix 816

DVIview may be obtained by sending $30 (to defray duplication costs), a blank tape, and a return mailer to Don Hosek. The program is still in the developmental stages, and enhancements will be made in the future. The program uses PK files.

Wash St GDDM driven IBM 3179 and 3279 terminals

Uses PXL files at 120dpi. Allows viewing of the page in eight parts normal size or three parts compressed.

W'mann IBM 3279, 3179-G

Previewing is provided by DVI82, the Weizmann driver for the Versatec plotter. The program uses PXL files.

▪ **UNIX**

Utah BBN Bitgraph

U Wash1 DMD5620

Uses GF, PK, or PXL files at 118dpi. `tpic` output is supported. The program consists of two parts: a program running on the host computer and another that is downloaded to the terminal.

▪ **VAX VMS**

Adelaide AED 512, ANSI-compatible, DEC ReGIS, DEC VT100, DEC VT220, Visual 500, 550

Uses PK or PXL files.

DECUS Tektronix 4014

Uses PK, GF, or PXL files.

INFN DEC ReGIS

Uses PXL files.

Talaris Talaris 7800

Utah BBN Bitgraph

Screen Previewers — Microcomputers and Workstations

▪ **Amiga**

Rad Eye

Uses PK files. Included with AmigaTEX.

▪ **Apollo**

Arbor

Uses GF, PK, and PXL files. Preview is available for $500.

U Ill X-11 Windows System

▪ **Atari ST**

TEXsys

Tools

▪ **Cadmus 9200**

U Köln

▪ **IBM PC**

Arbor, PTI EGA, MCGA, UGA, Hercules, Olivetti, Tecmar, Genius full page, ETAP Neftis, Toshiba 3100, AT&T 6300

Uses GF, PK, and PXL files as well as tuned PostScript fonts (the base set available with PostScript printers). Preview of integrated bit map graphics, font substitution, magnification on the fly, two-up display of pages, and searching for character strings are supported. Preview is available for $175.

Aurion, PTI EGA, CGA, VGA, Hercules Graphics Card, Wyse WY/700, Genius VHR Full Page Display, AT&T 6300

Uses fonts from the laser printer driver in PK or PXL format to display text. Magnification may be set on entry. Maxview is available for $125.

PTI

Uses fonts in GF, PK, or PXL format. On the fly magnification, on the fly inclusion of DVI files, font substitution, and 256 character fonts are supported. PTIVIEW is available for $149.

T A&M3 EGA, CGA, Hercules

The `cdvi` program is available for $175.

▪ **IBM PC/RT**

U Ill X-11 Windows

▪ **Integrated Solutions**

UCIrv1

Utah BBN Bitgraph

▪ **SUN**

Arbor

Uses GF, PK, and PXL files. Preview is available for $500.

UCB

UCIrv2

U Ill X-11 Windows, Sunview Window System

Uses GF, PK, and PXL files.

▪ **Vaxstation/Unix**

U Ill X-11 Windows

Uses GF, PK, and PXL files.

▪ **Vaxstation/VMS**

Arbor GPX(UIS)

Uses GF, PK, and PXL files.
Preview is available for $500.

INFN GPX(UIS)

Uses PXL files.

Philips GPX(UIS)

RTI GPX(UIS)

Uses PK files at 78, 94 and 112dpi. Written in ADA. Source is included.

Yale GPX(UIS)

Uses PK files at 300dpi.

Low-Resolution Printers on Multi-User Systems — Laser Xerographic, Electro-Erosion Printers

	Amdahl (MTS)	CDC Cyber	Data General MV	DEC-10	DEC-20	HP9000 500	IBM MVS	IBM VM/CMS	IBM VM/UTS	Prime	Siemens BS2000	Sym-bolics Lisp	UNIX	VAX VMS
Agfa P400							SEP	SEP			Saar		Saar SEP	SEP
Canon					Utah	Utah							Canon Utah	Utah
DEC LN03					Utah	Utah							Utah	DEC NLS Procyon Utah
Golden Laser 100					Utah	Utah							Utah	Utah
HP LaserJet Plus					Utah	T A&M2 Utah				OSU1			Arbor Utah	Arbor LasrPrt Utah
IBM 38xx, 4250, Sherpa							GMD1 URZ	GMD1 Wash St						
Imagen	Arbor UBC		T A&M1	Stanford Vander	Columb. Utah	Utah	Arbor	Arbor W'mann				MIT	Arbor U Md Utah	Arbor NLS Utah
Kyocera													MPAE	LasrPrt MPAE
PostScript printers					Utah	Arbor Utah		Arbor		OSU1		MIT	Arbor Carleton MIT Utah	Arbor DECUS Sydney Utah
QMS Lasergrafix	Arbor	U Wash2	T A&M1			T A&M2	Arbor GMD1	Arbor GMD1		OSU1 T A&M3	GMD1	MIT	Arbor MIT U Wash1	Arbor GA Tech T A&M3
Talaris														Talaris
Xerox Dover					CMU								Stanford	
Xerox 2700II		Bochum			OSU2 Xerox			ENS					Xerox	
Xerox 9700	Arbor U Mich						Arbor T A&M4	Arbor T A&M4	T A&M4				U Del	ACC Arbor T A&M4

Low-Resolution Printers on Multi-User Systems — Impact and Electrostatic Printers

	CDC Cyber	Cray	Data General MV	DEC-10	DEC-20	HP9000 500	IBM MVS	IBM VM	Prime	UNIX	VAX VMS
Apple ImageWriter					Utah	Utah				Utah	LSU Utah
DEC LA75, LP100					OSU2 Utah	Utah				Utah	Utah
Epson FX/RX					Utah	Utah				Utah	Utah
Facit 4542											INFN
Florida Data					MR						
MPI Sprinter					Utah	Utah				Utah	Utah
Okidata					Utah	Utah				Utah	Utah
Printronix			T A&M1		Utah	Utah				Utah	Utah
Toshiba			T A&M1		Utah	Utah				Utah	Procyon Utah
Varian											Sci Ap
Versatec	U Köln	PPC	T A&M1	GA Tech Vander	U Wash1		GMD1 U Milan2	W'mann	LLL	U Wash1	Caltech NLS

Low-Resolution Printers on Microcomputers and Workstations — Laser Xerographic, Electro-Erosion Printers

	Amiga	Apollo	Atari ST	HP1000	HP3000	HP9000 200	IBM PC	Integrated Solutions	Sun
Agfa P400							SEP		
Canon			Utah				Utah	Utah	Utah
Cordata LP300							PTI		
DEC LN03			Utah				Utah Wash U	Utah	Utah
Golden Laser 100			Utah				Utah	Utah	Utah
HP 2680				JDJW	PTI				
HP 2688A				JDJW		HP			
HP LaserJet Plus	Rad Eye	Arbor	TEXsys Tools	TRC Finl'd		MPAE	Arbor LasrPrt MPS,PTI Utah XOrbit	Utah	Utah
Imagen		Arbor OCLC	Utah				Arbor PTI Utah	Utah	Arbor Sun U Md Utah
Kyocera			LasrPrt				LasrPrt		
PostScript printers	Rad Eye	Arbor				Arbor	Arbor MPS PTI Utah	Utah	Arbor MIT Utah
QMS Kiss, Smartwriter	Rad Eye								
QMS Lasergrafix		Arbor Scan					Arbor PTI		Arbor MIT U Del
Xerox 9700		COS Scan							T A&M4

Low-Resolution Printers on Microcomputers and Workstations — Impact and Electrostatic Printers

	Amiga	Apollo	Atari ST	Cadmus 9200	HP1000	HP3000	IBM PC	Integrated Solutions	Sun
Apple ImageWriter	Rad Eye		Utah				MR Utah	Utah	Utah
Citizen 120-D	Rad Eye								
DEC LA75, LP100			Utah				Utah	Utah	Utah
Diablo						PTI			
Epson FX/RX	Rad Eye		TEXsys Tools Utah		JDJW	U Shef	Milan1 PTI U Shef Utah	Utah	Utah
Epson LQ	Rad Eye		TEXsys				PTI		
Fujitsu			TEXsys	U Köln					
GE 3000		COS							
HP DeskJet	Rad Eye								
MPI Sprinter			Utah				Utah	Utah	Utah
NEC	Rad Eye								
Okidata	Rad Eye		Utah				Utah	Utah	Utah
Printronix			Utah				T A&M1 Utah	Utah	Utah
Texas Instruments 855							T A&M1		
Toshiba			Utah				PTI Utah	Utah	Utah
Versatec									U Md

Typesetters

	Apollo	CDC Cyber	HP3000	IBM MVS	IBM PC	IBM VM/CMS	Siemens BS2000	Sperry 1100	Sun	UNIX	VAX VMS
Allied Linotype CRTronic											Procyon
Allied Linotype L100, L300P					PTI						
Allied Linotype L202					PTI						Procyon
Autologic APS-5, Micro-5	COS Scan				Arbor PTI				Arbor	Arbor	Arbor Interg'ph
Compugraphic 8400			U Shef		Arbor PTI						NLS
Compugraphic 8600		UNI.C			Arbor PTI	Wash St		U Wisc			NLS
Compugraphic 8800					Arbor						
Harris 7500										SARA	
Hell Digiset				GMD2			GMD2				

DVItoVDU 3.0 and PSPRINT 3.0

Peter Abbott & Andrew Trevorrow
Aston University, UK

TEX at Aston University

In order to provide at Aston University an integrated text processing and publishing service with TEX/LATEX as one of the prime elements it was necessary to make changes to the software in use. This article describes the latest versions of PSPRINT and DVItoVDU which are part of the service.

Before describing the changes to the software a brief description of the environment in which they are to be used is necessary. Aston's computing facilities available for text processing comprise the following elements:

- A DEC VAX computer system running VMS.
- PostScript printers (various Apple LaserWriters and a Linotronic 300), some connected to the University network but all with the potential to be connected.
- MS DOS PCs and Apple Macintoshes connected to the network.
- Various workstations including Apollo and Sun.
- A Microtek scanner connected to the network.
- A site licence for PCTEX.

The intention is to permit users to prepare material on the most suitable device available, including terminals on VAX processors and eventually UNIX systems, to proofread and produce draft copies on the nearest suitable output device and where high quality is required transmit the final copy to the Linotronic 300. Users are not constrained to use TEX/LATEX provided that their favourite software is capable of generating PostScript.

Current projects at Aston University will mean that within the next three years every workplace will be connected to the network thus allowing devices currently restricted to a small group of users or a department to be made more widely available.

Changes to DVItoVDU and PSPRINT

DVItoVDU is an interactive page previewer that drives a variety of commonly available terminals. PSPRINT is a PostScript driver that supports a variety of PostScript printers and can print a DVI file, a raw PostScript program, or an ordinary text file. Both programs run under VAX/VMS and are in the public domain. They are available for FTP or mail access from the TEX archive at Aston. (Elsewhere in this issue is an article by Peter Abbott which describes how to extract material from the archive.) All the necessary files are kept in:

```
[public.trevorrow.vms.dvitovdu]
[public.trevorrow.vms.psprint]
```

A recent TUGboat article (vol. 8, no. 1) described DVItoVDU 1.7 and PSPRINT 1.1 and suggested that further development was most unlikely. This prediction was obviously a little hasty. Significant changes have been carried out in the UK, initially at The Open University and more recently at Aston University.

DVItoVDU 3.0

Here are the most important changes to version 1.7:

- DVItoVDU can now handle PostScript fonts, assuming you have the necessary TFM files. There is a new `/tfm_directory` qualifier to specify the location of these files. So DVItoVDU can recognize a PostScript font, the TFM name must start with a particular string. The new `/psprefix` qualifier allows sites to specify this string (the default prefix is "`ps-`").
- Added `ZI` (Zoom In) and `ZO` (Zoom Out) commands. The former halves the current window dimensions and the latter doubles them. Although the same effects can be achieved by appropriate use of `H` and `V`, the new commands require less contemplation.
- Any `\special` commands on a page are now displayed by the `S` command rather than at the time the page is interpreted. This is much less annoying for those documents with lots of `\special` commands.
- The limitation requiring all PK/PXL files to contain a size substring of the same length has been removed. Sites that had to include a leading zero in some font names to overcome this limitation must now rename them (e.g., `$rename *.0635pk *.635pk`).
- Added `/hoffset` and `/voffset` qualifiers to allow shifting of page margins.

PSPRINT 3.0

Here are the most important changes to version 1.1:

- PSPRINT now supports the Linotronic 300 and DEC's PrintServer 40 as well as the Apple LaserWriter. A new `/device` qualifier indicates which type of printer to use. A separate command file is activated for each device.
- PSPRINT can now handle resident PostScript fonts. Like DVItoVDU, new `/tfm_directory` and `/psprefix` qualifiers have been added.

- There are a number of other new qualifiers: `/conserve_vm` can be used to conserve PostScript's virtual memory at the expense of downloading character bitmaps more often; `/queue` allows users to override the default queue; `/output` copies the PostScript code generated by PSPRINT into a given file rather than sending it to a printer; `/two` and `/wide` are variants of `/text` that print two "pages" (60 lines by 80/132 columns) on each sheet of paper; `/reverse` and `/noreverse` override the device-specific order in which pages are printed; `/increment` simplifies the printing of documents on both sides of the paper; `/hoffset` and `/voffset` allow margin shifting.
- Some new qualifiers are device-specific: `/size`, `/lowres` and `/cutmarks` for a Linotronic, and `/nobanner` and `/manualfeed` for a LaserWriter.
- As for DVItoVDU, the limitation requiring all font files to contain a size substring of the same length has been removed.
- Error messages now appear in the log file if PSPRINT is used in a batch job.
- A single temporary file is now sent to the print queue. This simplifies the PostScript code required for `/text` jobs and overcomes problems caused by print symbionts adding unwanted characters (such as formfeeds/linefeeds) between files in a multi-file print job.
- The PostScript prologue files used to start each job have been thoroughly revised (after reading Adobe's *PostScript Language Program Design*).

Unix versions of DVItoVDU and PSPRINT

Unix versions of DVItoVDU and PSPRINT are also available in the Aston archive. The files are kept in:

```
[public.trevorrow.pyramid.dvitovdu]
[public.trevorrow.pyramid.psprint]
```

The work was done on a Pyramid running OS/x in the Maths department at the University of Adelaide. Since they didn't have a Modula-2 compiler, both programs were translated into reasonably standard Pascal (plus a tiny bit of C to handle low-level terminal i/o). It shouldn't be too difficult to modify the code for another Unix machine.

Note that the Pyramid versions are based on DVItoVDU 1.7 and PSPRINT 1.1 for VAX/VMS and so are a little out-of-date. In particular, they do not support the use of PostScript fonts. Also, the documentation is nowhere near as comprehensive.

Additional facilities

Additional facilities have been created during the update to version 3.0. Here is a summary:

- SCREENVIEW (a modified version of Mark Damerell's CrudeType) reads a DVI file and creates an ordinary text file. Its primary use is the production of help screens and printed output from the one TEX source file, but it can also be used as a simple previewer.
- HEXIFY reads a binary PostScript file created by VersaScan and creates a new, editable file that can be sent to a PostScript printer's serial port using PSPRINT. (VersaScan runs on a Macintosh and can save a scanned image as PostScript, but the bitmap data contains 8-bit bytes. This file can be Kermited up to a VAX but cannot be sent to a PostScript printer's serial port as some control characters, such as CTRL-D, have a special meaning to the interpreter. HEXIFY replaces each byte of bitmap data with 2 hex digits.)
- A5BOOKLET reads a DVI file and creates two new DVI files that can be used to produce an A5 booklet suitable for folding and stapling. It is assumed the given DVI file has a page format suitable for A5 paper. The A5BOOKLET command uses Tom Rokicki's DVIDVI program to do the required pagination tricks.

These additional facilities can be found in the Aston archive in:

```
[public.trevorrow.vms.screenview]
[public.trevorrow.vms.hexify]
[public.trevorrow.vms.a5booklet]
```

Conclusion

The facilities described above are only the first steps in producing an integrated environment. Much work remains to be completed and the major outstanding items seen at present are:

- A house style for internal documentation.
- Standards for student work submission.
- Improved local (online?) help facilities and user documentation.
- An independent seamless interface for the user. E.g., a consistent interface for PSPRINT and DVItoVDU on VAX/VMS and Unix systems.
- The preview of PostScript code on screens.

It is hoped to be able to report on the development of these goals in a future issue.

Site Reports

A UK-Based TEX Mail Archive Server

Peter Abbott
Aston University UK

The computing facilities of the UK academic community are interlinked by JANET (Joint Academic NETwork), which is a private network based on X.25 protocols and which is administered on behalf of the community by the JNT (Joint Network Team). JANET links Universities, Polytechnics, Further Education centres, major research facilities and other related bodies. Gateways exist at various places on the network to give access to both public and private networks including PSS, EARN, Bitnet, Internet and UUCP, to name but a few.

Systems connected to JANET run the CBS (Colour Book Software), which provides a common set of facilities across a wide range of disparate computer systems. The major components of CBS are File Transfer (Blue Book), Interactive access (Green Book), Electronic mail (Grey Book) and to a lesser extent Job Transfer (Red Book). At Aston we hold an archive of TEX-related material for the benefit of our community and any other site that can gain access via the gateways. The archive currently (August 1988) contains approximately 200mbytes of data covering mainframes, minis, workstations and PCs with versions of TEX and LATEX for all these areas. The archive started life in the autumn of 1987, and since that date contributions have been received from wide and far, reflecting its value to the community.

Access from JANET sites to the archive is relatively simple and painless. The FTP facilities that are provided can be illustrated by a simple example. (I shall use the VAX/VMS notation but there are corresponding formats for UNIX, VM/CMS, NOS/VE, etc.). A user called `orinocco` is registered on a system with the name `uk.ac.wimbledon.common`. To extract files from the archive, `orinocco` signs on to his system and types the command `transfer`. The required parameters are input filename, output filename, remote username, remote username password. If we assume `orinocco` wishes to fetch the file `[public]000aston.readme` the sequence is

```
transfer
%_Input filename?
    uk.ac.aston.spock::[public]000aston.readme
%_Output filename? archive.list
%_Remote username? public
%_Remote username password? public

Transfer nnnnn has been queued.
```

Sometime later the file is available on the system at `uk.ac.wimbledon.common`. Failure to find or transfer the file and other error messages are notified via the normal VAX/VMS mail system.

Regrettably the FTP system is not totally machine-independent, and the notorious VAX/VMS file format `stream_lf` creates a number of difficulties for sites which are not running the VAX/VMS operating system. Similarly, the commercial world and many sites beyond the gateways do not have FTP software (and do not wish to implement it either).

Aston does offer a magnetic tape service to compensate for these problems but it is slow and time-consuming for all concerned, and unproductive when one considers the world-wide electronic network which is already in place.

This is not a new problem, and at Rochester a mail server was implemented to provide access to the LATEX Style collection. [Editor's note: The LATEX Style collection has moved to Clarkson, see page 294.] A similar mail service at Aston would open up the UK archive to a much wider community. I am pleased to say that such a mail service has been running on an experimental basis for some weeks now, and although not yet totally bug free does provide a useful service. Credit must be given to both Graham Toal, who put a mail server in place at Edinburgh for a limited period, and to Adrian Clarke, who is still developing the one that now runs at Aston. (Graham no longer recommends his server and refers all queries to the Aston system). Adrian is at the University of Essex, but undertakes the maintenance and development of the mail server (for which I continue to be grateful).

The Aston mail server is a batch job which runs on a VAX 8650 processor under the VAX/VMS operating system; eventually it will not be monitored, so errors will simply be filed in the normal 'black hole'. At this stage of the development cycle, sometimes even genuine mail succumbs, so if no reply is received after a suitable period you are recommended to try again. Log records are kept and common errors will be reported from time to time in UKTEX (the UK's equivalent to TEXhax). It is impossible to give estimates of the turnaround time

for any individual user; the server runs once per hour and the mail message are queued for transmission. The mail software makes a maximum of 30 attempts to send a message (10 at 10 minute intervals, 10 at 1 hourly intervals and 10 at 4 hourly intervals). This rather extended period is designed to overcome short-term network failures and for systems which are switched off for short periods of time or overnight. The cluster system at Aston is normally available 24 hours a day, seven days a week, with the occasional booked systems maintenance on a Wednesday morning and twice yearly maintenance checks by DEC.

Instructions on how to extract files from the archive are contained in a help file; this file is available by sending a mail message to

`texserver@uk.ac.aston.spock`

UK addresses on JANET are big-endian format and most users 'on the other side of a gateway' will need to specify it as `texserver@spock.aston.ac.uk`. The subject line in the incoming mail message is ignored, as is any text until a line starting with `---` (three minus or hyphen characters in columns 1 to 3); any text on that line is also ignored. The next line is the `name@return address` in UK format and the third line is the word `help` (in UPPER, lower or MiXeD case). For example:

```
--- (any text on this line is ignored)
name@address
help
```

The best rule to observe in quoting `name@address` is to use the format:

- JANET sites
  name@uk.ac.site.system
- Sites via earn-relay (Internet, Earn)
  name%little-endian%big-endian@earn-relay
- Sites via uk.ac.ukc (UUCP)
  name%little-endian%big-endian@uk.ac.ukc

Anyone who has problems getting mail back is welcome to send me (`abbottp@uk.ac.aston`) the message that they have tried; I will forward it to

`texserver@uk.ac.aston.spock`

with a copy to the originator showing the `name@address` format that is required. I do not guarantee to be able to solve every query but will do my best.

Atari ST Site Report

Klaus Guntermann
Technische Hochschule Darmstadt

Since our last report several changes have occurred. First we must say that the former distributor Kettler broke down in April 1988 (probably not because of their activities with TEX for the Atari ST). In the following months we had negotiations with several companies and since August 1988 there is a new distributor.

Furthermore the product has been updated a lot, the documentation has been extended (e.g. a local guide for LATEX is included now), and here we summarize the most important changes:

A new version of TEX is in distribution (currently 2.92). Now it supports search paths for input files, font files (`TFM`) and preloaded `FMT` files. The memory management has been redesigned to use all available memory (up to 65534 mem array elements).

The new preview driver for the monochrome monitor has the following enhancements:

- it is faster,
- it allows a two step reduction and back on the fly without disk access (for 4 : 1, 9 : 1 compression),
- it supports graphics inclusion for bitmap raster files,
  - the format of the files is simple and described in the documentation,
  - a conversion program for DEGAS pictures is included,
- the font search is customizable.

The new laser printer drivers (HP LaserJet+/ Series II or Kyocera) support graphics inclusion in both device dependent and "preview compatible" format (i.e. bitmap raster file).

The new integrating menu shell comes with the following features:

- it allows to select work file for all activities,
- it can call an editor, TEX, INITEX, the previewer, a printer driver or BIBTEX from pull down menu items, function keys, or soft keys on the screen,
- it is customizable
  - to call any editor,
  - to select the `FMT` file,
  - to specify paths for input files, `TFM` files and `FMT` files,
  - to predefine parameters and the initial working directory,

- the customized values may be saved and loaded, e.g. to switch between Plain and LaTeX or between different printer drivers.

An installation program for hard disk based versions is included.

New dot matrix drivers that support the "preview compatible" graphics inclusion are in preparation. These will run with less than 700 KB RAM available, even for a resolution of 360 dpi.

The new distributor will handle new requests as well as updates for former Kettler customers. Please direct inquiries for further information about stTeX to

TeXsys
Kranichweg 1, D-6074 Rödermark
Federal Republic of Germany
phone +49 6074 1617

Data General Site Report

Bart Childs

The distribution is stable. Most of the news is that out activities have been in making sure that we have the latest revision of all pieces of the system. We think we do.

My ineptitude with E-mail caused the printer charts to not be correct in the last issue. I hope this time I have successfully gotten the current information to Don Hosek.

The *environment* that gives a menu driven system has been rewritten from CLI macros into a WEB. This is much faster. We are planning on writing change files to make it available for UNIX and VMS. It has already been done for MS-DOS. This will help new users to begin using TeX quickly because all they have to do is remember `TeXe filename` to start it up, and select a number to edit, TeX, preview, or print. The menu is somewhat longer, because it also interfaces to several utilities and allows selection of other TeX files (and attendant macro packages), editor, printer, and switches.

There has been a lot of interest in the 64-bit TeX and I have been sending out the relevant (non-DG) changes frequently.

IBM VM/CMS Site Report

Dean Guenther
Washington State University

There are several changes and a few newcomers to the IBM VM/CMS distribution tape.

Thanks to Barbara Beeton, the most recent LaTeX (April 1988), AMS-TeX (version 1.1d), and Plain TeX (version 2.92) are now on the tape. Barbara also sent along the bug fixes for TeX and METAFONT, so I now have those two updated to versions 2.93 and 1.5 respectively.

Georg Bayer has updated his DVI3279 preview program. It no longer prints out its messages in German. Many thanks Georg!!

Eric Berg sent me the BibTeX .99 files. After quite a bit of work with Oren Patashnik, I finally got .99 up and running. It and its updated auxiliary files are all included.

Don Hosek has contributed several newcomers for the distribution tape. He has supplied a working version for PXTOPK, GFREAD, PKTYPE, and MFT. Besides those four, Don also modified GFTODVI to create the standard 1K blocked DVI instead of a 2K file; and he changed GFTOPXL so that it conforms to the standard 128 character convention for PXL version 1001 files.

I also updated Weave to version 2.9 (I can't remember who supplied the update) and I changed the default output filetype for GFTOPK from "GF" to "300GF". Oh yes, I also modified DVI2LIST to quit giving a disconcerting nonzero return code when it was including a page segment directly into IBM's Advanced Function Printing Data Stream (AFP/DS).

Chris Carruthers at the University of Ottawa sent me the Makeindex program. Developed at the University of Berkeley by Peehong Chen, it has been modified by Chris to run on Waterloo C on CMS. Chris has included a module for all those who would like to use Makeindex on CMS, but do not have Waterloo C.

Shashi Sathaye at the University of Kentucky has taken Nelson Beebe's drivers and added the code so that they will compile under Waterloo C. Mike Glendinning from the University of Manchester then took Shashi's mods and was able to get the Beebe drivers to compile under IBM C. I suspect both of these are now available through Nelson. Wayne Podaima, of the National Research Council in Canada, sent the PiCTeX macros, which I've also included on the tape.

MVS Site Report

Craig Platt
University of Manitoba

At the end of July, I sent the latest MVS TEX tape to Maria Code for distribution. It contains the following enhancements from the June, 1987 tape.

- TEX version 2.9
- METAFONT version 1.3
- BIBTEX version 0.99c
- GFtoPK, PKtype and MFT included.
- Dynamic file allocation.

The last item represents the biggest change, and refers to the way TEX file names are mapped to OS dataset names. MVS TEX users have always been hampered by the "DD name bottleneck", whereby the names of all files input or output by a job must be known prior to execution and pre-assigned to an 8-character DD name. This makes packages such as LATEX, that read and write lots of "auxiliary" files, cumbersome to use. The new release uses assembler routines developed by Richard Tilley and others at the University of Manitoba to allow dynamic (run-time) access to OS dataset names. Here is an example of how it works.

Suppose TEX sees the command `\input story` (or `\input story.tex`). Then several attempts are made to match this to the intended dataset. The first is to find a DD name, in this case, STORY. However, if the extension part of the name were other than .tex, then a combination name, formed from 5 characters of the "first name" and 3 characters of the extension would be used, as in the previous version of MVS TEX. For example `\input primes.contents` would be associated with DD name PRIMECON.

If no matching DD name is found, then TEX can search directly for a catalogued dataset with the name <prefix>.STORY.TEX, where <prefix> is a string supplied by the user in the PARM field of the program invocation. It will often be a user's logon id, but could be any legal OS dataset name prefix.

If this dataset is found, it is used. (Output files will be overwritten.) In case of a non-existent output file, a new file will be created, provided the user also supplies the name of a default output volume to put it on.

For input files, there is also a "library" option, which will look for a DD name of the form <ext>LIB, where <ext> comes from the first 5 characters of the extension part of the name. If found, this should point to a partitioned dataset which will be searched for a member with the "first name". In the case of story.tex, this means finding member STORY in the dataset allocated to DD name TEXLIB.

Another construction allows specification of a fully qualified OS dataset name in the TEX source. If a file name begins with a "sharp" character, '#', then the rest of the name is used without modification as a dataset name. For example, `\input #PLATT.STORY.TEX` will read the dataset PLATT.STORY.TEX regardless of what DD statements are supplied. This is useful if a user wants access to datasets with a different <prefix>, such as system supported macro libraries.

For partitioned datasets, there is yet another syntax. If the file name has the form aaa:bbb, then aaa specifies a partitioned dataset and bbb the member name. For example, to include a macro from a system library, a user might say

```
\input #sys1.macros.tex:today
```

which would correspond to the construction

```
DSN=SYS1.MACROS.TEX(TODAY)
```

in a DD statement.

