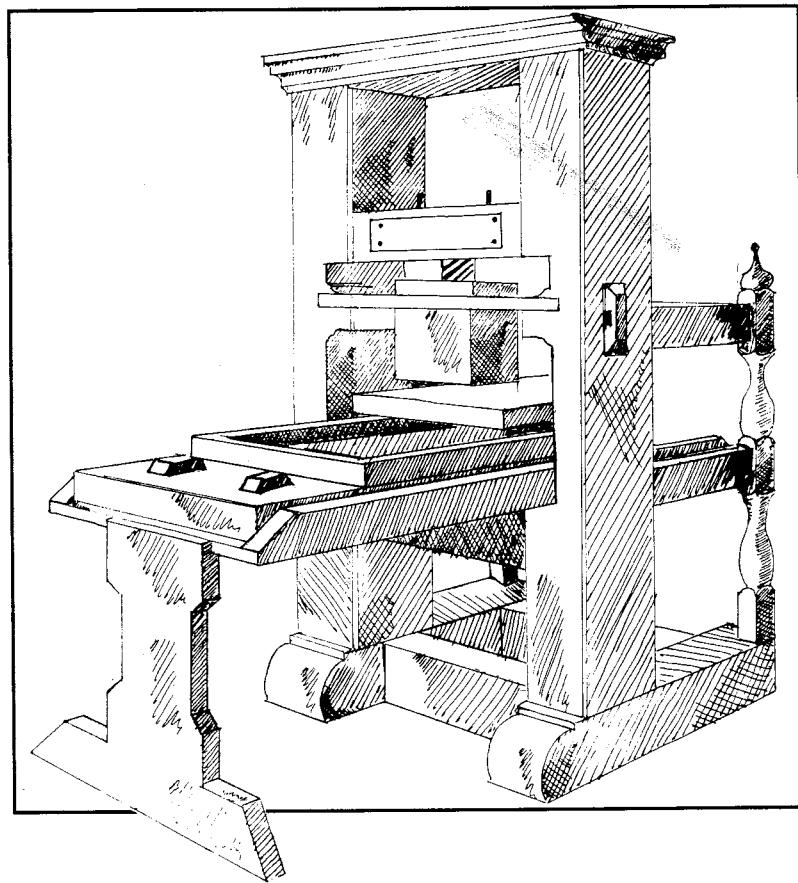


TUGBOAT

The Communications of the TeX Users Group



Volume 13, Number 2, July 1992

[Jenson] was so intent on legibility that he disregarded conformity to any standard—an innovation that modern designers might well consider.

Frederic W. Goudy,

*The Alphabet and Elements
of Lettering* (1942)

TUGBOAT

COMMUNICATIONS OF THE TEX USERS GROUP
EDITOR BARBARA BEETON

VOLUME 13, NUMBER 2
PROVIDENCE • RHODE ISLAND

JULY 1992
• U.S.A.

TUGboat

During 1992, the communications of the *TeX* Users Group will be published in four issues. One issue (Vol. 13, No. 3) will contain the Proceedings of the 1992 TUG Annual Meeting.

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

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

Submitting Items for Publication

The next regular issue will be Vol. 13, No. 4; deadlines are August 18, 1992, for technical items, and September 15, 1992, for reports and similar items; this issue is scheduled to be mailed in November. (Deadlines for future issues are listed in the Calendar, page 231.)

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

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

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

Reviewers

Additional reviewers are needed, to assist in checking new articles for completeness, accuracy, and presentation. Volunteers are invited to submit their names and interests for consideration; write to *TUGboat@Math.AMS.com* or to the Editor in care of the TUG office.

***TUGboat* Editorial Board**

Barbara Beeton, *Editor*

Victor Eijkhout, *Associate Editor, Macros*

Jackie Damrau, *Associate Editor, IATEX*

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

See page 131 for addresses.

Other TUG Publications

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

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

***TUGboat* Advertising and Mailing Lists**

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

Trademarks

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

APS μ5 is a trademark of Autologic, Inc.

DOS and MS/DOS are trademarks of MicroSoft Corporation

METAFONT is a trademark of Addison-Wesley Inc.
PC *TeX* is a registered trademark of Personal *TeX*, Inc.

PostScript is a trademark of Adobe Systems, Inc.

TeX and *AMS-TeX* are trademarks of the American Mathematical Society.

Textures is a trademark of Blue Sky Research.