I've used a similar scheme for METAFONT and BIBTEX, but the other programs in the package (e.g., TEXware, MFware) use just DD names, since (in most cases) they input or output only a fixed set of files.

So much for the good news. After sending off the tape, I noticed a small problem in the design parameters, which could affect LATEX users (sigh). When a text file is opened, e.g., by an `\openout` command, and no corresponding dataset exists, TEX creates a file with the default DCB parameters of RECFM=VB LRECL=256 BLKSIZE=6144. These values seemed reasonable, but I later discovered that it is quite easy to exceed the 256 character record length in a LATEX auxiliary file. For example, the command `\subsection{Running \TeX, \LaTeX and \AmSTeX.}` expands to a line of 446 characters in the .aux file! The result is that the expanded line gets split arbitrarily, usually in the middle of a control sequence, and you get a PASCAL/VS error message: "AMPX059E Text exceeds logical record length in file ... " in the SYSPRINT file.

I will try to fix this in the next release (perhaps even "by the time you read this"), by increasing the LRECL value to 512. In the meantime, for anyone who has already received the July tape, there are a couple of workarounds. One is to pre-allocate LATEX auxiliary files for each job before running TEX, using larger DCB values, avoiding the dynamic file creation. It is also possible to keep output lines short by judicious use of `\protect` to prevent

expansion of macros like \TeX. Finally, there is a quick fix that an installer can try. The assembler routines that perform the dynamic allocation are compiled separately into a PASCAL/VS segment called FILPROCS. There is a copy of the compiled segment in object format included on the tape as file number 36. It is possible to edit this file, changing the embedded string "LRECL=256" to "LRECL=512". The resulting object file can then be re-linked into the TEX load module with the IBM linkage editor (you need to REPLACE the csects GETMVSPA and GETDDN in the TEX module). This is admittedly a sketchy description, but if anyone needs help with it, I can send them detailed instructions.

In addition to enlarging LRECL, the next release should contain TEX 2.93 and METAFONT 1.5 (still awaiting trip/trap testing), but for future versions, I would like to try making the file parameters adjustable by the installer. For sites with PASCAL/VS, the WEB source can of course be edited and re-compiled, but this is not a solution for the many sites that don't. One suggestion is to provide a customization module in the form of an assembler subroutine. All MVS sites should then be able to edit and compile it, and re-link it into the TEX load module.

UnixTEX Site Report

Pierre A. MacKay

Since the last UnixTEX site report in January, 1988, many of the hopeful promises have been fulfilled. What was there called TEX-to-C has been renamed to the more comprehensive WEB-to-C, and compilation under this system is now the default. TEX, METAFONT, TEXware and BIBTEX are all supplied with WEB-to-C change files, and a good start has been made on MFware. The two conversion programs gftopk and pktogf are already done, as well as mft, gftype and pktype.

A few notes are in order on the way in which MFware is being approached. The basic WEBs are by this time perfectly stable, and since I am doing the translation myself, I have taken the liberty of making the changed programs more Unix-like and less reminiscent of TOPS20 or SAIL. Wherever possible, Unix command-line switches are used in place of the old dialog lines, and simple utilities are made to run silently by default. There is only one file of "extra" routines, mfwarext.c, which contains about the same lot of code as that used in TEXware. The test_access procedure is used to look for an input path in the appropriate environment variable in all cases, and the output file, if its name is automatically generated as in gftopk, is always deposited in the current working directory. The rather insignificant lot of output from gftopk and pktogf can be turned on by means of the command line switch -v; otherwise, these two programs run with no output to the screen at all. The gftype dialog has been replaced by the two command line switches -m for mnemonics, and -i for pixel image. The default is to produce neither of these voluminous outputs. The form of the gftype command for running the trap test is:

```
gftype -m -i trap.72270gf
```

It may be noted that the four programs so far discussed could be converted into true Unix style, by diverting "chatty" output to *stderr* and using *stdin* and *stdout* for the gf and pk files as appropriate. This has a sort of purist appeal, and would make it possible to run these programs in a pipe. I have been unable, however, to think of any scenario in which that would be useful, and it would eliminate the convenience of having the output file from the conversion programs supplied automatically with the desired filename. It is, of course, possible to force a non-standard output filename, the command line syntax for gftopk is:

```
gftopk [-v] gffile [pkfile]
```

The serious omission from all of this is gftodvi. If necessary, I will try to supply a change file for this program, but since Donald Knuth has announced his intention of rewriting the WEB, I would prefer not to spend much time on it.

In keeping with the attempt to suppress the use of pxl format, no attempt has been made, nor will any be made, to adapt pxl-related programs to WEB-to-C. The effort is better spent on making old pxl-reading programs read gf or pk format instead. Any successes in this line will be gratefully received and incorporated into the distribution. Remember, if you undertake to work on this problem, that gf fonts of 255 characters are becoming quite common.

Finally, a rather more serious confession about the change files for gftopk and pktogf. The WEBs for both programs supply a *preamble_comment* to replace the dated METAFONT comment in the original gf file. I have found it very useful to know the date of creation of any font, and am unwilling to lose this information, since I doubt that I am

alone in finding it valuable. I have therefore added to the change file some bits of code which insure that the original METAFONT comment, including the date and time, is passed through unchanged in place of the undated `gftopk` or `pktogf` comment.

The current versions of the principal programs on the distribution are TEX 2.93 (if you got the earliest copy of this by FTP from `SCORE.STANFORD.EDU`, get it again), METAFONT 1.5 (same caution) and BibTEX 0.99c. At the time of writing, WEB-to-C was at version 2.22. Tim Morgan's list of successes in the `README` for version 2.20 was (omitting the notes of detail):

Sun-3, SunOS 3.2, SunOS 3.4, 3.5, 4.0FCS
Sun-2, SunOS 3.2-4
Sun-4, SunOS 3.2-4, and SunOS 4.0FCS
Sequent Balance, Dynix 2.1.1
VAXen running 4.2, 4.3BSD, and Ultrix
Convex
Amdahl running UTS
Apollo, SR9.7 and SR10.0 (beta)
Ridge 32 running ROS 3.5 and C compiler version 2.1B
UNIXpc (aka 3b1 or PC7300) running System V version 3.51
MIPS R/1000, compiler version 1.21
Iris workstation
Celerity C1260, UNIX version 3.4.78
Pyramid 98x, running OSx64Q 4.0-870901, C Compiler CCOMP-4.0

To this list may be added MassComp, Tahoe, ELXSI, the Sun 386i and the Cray running Unicos. On most of these systems it really is possible to do the minimal editing of a file called `site.h`, and then type `make all`. I even managed, with a certain amount of arbitrary hacking, to get a full compilation on a VAX 11/780 running 4.1BSD.

One of the great advantages of WEB-to-C compilation is the ease with which special versions can be made up, both of altered and enhanced versions and of enlarged "gargantuan" TEX. A TEX with 200,000 "half-words" of box and general storage, with space for 9,500 macro names, and with other limits similarly expanded is available through the use of the `BIGTEX.PATCH` in the distribution. I have even tried a compilation with 1,000,000 "half-words" of general purpose memory, but that produced a 9-Megabyte core image, and was felt to be unneighborly. Big versions of TEX are genuine TEX, and are so convenient, especially for those who make heavy use of things like `pictex`, that we strongly advise everyone who can to switch to one.

There has been a good deal of mail about new versions of the undump program for preloaded TEX. I shall not repeat Ken Yap's analysis here, except to say that on most systems the C compilation of "gargantuan" TEX runs marginally faster than small TEX, for various reasons, and loads `fmt` files so fast that there is really little reason to use undump any longer. Here are the essential lines of a Bourne shell script that we use instead:

```
case ${3+toomany} in toomany )
 echo "Too many arguments!"
 echo "Usage: tex foo[.tex] [my[.fmt]]"
 echo "or latex foo[.tex]"
 echo "or slitex foo[.tex]"
 exit 1;;
esac
case $0 in
 */tex ) virtex '&'${2-plain} $1; exit;;
 tex ) virtex '&'${2-plain} $1; exit;;
 */latex ) virtex '&lplain' $1; exit;;
 latex ) virtex '&lplain' $1; exit;;
 */slitex ) virtex '&splain' $1; exit;;
 slitex ) virtex '&splain' $1; exit;;
esac
```

This script resides somewhere in the path for executable binaries, and is linked to the three names `tex`, `latex` and `slitex`. Notice that this approach gives you the opportunity to load your own private `fmt` file when you invoke the script under the name `tex`.

The modified program TEX-XET, for setting in two directions, can be compiled using `cxet.patch` and a first try at Antti Louko's multihyphen-TEX (Don Knuth has suggested "MulTEX" as a name for this) is also there.

There is lots more coming. I have received several new or upgraded graphics packages, which will make up a new `TeXgraphics` directory, and there seem to be new drivers ever other week. One of the most interesting is `crudetype` which answers the need for a readable line-printer output at sites where the use of bitmapping printers is rationed or excessively expensive. I have had some difficulty getting `crudetype` to pick up the `tfm` files it needs to calculate rough spacing, but once that is corrected, `crudetype` will be made a part of the distribution.

Finally, I take this opportunity to express my great debt to Elizabeth Tachikawa, whom many of you have met by telephone when you have called about the Unix TEX distribution. Without her management of all the administrative side of the distribution, it would simply be impossible. If the documentation for compilation and installation has

improved over the last year, and I think anyone in a position to compare would agree that it has, it is through her careful analysis of the problems that are phoned in, and through her constant review of every detail of the documentation and directory organization on the Unix TEX tapes.

Typesetting on Personal Computers

Recovering from a Hard-Disk Failure

Mitch Pfeffer and Alan Hoenig

I'm sure it's a corollary of Murphy's Law: The most precious part of your computer — the hard disk — is the part most prone to failure. Having gone through more that one Seagate hard disk in the past year, I decided to devise a strategy that would minimize my down time in the event of another hard-disk failure.

I realized the importance of backing up onto two different media (one of which should be removable) when a client lived through the following nightmare: He had been backing up his hard disk to floppies — but what he didn't know was that his floppy-disk drive was drifting out of alignment; immediately after writing a floppy, he could still read it back in on that same drive, so he suspected nothing. When his hard disk failed, and he tried to restore his system from the floppies, he found that his floppy-disk drive had now drifted still further out of alignment: not only couldn't he read his own floppies, but, because they were written with the heads out of alignment, nobody else could read them either. (Incidentally, it was a Priam hard disk that failed. Although Priams are considered to be highly-reliable drives, I've noticed that they fail in dusty environments.)

In addition to backing up to floppies, my solution is to use a pair of inexpensive ($250) hard disks in a system, and to copy just those files that have changed from my working hard disk to the backup hard disk every day, using DOS's `xcopy`. This only takes a few moments, and requires no fiddling with floppies. With this approach, all I need do to get back in operation if my main hard disk blows, is to shift two cables. This gives me an important advantage over a tape backup: If you use a tape backup and your hard disk blows, you can't run your system off the tape backup — you must first replace the drive, and then restore the contents of the tape to the new drive. Besides — at $250, the hard disk is cheaper than a tape drive, as well as faster and more convenient.

(Prices given in this article are dealer prices, which are often identical to mail-order prices.)

The drives I use are the Miniscribe 8438F: These are 30 Mb half-height RLL drives, with a moderately-fast 40 ms access time. (The type of work I do — programming, TEX'ing, and writing — doesn't benefit from a faster access time. TEX turned in the identical performance with this drive as with a $900 28 ms 60 Mb Priam.) I've been installing a pair of these drives in all the systems I've sold over the past several months, and not one has failed.

(I recently came across a different Miniscribe drive that looks even more attractive: the 3675. This is a 63 Mb, $275 drive, which has a 42 ms access time when formatted as two 30 Mb partitions (its normal access time is 61 ms). I hope to test this drive in future systems.)

To get the full 30 Mb out of the 8438F drive, you must use an RLL controller; normally, computers come with MFM controllers. RLL drives transfer information 50% faster than an MFM drive. I'm using Adaptec 2372A controller ($160); it features a 1-to-1 interleave, which means that an entire track can be read during a single rotation of the disk. The controller supports two $5\frac{1}{4}''$ floppy-disk drives, in addition to the two hard disks.

I also tested two other RLL controllers: the DTC-5287, and the Western Digital 1003-RA2.

The DTC controller performed well, but lacks the 1:1 interleave feature of the Adaptec; however, the DTC controller is rated for running in a system with a 12 MHz, 1 wait-state bus, while the Adaptec is rated for an 8 MHz, 1 wait-state bus. (The Adaptec rating is conservative: I've had no trouble running it in a 10 MHz, 0 wait-state system.) By the time you read this, DTC should be shipping their 7287 controller, which does support 1:1 interleaving.

My experience with the Western Digital RLL controller was dismal: The first two Western Digital controllers I received proved defective. When I finally got one that worked, I found that it took three times longer to read in files than the Adaptec.

Hardware Installation: The drives are shipped with their address-jumpers set to the lowest address position; this setting is for XT-class machines. For AT-class systems, you must move the jumper to the next-to-the-lowest position on both drives. The terminating resistor — a thin, colored, multi-pronged strip near the address-jumper — is removed from the drive that will be attached to the plug in the middle of wide cable. This drive will become the 'D:' drive.

Should the 'C:' drive (the drive at the end of the cable) blow, and you need to turn your 'D:' drive into your 'C:' drive, you'll need to insert the terminator resistor into the 'D:' drive, and move the two plugs from the 'C:' drive to 'D:' drive.

(To avoid having my clients insert the terminator into the 'D:' drive when the 'C:' drive blows, I'm considering creating a non-standard hard-disk control cable: it would be twisted going into the middle connector, and twisted again going into the connector at the end of the cable, effectively undoing the effect of the first twist. As always, the drive at the end of the cable is the only drive with the terminator; however, the drive at the end of the cable will now be the 'D:' drive. Should the 'C:' drive (attached to the connector in the middle of the cable) blow, the client need only move the plugs from the 'C:' drive to what was the 'D:' drive — the terminator is already in place.)

When plugging the power connectors into the drives, the connectors on the wires may ride up on the pins when you push the plug down into place. This would cause intermittent problems with the drives. To avoid this, I'd recommend that before you attempt to plug the connector to the drive, you first push each of the four leads down into the plug, and hold onto both the plug and the wires when pushing the plug into place. Support the printed-circuit board by placing your finger below the printed-circuit board when pushing the plug down. Once the plug is firmly in place, take a needle-nose pliers, or a hemostat, and push each of the four leads down onto the pins on the drive.

Software Installation: The Adaptec contains a built-in low-level format program, which is activated by typing 'g=c800:5' to the DOS debug program. As usual, this is followed by running fdisk on each drive. (For some reason, I found I had to do the low-level format twice on the 'D:' drive, before the fdisk would work properly.) Finally, do a 'format c:/s/v' and a 'format d:/s/v', to high-level format each drive; this also makes both drives bootable (you'll want to be able to boot off your 'D:' drive, should the 'C:' drive blow).

Operation: After installing all your software on the 'C:' drive, copy the entire contents of the 'C:' drive to the 'D:' drive by running the DOS 3.3 command 'xcopy c:\. d:\ /s/e/v'.

At the end of each day, you can copy just those files whose archive bit is 'on' (i.e., those files that have not yet been backed up to floppy — see below), by typing 'xcopy c:\. d:\ /a/s/e/v'. Or, to avoid copying unnecessary files, such as TEX's dvi and log files, you can run this batch file:

```
attrib -a c:\*.dvi /s
attrib -a c:\*.log /s
attrib -a c:\park!@#.cor /s
attrib -a c:\*.qex /s
attrib -a c:\*.qeb /s
xcopy c:\. d:\ /a/e/s/v
```

This clears the archive attribute of the superfluous files throughout the hard disk before doing the xcopy; the '/a' option tells xcopy to copy only those files whose archive bit is 'on'. (Microspell creates file with the qex extension after processing a tex file, and a file with a qeb extension results when you check a WEB file; park!@#.cor is the temporary file created by the Cordata driver.)

Note that if you delete a file from your 'C:' drive, its backup copy will still remain on the 'D:' drive; this would eventually lead to the 'D:' drive filling up with copies of deleted files. To solve this problem, I'm planning to toss together a program that deletes those files on 'D:' no longer found on 'C:', and invoke it at the start of the above batch file.

Fifth Generation Systems, makers of the Fastback backup program, just introduced a hardware card that automatically mirrors the two drives: any file that's saved or deleted from your working drive is automatically saved or deleted from the backup drive. If your working drive should fail, this card (call Counterpart) will sound an alarm and instantly switch you over to your back-up disk. This arrangement can be essential in many environments, such as on-line order entry. Fifth Generation supplied me with an evaluation unit, and I hope to have a report on this card in a future issue.

Note that the Miniscribe 8438F and the 3675 both lack an auto head-park feature, so before moving your system, be sure to park the heads. I use the park program included in the PC-Tools utility package. I found that the park program that comes with Disk Manager did not work with

the Adaptec, although it did work with the DTC controller.

Backing up to a removable device

The other component of my backup regime consists of backing up to floppies, using Fastback-Plus. Every month, I do a full backup of my entire hard disk to floppies; in doing so, Fastback turns off the archive bit of all the files. Every other day (or so), I do a differential backup, where Fastback copies just those files that have their archive bits 'on' — that is, just those files that have been changed or added since the last full backup; these are the same files that are copied to the `D:` drive by the batch file given above.

Fastback-Plus does have its problems: version 1.00 was unable to restore two out of the fifteen floppies it had created in backing up my hard disk (I did the backup with Fastback's read-after-writing verification turned on — see below). I suspect that the problem was caused by an imperfection in the way version 1.00 formatted new disks as it backed up. (If there's one program that had better be flawless, it's your backup program.)

Fastback also had trouble restoring files it had placed on a Bernoulli cartridge. If you have a Bernoulli box, I suggest you partition your hard disk into 20 Mb partitions, and backup the hard disk by `xcopy`'ing each partition to individual cartridges. This has the added advantage of not having to de-Fastback the files in order to use them.

I also noticed that if I formatted 360 K diskettes in my 1.2 Mb drive after running Fastback's installation routine, an inordinately high number of bad sectors were unjustly locked out. This problem went away after I reset the computer.

One of the nicer features of Fastback is that it allows you to exclude file and directories from the backup. I've set up my copy so that it excludes: the operating system kernel (`\command.com`, `\ibmbio.com`, and `\ibmdos.com` under PC-DOS); all the DOS programs (kept in `\DOS` on my system); and `*.dvi`, `*.log`, `*.qeb`, `*.qex`, and `park!@#.cor`. (Should my hard disk fail, I'll need to restore DOS and Fastback from their distribution diskettes anyway, in order to run Fastback to restore the other files.)

I run Fastback with 'write-verify' on, 'compression' set to 'save disks', and 'error correction' on. On my 386 system, Fastback takes about a minute per megabyte with these settings. The 'write-verify' option sounds like it offers more security than it really does: Fastback does not try to read back the information it wrote out to the floppy — all it does is compare what it read off your hard disk with the copy of that information it has in RAM.

If you use a hard-disk cache, make sure to turn it off before running a backup program — it will defeat the verification attempted by the program. (Personally, I don't bother when doing a differential backup, but before doing a full backup, I replace my `autoexec.bat` and `config.sys` files with the simplest possible versions and re-boot, to avoid any detrimental interactions that might occur between the backup program and, say, a resident program.)

For greater security, I alternate between two sets of floppies for both the full backups and the differential backups. To keep the differential sets straight, I move a Post-It marked 'Use Next' between the sets. Before doing a full backup, I put the last differential at the back of the box containing the most recent full backup, and then over-write the box containing the oldest full backup. This system not only allows easy recovery, but also allows me to dig up an early copy of a file if I find out that I've accidentally trashed a file, and backed up the trashed file.

The current version of Fastback-Plus is Version 2. It includes a separate verify feature: after the entire backup is complete, you run this option, and re-insert every diskette; Fastback will compare each file on the hard disk to the copy on your floppies (unfortunately, this still doesn't guard against an out-of-alignment floppy-disk drive). It also can be set to automatically delete unwanted history files.

Evaluation of K-Talk

C.G. van der Laan and J.R. Luyten
Rijksuniversiteit Groningen

We would like to announce the availability of a report entitled "Evaluation of K-Talk", RC-report 22, Groningen, 1988. Further information can be obtained from the authors. The Foreword of the report is reproduced below.

At the Rijksuniversiteit Groningen document preparation is done by text processors and document preparation systems 'at the desk', with possibly remote 'execution' and printing.

At the moment WordPerfect and TEX as representation of respectively text processors and document preparation systems enjoy the highest 'support category' — they are standards for the time being.

In practice most users start with WordPerfect and sometimes end with TEX.

Intermigration tools between these systems are therefore useful. K-Talk is a program that translates WordPerfect files into TEX files. This report aims to provide an answer to the question 'Is K-Talk a good tool?' Apart from 'the answer' — if any black and white answer is possible — we constructed a test collection of judiciously chosen document elements such that comparison can be made easily with future releases of K-Talk or similar products.

Furthermore is must be noted that translation of a WordPerfect document of only a few pages will produce a TEX file, preceded by a dozen or so pages of macros; moreover a macro library of considerable size is used. The average TEX user will usually not understand those macros. Adaptation of the produced document is not always simple. We also note that processing of the translated `.tex` file by e.g. LATEX or AMS-TEX is by no means trivial.

Training

Making Paragraphs

Alan Wittbecker

TEX, as perhaps you already know, is a typesetting program for the production of beautiful pages. Using TEX to produce beautiful pages is easy if you let TEX make the design decisions already built into the program. TEX is easier to understand, however, if you have an appreciation of the history of typography and book production, not to mention computer programming.

This article, as the first of a series, presents TEX in the simplest way — as a set of instructions for typesetting text. (The tutorial on page 276, by contrast, concentrates on TEX as a computer program.) TEX instructions are added to a file, called the source file, containing text. The text has been entered with words separated by spaces and groups of words separated by punctuation and blank lines, which represent the boundaries of phrases and paragraphs.

TEX instructions describe the procedure that TEX follows when it formats the text. The instructions and the text are entered using keyboard characters (ASCII characters), so that the file can be transferred easily to other computers. TEX formats a source file to a device-independent file (DVI file) that can be printed (after going through a DVI translator and output driver) on dot-matrix printers, laser printers, or typesetters.

A TEX instruction requires a special character, the backslash (\), to be recognized as an instruction. The body of the instruction is composed of alphabetic characters, usually a word; each instruction, in general, is ended with a space. Instructions are simply entered in the text.

Instructions are gobbled up — it is permissible to anthropomorphize TEX — when the text is formatted. Characters are read and printed as characters, unless they are special, like the backslash or percent sign (%), which is the comment character; but, even these can be printed with a specific instruction. Extra blanks between words are discarded because TEX calculates an optimum interword space — therefore, you can use blanks to arrange the source file. The carriage return from the terminal is converted to a blank.

TEX Makes Paragraphs

TEX considers the paragraph the basic unit of production. A paragraph is a short composition consisting of a group of sentences. The clear separation of paragraphs can distinguish thoughts, clarify content, and increase comprehension. A paragraph is commonly indicated by starting a new line and indenting that line. Sometimes space between paragraphs is also used to distinguish them.

After TEX reads text into its memory — by the mouthful! — paragraphs are examined for goodness according to a mathematical standard of beauty (based actually on calculations of "badness"), and then separated into lines.

TEX recognizes the beginning of a paragraph by reading an alphabetic or numeric character. The paragraph instruction itself (`\par`), however, comes at the end of a paragraph, instead of the beginning. All other TEX instructions precede the text they describe. A blank line also causes the end of a paragraph; actually, two consecutive keyboard carriage returns are translated as a `\par` instruction. Thus, the simplest source file only needs blank lines separating paragraphs. Figure 1 lists a source file. Notice the use of the percent sign to comment out information — nothing that follows a percent sign on a source line is even read.

```
%     figure 1. paragraphs
\TeX\ is part of a long tradition of
```

```
putting printed words on a page.
Many conventions of book and journal
production have roots in Mesopotamia
and Egypt, Greece and Italy.

Egyptian papyrus scrolls were written in
hieroglyphics in vertical columns
separated by thin black lines; sometimes
illustrations accompanied the text along
the top or bottom of the scroll, marked
off from text by double ruled lines.
As illustration assumed more importance,
it was placed within the text.
This format persisted through Greek and
Roman manuscripts to medieval and modern
books.

Although Greek scrolls were also written
in columns, characters were presented
continuously, in capitals, without breaks
between words.
Punctuation was usually nonexistent.
Breaks in thought were sometimes indicated
by an underlining stroke (known as a
paragraphos) or by a small blank space.
```

Figure 1. Paragraphs Source File

The output from this source file produces a formatted series of paragraphs (Figure 2). The lines in the source file are concatenated to fill in the full measure of the text area (6.5 inches). Additional spacing may be added between words to right justify them. Some words may be hyphenated to minimize the amount of spacing between words. The last line is filled in automatically with blank space.

TEX automatically makes basic decisions, called defaults, that determine the margins of a page, the kind and size of type, and the shapes of paragraphs. These decisions use default values. For paragraphs, the default amount of indention is twenty points; the default space between paragraphs is zero points. These defaults can be changed with specific instructions.

Paragraph Indent

The paragraph indention can be controlled with the `\parindent` instruction. The syntax is `\parindent` ⟨*dimen*⟩ where ⟨*dimen*⟩ is a parameter that has two parts: a number and a unit of measure. (The value of that number can be positive or negative — a negative value creates a hanging indention.)

A point is a unit of measure (1 inch = 72.27 points) developed by le Juene in 1737 for the metal type invented by Johann Gutenberg in 1440 — those small rectangular blocks of type required a fine measuring system. Type is traditionally measured in points. TEX performs measurements in points (`pt`), although dimensions can also be specified in `cm` (centimeters), `in` (inches), `mm` (millimeters), `em` (1 em ≈ width of capital letter M), `ex` (1 ex ≈ height of lower case x), as well as other units.

The indention can be suppressed for a paragraph with the `\noindent` instruction before the first text character. Then the first line begins at the left margin. Subsequent paragraphs are typeset with the normal indention. A special instruction for modifying a paragraph, such as `\noindent`, can also act to initiate a paragraph.

Paragraph Skip

The amount of vertical space between paragraphs can be controlled with the `\parskip` instruction. The syntax for this instruction is `\parskip` ⟨*dimen*⟩ where ⟨*dimen*⟩ is any number and valid unit of measure. The default value is `0pt`. When paragraphs are indented, extra space is optional; if the indention is set to zero, then extra space between them may be necessary.

Excerpt Paragraphs

An excerpt is a form of secondary text set off from the main text by margin indentions (and sometimes by type size or spacing before and after the text). The left margin is changed with a `\leftskip` instruction; the right margin with a `\rightskip` instruction. The default for both is zero. The syntax for each is `\leftskip` ⟨*dimen*⟩ where ⟨*dimen*⟩ can be any number and valid unit of measure.

A `\narrower` instruction indents the entire paragraph at *both* the left and right margins by the value of the `\parindent`. These instructions change the shape of all paragraphs that follow; that is, the values are *not* reset automatically. The effect of an unlimited instruction can be limited by putting that instruction within special delimiters. The open and close curly braces (`{}`) are TEX delimiters. A close brace must always match an open brace.

When these changes are included in the source file, the formatted version becomes more complex (see Figure 3). Notice the difference between the plain paragraphs in Figure 2 and the modified paragraph in Figure 4.

```
% Figure 3.  modified paragraphs
\parindent 0pt
\parskip 6pt
```

```
Papyrus, prepared from sliced reeds
pressed and glued together, was the
most commonly used book material in
Greece and Rome.
A crude vellum, made from animal
skins, had been known in old Egypt.

Pliny, the historian, relates that
Eumenes, King of Pergamum, wanted a
fine library for his city, but King
Ptolemy of Egypt, to avoid any
rivalry to the great library of
Alexandria, forbade the export of
papyrus to Pergamum (circa 170 B.C.).
Not to be foiled, Eumenes sponsored
the development of a finer, two-sided
vellum as a writing surface.

{\leftskip 10pt\rightskip 10pt
The library at Pergamum later became
an important center of culture.
It had 200,000 volumes when Antony
presented it as a gift to
Cleopatra---who made it part of the
Alexandrian library.\par} %par must
% be in braces for indents to work!

Moveable type has an honorable lineage.
A clay disk, dating from 1500 B.C.,
was found in the ruins of the palace
of Phaistos on Crete.
Later, in China in A.D.~1041, Pi-Sheng
developed type characters from hardened
clay.

Clay, however, did not hold up well
under repeated impressions.
By 1397 in Korea, type characters were
being cast in bronze.
Then, in 1440, Johann Gutenberg
demonstrated the commercial possibilities
of graphic reproduction with metal type.
\par
```

Figure 3. Altered Paragraphs Source

Hanging Paragraphs

Hanging paragraphs are the inverse of normal ones, where the first line is indented but the following ones are a full-measure wide. In a hanging paragraph, the first line is full-measure and the run-over lines are indented, usually by the amount of space of a paragraph indent. You must specify the indention value.

Hanging paragraphs are created by typing `\hangindent` ⟨*dimen*⟩ at the beginning of an nonindented paragraph (where `\parindent 0pt` or `\noindent` is used). For example, if ⟨*dimen*⟩ is given as `20pt`, all but the first line of the paragraph will be indented twenty points from the left margin; the first line will start at the left margin. To make a series of hanging paragraphs, you must end the previous paragraph, then state the hangindent, and finally start the paragraph with a `\noindent` (unless `\parindent 0pt` is set).

The number of full-measure lines is determined by `\hangafter 1`, the default. The parameter number determines the number of lines left full-measure wide. The number can be made negative with a minus sign — in fact, a `-1` and a `\hangindent 20pt` gives a normal paragraph). A sample source file is shown in Figure 5 and formatted in Figure 6. Note how instructions can be doubled.

```
% Figure 5. Hanging Paragraphs
\hangindent 30pt\noindent
Alphabet length. The horizontal measure,
in points, of the lower case alphabet
set in type of one size and face
(sometimes used to describe an optimum
width, e.g., 1.5 alphabet lengths).

\hangindent 30pt\noindent
Alignment. The way text lines up on a
column: align left (or flush left or
raggedright), align center, align right,
or justify (flush right and left).

\hangindent 30pt\noindent
Ascender. The part of a lowercase letter,
such as b or d, that extends above the
x-height (the height of a letter x).

\hangindent 30pt\noindent
Base line. An imaginary horizontal line
connecting the bottoms of capital letters
(not inclusive of the descenders of lower
case letters).

\hangindent 30pt\noindent
Body type. Type used for the text of a
work, as distinguished from display type,
which is used for chapter headings or
titles. The optimum size ranges from 10
to 12 points depending on style and use.
\par
```

Figure 5. Hanging Paragraphs Source

TEX is part of a long tradition of putting printed words on a page. Many conventions of book and journal production have roots in Mesopotamia and Egypt, Greece and Italy.

Egyptian papyrus scrolls were written in hieroglyphics in vertical columns separated by thin black lines; sometimes illustrations accompanied the text along the top or bottom of the scroll, marked off from text by double ruled lines. As illustration assumed more importance, it was placed within the text. This format persisted through Greek and Roman manuscripts to medieval and modern books.

Although Greek scrolls were also written in columns, characters were presented continuously, in capitals, without breaks between words. Punctuation was usually nonexistent. Breaks in thought were sometimes indicated by an underlining stroke (known as a paragraphos) or by a small blank space.

Figure 2. Formatted Paragraphs from Figure 1.

Papyrus, prepared from sliced reeds pressed and glued together, was the most commonly used book material in Greece and Rome. A crude vellum, made from animal skins, had been known in old Egypt.

Pliny, the historian, relates that Eumenes, King of Pergamum, wanted a fine library for his city, but King Ptolemy of Egypt, to avoid any rivalry to the great library of Alexandria, forbade the export of papyrus to Pergamum (circa 170 B.C.). Not to be foiled, Eumenes sponsored the development of a finer, two-sided vellum as a writing surface.

The library at Pergamum later became an important center of culture. It had 200,000 volumes when Antony presented it as a gift to Cleopatra—who made it part of the Alexandrian library.

Moveable type has an honorable lineage. A clay disk, dating from 1500 B.C., was found in the ruins of the palace of Phaistos on Crete. Later, in China in A.D. 1041, Pi-Sheng developed type characters from hardened clay.

Clay, however, did not hold up well under repeated impressions. By 1397 in Korea, type characters were being cast in bronze. Then, in 1440, Johann Gutenberg demonstrated the commercial possibilities of graphic reproduction with metal type.

Figure 4. Altered Paragraphs from Figure 3.

Alphabet length. The horizontal measure, in points, of the lower case alphabet set in type of one size and face (sometimes used to describe an optimum width, e.g., 1.5 alphabet lengths).

Alignment. The way text lines up on a column: align left (or flush left or raggedright), align center, align right, or justify (flush right and left).

Ascender. The part of a lowercase letter, such as b or d, that extends above the x-height (the height of a letter x).

Base line. An imaginary horizontal line connecting the bottoms of capital letters (not inclusive of the descenders of lower case letters).

Body type. Type used for the text of a work, as distinguished from display type, which is used for chapter headings or titles. The optimum size ranges from 10 to 12 points depending on style and use.

Figure 6. Hanging Paragraphs from Figure 5.

Item Paragraphs

Items are hanging paragraphs that "hang off" an identifier. The syntax for this instruction is `\item{\signif}` where ⟨*signifier*⟩ is any letter, number, or symbol with optional punctuation; the braces must be included if the ⟨*signifier*⟩ is more than one character.

A second level of indention for itemized lists is given by `\itemitem`, which indents twice the `\parindent` value. These instructions automatically end the previous paragraph. Refer to Figure 7 for an example.

```
% Figure 7  Items
\parskip 9pt  % spaces between pars
\item{1.}% curly braces contain number
Skillin, Marjorie, Robert Gay, et al.
1964.
Words Into Type.
New York: Appleton-Century-Crofts.
\item{2.}
Carter, Rob, Ben Day, and Philip Megs.
1985.
Typographic Design: Process and
Communication.
New York: Van Nostrand Reinhold Co.
\par
```

Figure 7. Item Paragraphs Source

Items are useful for lists, outlines, and bibliographies. Figure 8 shows a bibliography.

1. Skillin, Marjorie, Robert Gay, et al. 1964. Words Into Type. New York: Appleton-Century-Crofts.

2. Carter, Rob, Ben Day, and Philip Megs. 1985. Typographic Design: Process and Communication. New York: Van Nostrand Reinhold Co.

Figure 8. Formatted Items

The instructions presented in this article create paragraphs. Therefore, you should remember to end each one with a `\par` instruction or a blank line.

There is a lot more to paragraphs, including ragged margins, repetition of instructions for each paragraph, and special shapes, but that will be presented much later. The next of these training tutorials will address the contents of paragraphs: special characters, accents, fonts, and lines.

Macros

A Tutorial on `\futurelet`

Stephan v. Bechtolsheim

This is the second in a series of tutorials by this author. This time we will deal with `\futurelet`, a rather interesting instruction which causes many people unnecessary difficulties. This article is condensed from a draft of my books *Another Look at TEX*. See the end of this article for more information about the books.

Introduction

The `\futurelet` primitive is a TEX instruction allowing the user to look ahead. The term "look ahead" means that TEX will look at a future token and provide a copy of that token **without** absorbing it, i.e. without removing that token from the main token list. This operation allows the programmer to perform a test for "what token is coming" (to express it in a rather informal way) on the main token list. The token looked at through `\futurelet` will be removed later, typically as part of an argument of a later macro call as we will see shortly. It is **not** removed by the action of the `\futurelet` primitive.

Let us be more precise now; the `\futurelet` instruction has the following format:

`\futurelet` ⟨token$_1$⟩ ⟨token$_2$⟩ ⟨token$_3$⟩

Here is what TEX will do:

1. TEX will execute a `\let` ⟨token$_1$⟩ = ⟨token$_3$⟩. We therefore have generated a copy of ⟨token$_3$⟩ stored under the name of ⟨token$_1$⟩.
2. TEX removes ⟨token$_1$⟩ from the main token list.
3. TEX expands ⟨token$_2$⟩. This token is for all practical purposes a macro with the following properties:
   (a) The macro will use ⟨token$_1$⟩, which is a copy of ⟨token$_3$⟩, to find out what ⟨token$_3$⟩ is, in other words what token is to be expected later.
   (b) It will cause another macro to be expanded which will ultimately absorb ⟨token$_3$⟩. This other macro ordinarily depends on what ⟨token$_1$⟩ is.

There are many applications of `\futurelet`. We will here present only one example, although we will present it in quite some detail so the user will know how to apply `\futurelet` in different circumstances.

Using \futurelet in Macros with Optional Arguments

A typical application of \futurelet is the handling of macros with optional arguments as they are used, for instance, in LaTeX. By "optional argument" we mean an argument which in most cases is omitted, and is provided only occasionally in macro calls.

Defining the Problem

Let us give a specific example: we would like to define a macro \xx, which can be called in two different ways:

1. *With* optional argument as in \xx[opt]{arg} where opt is the optional argument enclosed in square brackets and arg is the regular argument.
2. *Without* optional argument as in \xx{arg} where arg is again the regular argument.

Before we discuss how this can be done in TeX, observe that we do not really have to use an optional argument. We could simply define two different macros \xxWithOpt for the case where an optional argument is given, and \xxNoOpt for the case where no optional argument is given:

```
\def\xxWithOpt [#1]#2{...}
\def\xxNoOpt #1{...}
```

How we can use \futurelet to find out whether an optional argument was given or not? We will define a macro \xx whose only function is to check whether there is an opening square bracket (optional argument is present) or not (no optional argument). The \xx macro will, after this has been determined, cause the \xxWithOpt macro to be invoked when there is an optional argument, and the \xxNoOpt macro to be called if there is no opening bracket. In other words the macros \xxWithOpt and \xxNoOpt do the "real work" while the only purpose of the \xx macro is to decide which of the two macros should be invoked.

Here is the completely worked out example.

```
% (1) First define two macros
%   \xxWithOpt and \xxNoOpt which
%   \xx will call.
%   These macros do "the real work".
% \xxWithOpt: optional argument ([#1],
%   enclosed in square brackets), and
%   regular argument ({#2},
%   undelimited).
% \xxNoOpt: assume no opt. argument,
%   but regular argument only {#1}.
\def\xxWithOpt [#1]#2{...}
\def\xxNoOpt #1{...}

% (2) The \xx macro has no parameter!
%   It only uses \futurelet to check
%   whether there is an optional
%   argument or not by checking
%   whether or not '[' follows \xx
%   in the input.
\def\xx {%
    \futurelet\xxLookedAtToken
                        \xxDecide
}

% (3) The \xxDecide macro, based on
%   the lookahead of \xx, calls
%   either \xxWithOpt or \xxNoOpt.
\def\xxDecide {%
    \ifx\xxLookedAtToken [%
        \let\next = \xxWithOpt
    \else
        \let\next = \xxNoOpt
    \fi
    \next
}
```

A Macro Call with Optional Argument

Let us now look at the following macro call of the \xx macro that we have defined: \xx[a]{b}. This generates the following token list:

\xx • [• a •] • { • b • }

Now \xx is expanded, yielding the following token list:

\futurelet • \xxLookedAtToken
• \xxDecide • [• a •] • { • b • }

Observe that \xxLookedAtToken corresponds to ⟨token$_1$⟩ of \futurelet, \xxDecide to ⟨token$_2$⟩ and [to ⟨token$_3$⟩ (see the format of \futurelet in the introduction above). Observe especially the value of ⟨token$_3$⟩: this is the token we are interested in. \xxDecide will test this token to check whether or not it is an opening square bracket, in order to decide whether to call \xxWithOpt or \xxNoOpt.

Next TeX executes the \futurelet, assigning [to \xxLookedAtToken, and then expands \xxDecide. This expansion leads to the following main token list:

\ifx • \xxLookedAtToken • [• \let
• \next • = • \xxWithOpt • \else
• \let • \next • = • \xxNoOpt • \fi
• \next • [• a •] • { • b • }

The \ifx conditional evaluates to true because \futurelet has just assigned an opening square bracket to \xxLookedAtToken. Therefore \let\next = \xxWithOpt is executed and the whole conditional (from \ifx through \fi) is removed from the main token list. This leads to the following new main

token list:

```
\next • [ • a • ] • { • b • }
```

Because `\next` is equivalent to `\xxWithOpt` this is equivalent to the following main token list:

```
\xxWithOpt • [ • a • ] • { • b • }
```

And this is of course exactly what we wanted: the macro `\xxWithOpt` is executed and the expansion of this macro will absorb the optional argument enclosed in square brackets and the mandatory argument enclosed in curly braces. Observe that, up to this point, the opening square bracket stayed on the main token list.

A Macro Call without Optional Argument

Let us now look at a macro call of `\xx` with **no** optional argument, as in `\xx{a}`. Here is a short description of what happens. `\xx` is expanded to yield the following token list:

```
\futurelet • \xxLookedAtToken
 • \xxDecide • { • a • }
```

Therefore an opening curly brace, not an opening square bracket, is assigned to `\xxLookedAtToken`. `\xxDecide` is now expanded and the conditional `\ifx • \xxLookedAtToken • [` this time evaluates to false. Therefore the assignment `\next = \xxNoOpt` will be executed. This leads to the following main token list:

```
\next • { • a • }
```

and now `\next` is the same as `\xxNoOpt`, exactly as we wanted it to be.

Looking at the Previous Example Once More, `\DblArg`

There are frequently cases where a macro requires two arguments, but both may be identical. In such a case, a macro may be defined with an optional argument, where the absence of the optional argument in the input is assumed to imply that an optional argument identical to the mandatory argument has been supplied. Using the notation of the previous example, this means that `\xxNoOpt{a}` is equivalent to `\xxWithOpt[a]{a}`.

The previous example can be easily modified to define a generic macro `\DblArg` so that the definition of `\xx` reads as follows:

```
\def\xx{\DblArg{\@xx}}
```

The call `\xx{1}` is converted into `\@xx[1]{1}` and the call `\xx[1]{2}` is converted into `\@xx[1]{2}`. Here is the definition of the macro `\DblArg`:

```
\catcode`@ = 11

% \DblArg
% =======
% #1: the macro to be called
% ultimately (\@xx above).
\def\DblArg #1{%
    \def\@DblArgTemp{#1}%
    \futurelet\@DblArgTok\@DblArg
}

% \@DblArg: if there was an opening
% square bracket then simply continue.
% Otherwise the main argument has
% to be duplicated to also become
% the argument enclosed in square
% brackets.
\def\@DblArg{%
    \ifx \@DblArgTok [%
        % Optional argument!
        \let\@DblArgTempA=\@DblArgTemp
    \else
        % No optional argument:
        % duplicate!
        \let\@DblArgTempA=\@DblArgB
    \fi
    \@DblArgTempA
}

% Read in the argument and duplicate
% to also become an optional argument.
\def\@DblArgB #1{\@DblArgTemp[#1]{#1}}

\catcode`@ = 12
```

Concluding Remark

This article is, as briefly mentioned in the introduction, an adaptation of a section of my books, *Another Look At TEX*, which I am currently finishing. These books, now two volumes totalling almost 1000 pages, grew out of my teaching and consulting experience. The main emphasis of the books is to give concrete and useful examples in all areas of TEX. (The section on `\futurelet` is 18 pages long. The chapter on `\halign` contains over 100 tables.) In *Another Look at TEX* you should be able to find an answer to almost any TEX problem.

Compact Matrix Display

Thomas Kneser
GWDG, Göttingen (FRG)
John S. McCaskill
MPI für biophysikalische Chemie,
Göttingen (FRG)

The following problem arose in the study of the molecular structure of RNA[1]:

One wishes to represent information about the probabilities that various pairs (i, j) from a sequence of length n over a finite alphabet occur. It is important to be able to locate accurately from the display which pair is involved and how probable it is on a logarithmic scale. A similar representation problem arises in displaying the strength of connection between pairs of units of neural networks[2].

For large n, the only compact way to represent the information on a line printer is to encode a different character for each of a finite number of probability levels. The information is then displayed as a matrix. This leads to rather ugly output which is not easily interpretable even if the characters are chosen with the amount of black ink increasing with increasing probability. However, for a first look this is the most efficient way to obtain the information. The problem then is to convert this character output to a high quality image for visual processing.

A succinct way of doing this is by drawing black boxes of varying sizes accurately positioned with lower left corners forming the square matrix of probabilities. TEX provides the opportunity to draw such structures by setting sequences of appropriate `\vrules` and to merge such plots with additional text and alphanumeric information. This merging leads as a side effect to the obvious advantage that a complete paper can easily be transferred through the networks by transmitting just a single file. Figure 2 shows that part of the input file which defines the matrix, containing line printer style character data. The related graphic output is shown in Figure 3.

Figure 1 shows the structure of one matrix element, with *wo* ranging from about *5pt* to *10pt* and $wi < wo$. The matrix elements can easily be coded as:

```
\hbox to \wo{\vrule height \wi width \wi
             depth 0pt \hfil}          (V1)
```

V1 is working fine for about $n < 50$ but for larger n problems arise concerning TEX's internal storage ("! TEX capacity exceeded ... "), even if the current page contains nothing other than the matrix and even if *mem_max*† is set to the maximum of $2^{16} - 1$.

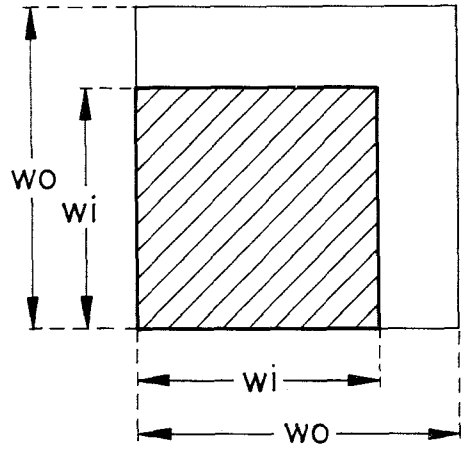


Figure 1

What is the reason for this rapid exhaustion of storage?

Clearly, TEX has to hold all the stuff defining the current page in *mem_array* until the `\shipout` operation is done. Taking this into account we have to ask the following question: how much memory does TEX use for one matrix element?

Analysing V1 following [3], we find that for each matrix element TEX generates the following list of *nodes*:

- 1 box_node
- 1 rule_node
- 1 glue_node

We define σ as the amount of storage TEX needs to allocate one rule matrix element. By summing up TEX's constants *box_node_size*, *rule_node_size* and *glue_spec_size* we get:

$$\sigma_1 = 7 + 4 + 4 = 15 \ memory\_words$$

This requirement is high compared with that for a *character token*, which requires only one *memory_word* to fill about the same area on paper.

Fortunately one finds that there is at least one alternative version with $\sigma < \sigma_1$:

```
\vrule height \wi width \wi depth 0pt
\rest=\wo
\advance\rest by -\wi
\kern\rest                             (V2)
```

† terms in italics with enclosed underline character refer to [3]

```
\centerline{PARTITION FUNCTION PROBABILITIES OF BINDINGS}\medskip
\centerline{log interval =     10.000}
\centerline{decreasing log (base 10) probabilities in order}
\hbox to \hsize{\hss\vbox{\halign{#\quad&\hfil#\cr
\eseq{ @}&    0 to  -0.5\cr\eseq{ *}& -0.5 to  -1.5\cr\eseq{ o}& -1.5 to  -2.5\cr
\eseq{ =}& -2.5 to  -3.5\cr\eseq{ +}& -3.5 to  -4.5\cr\eseq{ -}& -4.5 to  -5.5\cr
\eseq{ ;}& -5.5 to  -6.5\cr\eseq{ :}& -6.5 to  -7.5\cr\eseq{ ,}& -7.5 to  -8.5\cr
\eseq{ .}& -8.5 to  -9.5\cr\eseq{  }&  <  -9.5\cr}}\hss}%
\bigskip
\mrow{           1         2         3         4         5         6         7         8 }
\mrow{  12345678901234567890123456789012345678901234567890123456789012345678901234567890 }
\mrow{ 1G   .:,:,:=o-;;;:::;-;;;;;,,.,-*+--;;.                ,-:;-;:. :,,.  .,.   .:;;-G}
\mrow{ 2 G   .,,:=o;,,::..;---;;;;;,,.-*;,,,,.                ,  ,,. :,,.          ,:;oG}
\mrow{ 3  G   ,:,o;,,:::..-;;;;;;;;;;,*;,,,,..               . .:;::,,.           ,;=@G}
\mrow{ 4   U   ;=::::::::;-+====ooo=*-----;::                 :;;=o+;,          ,:-+o+;U}
\mrow{ 5    U   ;,:::::::;++====ooo=*o;;---;::                ;;;+=o;,        .,:-===*o-U}
\mrow{ 6     C   ,,:::::+-+========*o-;;--;;::                ,:;;+o+-,       .,:-===+o=;C}
\mrow{ 7      A   ,,:;;;::;+oooo===++--;---;::                 ::;oo=;,         ,;--=*=;A}
\mrow{ 8       U   ,;:;;;:;=o===oo=++++----;:;                 ::-+*+;.        .:--++==;U}
\mrow{ 9        A   ,:;;:;-==+o+++-+--;;;;;::                  :;;@=+;.       ..:;-o=+:A}
\mrow{10         C   ,::+@;;;;;;;;;-*:,,:;;;::                 ::@;-:.        .+=o=o:;;.C}
\mrow{ 1          C   ,:@+;;-;;;;;*-:,:::;;::                  :@;;-:.       +=o=o:::;.C}
\mrow{ 2           U   :-;;;;;;+-*-;:;--++++=:                 :@;:;;:,       +=o=o:.:;;.U}
\mrow{ 3            A   ;--++++=+-;--+==ooooo;,                :;+,:;;:,      ,;-+-; .:-;,A}
\mrow{ 4             U   --+===oo+--=-+=ooooo;,                ;--:::;;:,     :o--;. ,:;;,U}
\mrow{ 5              U   --+==o+--=o--o*oooo;,                :;-:::;;-.     .o;:,.  .:;;,U}
\mrow{ 6               C   -+===+;=o---+o====:;              ,,;-:::;-:.     o  .    ,:;;:C}
\mrow{ 7                G   +++=+-;--=@=---=-;       .   . , :.: .;;:,;,.:o        .:;;;G}
\mrow{ 8                 G   +-++-;;-@=---++-:..     .   ,      .:. :;;:,.:o        .:;;+G}
\mrow{ 9                  C   ---+oo-;-------,,    .   ,       ,:;;;+;:::;.       .:-+=:C}
\mrow{20                   U   -+oo==--------,.    .   ,       .;;:;-;:;--.       ,:-+=-U}
\mrow{ 1                    U   ====+;;;-;--+,.                .;:::;-;:::;       .:-+=;U}
\mrow{ 2                     U   ++++;;;;;-++,.     .          .:::;-;:::;       .:-+=;U}
\mrow{ 3                      U   ---;;;;;---,     .           .:::;-;:::;       .:-+=;U}
\mrow{ 4                       A   ;;;;;+==;;.    .           .:::;--:::;       .:-=o;A}
\mrow{ 5                        A   ;:;+==+;;,                .::,;--::--.       .:+oo;A}
\mrow{ 6                         A   ::==+-;;,        .       .:::;++;::;       .:=oo;A}
\mrow{ 7                          G   =+--;+;, .     .        .:::-==-o=-.       ,;+=o+G}
\mrow{ 8                           G   ;;;-+;,               .:::+++o=-,        ,;+oooG}
\mrow{ 9                            A   :;-;;.   .           .:::@+;,.          ,-+o:A}
\mrow{30                  @          C   ::;;.. .            ,:-@;+;:,          -:--:C}
\mrow{ 1                  @           C   ::;..              ,:@-;-;:.          -,;--:C}
\mrow{ 2                               U   ;-,              .:*-;---,.         ,.,;---U}
\mrow{ 3                                U   -,.            ,++;-++-,.          ,;+++U}
     .
     .
     .

\mrow{ 8         @                                                  G     ,           ,;+*G}
\mrow{ 9        @                                                    U               ,;+=+U}
\mrow{60       @                                                      G    .         ,;+@o+G}
\mrow{ 1                                                               A            .:@=+:A}
\mrow{ 2                                                                C          ,-+@:;;.C}
\mrow{ 3                                                                 C        ,-+@=.,. C}
\mrow{ 4                                                                  C      ,-+@=-    C}
\mrow{ 5                                                                   C     -+@=-,    C}
\mrow{ 6                                                                    C    +@=-,     C}
\mrow{ 7                                                                     C   @=-,      C}
\mrow{ 8                                                                      G            G}
\mrow{ 9                                                                       A           A}
\mrow{70                                                                        A          A}
\mrow{ 1                                                                      @   G        G}
\mrow{ 2                                                                     @     G       .G}
\mrow{ 3                                                                    @       G     .-G}
\mrow{ 4                                                                   @         G   .:-G}
\mrow{ 5                                                                  @           G   :;G}
\mrow{ 6                                                                 @             G   :G}
\mrow{ 7                                                                @               U   U}
\mrow{ 8                                                                                 U  U}
\mrow{ 9                                                                                  U U}
\mrow{80  @                                                                               CC}
\mrow{ 1 @                                                                                 C}
\mrow{ 2@                                                                                  C}
\mrow{  GGGUUCAUACCUAUUCGGCUUUUAAAGGACCUUUUCACGCGUAGCUAGCUACGCGAGGUGACCCCCCGAAGGGGGGUUUC }
```

Figure 2 : Input file SAMPLE1.TEX

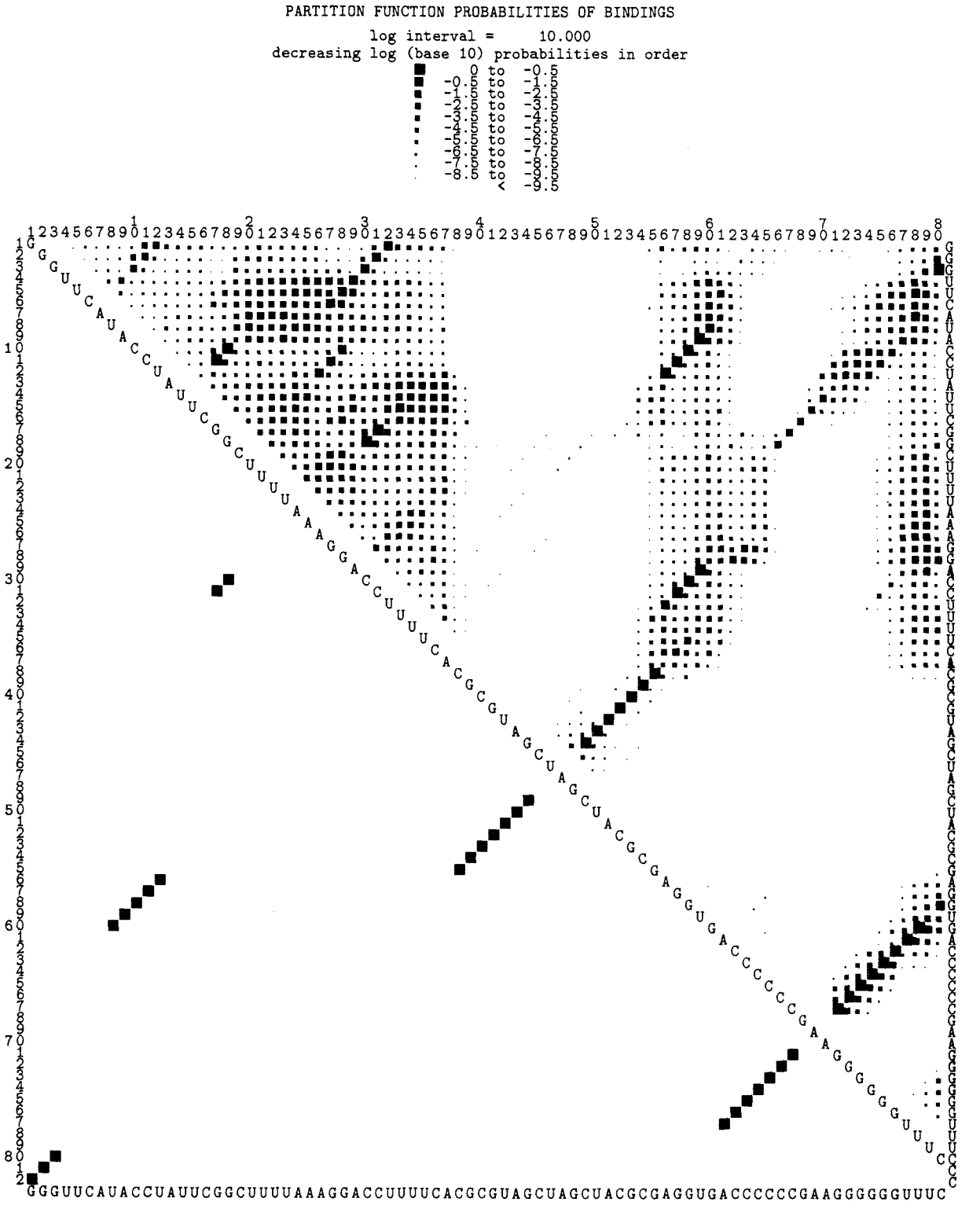


Figure 3 : Output related to SAMPLE1.TEX

Analyzing V2 we find:

- 1 rule_node
- 1 small_node (due to the \kern item)

and:

$$\sigma_2 = 4 + 2 = 6 \ memory\_words$$

since $small\_node\_size = 2$.

Comparing both σ values we get:

$$\sigma_2 = 0.4\sigma_1$$

Obviously V2 which is formulated more explicitly — and somewhat less elegantly — does the job much more economically. Therefore the macro \setrule in file CMD.TEX (Figure 4) contains V2.

Furthermore, storage requirements are reduced by summing up the space of horizontal sequences of empty matrix elements, using dimension register \accmt. Since the displayed matrices generally are sparse matrices this also leads to substantial memory saving.

Besides storage space, the requirement for CPU time is here also a limiting factor. We define T as the amount of CPU time TEX needs to process a particular matrix. Clearly T depends on the number of processed matrix elements but also heavily on the relation $N_E : N_R : N_O$ (number of empty elements, rules and \otherchars). Since we have developed CMD for an application concerning primarily sparse matrices, CMD should process empty elements faster than elements of the two other types. This is accomplished by the fact that CMD tests for spaces first (refer to Figure 4: the comparison of \next with \spacechar in macro \dodolist). If a space character is identified, CMD needs neither to expand macro \setrule nor to clamber through the \if-chain. However, T is still by a factor of 8 greater than the CPU time needed to set 'normal' running text; for SAMPLE1.TEX with

$$N_E : N_R : N_O = 5271 : 1386 : 420$$

we have measured $T \approx 16\,s$ (VAX 8650, TEX 2.0, Stanford version for VAX/VMS). About $0.3T$ are absorbed by the activities of macro \nonblank; here of course the \if-chain can be replaced by one simple \if...\else...\fi clause if we declare characters '.,;-+=*@' as active (catcode=13) — that is by effectively using indexing technique. But while CPU time tradeoff due to such a modification is small ($< 5\%$), the coding becomes significantly more difficult to maintain and to update.

After these TEXnical considerations let us still mention one possible modification of CMD: for some cases one may want the blacked rule to be shifted from the lower left corner of the element (ref. Figure 3) to the center of the element area. This can easily be accomplished by replacing the body of macro \setrule by:

```
\rest=\wo\advance\rest by -\wi
\rest=0.5\rest
\rulh=\wi\advance\rulh by \rest
\kern\rest
\vrule height\rulh width\wi depth-\rest
\kern\rest                                    (V3)
```

with an additional dimension register \rulh to hold the rule height.

Finally, we hope the proposed TEX coding can serve as an efficient general purpose tool for real discretized matrix display in a large number of different applications.

[1] McCaskill, J.S. "The Equilibrium Partition Function and Base Pair Binding Probabilities for RNA Structure" (1988) in press at Biopolymers

[2] Kohonen, T. "Self-organization and associative memory" Springer, Berlin, Heidelberg, New York (1984)

[3] Knuth, D.E. "TEX: The Program" Addison-Wesley, Reading, Massachusetts (1986)

Figure 4. The file CMD.TEX

```
\message{+++++ CMD Compact Matrix Display , Version 1.4 +++++}
%                                  Th. Kneser, John S. McCaskill
\newdimen\wo     % width of elements
\newdimen\wi     % width of \vrule in an element
\newdimen\mtacc  % accumulates length of successive empty elements
\newdimen\rest   % c(\rest) = c(\wo) - c(\wi)
\newdimen\zeropt%
\zeropt=0pt
%
\font\typw=CMTT8% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\typw            % WARNING: the current font should  n o t  be altered
\wo=0.65em       %          after this font related setting
                 % %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
\newdimen\hwaa\hwaa=.05em \newdimen\hwbb\hwbb=.10em
\newdimen\hwcc\hwcc=.15em \newdimen\hwdd\hwdd=.20em
\newdimen\hwee\hwee=.25em \newdimen\hwff\hwff=.30em
\newdimen\hwgg\hwgg=.35em \newdimen\hwhh\hwhh=.40em
\newdimen\hwii\hwii=.50em \newdimen\hwjj\hwjj=.60em
%     \hwXX: heights and widths of \vrules in matrix elements
%
\def\mrow#1{\hbox to \hsize{\hfil\dolist #1\endlist\mtelem\hfil}}%
%               % to set a centered matrix row
\def\eseq#1{\dolist #1\endlist\mtelem}%
%               % to set an inline sequence of matrix elements
\def\dolist{\afterassignment\dodolist\let\next=}
\def\dodolist{\ifx\next\endlist\let\next\relax%
              \else%
                \if\next\spacechar%
                   \global\advance\mtacc by \wo% to accumulate empty elements
                \else%
                   \ifdim\mtacc>\zeropt%
                      \mtelem%  process a sequence of empty elements
                   \fi%
                   \nonblank%   process one nonblank item
                \fi%
                \let\next\dolist%
                \fi%
              \next}%
% \dolist and \dodolist are derived from 'The TeXbook' Ex. 11.5,
% they parse a sequence of tokens (in the actual case the
% argument strings of \eseq and \mrow)
%
\def\nonblank{%  to set nonblank matrix elements
  \if\next .\wi=\hwaa\setrule%
  \else\if\next ,\wi=\hwbb\setrule%
  \else\if\next :\wi=\hwcc\setrule%
  \else\if\next ;\wi=\hwdd\setrule%
  \else\if\next -\wi=\hwee\setrule%
  \else\if\next +\wi=\hwff\setrule%
  \else\if\next =\wi=\hwgg\setrule%
  \else\if\next o\wi=\hwhh\setrule%
  \else\if\next *\wi=\hwii\setrule%
  \else\if\next @\wi=\hwjj\setrule%
  \else\otherchars%
\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi}% end of \def\nonblank
%
\def\mtelem{\kern\mtacc\global\mtacc=\zeropt}
%               % to set horizontal space equivalent
%               % to a sequence of empty elements
\def\setrule{\vrule height \wi width \wi depth \zeropt%
               \rest=\wo\advance\rest by -\wi%
               \kern\rest}
%               % to set a matrix element containing a rule
\def\otherchars{\hbox to \wo{\next\hss}}%
%               % to set nonblank & nonrule elements
\def\spacechar{ }% to be compared with \next
%
\baselineskip=\wo plus 1pt        % do  n o t  change
\lineskiplimit=0pt\parindent=0pt  % the current font here!
\obeyspaces
\input sample1
\bye
```

Typesetting Chess

Wolfgang Appelt
Gesellschaft für Mathematik und
Datenverarbeitung, Sankt Augustin

On a recent Sunday morning I was sitting in my garden reading a book on the latest chess world championship between Karpow and Kasparow. Though the book was surely good, giving a lot of explanations concerning the (probable) reasoning of the players, for me as a rather poor chess player, reading chess literature only casually, it presented some problems. There were rather few diagrams in the book showing the position on the board and sometimes for more than twenty moves only the usual textual chess notation (e.g., c2–c4, Kh8×g7) was used to describe the progress of the game.

Of course, for a chess expert it is no problem to read such a notation and to have always a precise visual picture of the position on the board in his mind, but for me as an untrained reader it was rather difficult. Usually, when reading such a book, you are supposed to have a chess board at hand and to perform the moves on the board to watch the progress of the play. However, such a procedure is not always completely satisfactory since it is often much more appropriate to compare simultaneously different positions on the board to understand what is really happening.

In other words, what I really would like was a lot more diagrams in chess books and I started to consider if TEX could be used for such a job. I went to my PC and two hours later I had a set of macros which were at least a starting point for such a task.

The general principles of these macros are described in this article. I shall not show all the details since, as it will be seen later on, there are still some problems left and the rather specific solutions I have chosen might not be of general interest.

The user interface of the macro set should be as close as possible to the conventional chess notation, i.e, you should be allowed to type for example

```
\move e2-e4  \move d7-d5
\move e4xd5  \move Sg8-f6
\showboard
```

and receive a diagram of the actual position on the board. (I shall use the so–called algebraic notation, which is commonly found in German chess literature, and I shall also use the German abbreviations for the pieces, e.g., "S" stands for "Springer" which means knight. However, a "translation" of the macros into the notation scheme of English or any other language should cause no major problems.)

Obviously, the chess board can be typeset as a table where each table entry represents exactly one field on the board. A macro for printing a chess board might therefore look like this:

```
\def\showboard{$$\vbox{\offinterlineskip
  \halign{\vrule\field##.&\field##.&...
           ...&\field##.\vrule\vrule\cr
    \noalign{\hrule}\noalign{\hrule}
    .\@a8&*.\@b8&.\@c8&*.\@d8&...&*.\@h8\cr
    \noalign{\hrule}
    *.\@a7&.\@b7&*.\@c7&.\@d7&...&.\@h7\cr
    \noalign{\hrule}
    .\@a6&*.\@b6&.\@c6&*.\@d6&...&*.\@h6\cr
    \noalign{\hrule}
    ⋮
    \noalign{\hrule}
    *.\@a1&.\@b1&*.\@c1&.\@d1&...&.\@h1\cr
    \noalign{\hrule}\noalign{\hrule}
    }}$$}
```

Each field on the chess board is represented by a macro, called \@a1, \@a2, ... \@a8, \@b1, ... \@h8, as it is the usual convention for naming the fields on a chess board. (Please note that the numbers 1...8 and the at-sign (@) have the *category code* 11, and therefore, e.g., \@b1 is a valid macro name.) Each of these "field macros" must either hold the value "void" or the value of the piece which is on this field at any time in the game. That means we have to keep track of an 8×8 matrix representing the position of each piece on the board. At the start of the game these definitions read as follows:

```
\def\@a8{\ST}\def\@b8{\SS}\def\@c8{\SL}
   \def\@d8{\SD}\def\@e8{\SK}\def\@f8{\SL}
   \def\@g8{\SS}\def\@h8{\ST}
\def\@a7{\SB}...\def\@h7{\SB}
\def\@a6{\void}\def\@b6{\void}
⋮
   \def\@g3{\void}\def\@h3{\void}
\def\@a2{\WB}...\def\@h2{\WB}%
\def\@a1{\WT}\def\@b1{\WS}...
   \def\@g1{\WS}\def\@h1{\WT}}
\def\void{}
```

The next step is providing a macro for performing a single move on the board. A move requires two actions:

— The "start position" of the moved piece must be updated, i.e., the macro \@xn (x ∈ {a...h}, n ∈ {1...8}), must be "cleared", and

— the "target position" of the moved piece must be redefined, i.e., the macro \@xn (x ∈ {a...h}, n ∈ {1...8}), must receive the value of the corresponding piece.

This can be achieved by the following macro:

```
\def\@move#1#2#3#4#5#6{% Syntax:
  % [KDTLSB][a-h][1-8][-x][a-h][1-8]
  % [...] means: select one
  \expandafter
      \def\csname @#2#3\endcsname{\void}%
  \ifx\colour\whitecolour\expandafter
      \def\csname @#5#6\endcsname
          {\csname W#1\endcsname}%
  \else\expandafter
      \def\csname @#5#6\endcsname
          {\csname S#1\endcsname}\fi}
```

The required syntax of the arguments is given as a comment in the macro. The macro first redefines the macro for the starting position to \void and then defines the value of the macro for the target position to the value of the moved piece. The value of this macro depends also on the fact if the move is performed by a white piece or by a black piece. For example, if white is the next player, the two consecutive moves e4×e5 and Sg8–f6, represented by the macro calls `\@move Be4xe5` and `\@move Sg8-f6` will produce the following definitions:

```
\def\@e4{\void} \def\@e5{WB}
\def\@g8{\void} \def\@f6{BS}
```

The next step is providing the macros for the user interface. The conventional algebraic chess notation first gives the value of the piece (in German either K, D, L, S or T), the start position (e.g., f2), followed by a dash or a "×" (if the move removes a piece from the board), followed by the target position. If the piece is a pawn the value of the piece is not given, i.e., a move may be denoted by "e2–e4", "e4×d5", "Sf2–g4" or "Sf2×g4". This is a very simple syntax, only the case of the missing value in the case of a pawn must be handled specially. The following macro can be used:

```
\def\move#1#2#3#4#5#6 {% Syntax:
  % {KDTLS}[a-h][1-8][-x][a-h][1-8]
  % {...} means: select zero or one
  \if#3-\@move B#1#2#3#4#5%
     \else\if#3x\@move B#1#2#3#4#5%
            \else\@move #1#2#3#4#5#6\fi\fi
  \ifx\colour\whitecolour\def\colour{S}%
      \else\def\colour{W}\fi
    }
```

Please note that the last parameter is a *delimited parameter*, i.e., the end of the argument sequence is denoted by a space. This makes the macro work with both five or six arguments.

Now we have assembled all together except the macros for the actual display of the fields of the board. The best way to print a field with or without a piece on it would be by using a special font, containing an empty white field, an empty black field, a white pawn on a white field, a white pawn on a black field, a black pawn on a white field etc. This would sum up to 26 different symbols which should be created by METAFONT. However, this would exceed a Sunday afternoon's entertainment and I therefore used the following, admittedly rather quick and dirty "approximation".

The macro `\showboard` uses the macro `\field` in the preamble of the `\halign`, and this `\field`–macro has two parameters as you may guess by looking at the table lines in the `\halign`. The first one is either empty or a "*", the latter case indicating that the field is a black one (see the "checkered" distribution of the asteriskes in the table lines). The second argument is one of the 64 "field macros". Having no special chess font available, the basic idea is the following:

— The displays of the chess pieces are created by putting together symbols from existing *Computer Modern* fonts.

— Black fields are created by constructing a quadratic pattern of characters from the (usually also available) *gray* font. (Making fat black squares out of `\vrules` or `\hrules` looks really ugly; I tried this first.)

There is only one complication which must be taken care of: If a piece has to be put on a black field, this field must not be completely filled with the background pattern, but there must be left some white space in the middle where the piece is displayed. The `\field`–macro may there look as follows:

```
\newif\ifblackfield
\def\field#1.#2.{\def\next{#1}%
   \ifx\next\empty\blackfieldfalse
       \else\blackfieldtrue\fi
   \ifblackfield\edef\next{#2}%
       \ifx\next\empty\vrule\fieldstrut
           \hbox to \fieldwidth{\hfill
                \emptyblackfield\hfill}%
       \else\vrule\fieldstrut\blackborder
            \setpiece{#2}\fi
   \else\vrule\fieldstrut\setpiece{#2}\fi}
\def\fieldstrut{\vrule height\fieldheight
```

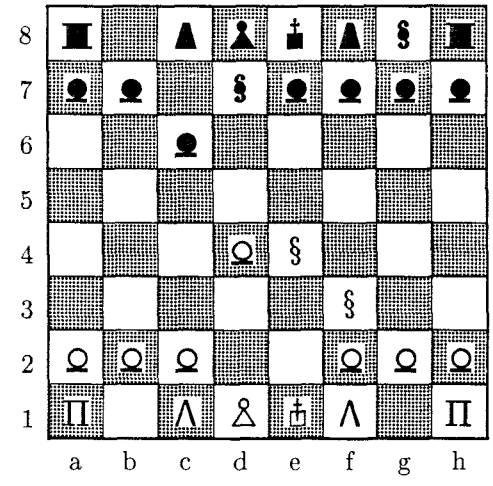
```
                depth\fielddepth width0pt}
\def\setpiece#1{\hbox to \fieldwidth
                    {\hfill#1\hfill}}
```

The macro `\emptyblackfield` which appears in the `\field`–macro is basically a `\vbox`, containing several lines of characters from the *gray* font, and the macro `\blackborder` is rather similar, but it leaves some white space in the middle. Instead of describing these two macros and the macros displaying the different chess pieces in detail, I shall give a small example, whereby you might guess what symbols of the *Computer Modern* fonts I used for the pieces. For example, the input text

```
\move e2-e4  \move c7-c6
\move d2-d4  \move d7-d5
\move Sb1-d2 \move d5xe4
\move Sd2xe4 \move Sb8-d7
\move Sg1-f3
\showboard
```

will give the the following diagram:

Board after White's 5. move



Please note, that the `\showboard`–macro has been slightly extended to print a number and a character, respectively, on the left side and on the bottom of the board. Furthermore, a heading for the diagram is printed, telling the number of the move and the name ("Black" or "White") of the last player. This requires also a small extension of the `\move`–macro to keep track of this information.

Just to give you an impression what the macros for the display of the different chess pieces look like, one example: The definition for a white pawn is:

```
\def\WB{\together{\bbbsym\char14 }%
     {\kern -1pt\hbox{\vrule
      height 1.4pt depth 0pt width 8pt}}}
```

where `\together` is a macro with two parameters which are symbols, characters or rules, which are to be printed atop of each other, and `\bbbsym` is the symbol font at `\magstep3`.

If there is a special chess font available the definitions of the macros displaying the pieces are just selections of characters from this font, e.g.

```
\def\WB{{\chessfont
  \ifblackfield\char11 \else\char39 \fi}}
```

and the definition of the `\field`–macro would become a bit simpler.

Though the macros shown above will not give a professional environment for typesetting chess books they may be used as a good starting point for such a task. The most obvious improvement is, of course, the creation of a set of special symbols by METAFONT. This should be a rather simple task, even for people with a rather limited experience in typographic design.

Furthermore, the macros `\move` and `\@move`, respectively, have to be extended to handle the so–called *castling*, denoted by 0–0 or 0–0–0, and the special pawn move called *capture en passant*, denoted often by, e.g., f5×g6(e.p.). These extensions are rather straightforward.

It is also a simple extension to the `\move`–macro to make it print its arguments. For example, before printing the board the above shown input could give an additional listing of the game in the form:

1.	**e2–e4**	**c7–c6**
2.	**d2–d4**	**d7–d5**
3.	**Sb1–d2**	**d5×e4**
4.	**Sd2×e4**	**Sb8–d7**
5.	**Sg1–f3**	

Making an even more ambitious step, it should also be possible to extend the macros shown above to check if a move is legal or not. For example, if you enter `\move Sg3xf5` you should get an error message if there is no "Springer" on g3 or no piece on f5. In other words, it should be possible to develop a set of TEX macros which know the legal moves of a chess game and which detect typos.

To summarize: TEX can be used for an integrated typesetting of chess games (in conventional notation) and of chess diagrams. Exploiting TEX's macro facilities it should be possible to eliminate typos which can be a great embarassment to the readers. Even if typos are less frequent in well-typeset chess books due to careful proofreading — you will probably find typos more frequently in the

chess columns of newspapers — such an approach might be able to improve the quality and to reduce the costs of chess literature.

Editor's note: The gray font referred to here is normally used to test METAFONT proof characters — it is the font that appears in the character illustrations in Volume E of *Computers & Typesetting*. Unlike ordinary METAFONT fonts, the gray font is device-dependent. That is, different versions, *with different* .TFM *files*, will be used to produce output on devices with different print characteristics, including resolution.

Dr. Appelt originally prepared this article using a laser printer with 300 dots-per-inch resolution; the typesetter on which TUGboat camera copy is prepared has a final resolution of over 1000 dots per inch, although fonts for it are created at 723 dots per inch. Attempts to install a suitable typesetter-specific gray font failed, so the figure of the chessboard has been pasted in from the laser printer copy that Dr. Appelt supplied.

Anyone attempting to use the macros defined in this article, or doing anything else that requires the gray font (including METAFONT), should be aware of this restriction.

Equation Numbering in Plain TEX

J. E. Pittman

A few simple macros can provide facilities for automatic equation numbering with (limited) forward referencing. A backward (after the equation has been displayed) reference to an equation is made in the text by the use of the `\referenceequation{`*name*`}` macro, which generates the appropriate number and inserts it into the text. The `\referenceequation` macro will also work correctly if it is used 'just' before the referenced equation, i.e., as long as there are no numbered equations between the referenced equation and the point of reference.

A forward (before the equation has been displayed) reference to an equation is made by the use of the `\forwardreferenceequation{`*name*`}{`*n*`}` macro, where *n* is the number of numbered equations that will be displayed between the point of reference and the referenced equation.

Within displayed equations, the `\eqname{`*name*`}` macro can be used in same manner that the `\eqno` *text* macro is normally used. Note: `\eqno` is documented in chapter 19 of The TEXbook.

If an equation is to be numbered but not referenced, the `\eqnum` macro can be used in place of the `\eqname{`*name*`}` macro.

Figure 1 gives an example of the way in which these macros are normally used.

This method of equation numbering is limited due to the requirement of equation counting for forward referencing, however, it works well for most applications and does not require more than one pass through the input file(s).

The following input:

```
% --- example ---
Equation \forwardreferenceequation{byhalves}{2}
gives a simple example of a convergent
infinite series.
$$
E = mc^2        \eqname{emc2}
$$
$$
A = A           \eqnum
$$
$$
1 = \sum_{n=1}^\infty 2^{-n}
  = {1 \over 2} + {1 \over 4} +
    {1 \over 8} + \cdots    \eqname{byhalves}
$$
\TeX\ reduces the task of typesetting
Einstein's famous equation
(\referenceequation{emc2}) to pure
simplicity.
\par
```

Produces:

Equation 3 gives a simple example of a convergent infinite series.

$$E = mc^2 \tag{1}$$

$$A = A \tag{2}$$

$$1 = \sum_{n=1}^{\infty} 2^{-n} = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots \tag{3}$$

TEX reduces the task of typesetting Einstein's famous equation (1) to pure simplicity.

Figure 1. Example of equation numbering macro use

```
% --- macros ---
%
\newcount\equationnumber      \equationnumber=0
%
\def\eqnum{\relax
   \global\advance\equationnumber by 1
   \equationnumberformat{\the\equationnumber}%
   }%
%
\def\eqname#1{\relax
   \count255=\equationnumber
   \assignnumber{EN#1}\equationnumber
   \global\equationnumber=\count255
   \global\advance\equationnumber by 1
   \ifnum\csname EN#1\endcsname=\equationnumber
   \else
      \message{The equation number for ``#1'' is incorrect!}%
   \fi
   \equationnumberformat{\csname EN#1\endcsname}%
   }%
%
\def\equationnumberformat#1{\eqno(\equationnumbertype{#1})}%
%
\def\equationnumbertype#1{\number#1\relax}%
%
\def\referenceequation#1{\relax
   \assignnumber{EN#1}\equationnumber
   \equationnumbertype{\csname EN#1\endcsname}%
   }%
%
\def\forwardreferenceequation#1#2{\relax
   \global\advance\equationnumber by #2
   \assignnumber{EN#1}\equationnumber
   \global\advance\equationnumber by -1
   \global\advance\equationnumber by -#2
   \referenceequation{#1}%
   }%
%
% Macro for numbering, parameters are the csname text and a counter.
%
\def\assignnumber#1#2{\relax
   \ifnum0<0\csname#1\endcsname
   \else
      \global\advance#2 by 1
      \expandafter\expandafter\expandafter
            \xdef\csname#1\endcsname{\the#2}%
   \fi
   }%
```

Figure 2. Listing of the macros for equation numbering

Loopy.TeX

J. E. Pittman

Recently, I encountered an application that required a set of nested loops and local-only assignments and definitions. TEX's `\loop...\repeat` construction proved to be inadaquate because of the requirement that the inner loop be grouped. To solve the problem, I wrote a general purpose integer 'for loop' macro, the syntax of which is simply:

`\forcount\`*csname* `=` *start* `to`
finish `by` *increment* `do`
body of the loop
`\endfor\`*csname*

The *csname* given above must be defined as a count register by a `\countdef`, `\newcount`, or `\declarecount` macro.

The 'for loop' macro utilizes general-purpose while and while-not loop macros, the syntax of both is:

`\while\`*csname conditional* `do`
body of the loop
`\endwhile\`*csname*

The *csname* can be any control sequence name that is locally unique.

A listing of the file `loopy.tex` is given in figure 1. An example file which generates a simple multiplication table and its output are shown in figures 2 and 3.

The definitions of a set of 'declare' macros, which function like non-global 'new' macros, is given in figure 4.

```
\def\forcount #1{\relax
   \def
         \for #1=##1to ##2by ##3do
            ##4%
            \endfor #1%
      {\relax
      #1=##1\relax
      \ifnum ##3>0
         \whilenot #1\ifnum ##2<#1do
            ##4%
            \advance #1 by ##3\relax
            \endwhilenot #1%
      \else
         \while #1\ifnum ##2<#1do
            ##4%
            \advance #1 by ##3\relax
            \endwhile #1%
      \fi
      }%
   \for #1%
   }%
%
\let\endwhilenot=\fi
%
\def\whilenot #1{\relax
   \def
         \whilenotloop#1 ##1do
            ##2%
            \endwhilenot #1%
      {\relax
         \expandafter\def\csname whilenotbody\string#1\endcsname{##2}%
         \expandafter\def\csname whilenotloop\string#1\endcsname
            {\relax
               ##1%
                  \let\next=\relax
               \else
```

```
                \csname whilenotbody\string#1\endcsname
                \expandafter\let\expandafter\next
                      \csname whilenotloop\string#1\endcsname
             \fi
             \next
             }%
          \csname whilenotloop\string#1\endcsname
          }%
   \whilenotloop#1
   }%
%
\let\endwhile=\fi
%
\def\while #1{\relax
   \def
         \whileloop#1 ##1do
            ##2%
            \endwhile #1%
      {\relax
         \expandafter\def\csname whilebody\string#1\endcsname{##2}%
         \expandafter\def\csname whileloop\string#1\endcsname
            {\relax
               ##1%
                  \csname whilebody\string#1\endcsname
                  \expandafter\let\expandafter\next
                        \csname whileloop\string#1\endcsname
               \else
                  \let\next=\relax
               \fi
               \next
               }%
         \csname whileloop\string#1\endcsname
         }%
   \whileloop#1
   }
```

Figure 1. Listing of the macros for looping.

```
\beginboxes{}
   \declarecount\x
   \declarecount\y
   \declarecount\z
   \column{\leftrulewidth=1.2pt \rightrulewidth=1.2pt}
   \forcount\x = 1 to 11 by 1 do
      \column{\leftrulewidth=0pt \rightrulewidth=0.4pt}
      \endfor\x
   \column{\leftrulewidth=0pt \rightrulewidth=1.2pt}
   \row{\toprulewidth=1.2pt \bottomrulewidth=1.2pt}
   \entry{$\times$}
   \forcount\x = 1 to 12 by 1 do
      \entry{\number\x}
      \endfor\x
   \forcount\y = 1 to 12 by 1 do
      \ifnum\y=12
```

```
      \row{\toprulewidth=0pt \bottomrulewidth=1.2pt}
   \else
      \row{\toprulewidth=0pt \bottomrulewidth=0.4pt}
   \fi
   \entry{\number\y}
   \forcount\x = 1 to 12 by 1 do
      \z=\x
      \multiply\z by \y
      \entry{\number\z}
      \endfor\x
   \endfor\y
\endboxes
```

Figure 2. Listing of a loopy example.

×	1	2	3	4	5	6	7	8	9	10	11	12
1	1	2	3	4	5	6	7	8	9	10	11	12
2	2	4	6	8	10	12	14	16	18	20	22	24
3	3	6	9	12	15	18	21	24	27	30	33	36
4	4	8	12	16	20	24	28	32	36	40	44	48
5	5	10	15	20	25	30	35	40	45	50	55	60
6	6	12	18	24	30	36	42	48	54	60	66	72
7	7	14	21	28	35	42	49	56	63	70	77	84
8	8	16	24	32	40	48	56	64	72	80	88	96
9	9	18	27	36	45	54	63	72	81	90	99	108
10	10	20	30	40	50	60	70	80	90	100	110	120
11	11	22	33	44	55	66	77	88	99	110	121	132
12	12	24	36	48	60	72	84	96	108	120	132	144

Figure 3. Output of figure 2.

```
\def\declarecount {\allocate0\countdef}%
\def\declaredimen {\allocate1\dimendef}%
\def\declareskip  {\allocate2\skipdef}%
\def\declaremuskip{\allocate3\muskipdef}%
\def\declarebox   {\allocate4\chardef}%
\def\declaretoks  {\allocate5\toksdef}%
%
\def\allocate#1#2#3{\relax
   \advance\count1#1 by 1
   \ifnum\count1#1<\count19
   \else
      \errmessage{No room for \string#3!}%
   \fi
   #2#3=\count1#1
   }
```

Figure 4. Listing of the declare macros.

A Page Make-up Challenge

David F. Rogers

The Problem

I have been involved with typesetting relatively complex mathematical and engineering textbooks using TEX since late 1982. These are books that are typically 5–600 pages long with an average of more than a figure per page. Further, the text is liberally endowed with large complex display equations and tables both normal and turned. TEX's page make-up abilities are woefully lacking for this application. This lack is perhaps understandable since Knuth's design goal was to develop a system capable of consistently and beautifully typesetting the volumes of the *Art of Computer Programming*. *Art of Computer Programming* volumes contain few, if any, figures, large display equations or turned tables in the text.

Publishers impose rather stringent page make-up requirements for figure placement in engineering, science and mathematical textbooks. Typical requirements in priority order are:

1. Numbered figures must be inserted in numerical sequence.
2. Numbered figures must be inserted after the first reference to the figure.
3. Numbered figures are to be placed flush left at the top or bottom of the page with minimum $1\frac{1}{2}$ pc and maximum $2\frac{1}{2}$ pc above and/or below the text.
4. Numbered figures should be visible from the first reference.
5. If page make-up places a numbered figure several pages after its first reference, then and only then, may it be placed *before* its first reference. However, it *must be visible* from its first reference.

Figure caption rules somewhat further complicate the problem. Examples for a standard 29 pc page width are:

1. Numbered figures 19 to 29 pc wide have the figure caption positioned flush left 1 pc below the figure × the page measure (hsize).
2. Numbered figures less than 19 pc wide have the figure caption positioned 1 pc to the right of the figure × the remainder of the page width and base-aligned with the figure.
3. Sequentially numbered figures less than 13 pc wide are placed side-by-side in 13 pc wide boxes separated by a 3 pc space. Figure captions are placed flush left 1 pc below each box.

Considering that TEX's output routines do not look ahead very well, it is easy to see that such page make-up rules seriously complicate the task of typesetting and making-up a book of this nature.

The Current Solution

In applications of this nature both Plain TEX's `\topinsert` and `\midinsert` commands are known not to work. Further, LATEX's floating insert commands also do not work. Consequently, it is necessary to essentially do the page make-up by hand *using a computer*! The technique (TEXnique??) is conceptually simple and very labor intensive. The manuscript is broken up into 30–50 page segments within chapter boundaries (a chapter is assumed to always begin on a recto page). The segment is TEX'd. For the pages preceding the first figure reference, white space is inserted or deleted both to balance the length of facing pages and to keep the page length within acceptable limits. No, TEX does not always do it quite correctly, i.e. according to the page make-up rules. Immediately after the first figure reference, white space equal to the figure size is inserted along with the figure caption. If there is sufficient space at the bottom of the page containing the figure reference, called the current page, it is inserted there, if not, it is moved to the top of the next page.

To reiterate, conceptually this technique is quite easy; in practice it is quite difficult. Adding white space and the figure caption to the bottom of the current page must be done by measuring up from the bottom of the page, finding the exact end of line corresponding to the required figure space plus the space occupied by the figure caption and inserting the white space and figure caption at that point in the manuscript. To prevent TEX from reformatting the pages to this point a `\vfill\eject` is placed at the bottom of the previous page. This, of course, does not always work. TEX occasionally decides that the previous material is best presented with an incomplete last line! When this happens material must be moved — *word-by-word* — from the current page to ahead of the `\vfill\eject` on the previous page until the result is correct. A similar technique is used when the figure is placed at the top of a page. A combination of these techniques is used when both a top and bottom figure appear on the same page. The *fun* really begins when a page contains large display equations or large numbers of display equations and both top and

bottom figures. The result is a long, possibly nonconverging, iteration process.

As Reference 1 illustrates, doing a book of this type with TEX is quite possible. It is just a bit painful. Unfortunately, it is also not cost effective. Currently, it is less expensive for a book publisher to simply typeset the manuscript using TEX without including figure captions or spaces and use the traditional Xacto knife and glue pot page make-up technique. That offends me as I am sure it does you.

The Challenge

My initial thought was to simply write a `\bottominsert` macro similar to the `\topinsert` and `\midinsert` macros. However, discussions with output routine gurus at the recent Montreal TUG meeting have convinced me that this will not work, at least not very well.

The Challenge then is for the output macro gurus to write a figure placement macro that incorporates items 1–3 above (4 and 5 can be handled manually). The suggested calling sequence for the macro is

```
\figplace#1#2#3#4
```

where

`#1` is the vertical dimension of the white space to be left for the figure.

`#2` is the horizontal dimension of the white space to be left for the figure.

`#3` is the figure number.

`#4` is the figure caption.

The assumption is that a custom figure caption macro is used within the figure placement macro. A sample figure caption macro might be:

```
% define a figure caption macro.
% #1 is the figure number.
% #2 is the caption.
% the caption is to be set in a 'box'
%    left and right justified 1em to
%    the right of the figure number.
% the size of the box containing the word
%    Figure; its number and the 1em
%    skip are found in box0.
% box1 is \hsize less the width of box0.
% a \vtop is used along with an \halign
%    to obtain the flush left and right effect.
% \spaceskip is used to help in preventing
%    overfull lines.
\def\figcap#1#2{{%
  \setbox0=\hbox{{\bf Figure #1}\hskip 1em}%
  \setbox1=\vtop{%
    \advance \hsize by-\wd0 \noindent
    \spaceskip=.3em plus.2em minus.2em #2}%
  \halign{## & ## \cr
    \box0 & \box1 \cr}}%
\bigskip
}
```

Unfortunately, other commitments as well as my current level of expertise prevent me from attempting this job.

References

1. Rogers, David F. *Procedural Elements for Computer Graphics*, McGraw-Hill Book Co., New York, 1985.

LaTeX

Contents of LaTeX Style Collection as of 24 September 1988

Michael DeCorte
Clarkson University

The LaTeX style collection has been moved. It now resides at `sun.soe.clarkson.edu`. The collection will now be maintained by Michael DeCorte, a Clarkson University computer science student. As always, submissions are encouraged. Mail should be sent to

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

A few changes have been made in the collection that should be noted. First, the names of the files that users should access are `Index`, `Description` and `Readme`. Their contents are described below. Also, if a `.doc` file exists it will be renamed to `.sty` and the original `.sty` will be removed. This change does not affect the style file's use. If you must squeeze the last bit of performance out of your machine, get `docsty.shar`.

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.

Non-Internet users: how to retrieve by mail

To retrieve files or help documentation, send mail to `archive-server@sun.soe.clarkson.edu` with the body of the mail message containing the command `help` or `index` or `send`. The `send` command must be followed by `latex-style` and then the files you want. Users who are not in the uucp maps database are strongly encourage to include a `path` command followed by a path from clarkson to you. If you don't include a `path` command, your mail may not get to you and will definitely be delayed as Michael will have to mail it by hand. You should realize that Clarkson does not have a uucp connection; therefore you must send it to an Internet or Bitnet host that does have a uucp connection. For example, `host!user@uunet.uu.net` should send:

```
To: archive-server@clarkson.edu
Subject:
path host!user@uunet.uu.net
send latex-style Readme Index
send latex-style resume.sty
```

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

- Bitnet users: Texas A&M maintains a list- 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 tamvm1 get tex filelist`
- UK users: A comprehensive repository of TeX-related material is maintained at Aston University. An article by Peter Abbott in this issue (page 263) gives general instructions for accessing the archive.

Additional volunteers should contact Michael.

Distribution for IBM PC and clone users

There are two sources.

- David W. Hopper

Sample ftp session for Internet users

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

446 Main Street
Toronto, Ontario
Canada M4C 4Y2

has LaTeX style files only.

1. Either one 1.2 MB diskette or three 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 utilities. For a list or other info send a SASE.

1. 360 KB diskettes, blank and formatted.
2. A stamped, self-addressed mailer, and
3. $1.50 per disk. If you live outside North America, skip the stamps and send additional money or International Reply Coupons.

As a convenience for people who have more money than floppies, Jon will supply everything for $6.00 per disk to U.S./Canada/Mexico addresses.

Contents of the collection

`Description` A list of all the files in alphabetical ordering with a brief description of each

`Index` A list of all the files in reverse chronological ordering

`Readme` Documentation on how to get files

`a4.sty` Set page size to A4

`a4wide.sty` Adjusts width to suit A4

`a5.sty` Sets A5 page size (use only with 10pt)

`a5comb.sty` Same, but for spirally-bound documents (bigger inner margins)

`aaai-instructions.tex` Instructions to authors

`aaai-named-0.99.bst` BibTeX style to accompany `aaai.sty`, for version 0.99

`aaai-named-0.98.bst` For version 0.98

`aaai.sty` Style file for AAAI conference 1987

`acm.bst` ACM BibTeX style

`agugrl.sty` AGU Geophysical Research Letters style, sample

`agujgr.sty` AGU Journal of Geophysical Research style, sample

`album.shar` Style for printing cassette labels

`alltt.sty` Like verbatim, but permits other commands inside

`amssymbols.sty` Load AMS symbol fonts

`apalike.sty` American Psychological Association style files

`apalike.bst` BibTeX file for `apalike` for version 0.99

`article.txt` Standard files in text format with `art10.txt` places to make language specific `art11.txt` changes indicated `art12.txt`

`biihead.sty` Underlined heading

`boxedminipage.sty` Puts a box round a minipage

`bsf.sty` Provide access to bold sans serif fonts in LaTeX

`btxdoc.tex` Documentation on how to use BibTeX

`btxhak.tex` Documentation on how to program BibTeX

`btxdoc.bib` Bibliography needed to LaTeX `btxdoc.tex` and `btxhax.tex`

`captcont.sty` Auxiliary file needed by files described in `local-suppl`

`cyrillic.sty` Load cyrillic font

`dayofweek.tex` Macros to compute day of week and phase of moon. Examples of how to use TeX arithmetic capabilities.

`deproc.sty` DECUS Proceedings style

`deprocldc.tex` Paper that describes the above

`docsty.shar` Program to convert `.doc` to `.sty` by stripping comments

`doublespace.sty` Double spacing in text

`draft.sty` Draft option for documents for "debugging"

`drafthead.sty` Prints DRAFT in heading

`drop.sty` Style for making large dropped initials for starting paragraphs

`dvidoc.shar1` DVI to character device filter for Unix BSD systems

`dvidoc.shar2` part 2 of the above file

`eepic10.shar` A picture environment that used tpic specials
`epic.shar1` An extended picture environment
`epic.shar2` part 2 of the above file
`espo.sty` Style file for Esperanto
`fig2epic1c.shar` Converts fig code to epic or eepic files
`fixup.sty` Fixup Plain's `\bigl`, etc., to track LaTeX size changes
`fnpara.tex` Sets footnotes as paragraphs
`format.sty` Print FP numbers in fixed format
`fullpage.sty` Get more out of a page
`geophysics.sty` Geophysics journal style
`german.sty` Style file for German
`ieeetr.bst` IEEE Transactions BibTeX style
`insertplot.readme` Documentation on `insertplot.sty`
`insertplot.sty` For inserting PostScript in files printed with Arbor DVIPS
`ist21.sty` IST21 document style option for cover page
`latex.bug` Latest listing of bugs found in LaTeX
`layout.readme` Prints nice diagram showing page parameters
`layout.tex`
`lcustom.tex` Useful macros and definitions for LaTeX
`lfonts_ams.readme` Use AMS symbols in LaTeX
`lfonts_ams.tex`
`lgraph.shar` Data to graph command filter in Pascal
`local-suppl.tex` Supplement to local guide; describes `tgrind`, `sfwmac`, `trademark`, `lcustom`, `xxxcustom`, and `xxxslides`
`manual.readme` Like "book" but for manuals.
`manual.sty` Need to look at "book"
`man10.sty` for documentation
`man11.sty`
`man12.sty`
`memo.sty` Memo style option
`merge.sty` Form letter option to LaTeX letter style
`mfr.sty` Modifier to `memo.sty`
`mitthesis.sty` Massachusetts Institute of Technology thesis format
`mitthesis-sample.tex` sample for above
`natsci.bst` Natural sciences generic BibTeX style
`natsci.sty` Formats citations created with `natsci.bst`
`newalpha.bst` Modified alphabetic BibTeX style
`nl.sty` Style file customized for Dutch
`nopagenumbers.sty` Remove page numbers
`pcwritex.shar` PC-Write to TeX interface. Contains control characters.
`pslatex.shar` Macros to build pslatex, a LaTeX that uses printer resident PostScript fonts. Requires dvi2ps that understands PostScript fonts.
`remark.sty` Like newtheorem but no `\it`
`resume.sty` Format for doing resumes
`resume-sample.tex` Sample file for above
`romanneg.sty` Roman-numbered pages get negative page numbers (useful when selecting only part of a document to be printed)
`rscsencode.shar` RSCS en/decoder
`sc21.sty` ISO/TC97/SC21 document style
`sc21-wg1.sty` Option for cover page
`schedule.sty` Style for generating schedule sheets
`sfwmac.sty` Useful macros for Unix documentation
`shapiro-btxbst-0.98.sty`
`shapiro-btxbst-0.98.readme`
`shapiro-makebst.sh` A master file for BibTeX styles with standard styles and some new ones. Also a Unix sh script to generate the styles
`showlabels.sty` Shows labels and references to them
`select.tex` Selectively print pages in a TeX document
`semitic.sty` Used to set Semitic languages
`siam.bib` BibTeX file for `siam.tex`
`siam.bst` SIAM BibTeX style
`siam.sty` SIAM LaTeX style file
`siam.tex` Documentation for `siam.sty`
`siam10.sty`

`siam11.sty`
`siam12.sty`
`slem.sty` Change `\sl` to `\em`
`spacecites.sty`
Modified to give spacing between citations
`suthesis.sty` Stanford U thesis style
`svma.sty` Style for Springer-Verlag reports, multi-author
`svsa.sty` Springer-Verlag, single author
`svma.tex` the manual for svma
`tabledoc.tex` Documentation for `tables.sty`
`tables.sty` Ruled and unruled tables made easy
`texindex.shar`
Style file and processor for index entries for VMS
`texnames.sty` Define a couple more TEX names
`tgrind.sty` Tgrind macros for LATEX instead of TEX
`threepart.sty`
Three part page headers
`titlepage.txt`
Style file in text format to go with `article.txt`
`trademark.sty`
Definitions of common trademarks
`uct10.sty` U of California thesis style
`uct11.sty`
`uct12.sty`
`ucthesis.sty`
`ucthesis.readme`
`uuencode.shar`
uu en/decoder to assist file transfers
`vdm.sty` Vienna Development Method LATEX style
`vdm.tex` documentation on above
`wsltex.shar` Wordstar to LATEX filter, C and Pascal versions
`xxxcustom.tex`
Supplementary macros for `xxx-tex`, for some `xxx`
`xxxslides.sty`
Supplementary macros for SLITEX, includes `slides.sty`

The LATEX Column

Jackie Damrau
University of New Mexico

Please keep those questions or helpful hints coming. Remember, they will be answered as soon as possible via electronic mail (if possible) and then published in the next TUGboat. Until then, happy LATEXing.

Question 1

```
\begin{titlepage} \\
\maketitle \\
\end{titlepage}
```

Although this is not a usage mentioned in The Manual, it is not discouraged thereby and it is, I feel, a very reasonable thing for Ms/Mr Naive User to input; indeed, it works fine–except that the page gets "headed and footed" and number "0" (contradicting the first paragraph of Section 5.3.1).

This happens because `\maketitle` starts with (at least) one `\newpage`: I should like to query whether these are needed.

Also, whilst on the subject of `\maketitle`, why does it zealously "zero out" `\@title`, `\@author`, etc. and even itself! What dreadful consequences would ensure if this was not done? Is it perhaps, merely to prevent such terrible solecisms as having the title appear more than once in a document?

Chris Rowley
The Open University
JANET: ca_rowley\@vax.acs.open.ac.uk

Answer from Leslie Lamport:

From page 84:

> The `\maketitle` command ... is described in Sections 2.2.2 and C.4.3. You can also create your own title page with the `titlepage` environment. ...
>
> You are completely responsible for what appears on a title page made with the `titlepage` environment.

While this admittedly doesn't explicitly say that you **CAN'T** use `\maketitle` in a titlepage environment, it should at least lead the reader to suspect that this might be the case.

I can see no reason to use two instances of the `\maketitle` command defined by the standard styles, so that command "zeros out" the definitions to save a little space. The values of `\@title`, etc. would be of use only to commands defined by a document style; any style designer who wants to use the values elsewhere should redefine `\maketitle` so it preserves them.

A new implementation of the array– and tabular–environments

Frank Mittelbach
Johannes Gutenberg Universität
D–6500 Mainz

Abstract

This article describes a new implementation of the LaTeX array– and tabular–environments. The special merits of this implementation are further options to format columns and the fact that fragile LaTeX–commands don't have to be `\protect`'ed any more within those environments.

At the same time it shows a new — and in our opinion sensible — way of documenting TeX-macros: This article is the style–file that is to be used. All we need in addition to it is a short TeX–program which visualizes the comments and puts the definitions in verbatim mode.

Introduction

First we will define the current version of this file:

```
\typeout{Style-Option: 'array' v1.9g \space\space <24.6.88> (F.M.)}
\typeout{English documentation dated \space\space <24.6.88> (F.M.)}
```

This new implementation of the array– and tabular–environments is part of a larger project in which we are trying to improve the LaTeX-code in some aspects and to make LaTeX even easier to handle. At the moment we are experimenting with a version where all commands are automatically robust.

The reader should be familiar with the general structure of the environments mentioned above. Further information can be found in LAMPORT [3]. The additional options which can be used in the preamble as well as those which now have a slightly different meaning are described in Table 1.

`p{width}`	Defines a column of width `width`. Every entry will be centered in proportion to the rest of the line. It is somewhat like `\parbox{width}`. In the original definition `p{..}` was a topaligned parbox.
`t{width}`	Equivalent to `\parbox[t]{width}`, the former `p`–option.
`b{width}`	Coincides with `\parbox[b]{width}`.
`>{decl.}`	Can be used before an `l`, `r`, `c`, `p`, `t` or a `b` option. It inserts `decl.` directly in front of the entry of the column.
`<{decl.}`	Can be used after an `l`, `r`, `c`, `p{..}`, `t{..}` or a `b{..}` option. It inserts `decl.` right after the entry of the column.
`|`	Inserts a vertical line. The distance between two columns will be enlarged by the width of the line in contrast to the original definition of LaTeX.
`!{decl.}`	Can be used anywhere and corresponds with the `|` option. The difference is that `decl.` is inserted instead of a vertical line, so this option doesn't suppress the normally inserted space between columns in contrast to `@{...}`.

Table 1: The new preamble options.

Additionally we introduce a new parameter called `\extrarowheight`. If it takes a positive length, the value of the parameter is added to the normal height of every row of the table, while the depth will remain the same. This is important for tables with horizontal lines because those lines normally touch the capital letters. For example, we used `\extrarowheight=1pt` in Table 1.

We will discuss a few examples using the new preamble options before dealing with the implementation.

- If you want to use a special font (for example `\bf`) in a flushed left column, this can be done with `>{\bf}l`. You do not have to begin every entry of the column with `\bf` any more.
- In columns which have been generated with `p`, `t` or `b`, the default value is `\parindent=0pt`. This can be changed with `>{\parindent=1cm}p`.
- The `<`–option was originally developed for the following application: `>{$}c<{$}` generates a column in math mode in a tabular–environment. If you use this type of a preamble in an array–environment, you get a column in LR mode because the additional `$`'s cancel the existing `$`'s.
- One can also think of more complex applications. A problem which has been mentioned several times in TEXhax can be solved with `>{\centerdots}c <{\endcenterdots}`. To center decimals at their decimal points you (only?) have to define the following macros:

```
{\catcode`\.=\active\gdef.{\egroup\setbox2=\hbox\bgroup}}
\def\centerdots{\catcode`\.=\active\setbox0=\hbox\bgroup}
\def\endcenterdots{\egroup\ifvoid2 \setbox2=\hbox{0}\fi
   \ifdim \wd0>\wd2 \setbox2=\hbox to\wd0{\unhbox2\hfill}\else
     \setbox0=\hbox to\wd2{\hfill\unhbox0}\fi
   \catcode`\.=12 \box0.\box2}
```

- Using `c!{\hspace{1cm}}c` you get space between two columns which is enlarged by one centimeter, while `c@{\hspace{1cm}}c` gives you exactly one centimeter space between two columns.

These examples should be sufficient to demonstrate the use of the new preamble options.

It is obvious that those environments will consist mainly of an `\halign`, because TEX typesets tables using this primitive. That is why we will now take a look at the algorithm which determines a preamble for a `\halign` starting with a given user preamble using the options mentioned above.

The construction of the preamble

The most interesting macros of this implementation are without doubt those which are responsible for the construction of the preamble for the `\halign`. The underlying algorithm was developed by LAMPORT (resp. KNUTH, see TEXhax V87#??), and it has been extended and improved.

The user preamble will be read token by token. A token is a single character like `c` or a block enclosed in `{...}`. For example the preamble of `\begin{tabular}{lc||c@{\hspace{1cm}}}` consists of the token `l`, `c`, `|`, `|`, `@` and `\hspace{1cm}`.

The currently used token and the previous one are needed to decide on how the construction of the preamble has to be continued. In the example mentioned above the `l` causes the preamble to begin with `\hskip\tabcolsep`. Furthermore `#\hfil`

would be appended to define a flush left column. The next token is a c. Because it was preceded by an l it generates a new column. This is done with `\hskip\tabcolsep & \hskip\tabcolsep`. The column which is to be centered will be appended with `\hfil #\hfil`. The token | would then add a space of `\hskip\tabcolsep` and a vertical line because the last tokens was a c. The following token | would only add a space `\hskip\doublerulesep` because it was preceded by the token |. We will not discuss our example further but rather take a look at the general case of constructing preambles.

The example shows that the desired preamble for the `\halign` can be constructed as soon as the actions of all combinations of the preamble tokens are specified. There are 18 such tokens so we have $19 \cdot 18 = 342$ combinations if we count the beginning of the preamble as a special token. Fortunately, there are many combinations which generate the same spaces, so we can define token classes. We will identify a token within a class with a number, so we can insert the formatting (for example of a column). Table 2 lists all token classes and their corresponding numbers.

token	\@chclass	\@chnum	token	\@chclass	\@chnum
c	0	0	Start	4	—
l	0	1	@-arg	5	—
r	0	2	!	6	—
p-arg	0	3	@	7	—
t-arg	0	4	<	8	—
b-arg	0	5	>	9	—
|	1	0	p	10	3
!-arg	1	1	t	10	4
<-arg	2	—	b	10	5
>-arg	3	—			

Table 2: Classes of preamble tokens

`\@chclass` `\@chnum` `\@lastchclass` The class and the number of the current token are saved in the count registers `\@chclass` and `\@chnum`, while the class of the previous token is stored in the count register `\@lastchclass`. All of the mentioned registers are already allocated in `latex.tex`. This is why the following three lines of code are commented out. Later, throughout the text, I will not mention again explicitly whenever I use a % sign that these parts are already defined in `latex.tex`.

```
% \newcount \@chclass
% \newcount \@chnum
% \newcount \@lastchclass
```

`\@addtopreamble` We will save the already constructed preamble for the `\halign` in the global macro `\@preamble`. This will then be enlarged with the command `\@addtopreamble`.

```
\def\@addtopreamble#1{\xdef\@preamble{\@preamble #1}}
```

The character class of a token

`\@testpach` With the help of `\@lastchclass` we can now define a macro which determines the class and the number of a given preamble token and assigns them to the registers `\@chclass` and `\@chnum`.

```
\def\@testpach#1{\@chclass
```

First we deal with the cases in which the token (#1) is the argument of !, @, < or >. We can see this from the value of \@lastchclass:

```
\ifnum \@lastchclass=6 \@ne \@chnum \@ne \else
 \ifnum \@lastchclass=7 5 \else
  \ifnum \@lastchclass=8 \tw@ \else
   \ifnum \@lastchclass=9 \thr@@
```

Otherwise we will assume that the token belongs to the class 0 and assign the corresponding number to \@chnum if our assumption is correct.

```
\else \z@
```

If the last token was a p, t or a b, \@chnum already has the right value. This is the reason for the somewhat curious choice of the token numbers in class 10.

```
\ifnum \@lastchclass = 10 \else
```

Otherwise we will check if #1 is either a c, l or an r.

```
\@chnum
\if #1c\z@ \else
 \if #1l\@ne \else
  \if #1r\tw@ \else
```

If it is a different token, we know that the class was not 0. We assign the value 0 to \@chnum because this value is needed for the |–token. Now we must check the remaining classes. Note that the value of \@chnum is insignificant here for most classes.

```
\z@ \@chclass
\if #1|\@ne \else
 \if #1!6 \else
  \if #1@7 \else
   \if #1<8 \else
    \if #1>9 \else
```

The remaining permitted tokens are p, t and b (class 10).

```
10
\@chnum
\if #1p\thr@@ \else
 \if #1t4 \else
  \if #1b5 \else
```

Now the only remaining possibility is a forbidden token, so we choose class 0 and number 0 and give an error message. Then we finish the macro by closing all \if's.

```
\z@ \@chclass \z@ \@preamerr \z@ \fi \fi \fi \fi
\fi \fi  \fi  \fi  \fi  \fi  \fi \fi \fi \fi \fi \fi}
```

Multiple columns (∗–form)

\@xexpast Now we discuss the macro that deletes all forms of type ∗{*N*}{*String*} from a user preamble and replaces them with *N* copies of *String*. Nested ∗–expressions are dealt with correctly; that means ∗–expressions are not substituted if they are in explicit braces, as in @{∗}.

This macro is called via \@xexpast⟨*preamble*⟩∗0x\@@. The ∗–expression ∗0x is being used to terminate the recursion, as we shall see later, and \@@ serves as an argument delimiter. \@xexpast has four arguments. The first one is the part of the user preamble before the first ∗–expression while the second and third ones are the arguments of the first ∗–expression (that is *N* and *String* in the notation mentioned above). The fourth argument is the rest of the preamble.

```
\def\@xexpast#1*#2#3#4\@@{%
```

The number of copies of *String* that are to be produced (#2) will be saved in a count register.

```
\@tempcnta #2
```

We save the part of the preamble which does not contain a *–form (#1) in a *Plain*TEX token register. We also save *String* (#3) using a LATEX token register.

```
\toks@={#1}\@temptokena={#3}%
```

Now we have to use a little trick to produce N copies of *String*. We could try `\def\@tempa{#1}` and then N times `\edef\@tempa{\@tempa#3}`. This would have the undesired effect that all macros within #1 and #3 would be expanded, although, for example, constructions like `@{..}` are not supposed to be changed. That is why we `\let` two control sequences to be equivalent to `\relax`.

```
\let\@thetoksz\relax \let\@thetoks\relax
```

Then we ensure that `\@tempa` contains `{\@thetoksz\@thetoks...\@thetoks}` (the macro `\@thetoks` exactly N times) as substitution text.

```
\def\@tempa{\@thetoksz}%
\ifnum\@tempcnta >0 \@whilenum\@tempcnta >0\do
  {\edef\@tempa{\@tempa\@thetoks}\advance \@tempcnta \m@ne}%
```

If N was greater than zero we prepare for another call of `\@xexpast`. Otherwise we assume we have reached the end of the user preamble, because we had appended `*0x\@@` when we first called `\@xexpast`. In other words: if the user inserts `*{0}{..}` in his preamble, LATEX ignores the rest of it.

```
\let \@tempb \@xexpast \else
\let \@tempb \@xexnoop \fi
```

Now we will make sure that the part of the user preamble, which was already dealt with, will be saved again in `\@tempa`.

```
\def\@thetoksz{\the\toks@}\def\@thetoks{\the\@temptokena}%
\edef\@tempa{\@tempa}%
```

We have now evaluated the first *–expression, and the user preamble up to this point is saved in `\@tempa`. We will put the contents of `\@tempa` and the rest of the user preamble together and work on the result with `\@tempb`. This macro either corresponds to `\@xexpast`, so that the next *–expression is handled, or to the macro `\@xexnoop`, which only ends the recursion by deleting its argument.

```
\expandafter \@tempb \@tempa #4\@@}
```

`\@xexnoop` So the first big problem is solved. Now it is easy to specify `\@xexnoop`. Its argument is delimited by `\@@` and it simply expands to nothing.

```
% \def\@xexnoop#1\@@{}
```

The insertion of declarations (>, <, !, @)

The preamble will be enlarged with the help of `\xdef`, but the arguments of >, <, ! and @ are not supposed to be expanded during the construction (we want an implementation that doesn't need a `\protect`). So we have to find a way to inhibit the expansion of those arguments.

We will solve this problen with token registers. We need one register for every ! and @, while we need two for every c, l, r, t, p or b. This limits the number of columns of a table because there are only 256 token registers. But then, who needs tables with more than 100 columns?

One could also find a solution which only needs two or three token registers by proceeding similarly as in the macro `\@xexpast` (see page 301). The advantage of our approach is the fact that we avoid some of the problems that arise with the other method[1].

[1]Maybe there are also historical reasons.

So how do we proceed? Let us assume that we had `!{foo}` in the user preamble and say we saved `foo` in token register 5. Then we call `\@addtopreamble{\@thetoks5}` where `\@thetoks` is defined in a way that it does not expand (for example it could be equivalent to `\relax`). Every following call of `\@addtopreamble` leaves `\@thetoks5` unchanged in `\@preamble`. If the construction of the preamble is completed we change the definition of `\@thetoks` to `\the\toks` and expand `\@preamble` for the last time. During this process all parts of the form `\@thetoks`⟨*Number*⟩ will be substituted by the contents of the respective token registers.

As we can see from this informal discussion the construction of the preamble has to take place within a group, so that the token registers we use will be freed later on. For that reason we keep all assignments to `\@preamble` global; therefore the replacement text of this macro will remain the same after we leave the group.

`\count@` We further need a count register to remember which token register is to be used next. This will be initialized with -1 if we want to begin with the token register 0. We use the *Plain*TEX scratch register `\count@` because everything takes place locally. All we have to do is insert `\@thetoks \the\count@` into the preamble. `\@thetoks` will remain unchanged and `\the\count@` expands into the saved number.

`\prepnext@tok` The macro `\prepnext@tok` is in charge of preparing the next token register. For that purpose we increase `\count@` by 1:

```
\def\prepnext@tok{\advance \count@ \@ne
```

Then we locally delete any contents the token register might have.

```
\toks\count@={}}
```

`\save@decl` During the construction of the preamble the current token is always saved in the macro `\@nextchar` (see the definition of `\@mkpream` on page 304). The macro `\save@decl` saves it into the next free token register, i.e. in `\toks\count@`.

```
\def\save@decl{\toks \count@ = \expandafter
      {\expandafter \relax \@nextchar}}
```

The reason for the use of `\relax` is the following hypothetical situation in the preamble: `..\the\toks1\the\toks2..` TEX expands `\the\toks2` first in order to find out if the digit 1 is followed by other digits. E.g. a 5 saved in the token register 2 would lead TEX to insert the contents of token register 15 instead of 1 later on.

What should happen if we want to add another column to the preamble, i.e. if we have found a `c`, `l`, `r`, `t`, `p` or `b` in the user preamble ? In this case we have the problem that the token register from `>{..}` and `<{..}` has to be inserted at this moment because formatting instructions like `\hfil` have to be set around them. On the other hand it is not known yet, if any `<{..}` instruction will appear in the user preamble at all.

We solve this problem by adding two token registers at a time. This explains why we have freed the token registers in `\prepnext@tok`.

`\insert@column` We now define the macro `\insert@column` which will do this work for us.

`\@sharp`

```
\def\insert@column{%
```

Here, we assume that the count register `\@tempcnta` has saved the value `\count@` -1.

```
\@thetoks \the\@tempcnta
```

Next follows the # sign which specifies the place where the text of the column shall be inserted. To avoid errors during the expansions in `\@addtopreamble` we hide this sign in the command `\@sharp` which is temporarily occupied with `\relax` during the build-up of the preamble. To remove unwanted spaces before and after the column text, we set an `\ignorespaces` in front and a `\unskip` afterwards.

```
\ignorespaces \@sharp \unskip
```

Then the second token register follows whose number should be saved in \count@.

```
\@thetoks \the\count@}
```

The separation of columns

\@addamp In the preamble a & has to be inserted between any two columns; before the first column there should not be a &. As the user preamble may start with a | we have to remember somehow if we have already inserted a # (i.e. a column). This is done with the boolean variable \if@firstamp that we test in \@addamp, the macro that inserts the &.

```
%    \newif \@iffirstamp
%    \def\@addamp{\if@firstamp \@firstampfalse
%                 \else \@addtopreamble &\fi}
```

\@acol \@acolampacol \col@sep We will now define some abbreviations for the extensions that appear most often in the preamble build-up. Here \col@sep is a dimen register which is set equivalent to \arraycolsep in an array–environment; otherwise it is set equivalent to \tabcolsep.

```
\newdimen\col@sep
\def\@acol{\@addtopreamble{\hskip\col@sep}}
%    \def\@acolampacol{\@acol\@addamp\@acol}
```

The macro \@mkpream

\@mkpream Now we can define the macro which builds up the preamble for the \halign. First we initialize \@preamble, \@lastchclass and the boolean variable \if@firstamp.

```
\def\@mkpream#1{\gdef\@preamble{}\@lastchclass 4 \@firstamptrue
```

During the build-up of the preamble we cannot directly use the # sign; this would lead to an error message in the next \@addtopreamble call. Instead, we use the command \@sharp at places where later a # will be. This command is at first given the meaning \relax; therefore it will not be expanded when the preamble is extended. In the macro \@array, shortly before the \halign is carried out, \@sharp is given its final meaning.

We deal with the commands \@startpbox and \@endpbox in a similar way, although the reason is different here: these macros expand to many tokens which would delay the build-up of the preamble.

```
\let\@sharp\relax \let\@startpbox\relax \let\@endpbox\relax
```

Now we remove possible *-forms in the user preamble with the command \@xexpast. As we already know, this command saves its result in the macro \@tempa.

```
\@xexpast #1*0x\@@
```

Afterwards we initialize all registers and macros that we need for the build-up of the preamble. Since we want to start with the token register 0, \count@ has to contain the value −1.

```
\count@\m@ne
\let\@thetoks\relax
```

Then we call up \prepnext@tok in order to prepare the token register 0 for use.

```
\prepnext@tok
```

To evaluate the user preamble (without stars) saved in \@tempa we use the LaTeX–macro \@tfor. The strange-looking construction with \expandafter is based on the fact that we have to put the replacement text of \@tempa and not the macro \@tempa to this LaTeX–macro.

```
\expandafter \@tfor \expandafter \@nextchar
 \expandafter :\expandafter =\@tempa \do
```

The body of this loop (the group after the \do) is executed for one token at a time, whereas the current token is saved in \@nextchar. At first we evaluate the current token with the already defined macro \@testpach, i.e. we assign to \@chclass the character class and to \@chnum the character number of this token.

```
{\@testpach\@nextchar
```

Then we branch out depending on the value of \@chclass into different macros that extend the preamble appropriately.

```
\ifcase \@chclass \@classz \or \@classi \or \@classii
  \or \save@decl \or \or \@classv \or \@classvi
  \or \@classvii \or \@classviii  \or \@classix
  \or \@classx \fi
```

Two cases deserve our special attention: Since the current token cannot have the character class 4 (start) we have skipped this possibility. If the character class is 3, only the content of \@nextchar has to be saved into the current token register; therefore we call up \save@decl directly and save a macro name. After the preamble has been extended we save the value of \@chclass in the counter \@lastchclass to assure that this information will be available during the next run of the loop.

```
\@lastchclass\@chclass}%
```

After the loop has been finished space must still be added to the created preamble, depending on the last token. Depending on the value of \@lastchclass we perform the necessary operations.

```
\ifcase\@lastchclass
```

If the last class equals 0 we add a \hskip\col@sep.

```
    \@acol
```

If it equals 1 we do not add any additional space so that the horizontal lines do not exceed the vertical ones.

```
\or
```

Class 2 is treated like class 0 because a <{...} can only directly follow after class 0.

```
\or \@acol
```

Most of the other possibilities can only appear if the user preamble was defective. Class 3 is not allowed since after a >{..} there must always follow a c, l, r, p, t or b. We report an error and ignore the declaration given by {..}.

```
\or \@preamerr \thr@@
```

If \@lastchclass is 4 the user preamble has been empty. To continue, we insert a # in the preamble.

```
\or \@preamerr \tw@ \@addtopreamble\@sharp
```

Class 5 is allowed again. In this case (the user preamble ends with @{..}) we need not do anything.

```
\or
```

Any other case means that the arguments to @, !, <, >, p, t or b have been forgotten. So we report an error and ignore the last token.

```
\else  \@preamerr \@ne \fi
```

Now that the build-up of the preamble is almost finished we can insert the token registers and therefore redefine \@thetoks. The actual insertion, though, is performed later.

```
\def\@thetoks{\the\toks}}
```

The macros \@classz to \@classx

The preamble is extended by the macros \@classz to \@classx which are called by \@mkpream depending on \@lastchclass (i.e. the character class of the last token).

\@classx First we define \@classx because of its important rôle. When it is called we find that the current token is p, t or b. That means that a new column has to start.

```
\def\@classx{%
```

Depending on the value of \@lastchclass different actions must take place:

```
  \ifcase \@lastchclass
```

If the last character class was 0 we separate the columns by \hskip\col@sep followed by & and another \hskip\col@sep.

```
    \@acolampacol
```

If the last class was class 1 — meaning that a vertical line was drawn, — before this line a \hskip\col@sep was inserted. Therefore there has to be only a & followed by \hskip\col@sep. But this & may be inserted only if this is not the first column. This process is controlled by \if@firstamp in the macro \addamp.

```
  \or \@addamp \@acol
```

Class 2 is treated like class 0 because <{...} can only follow after class 0.

```
  \or \@acolampacol
```

Class 3 requires no actions because everything necessary has been done by the preamble token >.

```
  \or
```

Class 4 means that we are at the beginning of the preamble. Therefore we start the preamble with \hskip\col@sep and then call \@firstampfalse. This makes sure that a later \@addamp inserts the character & into the preamble.

```
  \or \@acol \@firstampfalse
```

For class 5 tokens only the character & is inserted as a column separator. Therefore we call \@addamp.

```
  \or \@addamp
```

Other cases are impossible. For an example \@lastchclass = 6 — as it might appear in a preamble of the form ...!p... — p would have been taken as an argument of ! by \@testpach.

```
  \fi}
```

\@classz If the character class of the last token is 0 we have c, l, r or an argument of t, b or p. In the first three cases the preamble must be extended the same way as if we had class 10. The remaining two cases do not require any action because the space needed was generated by the last token (i.e. t, b or p). Since \@lastchclass has the value 10 at this point nothing happens when \@classx is called. So the macro \@classz may start like this:

```
\def\@classz{\@classx
```

Acording to the definition of \insert@column we must store the number of the token register in which a preceding >{..} might have stored its argument into \@tempcnta.

```
  \@tempcnta \count@
```

To have \count@ = \@tmpcnta + 1 we prepare the next token register.

```
  \prepnext@tok
```

Now the preamble must be extended with the column whose format can be determined by \@chnum.

```
  \@addtopreamble{\ifcase \@chnum
```

If \@chnum has the value 0 a centered column has to be generated. So we begin with stretchable space.

```
    \hfil
```

The command \d@llar follows expanding into nothing (in the tabular-environment) or into $. By providing an appropriate setting of \d@llar we achieve that the contents of the columns of an array-environment are set in math mode while those of a tabular-environment are set in LR mode.

```
    \d@llar
```

Now we insert the contents of the two token registers and the symbol for the column entry (i.e. # or more precisely \@sharp) using \insert@column.

```
\insert@column
```

We end this case with another \d@llar \hfil.

```
\d@llar \hfil
```

The templates for l and r (i.e. \@chnum 1 or 2) are generated the same way. Since one \hfil is missing the text is moved to the relevant side.

```
\or \d@llar \insert@column \d@llar \hfil
\or \hfil \d@llar \insert@column \d@llar
```

The templates for p, t and b mainly consist of a box. In case of p it is generated by \vcenter. This command is allowed only in math mode. Therefore we start with a $.

```
\or $\vcenter
```

The part of the templates which is the same in all three cases (p, t and b) is built by the macros \@startpbox and \@endpbox. \@startpbox has an argument: the width of the column which is stored in the current token (i.e. \@nextchar). Between these two macros we find the well-known \insert@column.

```
\@startpbox{\@nextchar}\insert@column \@endpbox $%
```

The templates for t and b are generated in the same way though we do not need the $ characters because we use \vtop or \vbox.

```
\or \vtop \@startpbox{\@nextchar}\insert@column \@endpbox
\or \vbox \@startpbox{\@nextchar}\insert@column \@endpbox
```

Other values for \@chnum are impossible. Therefore we end the arguments to \@addtopreamble and \ifcase. Before we come to the end of \@classz we have to prepare the next token register.

```
\fi}\prepnext@tok}
```

\@classix In case of class 9 (>–token) we first check if the character class of the last token was 3. If so, we have a user preamble of the form ..>{...}>{...}.. which is not allowed. We only give an error message and continue. So the declarations defined by the first >{...} are ignored.

```
\def\@classix{\ifnum \@lastchclass = \thr@@
      \@preamerr \thr@@ \fi
```

Furthermore, we call up \@classx because afterwards always a new column is started by c, l, r, p, t or b.

```
\@classx}
```

\@classviii If the current token is a < the last character class must be 0. In this case it is not necessary to extend the preamble. Otherwise we output an error message, set \@chclass to 6 and call \@classvi. This assures that < is treated like !.

```
\def\@classviii{\ifnum \@lastchclass >\z@
      \@preamerr 4\@chclass 6 \@classvi \fi}
```

\@arrayrule There are only two incompatibilities with the original definition: the p–option mentioned earlier and the definition of \@arrayrule. In the original a line without width[2] is created by multiple \hskip .5\arrayrulewidth. We only insert a vertical line into the preamble. This is done to prevent problems with TEX's main memory when generating tables with many vertical lines in them (especially in the case of floats).

```
\def\@arrayrule{\@addtopreamble \vline}
```

[2]So the space between cc and c|c is equal.

\@classvii As a consequence it follows that in case of class 7 (@ token) the preamble need not be extended. In the original definition \@lastchclass = 1 is treated by inserting \hskip .5\arrayrulewidth. We only check if the last token was of class 3 which is forbidden.

```
\def\@classvii{\ifnum \@lastchclass = \thr@@
```

If this is true we output an error message and ignore the declarations stored by the last >{...}, because these are overwritten by the argument of @.

```
    \@preamerr \thr@@ \fi}
```

\@classvi If the current token is a regular ! and the last class was 0 or 2 we extend the preamble with \hskip\col@sep. If the last token was of class 1 (for instance |) we extend with \hskip\doublerulesep because the construction !{...} has to be treated like |.

```
\def\@classvi{\ifcase \@lastchclass
          \@acol
      \or \@addtopreamble{\hskip \doublerulesep}%
      \or \@acol
```

Now \@preamerr... should follow because a user preamble of the form ..>{..}!.. is not allowed. To save memory we call \@classvii instead which also does what we want.

```
      \or \@classvii
```

If \@lastchclass is 4 or 5 nothing has to be done. Classes 6 to 10 are not possible. So we finish the macro.

```
      \fi}
```

\@classii \@classiii In the case of character classes 2 and 3 (i.e. the argument of < or >) we only have to store the current token (\@nextchar) into the corresponding token register since the preparation and insertion of these registers are done by the macro \@classz. This is equivalent to calling \save@decl in the case of class 3. To save command identifiers we do this call up in the macro \@mkpream (see page 304).

Class 2 exhibits a more complicated situation: the token registers have already been inserted by \@classz. So the value of \count@ is too high by one. Therefore we decrease \count@ by 1.

```
\def\@classii{\advance \count@ \m@ne
```

Next we store the current token into the correct token register by calling \save@decl and then increase the value of \count@ again. At this point we can save memory once more (at the cost of time) if we use the macro \prepnext@tok.

```
    \save@decl\prepnext@tok}
```

\@classv If the current token is of class 5 then it is an argument of a @ token. It must be stored into a token register.

```
\def\@classv{\save@decl
```

We extend the preamble with a command which inserts this token register into the preamble when its construction is finished. This argument should be in math mode if it is used in an array-environment. Therefore we surround it with \d@llar's.

```
    \@addtopreamble{\d@llar\@thetoks\the\count@\d@llar}%
```

Finally we must prepare the next token register.

```
    \prepnext@tok}
```

\@classi In the case of class 0 we generated the necessary space between columns by using the macro \@classx. Analogously the macro \@classvi can be used for class 1.

```
\def\@classi{\@classvi
```

Depending on \@chnum a vertical line

```
    \ifcase \@chnum \@arrayrule
```

or (in case of !{...}) the current token — stored in \@nextchar — has to be inserted into the preamble. This corresponds to calling \@classv.

```
\or \@classv \fi}
```

\@startpbox In \@classz the macro \@startpbox is used. The width of the parbox is passed as an argument. \vcenter, \vtop or \vbox is already in the preamble. So we start with the braces for the desired box.

```
\def\@startpbox#1{\bgroup
```

The argument is the width of the box. This information has to be assigned to \hsize. Then we assign default values to several parameters used in a parbox.

```
\hsize #1 \@arrayparboxrestore
```

Our main problem is to obtain the same distance between succeeding lines of the parbox. We have to remember that the distance between two parboxes should be defined by \@arstrut. That means that it can be greater than the distance within a parbox. Therefore it is not enough to set a \@arstrut at the beginning and at the end of the parbox. This would dimension the distance between first and second line and the distance between the two last lines of the parbox incorrectly. To prevent this we set an invisible rule of height \@arstrutbox at the beginning of the parbox. This has no effect on the depth of the first line. At the end of the parbox we set analogously another invisible rule which affects only the depth of the last line.

```
\vrule \@height \ht\@arstrutbox \@width \z@}
```

\@endpbox If there are any declarations defined by >{...} and <{...} they now follow in the macro \@classz — the contents of the column in between. So the macro \@endpbox must insert the specialstrut mentioned earlier and then close the group opened by \@startpbox.

```
\def\@endpbox{\vrule \@width \z@ \@depth \dp\@arstrutbox \egroup}
```

Building and calling \halign

\@array Now that we have discussed the macros needed for the evaluation of the user preamble we can define the macro \@array which uses these macros to create a \halign. It has two arguments. The first one is a position argument which can be t, b or c; the second one describes the preamble wanted, e.g. it has the form |c|c|c|.

```
\def\@array[#1]#2{%
```

First we define a strut whose size basically corresponds to a normal strut multiplied by the factor \arraystretch. This strut is then inserted into every row and enforces a minimal distance between two rows. Nevertheless, when using horizontal lines, large letters (like accented capital letters) still collide with such lines. Therefore at first we add to the height of a normal strut the value of the parameter \extrarowheight.

```
\@tempdima \ht\strutbox
\advance \@tempdima by\extrarowheight
\setbox \@arstrutbox \hbox{\vrule
          \@height \arraystretch \@tempdima
          \@depth \arraystretch \dp\strutbox
          \@width \z@}%
```

Then we open a group, in which the user preamble is evaluated by the macro \@mkpream. As we know this must happen locally. This macro creates a preamble for a \halign and saves its result globally in the control sequence \@preamble.

```
\begingroup
\@mkpream{#2}%
```

We again redefine \@preamble so that a call up of \@preamble now starts the \halign. Thus also the arguments of >, <, @ and !, saved in the token registers, are inserted into the preamble. The \tabskip at the beginning and end of the preamble is set to 0pt

(in the beginning by the use of \ialign). Also the command \@arstrut is built in, which inserts the \@arstrutbox, defined above. Of course, the opening brace after \ialign has to be implicit as it will be closed in \endarray or another macro.

```
\xdef\@preamble{\ialign \@halignto
                \bgroup \@arstrut \@preamble
                        \tabskip \z@ \cr}%
```

What we have not explained yet is the macro \@halignto that was just used. Depending on its replacement text the \halign becomes a \halign to ⟨*dimen*⟩. Now we close the group again. Thus \@startpbox and \@endpbox as well as all token registers get their former meaning back.

```
\endgroup
```

Now we decide, depending on the position argument, in which box the \halign is to be put. (\vcenter may be used because we are in math mode.)

```
\if #1t\vtop \else \if #1b\vbox \else \vcenter \fi \fi
```

Now another implicit opening brace appears; then definitions which shall stay local follow. While constructing the \@preamble in \@mkpream the # sign must be hidden in the macro \@sharp which is \let to \relax at that moment (see definition of \@mkpream on page 304). All these now get their actual meaning.

```
\bgroup
\let \@sharp ##\let \protect \relax
```

With the above defined struts we fix the distance between rows by setting \lineskip and \baselineskip to 0pt. Since $'s have to be set around every column in the array-environment the parameter \mathsurround should also be set to 0pt. This prevents additional space between the rows. The *Plain*TEX-macro \m@th does this.

```
\lineskip \z@
\baselineskip \z@
\m@th
```

We also have to assign a special meaning (which we still have to specify) to the line separator \\, and redefine the command \par in such a way that empty lines in \halign cannot do any damage. We succeed in doing the latter by choosing something that will disappear when expanding. After that we only have to call up \@preamble to start the desired \halign.

```
\let\\ \@arraycr  \let\par\@empty \@preamble}
```

\extrarowheight The dimen parameter used above also needs to be allocated. As a default value we use 0pt, to ensure compatibility with standard LATEX.

```
\newdimen \extrarowheight
\extrarowheight=0pt
```

\@arstrut Now the insertion of \@arstrutbox through \@arstrut is easy since we know exactly in which mode TEX is while working on the \halign preamble.

```
\def\@arstrut{\unhcopy\@arstrutbox}
```

The line separator \\

\@arraycr In the macro \@array the line separator \\ is \let to the command \@arraycr. Its definition starts with a special brace which I have copied directly from the original definition. This is necessary because the \futurelet in \@ifnextchar might expand a following & token in a construction like \\ &. This would otherwise end the alignment template at a wrong time. For further information see [1, Appendix D].

```
\def\@arraycr{{\ifnum 0='}\fi
```

Then we test whether the star form is being used and ignore a possible star (I disagree with this procedure, because a star does not make any sense here).

```
\@ifstar \@xarraycr \@xarraycr}
```

\@xarraycr In the command \@xarraycr we test if an optional argument exists.

```
\def\@xarraycr{\@ifnextchar [%
```

If it does, we branch out into the macro \@argarraycr; if not, we close the special brace (mentioned above) and end the row of the \halign with a \cr.

```
    \@argarraycr {\ifnum 0='{\fi}\cr}}
```

\@argarraycr If additional space is requested by the user this case is treated in the macro \@argarraycr. First we close the special brace and then we test if the additional space is positive.

```
\def\@argarraycr[#1]{\ifnum0='{\fi}\ifdim #1>\z@
```

If this is the case we create an invisible vertical rule with a depth of \dp\@arstrutbox+⟨*wanted space*⟩. Thus we achieve that all vertical lines specified in the user preamble by a | are now generally drawn. Then the row ends with a \cr.

If the space is negative we end the row at once with a \cr and move back up with a \vskip.

While testing these macros I found out that the \endtemplate created by \cr and & is something like an \outer primitive and therefore it should not appear in incomplete \if statements. Thus the following solution was chosen, to hide the \cr in other macros when TEX is skipping conditional text.

```
    \@xargarraycr{#1}\else \@yargarraycr{#1}\fi}
```

\@xargarraycr The following macros were already explained above.
\@yargarraycr

```
\def\@xargarraycr#1{\unskip
  \@tempdima #1\advance\@tempdima \dp\@arstrutbox
  \vrule \@depth\@tempdima \@width\z@ \cr}
\def\@yargarraycr#1{\cr\noalign{\vskip #1}}
```

Spanning several columns

\multicolumn If several columns should be held together with a special format the command \multicolumn must be used. It has three arguments: the number of columns to be covered, the format for the result column, and the actual column entry.

```
\def\multicolumn#1#2#3{%
```

First we combine the given number of columns into a single one; then we start a new block so that the following definition is kept local.

```
    \multispan{#1}\begingroup
```

Since a \multicolumn should only describe the format of a result column, we redefine \@addamp in such a way that one gets an error message if one uses more than one c, l, r, p, t or b in the second argument. One should consider that this definition is local to the build-up of the preamble; an array– or tabular–environment in the third argument of the \multicolumn is therefore worked through correctly as well.

```
    \def\@addamp{\if@firstamp \@firstampfalse \else
                \@preamerr 5\fi}%
```

Then we evaluate the second argument with the help of \@mkpream. Now we still have to insert the contents of the token register into the \@preamble, i.e. we have to say \xdef\@preamble{\@preamble}. This is achieved more compactly by writing:

```
    \@mkpream{#2}\@addtopreamble\@empty
```

After the \@preamble is created we forget all local definitions and contents of the token registers.

```
    \endgroup
```

In the special situation of \multicolumn \@preamble is not needed as preamble for a \halign but it is directly inserted into our table. Thus instead of \sharp there has to be the column entry (#3) wanted by the user.

```
\def\@sharp{#3}%
```

Now we can pass the \@preamble to TEX. For safety we start with an \@arstrut. This should usually be in the template for the first column; however we do not know if this template was overwritten by our \multicolumn.

```
\@arstrut \@preamble \ignorespaces}
```

The Environment Definitions

After these preparations we are able to define the environments. They differ only in the initialisations of \d@llar, \col@sep and \@halignto.

\@halignto
\d@llar

In order to conserve the save stack we assign the replacement texts for \@halignto and \d@llar each time globally.

\array

Our new definition of \array then reads:

```
\def\array{\col@sep\arraycolsep
  \gdef\d@llar{$}\gdef\@halignto{}%
```

Since there might be an optional argument we call another macro which is also used by the other environments.

```
  \@tabarray}
```

\@tabarray

This macro tests for a optional bracket and then calls up \@array or \@array[c] (as default).

```
\def\@tabarray{\@ifnextchar[{\@array}{\@array[c]}}
```

\tabular
\tabular*

The environments tabular and tabular* differ only in the initialisation of \@halignto. Therefore we define

```
\def\tabular{\gdef\@halignto{}\@tabular}
```

and analogously

```
\expandafter\def\csname tabular*\endcsname#1{%
      \gdef\@halignto{to#1}\@tabular}
```

\@tabular

The rest of the job is carried out by the \@tabular macro:

```
\def\@tabular{%
```

First of all we have to make sure that we start out in hmode. Otherwise we might find our table dangling by itself on a line.

```
  \leavevmode
```

It should be taken into consideration that the macro \@array must be called in math mode. Therefore we open a box, insert a $ and then assign the correct values to \col@sep and \d@llar.

```
  \hbox \bgroup $\col@sep\tabcolsep \gdef\d@llar{}%
```

Now everything tabular specific is done and we are able to call the \@tabarray macro.

```
  \@tabarray}
```

\endarray

When the processing of array is finished we have to close the \halign and afterwards the surrounding box selected by \@array. To save token space we then redefine \@preamble because its replacement text isn't needed any longer.

```
\def\endarray{\crcr \egroup \egroup \gdef\@preamble{}}
```

\endtabular
\endtabular*

To end a tabular or tabular* environment we call up \endarray, close the math mode and then the surrounding \hbox.

```
\def\endtabular{\endarray $\egroup}
\expandafter\let\csname endtabular*\endcsname=\endtabular
```

Last-minute definitions

If this file is used as a style file we should \let all macros to \relax that were used in the original but are no longer necessary.

```
\let\@ampacol=\relax        \let\@expast=\relax
\let\@arrayclassiv=\relax   \let\@arrayclassz=\relax
\let\@tabclassiv=\relax     \let\@tabclassz=\relax
\let\@arrayacol=\relax      \let\@tabacol=\relax
\let\@tabularcr=\relax      \let\@@endpbox=\relax
\let\@argtabularcr=\relax   \let\@xtabularcr=\relax
```

\@preamerr We also have to redefine the error routine \@preamerr since new kinds of errors are possible. The code for this macro is not perfect yet; it still needs too much memory.

```
\def\@preamerr#1{\def\@tempd{{..} at wrong position: }%
   \@latexerr{%
   \ifcase #1 Illegal pream-token (\@nextchar): `c' used\or %0
    Missing arg: token ignored\or                          %1
    Empty preamble: `l' used\or                            %2
    >\@tempd token ignored\or                              %3
    <\@tempd changed to !{..}\or                           %4
    Only one colum-spec. allowed.\fi}\@ehc}                %5
```

\@tfor Testing this implementation an error was found in the definition of the LaTeX macro \@tfor. It was not implemented according to its specification. The assignment to \@fortmp must not take place via \xdef. A \def has to be used because #2 should not be expanded. Since this mistake does not show up when \@tfor is used in latex.tex, it does not seem to have been noticed.

```
\def\@tfor#1:=#2\do#3{\def\@fortmp{#2}\ifx\@fortmp\@mpty
      \else\@tforloop#2\@nil\@nil\@@#1{#3}\fi}
```

References

[1] D. E. KNUTH. The TeXbook (Computers & Typesetting Volume A). Addison-Wesley, Reading, Massachusetts, 1986.

[2] D. E. KNUTH. TeX: The program (Computers & Typesetting Volume B). Addison-Wesley, Reading, Massachusetts, 1986.

[3] L. LAMPORT. LaTeX – A Document Preparation System. Addison-Wesley, Reading, Massachusetts, 1986.

[4] L. LAMPORT. latex.tex, Version 2.09 of (15. Sept. 87).

Index

Calendar

1988

Nov 4 UK TEX Users Group, inaugural meeting. University of Nottingham, England.

Nov 24 NTG — Nederlandse TEX Gebruikers, meeting. ENR, Petten, The Netherlands. (See report, page 316.)

Dec 5 – 9 ACM Conference on Document Processing Systems, Santa Fe, New Mexico. For information, contact Peter Orbeton, (617) 577-8500 or Orbeton.chi@xerox.com.

1989

University of New Mexico, Albuquerque

Jan 9 – 13 Intensive Beginning/Intermed. TEX

Florida State Supercomputer Research Institute, Tallahassee

Jan 9 – 13 Advanced TEX/Macro Writing

Jan 9 – 13 Intensive Introduction to LATEX

California State University, Northridge

Jan 9 – 13 Output Routines

Jan 17 **TUGboat Volume 10, No. 1:** Deadline for receipt of manuscripts.

Apr 15 Protext V Conference. October 4–6, Boston, Massachusetts. Call for papers: Submit 1 page abstract and paper (6 page maximum) to Programme Committee, Protext Conference, INCA, P. O. Box 2, Duń Laoghaire, Ireland.

May 1 **TUGboat Volume 10, No. 2:** Deadline for receipt of manuscripts (tentative).

Jun 29 – 30 NTG — Nederlandse TEX Gebruikers, "TEX happening". Utrecht, The Netherlands. (See report, page 316.)

Jul 30 – Aug 4 ACM SIGGRAPH '89, Boston, Massachusetts. Contact: Chris Herot or Branko Gerovac, (312) 644-6610.

TEX Users Group 1989 Conference
— Tenth Anniversary —
Stanford University, Stanford, California

Aug 14 – 18 Short Courses: to be announced

Aug 21 – 23 **TUG Annual Meeting**

Aug 24 – 25 Short Courses: to be announced

Sep 11 **TUGboat Volume 10, No. 3:** Deadline for receipt of manuscripts (tentative).

Oct 4 – 6 Protext V Conference: 5th International Conference on Computer-Aided Text Processing and its Applications. Boston, Massachusetts. For information, contact Protext Conference, INCA, P. O. Box 2, Duń Laoghaire, Ireland; +353-1-613749.

Oct 12 – 13 RIDT'89 — Raster Imaging and Digital Typography. Ecole Polytechnique Fédérale, Lausanne, Switzerland. For information, contact Prof. R.D. Hersch, Lausanne, Switzerland; (4121) 47 43 57/693 43 57 or hersch@elde.epfl.ch; or Debra Adams, (415) 494-4022 or adams.pa@Xerox.com. (See announcement, page 316.)

For additional information on the events listed above, contact the TUG office (401-751-7760) unless otherwise noted.

Status as of 17 October 1988

Dutch TEX Users Group

In late June, a Dutch TEX Users group was formed. Its name is 'Nederlandse TEX Gebruikers' or NTG for short. At the moment roughly 60 persons are members. Whether institutions as well as individuals can become members is a minor organisational detail, and will be settled in due time. The members come from universities (computing centres and press departments as well as faculties), publishers, typesetters, software houses, governmental departments, telephone company, energy centre, philips, computer companies, printer/supplies companies, to name but a few.

Of course, the principal aim of the user group is to facilitate the use of TEX and related products. As a result we make use of a listserver — TEX-NL — to submit (elementary) problems to the community and to transmit the answers, apart from the TEXhax general possibility. At the moment two (volunteer) TEXperts are willing to answer the various questions.

Another activity is to pass on information, e.g. from similar groups, and cooperate with related groups such as the Dutch SGML group and other (European) TEX user groups. At the first meeting, people with related interests did meet, and as a consequence several small working groups have been formed.

One group considers education and courses, another evaluates products, a third is busy with the various aspects of fonts, the next considers the migration from text processors to TEX and experiences the relation with SGML, etc. Of course, one group is busy with the peculiarities of the Dutch language in relation to TEX, especially public domain hyphenation and template `.sty` files. In total, 12 working groups have been formed, and hopefully all will be active. The aim of these working groups is to become a source of knowledge and experience, and if necessary to develop material.

Apart from this 'organised' form, individuals of course — whether they like it or not — experience publishing with (LA)TEX. The next NTG meeting is in Petten on 24 November at ENR. (Contact

G.J.H. van Nes — secretary and host
ENR Rekencentrum,
Postbus 1,
1755 ZG Petten, The Netherlands,
+31/2246 4185,
Bitnet: `VANNES@HPEENR51`
DECnet: `enr001::vannes`

for further information).

On 29 and 30 June 1989 a general Dutch TEX-happening is scheduled at Utrecht, where the users group will show, teach and talk about the features of TEX, etc., as a tool for document preparation.

C.G. van der Laan (chairman)
Rekencentrum RUG
Landleven 1, 9700 AV
Groningen, The Netherlands
+31/50 633374 or
+31/50 633440
Bitnet: `cgl@hgrrug5`
DECnet: `rugr86::cgl`

RIDT'89 — International Workshop on Raster Imaging and Digital Typography

Ecole Polytechnique Fédérale
Lausanne, Switzerland
October 12–13, 1989

Raster image processors for non-impact printers and plotters require highly sophisticated algorithms and performant hardware. Outline character acquisition, design, manipulation and rasterization, as well as graphic and image rendering are of major concern to scientists and engineers involved in the development of raster imaging devices.

Authors are invited to submit papers describing original research results on the most relevant and recent developments in digital typography and raster imaging. Contributions are welcomed on any of the topic areas covered by the above theme. This includes (and is not limited to):

- Shape acquisition (curve fitting)
- Shape manipulation
- Character design
- Character representation and transformation
- Measuring type quality
- Character structures (generation/recognition)
- Page description languages
- Rasterization algorithms
- Rasterization accuracy
- Fast rasterization hardware

Call for Papers

Submit extended abstracts (2–3 pages) or full papers in English by January 15, 1989. Authors will be notified by March 15, 1989. Camera ready full papers are due by April 29, 1989.

The abstracts should contain: Title, Authors, Authors' affiliation, Keywords, Main Text, References, and Authors' Biography. All accepted papers will be published in the Conference Proceedings.

Potential contributors are encouraged to give advance notice of their intention to submit a paper. Send abstracts to:

Prof. Roger D. Hersch
LSP/EPFL
37, avenue de Cour
CH-1007 Lausanne
Switzerland
(4121) 47 43 57
Bitnet: HERSCH@ELDE.EPFL.CH

Prof. Hersch is chairman of the Program Committee. Vice-chairpersons are Debra Adams, Xerox PARC (adams.pa@Xerox.com), and Jacques André, INRIA/IRISA (jandre@irisa.uucp). Chuck Bigelow and Richard Southall are also on the Committee.

Late-Breaking News

TEXhax Moves North

Pierre A. MacKay

The TEXhax mail digest is about to move north to the University of Washington. Since its beginning, this digest has been maintained entirely out of Stanford University, but it will now be added to the other services of the TEX Users Group, and we will be working toward a close association and integration of the electronic mail digest with the published issues of TUGboat.

The TEX Users Group has provided funding for an editorial position so that it will be possible to work over each issue, and to develop data-bases, macro libraries and other similar aids from the correspondence.

The new moderators will be Pierre MacKay and Tiina Modisett. For the first months after November 1, our principal effort will be to live up to the standards already set by previous moderators, but we are already drawing up plans for the other services suggested above. We are deeply grateful to the moderators who have kept TEXhax going during the past, and most particularly to Malcolm Brown, who is aiding us through this period of transition.

Late-Breaking News

Production Notes

Barbara Beeton

Input and input processing

Electronic input for articles in this issue was received in several forms: mail, floppy disk, and file transfers to the AMS computer. One article was accepted in the form of camera copy (see the section on output), as were several figures that required special fonts or that could not be prepared on the American Mathematical Society's typesetter for other reasons.

Authors who had written articles previously for TUGboat typically submitted files that were fully tagged and ready for processing with the TUGboat macros — `tugbot.sty` for PLAIN-based files and `ltugbot.sty` for those using LATEX. (When possible, a copy of the file actually used for production is returned to the author, along with the current version of the macros, if the author has requested them. This seems to provide authors with incentive to write again for TUGboat.)

Articles in which no, or limited, TEX coding was present were tagged according to the `tugbot.sty` conventions. 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 identify macro definitions that conflicted with ones already defined for TUGboat. In the case of LATEX-based articles, it was not necessary to consider interactions with other articles. (`\documentstyle{article}` is the basis for `ltugbot.sty`; I have not yet devised a method of processing multiple articles in a stream, so each is processed separately, and physical pasteup is used where required to merge partial pages.) For PLAIN-based articles, the side-effects of an author's own definitions can usually be kept to a minimum by posting `\begingroup...\endgroup` around the article.

Most submissions for this issue were PLAIN. For these items, test runs of TEX separately and in groups were used to determine the arrangement and page numbers (to satisfy any possible cross references). The final processing of these articles was a single TEX run, with ranges of page numbers skipped where LATEX-based items would be inserted. LATEX items, as mentioned above, were processed individually, and arranged in the proper order after camera copy was produced.

The following articles were prepared using LaTeX; all others (except for items received as camera copy, for which see below) used the regular `tugbot.sty`.

- Peter Abbott, *A UK-based TeX mail archive server*, page 263.
- Jackie Damrau, *The LaTeX user's column*, page 297.
- Frank Mittelbach, *A new implementation of the array- and tabular-environments*, page 298.
- Dominik Wujastyk, *Further faces*, page 246.

Output

Camera copy for this issue of TUGboat was prepared on the devices indicated, and can be taken as representative of the output produced by those devices. The bulk of this issue was at the American Mathematical Society on a VAX 8600 (VMS) and output on an APS-μ5 using resident CM fonts and additional downloadable fonts for special purposes. The items listed below were received as camera copy; they were prepared on the devices indicated. The output devices used to prepare the advertisements were not usually identified; anyone interested in determining the device used for a particular ad should inquire of the advertiser.

- Unidentified:
  - all advertisements. Some of the ads were received in a size larger than permitted; these were reduced photographically using the PMT process.
  - Wolfgang Appelt; in *Typesetting chess*, p. 284, the chessboard on the third page has been pasted in from his original copy, which was produced on an unidentified laser printer.
  - Thomas Kneser; in *Compact matrix display*, p. 279, figure 1 was reduced from an ink drawing and pasted in.
- Apple LaserWriter II NT/X (300 dpi): Graeme McKinstry, *Some typesetting conventions*, p. 236; VAX/VMS; run with LaTeX using PostScript Times and Helvetica fonts, with Computer Modern for Figure 1.

A Thank You Note

Two of TUG's most diligent volunteers, both of whom took up their posts at the 1986 TUG meeting, are "retiring" from their adopted tasks: Malcolm Brown, the TeXhax moderator, and Ken Yap, creator and maintainer of the LaTeX-style collection. Malcolm remains on the staff at Stanford. Ken is looking forward to his doctorate from the University of Rochester next Spring (he must first complete his dissertation, in LaTeX, of course), and after that, to a change of scenery. It's been a delight to work with both of them. They've done a super job, and it shouldn't go unrecognized. Thank you both!

Barbara Beeton

TUG Business

Accountant's Review of TUG Financial Records

Years ended December 31, 1987 and 1986

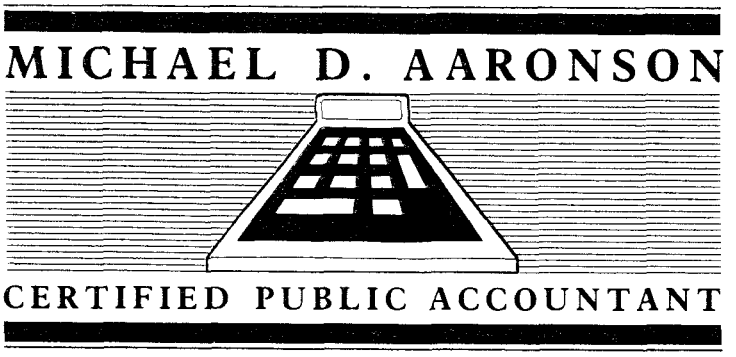
The following three pages contain the report of the accountant's review of the TUG financial records for the two years ending December 31, 1987 and 1986.

TeX Users Group

Statements of Cash Receipts and Disbursements

Years Ended December 31, 1987 and 1986

(with accountant's review report thereon)



To the Board of Directors
TEX Users Group
Providence, Rhode Island

I have reviewed the accompanying statements of cash receipts and disbursements of the TEX Users Group for the year ended December 31, 1987 and 1986 in accordance with standards established by the American Institute of Certified Public Accountants. All information included in these statements are the representation of the management of the TEX Users Group.

The statements of cash receipts and disbursements are a summary of the cash activities of the Group and do not include certain transactions that would be included if the Group prepared its financial statements on the accrual basis as contemplated by generally accepted accounting practices.

A review consists principally of inquiries of management personnel and analytical procedures applied to financial data. It is substantially less in scope than an examination in accordance with generally accepted auditing standards, the objective of which is the expression of an opinion regarding the financial statements taken as a whole. Accordingly, I do not express such an opinion.

However, based on my review, I am not aware of any material modifications that should be made to the accompanying statements of cash receipts and disbursements of the TEX Users Group for the years ended December 31, 1987 and 1986.

Michael Aaronson, CPA

Providence, Rhode Island
May 27, 1988

This report contains three (3) pages.

71 Princeton Avenue — Providence, Rhode Island 02907 — (401) 521-1160

TEX Users Group

Statements of Cash Receipts and Disbursements
Years Ended December 31, 1987 and 1986

	1987	1986
Receipts:		
Dues	$161,596	$ 97,420
Meetings and courses	207,345	204,831
Sale of publications	169,366	167,690
Advertising	16,345	7,999
Contributions (Note 2)	12,423	8,807
Interest	11,699	5,298
Miscellaneous	-0-	1,190
Total receipts	578,774	493,235
Disbursements		
Newsletter	39,418	29,698
Meetings and courses	135,475	88,500
Cost of publications	86,120	121,034
Administrative costs	147,054	134,881
Contribution (Note 3)	-0-	18,750
Exhibits	2,375	1,453
Total disbursements	410,442	394,316
Excess of receipts over disbursements	168,332	98,919
Cash balances, beginning of year	143,503	44,584
Cash balances, end of year (Note 4)	$311,835	$143,503

See accompanying accountant's review report and
notes to cash receipts and disbursements statements.

TEX Users Group

Notes to Cash Receipts and Disbursements Statements
Years Ended December 31, 1987 and 1986

1. Description of organization and summary of significant accounting policies

   a) Description of organization:

   The TEX Users Group is a Rhode Island nonprofit corporation. An application for tax exemption is pending at the Internal Revenue Service. The TEX Users Group (TUG) provides information and technical assistance to the users of TEX, a sophisticated typesetting computer application through the publication of a newsletter and the conduct of conferences and TEX courses.

   b) Summary of significant accounting policies:

   Accounting method: TUG used the cash basis of accounting through December 31, 1987. Effective January 1, 1988 the organization switched to the accrual basis of accounting.

2. Contributions (income)

   Contributions consist of royalty fees assigned to TUG by the author.

3. Contribution (expense)

   The $18,750 contribution for 1986 consisted of support TUG gave to a TEX project run by a California university.

4. Cash balances

   Cash balances at December 31, 1987 and 1986 consisted of:

	1987	1986
Cash checking	$ (1,490)	$ 580
Cash money market funds	213,325	142,923
Certificate of deposit	100,000	-0-
Total	$311,835	$143,503

Institutional Members

Addison-Wesley Publishing Company, *Reading, Massachusetts*

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*

Brookhaven National Laboratory, *Upton, New York*

Brown University, *Providence, Rhode Island*

California Institute of Technology, *Pasadena, California*

Calvin College, *Grand Rapids, Michigan*

Centre Inter-Régional de Calcul Électronique, CNRS, *Orsay, France*

City University of New York, *New York, New York*

College of St. Thomas, Computing Center, *St. Paul, Minnesota*

College of William & Mary, Department of Computer Science, *Williamsburg, Virginia*

COS Information, *Montreal, P. Q., Canada*

Data General Corporation, *Westboro, Massachusetts*

DECUS, L&T Special Interest Group, *Marlboro, Massachusetts*

Department of National Defence, *Ottawa, Ontario, Canada*

Digital Equipment Corporation, *Nashua, New Hampshire*

dit Company, Ltd., *Tokyo, Japan*

Edinboro University of Pennsylvania, *Edinboro, Pennsylvania*

Electricité de France, *Clamart, France*

Environmental Research Institute of Michigan, *Ann Arbor, Michigan*

European Southern Observatory, *Garching bei München, Federal Republic of Germany*

Fermi National Accelerator Laboratory, *Batavia, Illinois*

Försvarets Materielverk, *Stockholm, Sweden*

General Motors Research Laboratories, *Warren, Michigan*

Geophysical Company of Norway A/S, *Stavanger, Norway*

Grinnell College, Computer Services, *Grinnell, Iowa*

GTE Laboratories, *Waltham, Massachusetts*

Hartford Graduate Center, *Hartford, Connecticut*

Harvard University, Computer Services, *Cambridge, Massachusetts*

Hewlett-Packard Co., *Boise, Idaho*

Hobart & William Smith Colleges, *Geneva, New York*

Humboldt State University, *Arcata, California*

Hutchinson Community College, *Hutchinson, Kansas*

IBM Corporation, Scientific Center, *Palo Alto, California*

Illinois Institute of Technology, *Chicago, Illinois*

Imagen, *Santa Clara, California*

Informatika, *Hamburg, Federal Republic of Germany*

Institute for Advanced Study, *Princeton, New Jersey*

Institute for Defense Analyses, Communications Research Division, *Princeton, New Jersey*

Intevep S. A., *Caracas, Venezuela*

Iowa State University, *Ames, Iowa*

Istituto di Cibernetica, Università degli Studi, *Milan, Italy*

Kuwait Institute for Scientific Research, *Safat, Kuwait*

The Library of Congress, *Washington, DC*

Los Alamos National Laboratory, University of California, *Los Alamos, New Mexico*

Louisiana State University, *Baton Rouge, Louisiana*

Marquette University, Department of Mathematics, Statistics, and Computer Science, *Milwaukee, Wisconsin*

Massachusetts Institute of Technology, Artificial Intelligence Laboratory, *Cambridge, Massachusetts*

Massachusetts Institute of Technology, Information Services, *Cambridge, Massachusetts*

Mathematical Reviews, American Mathematical Society, *Ann Arbor, Michigan*

Max Planck Institut für Mathematik, *Bonn, Federal Republic of Germany*

Max Planck Institute Stuttgart, *Stuttgart, Federal Republic of Germany*

McGill University, *Montreal, Quebec, Canada*

National Cancer Institute, *Frederick, Maryland*

National Center for Atmospheric Research, *Boulder, Colorado*

National Institutes of Health, *Bethesda, Maryland*

National Research Council Canada, Computation Centre, *Ottawa, Ontario, Canada*

National Semiconductor Corporation, *Santa Clara, California*

New Jersey Institute of Technology, *Newark, New Jersey*

New York University, Academic Computing Facility, *New York, New York*

Nippon Telegraph & Telephone Corporation, Software Laboratories, *Tokyo, Japan*

Northeastern University, Academic Computing Services, *Boston, Massachusetts*

Online Computer Library Center, Inc. (OCLC), *Dublin, Ohio*

Pennsylvania State University, Computation Center, *University Park, Pennsylvania*

Personal TEX, Incorporated, *Mill Valley, California*

Purdue University, *West Lafayette, Indiana*

QMS, Inc, *Mobile, Alabama*

Queens College, *Flushing, New York*

Research Triangle Institute, *Research Triangle Park, North Carolina*

RE/SPEC, Inc., *Rapid City, South Dakota*

Rice University, Department of Computer Science, *Houston, Texas*

Royal Marsden Hospital, *Surrey, England*

Ruhr Universität Bochum, *Bochum, Federal Republic of Germany*

Rutgers University, Hill Center, *Piscataway, New Jersey*

St. Albans School, *Mount St. Alban, Washington, D.C.*

Sandia National Laboratories, *Albuquerque, New Mexico*

SAS Institute, *Cary, North Carolina*

I. P. Sharp Associates, *Palo Alto, California*

Smithsonian Astrophysical Observatory, Computation Facility, *Cambridge, Massachusetts*

Software Research Associates, *Tokyo, Japan*

Sony Corporation, *Atsugi, Japan*

Space Telescope Science Institute, *Baltimore, Maryland*

Springer-Verlag, *Heidelberg, Federal Republic of Germany*

Stanford Linear Accelerator Center (SLAC), *Stanford, California*

Stanford University, Computer Science Department, *Stanford, California*

Stanford University, ITS Graphics & Computer Systems, *Stanford, California*

State University of New York, Department of Computer Science, *Stony Brook, New York*

Stratus Computer, Inc., *Marlboro, Massachusetts*

Syracuse University, *Syracuse, New York*

Talaris Systems, Inc., *San Diego, California*

Texas A & M University, Computing Services Center, *College Station, Texas*

Texas A & M University, Department of Computer Science, *College Station, Texas*

Tribune TV Log, *Glens Falls, New York*

TRW, Inc., *Redondo Beach, California*

Tufts University, *Medford, Massachusetts*

TV Guide, *Radnor, Pennsylvania*

TYX Corporation, *Reston, Virginia*

UNI.C, Danmarks EDB-Center, *Aarhus, Denmark*

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, Berkeley, Academic Computing Services, *Berkeley, California*

University of California, Berkeley, Computer Science Division, *Berkeley, California*

University of California, Irvine, Department of Mathematics, *Irvine, California*

University of California, Irvine, Information & Computer Science, *Irvine, California*

University of California, San Diego, *La Jolla, California*

University of California, San Francisco, *San Francisco, California*

University of Canterbury, *Christchurch, New Zealand*

University of Chicago, Computation Center, *Chicago, Illinois*

University of Chicago, Computer Science Department, *Chicago, Illinois*

University of Crete, Institute of Computer Science, Research Center, *Heraklio, Crete, Greece*

University of Delaware, *Newark, Delaware*

University of Exeter, Computer Unit, *Exeter, Devon, England*

University of Glasgow, *Glasgow, Scotland*

University of Groningen, *Groningen, The Netherlands*

University of Illinois at Chicago, Computer Center, *Chicago, Illinois*

University of Illinois at Urbana-Champaign, Computer Science Department, *Urbana, Illinois*

University of Kansas, Academic Computing Services, *Lawrence, Kansas*

University of Maryland, *College Park, Maryland*

University of Massachusetts, *Amherst, Massachusetts*

University of North Carolina, School of Public Health, *Chapel Hill, North Carolina*

University of Oslo, Institute of Informatics, *Blindern, Oslo, Norway*

University of Ottawa, *Ottawa, Ontario, Canada*

University of Southern California, Information Sciences Institute, *Marina del Rey, California*

University of Stockholm, Department of Mathematics, *Stockholm, Sweden*

University of Texas at Austin, Physics Department, *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*

Vanderbilt University, *Nashville, Tennessee*

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*

10

TUG's Tenth

Plans are well underway for a *gala* celebration at TUG's 10th Annual Meeting at Stanford next August 21–23. In addition to a very ambitious program (see Cover 3 of this issue), we're going to have a *birthday party*—cake and all! And you're all invited! We'd like to see as many "old faces" as possible and look forward to welcoming a lot of newcomers. Don Knuth, *the* Grand Wizard, will be the keynote speaker!

A souvenir program is planned, with pictures gathered from the last 9 years of meetings. So, if you have any special photographs that we could consider for inclusion in the "souvenir program", send them to Ray Goucher at the TUG office. Mark them clearly with your name and address, so that we can return them to you.

Request for Information

The TEX Users Group maintains a database and publishes a membership list containing information about the equipment on which TEX is (or will be) installed and about the applications for which TEX is used. This list is updated periodically and distributed to members with TUGboat, to permit them to identify others with similar interests. Thus, it is important that the information be complete and up-to-date.

Please answer the questions below, in particular those regarding the status of TEX and the hardware on which it runs. (Operating system information is particularly important in the case of IBM mainframes and VAX.) This hardware information is used to group members in the listings by computer and output device.

If accurate information has already been provided by another TUG member at your site, indicate that member's name and the same information will be repeated automatically under your name. If your current listing is correct, you need not answer these questions again. Your cooperation is appreciated.

- *Send completed form with remittance* (checks, money orders, UNESCO coupons) to:
  TEX Users Group
  P. O. Box 594
  Providence, Rhode Island 02901, U.S.A.
- *For foreign bank transfers* direct payment to the TEX Users Group, account #002-031375, at:
  Rhode Island Hospital Trust National Bank
  One Hospital Trust Plaza
  Providence, Rhode Island 02903-2449, U.S.A.
- *General correspondence* about TUG should be addressed to:
  TEX Users Group
  P. O. Box 9506
  Providence, Rhode Island 02940-9506, U.S.A.

Name: _____
Home [ ]
Bus. [ ] Address: _____
_____
_____
_____

QTY	ITEM	AMOUNT
	1989 TUGboat Subscription/TUG Membership (Jan.–Dec.) – **North America** New (first-time): [ ] $35.00 each Renewal: [ ] $45.00; [ ] $35.00 – reduced rate if renewed before February 1, 1989	
	1989 TUGboat Subscription/TUG Membership (Jan.–Dec.) – **Outside North America** New (first-time): [ ] $45.00 each Renewal: [ ] $50.00; [ ] $45.00 – reduced rate if renewed before February 1, 1989	
	TUGboat back volumes: 1980 1981 1982 1983 1984 1985 1986 1987 1988 Circle volume(s) desired: vol. 1 vol. 2 vol. 3 vol. 4 vol. 5 vol. 6 vol. 7 vol. 8 vol. 9 Indiv. issues $18.00 ea.: $18 $50 $35 $35 $35 $50 $50 $50 $50	

Air mail postage is included in the rates for all subscriptions and memberships outside North America.
Quantity discounts available on request.

TOTAL ENCLOSED: _____
(*Prepayment in U.S. dollars required*)

Membership List Information

Institution (if not part of address): _____
_____

Title: _____
Phone: _____
Network address: _____
[ ] Arpanet [ ] BITnet
[ ] CSnet [ ] uucp
[ ] JANET [ ] other _____

Specific applications or reason for interest in TEX:

My installation can offer the following software or technical support to TUG:

Please list high-level TEX users at your site who would not mind being contacted for information; give name, address, and telephone.
_____
_____

Date: _____

Status of TEX: [ ] Under consideration
[ ] Being installed
[ ] Up and running since: _____
Approximate number of users: _____

Version of TEX:
[ ] Pascal
[ ] C
[ ] other (describe)
From whom obtained: _____

Hardware on which TEX is used:

Computer(s)	Operating system(s)	Output device(s)
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____

Proceedings of the Ninth Annual Meeting

of the TEX Users Group

McGill University, Montréal

August 21–24, 1988

Contents

Public Domain TEX

The authorized and current versions TEX software are available from *Maria Code - Data Processing Services* by special arrangement with Stanford University and other contributing universities. The standard distribution tape contains the source of TEX and METAFONT, the macro libraries for $\mathcal{AMS}$-TEX, LATEX, SliTEX and HP TEX, sample device drivers for a Versetec and LN03 printers, documentation files, and many useful tools.

Since these are in the public domain, they may be used and copied without royalty concerns. They represent the official versions of TEX. 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" TEX 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 TEX 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. If you want to pay with Visa or MasterCard, include your number, card type, expiration date and signature.

TEX Order Form

TEX Distribution tapes:

____ Standard ASCII format
____ Standard EBCDIC format
____ Special VAX/VMS format Backup
____ Special DEC 20/TOPS 20 Dumper format
____ Special IBM VM/CMS format
____ Special IBM MVS format

Font Library Tapes (GF files)

____ 300 dpi VAX/VMS format
____ 300 dpi generic format
____ IBM 3820/3812 MVS format
____ IBM 3800 CMS format
____ IBM 4250 CMS format
____ IBM 3820/3812 CMS format

Tape prices: $92.00 for first tape,
$72.00 for each additional tape.

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	37.00	4	____
METAFONT book (vol. C) softcover	22.00	2	____
METAFONT the Program (vol. D) hardcover	37.00	4	____
Computer Modern Typefaces (vol. E) hardcover	37.00	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	____

* 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

____ domestic priority mail rate $1.50/lb.
____ air mail to Canada and Mexico: rate $2.00/lb.
____ export surface mail (all countries): rate $1.50/lb.
____ air mail to Europe, South America: rate $5.00/lb.
____ 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 organizaton a member of TUG? _____

Have You Met Your Mac?

We'd like to suggest a small heresy—that the Apple Macintosh, with our *Textures* software, might just possibly be the best vehicle for TEX users right now. Not just the best low-cost alternative—given the choice of any system at any price, you could rightly choose the Macintosh and *Textures*.

If you think the Macintosh is a toy, look at the Macintosh II. It's definitely not for kids—16MHz 68020, memory to 8MB, disk to 300MB, large sharp screens. And *Textures* is not a micro-TEX; your TEX files will run unchanged.

Textures gives you TEX at your fingertips—responsive, integrated. Go from editing this copy, through TEX, to previewing the finished page—one keystroke, three seconds. Go from page to page—one mouse-click, one second. Scan the whole page for form, proof it at 12 or 14 or 20 points, click the magnifier and check that equation at LaserWriter resolution, instantly. Make your current macro set preloaded in seconds, anytime.

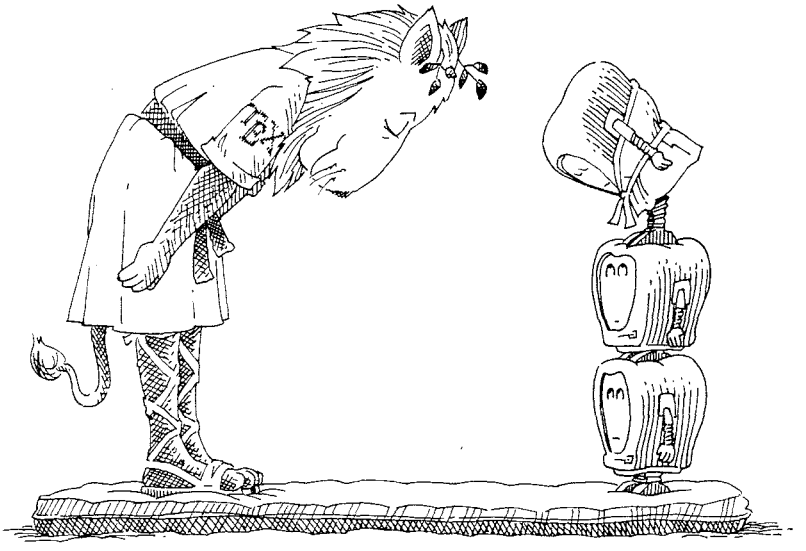
TEX is the world's most capable typesetting software, but TEX doesn't do pictures. Adobe Illustrator *does* do pictures, with a line quality finer than any technical pen. Or use MacDraw from Claris for technical drawings; learn it in less than one hour. Image Studio from Letraset does halftones, hand-painted or scanned. All world-class programs, all only on the Macintosh. With these tools (and many others), *Textures* does pictures—on screen, on paper, beautifully.

We're convinced that *Textures* and the Macintosh are worth a serious trial from anyone working with TEX. We'll make it easy for you to see what we mean, at our risk. If there's a Macintosh in your neighborhood, order a copy on approval. If no Macintosh is nearby, let us arrange a demonstration at a dealer in your area. You'll like what you see.

Call us on it.

Textures
for the Apple Macintosh

Single Copy $495
Educational $395



Blue Sky Research
534 SW Third Avenue
Portland, Oregon 97204
800/622-8398, 503/222-9571
Telex 9102900911

Tired of Waiting for TEX?

Wait no more! Now you can preview and edit as TEX is running—on the Commodore Amiga. Only the Amiga gives you a true multitasking TEX environment. Run TEX, METAFONT, preview, an editor, communications program, printer driver, and C compiler simultaneously. Switch from editing a file to TEXing and previewing that file with a single keystroke—often less than a second from editor to previewed page. And there is no need to wait until TEX finishes the entire document; you can preview any page TEX has finished, find errors, and make corrections with your editor, as TEX continues processing.

AmigaTEX provides significantly more functionality for significantly less money than virtually any other microcomputer implementation. It is the fastest TEX available for personal computers. The basic package includes fonts, macro packages, and utilities that cost extra on other implementations. And Radical Eye Software is committed to providing the best support possible.

Check out these features, and send for your free demo disk today!

AmigaTEX

- Fast! Does TEXbook in 51 minutes.
- Fast! Preview a page a second.
- Fast! Print up to eight pages per minute.
- Fast! Preview documents even as they are being TEXed.
- Fast! Even faster with a 68020 or 68030.
- Supports Amiga special characters and foreign keymaps.
- Hard disk not required; runs nicely with two floppies.
- Includes LaTEX, SliTEX, BibTEX, and preview.
- All ten disks of programs, macros and fonts: Only $200!

AmigaMETAFONT

- Includes interactive screen support and CM font sources.
- Only $75.

Printer Drivers

- Supporting HP LaserJet Plus and Series II, PostScript, QMS KISS and SmartWriter, HP DeskJet, Epson LQ series, NEC Pinwriter series, Epson MX and FX series, ImageWriter II, and others.
- Includes the full set of CM, LaTEX and SliTEX fonts.
- Only $100 each.

Radical Eye Software
Box 2081 Stanford, CA 94309 (415) 326-5312

MacroTEX:

Makes Table Making Easy. . .

A MacroTEX Table

Here	are	some	table	features
A	line	with	gaps	in it
Multi column	An interesting feature is that text will wrap in the width of the multicolumn environment.			
23.456	double	lines	are	here
345.33	here	are	thick	lines

```
\horizontallines
\verticallines
\tabletitle{5}{A Macro\TeX\ Table}
\table*
Here&are&some&table&features\cr
\gapsinhline{24}
A&line&with&gaps&in it\cr
Multi column
&\multicol{4}{o}{An interesting
feature is that text will wrap in
the width of the multicolumn
environment.}\cr
\lineup{23}{.456}
&double\dblvline&lines&are
&here\dblhline\cr
\lineup{345}{.33}
&here\thickvline{2pt}&are
&thick&lines\thickhline{2pt}\cr
\endtable
```

And lots more too. . .

Table of Contents Generation
Cross-Referencing
Slide Making
Revision Bars
Drop Caps
Indexing
Glossary
Bibliography
Document styles

Call or Write
for more information

TEXnology Inc.

57 Longwood Avenue, Brookline, MA 02146

(617) 738-8029 *Source Code Included, Written in Plain TEX, $200, Site Licenses Available*

Index of Advertisers

TEX Users Group Membership List — Supplement

December 1988

This supplementary list, compiled on 20 October 1988, includes the names of all persons who have become members of TUG or whose addresses have changed since publication of the last membership list update, as of 16 June 1988 and bound into TUGboat Vol. 9, No. 2. The last full membership list, as of 15 February 1988, is bound into TUGboat Vol. 9, No. 1. All institutional members are listed. Total membership: 147 institutional members and 3,172 individuals affiliated with more than 1,350 colleges and universities, commercial publishers, government agencies, and other organizations throughout the world having need for an advanced composition system.

The following information is included for each listing of an individual member, where it has been provided:

Name and mailing address

Telephone number

Network address

Title and organizational affiliation, when that is not obvious from the mailing address

Computer and typesetting equipment available to the member, or type of equipment on which his organization wishes to (or has) installed TEX

Uses to which TEX may be put, or a general indication of why the member is interested in TEX

CONTENTS

Recipients of this list are encouraged to use it to identify others with similar interests, and, as TUG members, to keep their own listings up-to-date in order for the list to remain as useful as possible. New or changed information may be submitted on the membership renewal form bound into the back of a recent issue of TUGboat. Comments on ways in which the content and presentation of the membership list can be improved are welcome.

This list is intended for the private use of TUG members; it is not to be used as a source of names to be included in mailing lists or for other purposes not approved by TUG. Additional copies are available from TUG. Mailing lists of current TUG membership are available for purchase. For more information, contact Ray Goucher, TUG Executive Director.

Distributed with TUGboat Volume 9 (1988), No. 3. Published by

TEX Users Group
P. O. Box 9506
Providence, R.I. 02940-9506, U.S.A.

TUG Board of Directors

Officers

CHILDS, S. Bart
Dept of Computer Science
Texas A & M University
College Station, TX 77843-3112
409-845-5470
bart@cssun.tamu.edu
Bitnet: Bart@TAMLSR
President; Finance Committee;
DG MV Site Coordinator

FURUTA, Richard
Department of Computer Science
University of Maryland
College Park, MD 20742
301-454-1461
furuta@mimsy.umd.edu
Vice President; Finance Committee;
Laser-Lovers moderator

HOENIG, Alan
17 Bay Avenue
Huntington, NY 11743
516-385-0736
Secretary; Finance Committee

NESS, David
TV Guide
Radnor, PA 19088
215-293-8860
Treasurer; Finance Committee

Other Board Members

BECK, Lawrence A.
Grumman Data Systems
R & D, MS D12-237
Woodbury, NY 11797
516-682-8478
X3V1.8 Coordinator

BEETON, Barbara
American Mathematical Society
P. O. Box 6248
Providence, RI 02940
401-272-9500 x299
BNB@Math.AMS.com;
BNB@XX.LCS.MIT.edu
Editor, TUGboat

CARNES, Lance
163 Linden Lane
Mill Valley, CA 94941
415-388-8853
Site Coordinator for "small" systems

CLARK, Malcolm W.
Imperial College
Computer Center
Exhibition Rd
London SW7 2BP, England
01-589-5111 x 4949
Janet: TeXline@UK.AC.IC.CC.VAXA

CRAWFORD, John M.
Computing Services Center
College of Business
Ohio State University
1775 College Road
Columbus, OH 43210
614-292-1741
Crawford-J@Ohio-State.edu;
Bitnet: TS0135@OHSTVMA
International Coordinator;
Prime 50 Series Site Coordinator

DYER, Allen R.
13320 Tridelphia Road
Ellicott City, MD 21043
(301) 531-3965

FARRELL, Shawn
Computing Centre
McGill University
805 Sherbrooke St W
Montréal H3A 2K6, Québec Canada
514-398-3676
Bitnet: CCSF@MCGILLA

FOX, Jim
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@uwacdc
CDC Cyber Site Coordinator

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

GIROUARD, Regina
American Mathematical Society
P. O. Box 6248
Providence, RI 02940
401-272-9500 x224
RMG@Math.AMS.com

GOUCHER, Raymond
TEX Users Group
P. O. Box 9506
Providence, RI 02940-9506
401-751-7760
REG@Math.AMS.com
Executive Director

GUENTHER, Dean
Computing Service Center
Washington State University
Pullman WA 99164-1220
509-335-0411
Bitnet: Guenther@WSUVM1
Finance Committee; IBM VM/CMS
Site Coordinator; Annual Meeting
Program Coordinator

HENDERSON, Doug
Division of Library Automation
Office of the President
University of California, Berkeley
300 Lakeside Drive, Floor 8
Oakland, CA 94612-3550
415-987-0561
Bitnet: dlatex@ucbcmsa
Metafont Coordinator

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

KELLERMAN, David
Northlake Software
812 SW Washington
Portland, OR 97205
503-228-3383
Usenet: imagen!negami!davek
VAX (VMS) Site Coordinator

KNUTH, Donald E.
Department of Computer Science
Stanford University
Stanford, CA 94305
DEK@Sail.Stanford.edu

MacKAY, Pierre A.
Northwest Computer Support Group
University of Washington
Mail Stop DW-10
Seattle, WA 98195
206-543-6259
MacKay@June.CS.Washington.Edu
UNIX Site Coordinator

PIZER, Arnold
Department of Mathematics
University of Rochester
Rochester, NY 14627
716-275-4428

PLATT, Craig R.
Dept of Math & Astronomy
Machray Hall
University of Manitoba
Winnipeg R3T 2N2, Manitoba Canada
204-474-9832
Bitnet: platt@uofmcc;
Bitnet: platt@ccm.UManitoba.CA;
CSnet: platt@uofm.cc.cdn
IBM MVS Site Coordinator

THIELE, Christina
Canadian Journal of Linguistics
Carleton University
Ottawa K1S 5B6, Ontario Canada
Bitnet: WSSCAT@CARLETON

WHIDDEN, Samuel B.
American Mathematical Society
P. O. Box 6248
Providence, RI 02940
401-272-9500
sbw@Math.AMS.com
Finance Committee

ZAPF, Hermann
Seitersweg 35
D-6100 Darmstadt
Federal Republic of Germany

Membership List — Supplement

Addresses and telephone numbers of individuals serving in more than one capacity are listed only once. Unless indicated otherwise, network address are for the Internet.

Site Coordinators

- **CDC Cyber**

  FOX, Jim
  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@uwacdc
  CDC Cyber Site Coordinator

- **DG MV**

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

- **IBM MVS**

  PLATT, Craig R.
  Dept of Math & Astronomy
  Machray Hall
  University of Manitoba
  Winnipeg R3T 2N2, Manitoba Canada
  204-474-9832
  Bitnet: platt@uofmcc;
  Bitnet: plattCSnet: platt@uofm.cc.cdn

- **IBM VM/CMS**

  GUENTHER, Dean
  Computing Service Center
  Washington State University
  Pullman WA 99164-1220
  509-335-0411
  Bitnet: Guenther@WSUVM1
  Annual Meeting Program Coordinator

- **Prime 50 Series**

  CRAWFORD, John M.
  Computing Services Center
  College of Business
  Ohio State University
  1775 College Road
  Columbus, OH 43210
  614-292-1741
  Crawford-J@Ohio-State.edu;
  Bitnet: TS0135@OHSTVMA
  International Coordinator

- **"small" systems**

  CARNES, Lance
  163 Linden Lane
  Mill Valley, CA 94941
  415-388-8853

- **UNIX**

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

- **VAX (VMS)**

  KELLERMAN, David
  Northlake Software
  812 SW Washington
  Portland, OR 97205
  503-228-3383
  Usenet: imagen!negami!davek

Committees

- **Annual Meeting Program Committee**

  FARRELL, Shawn
  514-398-3676
  Bitnet: CCSF@MCGILLA
  (Board of Directors)

  GUENTHER, Dean R.
  509-335-0411
  Bitnet: GUENTHER@WSUVM1
  (Board of Directors)
  Annual Meeting Program Coordinator

  THIELE, Christina
  Bitnet: WSSCAT@CARLETON
  (Board of Directors)

- **Bylaws Committee**

  BEETON, Barbara
  401-272-9500 x299
  BNB@Math.AMS.com
  (Board of Directors)

  DYER, Allen R.
  13320 Tridelphia Road
  Ellicott City, MD 21043
  (301) 531-3965
  (Board of Directors)
  Chair

  GOUCHER, Raymond
  401-751-7760
  reg@Math.AMS.com
  (Board of Directors)
  (ex officio)

  KNUTH, Donald E.
  (Board of Directors)
  (ex officio)

  PRICE, Lynne A.
  Hewlett-Packard
  3200 Hillview Avenue
  Palo Alto, CA 94304
  408-857-4075

  WHIDDEN, Samuel B.
  401-272-9500
  sbw@Math.AMS.com
  (Board of Directors)

- **Committee on Local User Groups**

  CLARK, Malcolm W.
  Janet: TeXline@UK.AC.IC.CC.VAXA
  (Board of Directors)

  GOUCHER, Raymond
  401-751-7760
  REG@Math.AMS.Com
  (Board of Directors)
  Chair

  HENDERSON, Doug
  415-642-9485
  Bitnet: dlatex@ucbcmsa
  (Board of Directors)

  THIELE, Christina
  Bitnet: WSSCAT@CARLETON
  (Board of Directors)

- **Membership Committee**

  ARMSTRONG, Mary
  TEX Users Group
  P. O. Box 9506
  Providence, RI 02940-9506
  401-751-7760
  (ex officio)

  CLARK, Malcolm W.
  Janet: TeXline@UK.AC.IC.CC.VAXA
  (Board of Directors)

  GIROUARD, Regina
  401-272-9500 x224
  RMG@Math.AMS.com
  (Board of Directors)

- **Nominating Committee**

  BARNHART, Elizabeth
  National EDP Department
  TV Guide
  Radnor, PA 19088
  215-293-8890

  MacKAY, Pierre A.
  Department of Computer
  Science, FR-35
  University of Washington
  Seattle, WA 98195
  206-543-6259
  MacKay@June.CS.Washington.Edu
  (Board of Directors)

  NAUGLE, Norman
  Mathematics Department
  Texas A & M University
  College Station, TX 77843-3112
  409-845-3104

- **Output Device Standards Committee**

  McGAFFEY, Robert W.
  Martin Marietta Energy Systems, Inc.
  Building 9104-2
  P. O. Box Y
  Oak Ridge, TN 37831
  615-574-0618
  McGaffey%ORN.MFEnet@nmfecc.arpa

- **1989 Scholarship Committee**

  SHARLOW, Larry
  Orban Associates Inc
  645 Bryant St
  San Francisco, CA 94107
  415-957-1063

  WITTBECKER, Alan
  TEX Users Group
  P. O. Box 9506
  Providence, RI 02940
  401-751-7760
  aew@Math.AMS.com

- **TUGboat Editorial Committee**

  BEETON, Barbara
  TUGboat@Math.AMS.Com
  (Board of Directors)
  Editor

  DAMRAU, Jackie
  Dept of Math & Statistics
  Univ of New Mexico
  Albuquerque, NM 87131
  505-277-4623
  damrau@dbitch.unm.edu;
  Bitnet: damrau@bootes
  Associate Editor, LATEX

  EPPSTEIN, Maureen
  Administrative Publications
  Stanford University
  Encina Hall, Room 200
  Stanford, CA 94305
  415-725-1717
  as.mve@Forsythe.Stanford.Edu
  Associate Editor for Applications

  HOENIG, Alan
  (Board of Directors)
  Associate Co-Editor, Typesetting on Personal Computers

  HOSEK, Don
  Platt Campus Center
  Harvey Mudd College
  Claremont, CA 91711
  BITnet: DHosek@YMIR
  Associate Editor for Output Devices

  JÜRGENSEN, Helmut
  Dept of Computer Science
  Univ of Western Ontario
  London N6A 5B7, Ontario, Canada
  519-661-3560
  Bitnet: A505@UWOCC1
  UUCP: helmut@deepthot
  Associate Editor for Software

  MANN, Laurie D.
  Stratus Computer
  55 Fairbanks Blvd
  Marlboro, MA 01752
  617-460-2610
  uucp: harvard!anvil!es!Mann
  Associate Editor for Training issues

  PFEFFER, Mitch
  Suite 90
  148 Harbor View South
  Lawrence, NY 11559
  516-239-4110
  Associate Co-Editor, Typesetting on Personal Computers

  TOBIN, Georgia K.M.
  The Metafoundry
  OCLC Inc., MC 485
  6565 Frantz Road
  Dublin, OH 43017
  614-764-6087
  Associate Editor for Fonts

  WITTBECKER, Alan
  TEX Users Group
  P. O. Box 9506
  Providence, RI 02940
  401-751-7760
  TUGboat@Math.AMS.com

TUG Institutional Members

Addison-Wesley Publishing Company
Reading, Massachusetts

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

Brookhaven National Laboratory
Upton, New York

Brown University
Providence, Rhode Island

California Institute of Technology
Pasadena, California

Calvin College
Grand Rapids, Michigan

Centre Inter-Régional de Calcul Électronique, CNRS
Orsay, France

City University of New York
New York, New York

College of St. Thomas
Computing Center
St. Paul, Minnesota

College of William & Mary
Department of Computer Science
Williamsburg, Virginia

COS Information
Montreal, P. Q., Canada

Data General Corporation
Westboro, Massachusetts

DECUS, L&T Special Interest Group
Marlboro, Massachusetts

Department of National Defence
Ottawa, Ontario, Canada

Digital Equipment Corporation
Nashua, New Hampshire

dit Company, Ltd.
Tokyo, Japan

Edinboro University of Pennsylvania
Edinboro, Pennsylvania

Electricité de France
Clamart, France

Environmental Research Institute of Michigan
Ann Arbor, Michigan

European Southern Observatory
Garching bei München, Federal Republic of Germany

Fermi National Accelerator Laboratory
Batavia, Illinois

Försvarets Materielverk
Stockholm, Sweden

General Motors Research Laboratories
Warren, Michigan

Geophysical Company of Norway A/S
Stavanger, Norway

Grinnell College
Computer Services
Grinnell, Iowa

GTE Laboratories
Waltham, Massachusetts

Hartford Graduate Center
Hartford, Connecticut

Harvard University
Computer Services
Cambridge, Massachusetts

Hewlett-Packard Co.
Boise, Idaho

Hobart & William Smith Colleges
Geneva, New York

Humboldt State University
Arcata, California

Hutchinson Community College
Hutchinson, Kansas

IBM Corporation
Scientific Center
Palo Alto, California

Illinois Institute of Technology
Chicago, Illinois

Imagen
Santa Clara, California

Informatika
Hamburg, Federal Republic of Germany

Institute for Advanced Study
Princeton, New Jersey

Institute for Defense Analyses
Communications Research Division
Princeton, New Jersey

Intevep S. A.
Caracas, Venezuela

Iowa State University
Ames, Iowa

Istituto di Cibernetica
Università degli Studi
Milan, Italy

Kuwait Institute for Scientific Research
Safat, Kuwait

The Library of Congress
Washington, DC

Los Alamos National Laboratory
University of California
Los Alamos, New Mexico

Louisiana State University
Baton Rouge, Louisiana

Marquette University
Department of Mathematics, Statistics
and Computer Science
Milwaukee, Wisconsin

Massachusetts Institute of Technology
Artificial Intelligence Laboratory
Cambridge, Massachusetts

Massachusetts Institute of Technology
Information Services
Cambridge, Massachusetts

Mathematical Reviews
American Mathematical Society
Ann Arbor, Michigan

Max Planck Institut für Mathematik
Bonn, Federal Republic of Germany

Max Planck Institute Stuttgart
Stuttgart, Federal Republic of Germany

McGill University
Montreal, Quebec, Canada

National Cancer Institute
Frederick, Maryland

National Center for Atmospheric Research
Boulder, Colorado

National Institutes of Health
Bethesda, Maryland

National Research Council Canada
Computation Centre
Ottawa, Ontario, Canada

National Semiconductor Corporation
Santa Clara, California

New Jersey Institute of Technology
Newark, New Jersey

New York University
Academic Computing Facility
New York, New York

Nippon Telegraph & Telephone Corporation
Software Laboratories
Tokyo, Japan

Northeastern University
Academic Computing Services
Boston, Massachusetts

Online Computer Library Center, Inc. (OCLC)
Dublin, Ohio

Pennsylvania State University
Computation Center
University Park, Pennsylvania

Personal TEX, Incorporated
Mill Valley, California

Purdue University
West Lafayette, Indiana

QMS, Inc
Mobile, Alabama

Queens College
Flushing, New York

Research Triangle Institute
Research Triangle Park, North Carolina

RE/SPEC, Inc.
Rapid City, South Dakota

Rice University
Department of Computer Science
Houston, Texas

Royal Marsden Hospital
Surrey, England

Ruhr Universität Bochum
Bochum, Federal Republic of Germany

Rutgers University
Hill Center
Piscataway, New Jersey

St. Albans School
Mount St. Alban, Washington, D.C.

Sandia National Laboratories
Albuquerque, New Mexico

SAS Institute
Cary, North Carolina

I. P. Sharp Associates
Palo Alto, California

Smithsonian Astrophysical Observatory
Computation Facility
Cambridge, Massachusetts

Software Research Associates
Tokyo, Japan

Sony Corporation
Atsugi, Japan

Space Telescope Science Institute
Baltimore, Maryland

Springer-Verlag
Heidelberg, Federal Republic of Germany

Stanford Linear Accelerator Center (SLAC)
Stanford, California

Stanford University
Computer Science Department
Stanford, California

Stanford University
ITS Graphics & Computer Systems
Stanford, California

State University of New York
Department of Computer Science
Stony Brook, New York

Stratus Computer, Inc.
Marlboro, Massachusetts

Syracuse University
Syracuse, New York

Talaris Systems, Inc.
San Diego, California

Texas A & M University
Computing Services Center
College Station, Texas

Texas A & M University
Department of Computer Science
College Station, Texas

Tribune TV Log
Glens Falls, New York

TRW, Inc.
Redondo Beach, California

Tufts University
Medford, Massachusetts

TV Guide
Radnor, Pennsylvania

TYX Corporation
Reston, Virginia

UNI.C
Danmarks EDB-Center
Aarhus, Denmark

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, Berkeley
Academic Computing Services
Berkeley, California

University of California, Berkeley
Computer Science Division
Berkeley, California

University of California, Irvine
Department of Mathematics
Irvine, California

University of California, Irvine
Information & Computer Science
Irvine, California

University of California, San Diego
La Jolla, California

University of California, San Francisco
San Francisco, California

University of Canterbury
Christchurch, New Zealand

University of Chicago
Computation Center
Chicago, Illinois

University of Chicago
Computer Science Department
Chicago, Illinois

University of Crete
Institute of Computer Science
Research Center
Heraklio, Crete, Greece

University of Delaware
Newark, Delaware

University of Exeter
Computer Unit
Exeter, Devon, England

University of Glasgow
Glasgow, Scotland

University of Groningen
Groningen, The Netherlands

University of Illinois at Chicago
Computer Center
Chicago, Illinois

University of Illinois at Urbana-Champaign
Computer Science Department
Urbana, Illinois

University of Kansas
Academic Computing Services
Lawrence, Kansas

University of Maryland
College Park, Maryland

University of Massachusetts
Amherst, Massachusetts

University of North Carolina
School of Public Health
Chapel Hill, North Carolina

University of Oslo
Institute of Informatics
Blindern, Oslo, Norway

University of Ottawa
Ottawa, Ontario, Canada

University of Southern California
Information Sciences Institute
Marina del Rey, California

University of Stockholm
Department of Mathematics
Stockholm, Sweden

University of Texas at Austin
Physics Department
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

Vanderbilt University
Nashville, Tennessee

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

(147 institutional members as of 20 Oct 1988)

TEX Consulting and Production Services

North America

ALDINE PRESS
12625 La Cresta Drive, Los Altos Hills, CA 94022;
(415) 948-2149
Composition services for mathematical and technical books.

AMERICAN MATHEMATICAL SOCIETY
P. O. Box 6248, Providence, RI 02940;
(401) 272-9500, ext. 224
Typesetting from DVI files on an Autologic APS Micro-5. Times Roman and Computer Modern fonts. Composition services for mathematical and technical books and journal production.

ARBORTEXT, Inc.
535 W. William, Suite 300, Ann Arbor, MI 48103;
(313) 996-3566
Typesetting from DVI files on an Autologic APS-5. Computer Modern and standard Autologic fonts. TEX installation and applications support. TEX-related software products.

ARCHETYPE PUBLISHING, Inc., Lori McWilliam Pickert
P. O. Box 6567, Champaign, IL 61821; (217) 359-8178
Experienced in producing and editing technical journals with TEX; complete book production from manuscript to camera-ready copy; TEX macro writing including complete macro packages; consulting.

HOENIG, Alan
17 Bay Avenue, Huntington, NY 11743; (516) 385-0736
TEX typesetting services including complete book production; macro writing; individual and group TEX instruction.

KUMAR, Romesh
1549 Ceals Court, Naperville, IL 60565; (312) 972-4342
Beginners and intermediate group/individual instruction in TEX. Development of TEX macros for specific purposes. Using TEX with FORTRAN for custom-tailored software. Flexible hours, including evenings and weekends.

OGAWA, Arthur
920 Addison, Palo Alto, CA 94301; (415) 323-9624
Experienced in book production, macro packages, programming, and consultation. Complete book production from computer-readable copy to camera-ready copy.

RICHERT, Norman
1614 Loch Lake Drive, El Lago, TX 77586;
(713) 326-2583
TEX macro consulting.

TEXNOLOGY, Inc., Amy Hendrickson
57 Longwood Ave., Brookline, MA 02146;
(617) 738-8029.
TEX macro writing (author of MacroTEX); custom macros written to meet publisher's or designer's specifications; instruction.

Outside North America

BAZARGAN, Kaveh
Optics Section, Blackett Laboratory, Imperial College of Science and Technology, London SW7 2BZ, U.K.;
(01) 589 5111 ext. 6841
Instruction in TEX, for beginner and intermediate levels. Custom macros, including complex tables.

TEXPERT SYSTEMS Ltd., (Malcolm Clark and Cathy Booth)
5, Northernhay Square, Exeter, EX4 3ES, Devon, U.K.;
(0392) 76091
The Complete TEX Service: consultation, all levels of document production, printing, customised style sheets (macros) to simplify text input, comprehensive training courses (will travel).

TEXWORKS Pty. Ltd.
157 Danks Street, Albert Park, Melbourne, Victoria 3206, Australia; 61 3 699 4083
Commercial versions of TEX and screen previewers for computers including PC, Macintosh, Amiga, Unix, VAX, and TEX-related products; print drivers for dot matrix printers, 300 dpi lasers, and phototypesetters (Autologic, Compugraphic and Linotronic); assistance with information on public domain software; consulting for typesetters and other TEX users.

VARDAS
88, North Rd., St. Andrews, Bristol, BS6 5AJ, U.K.;
(0272) 428709
TEX consultation, instruction, macro writing, book production, consulting.

Information about this service can be obtained from Ray Goucher, TEX Users Group, P. O. Box 9506, Providence, RI 02940, (401) 751-7760.