UNIX is a registered trademark of UNIX Systems Laboratories, Inc.

General Delivery

Changing \TeX ?

Malcolm Clark

The topic of a 'future' \TeX has been a source of discussion ever since \TeX managed to slip out from Knuth's office. And twice there have been 'official' developments to the fundamental model: first with the transition from the original SAIL implementation through Pascal and eventually WEB—to what became known as $\text{\TeX}82$ when it was first released; and more recently the mild tinkering (by comparison) which is colloquially termed $\text{\TeX}3$. Knuth has made quite clear that he will not change \TeX again, and that while he is happy for anyone else to embed the program code in other applications, those applications are not to be termed \TeX . In passing I recall how much time I spent between 1984 and 1989 extolling one of the great merits of \TeX —that it was fixed and need not be relearned, unlike the tide of other (and commercial) software; only to have to retract a little in 1989. But this time I'm even more convinced.

But still, there are repeated reminders that \TeX is not perfect and that in order to stem the tide of newer, and obviously more desirable, systems for typesetting, document preparation, and electronic publishing, some radical surgery is needed. It seems hardly surprising to me that later systems should not surpass \TeX in their quality: I am more surprised that they often fail to do so. For example, given the relative ease with which the mathematical typesetting of \TeX might be extracted and used elsewhere, it comes as more of a surprise that even now, few other systems approach the standard which is already 14 years old. In my cynical moments I suspect that the world may not be as concerned with quality as \TeX claims to be.

How can these many suggestions for enhancement or improvement be accommodated? In the first place, we have to acknowledge that many 'enhancements' have taken place: from the very beginning there were variants—the shadowy Tyx, and latterly VORTEX, MLTEX, VTEx and so on. They appear to meet Knuth's requirement that they are not called \TeX . They each offer some extensions to meet some perceived lack or need (although VORTEX's pedigree is a little different). But what really are the most significant \TeX developments over the last ten years? The important

development has been LATEX—a change to the user interface, rather than an enhancement of some concept of typographic quality. Like its host, \TeX , it is an imperfect tool, but pragmatists recognize its widespread generality and applicability.

The major extensions of BIBTEX and MakeIndex were designed for use with LATEX. If there are two things we need right now, they are 'finished' versions of BIBTEX and MakeIndex for every platform that runs LATEX—and the documentation that goes with them. If the new books which are appearing managed to include these support facilities, they would be doing us all a service. (And if BIBTEX format were to be an output format of the many on-line bibliographies, that would be even nicer.)

In another area of 'support' facilities, it seems to me that one of the great imponderables is slowly disappearing: I almost always know how to invoke (IA)TeX, but figuring out how to magic up the printer driver is often a mystery. As POSTSCRIPT continues its inexorable growth, Tom Rockiki's DVIPS appears to be emerging as the most widely supported driver. This gives us the chance of *de facto* standardization and the chance to use the \special command without grief (probably to incorporate some graphics). If we couple this with the various applications which convert POSTSCRIPT to what your laser printer or screen can display (and some of these are public domain) we are heading for a device independence which was only a dream a few years ago.

Am I just being complacent? Is it sufficient to accommodate increased functionality and an improved user interface within the present shell?—one of the things which the LATEX3 project should deliver. Should I rather be worrying that it is difficult to create magazines like Newsweek with (IA)TeX? Would I be grateful for a copy of Frame, Interleaf, Grif, 3B2, or even Quark XPress? I quail before the thought of all that re-learning, frustration, the need to keep up with upgrades, the inability to swap documents easily between my Vax, SparcStation, Macintosh and the crusty MS-DOS machine gathering dust in the corner. Will I have to learn how to design documents myself? Life is too short. In future columns I will discuss how Shakespeare's Sonnets need drastic re-writing, and how Mozart's Magic Flute requires a change in the underlying paradigm to accommodate the developments in rap music and sampling.

But the 'stasis' strategy only retains the existing users: would anyone volunteer to adopt (IA)TeX from scratch? The one area we have done the most effective and consistent job is in telling everyone

how difficult (I^A)T_EX is to use, and what a dreadful typeface Computer Modern is. It used to be the English who had a reputation for under-statement and self-deprecation. (I^A)T_EXies have easily overtaken them. Why should this be?

Somewhere in this hyperbole serious questions are lurking. To what extent should TUG be pursuing the 'future' of T_EX? And which future? If we examine the TUG Bylaws, we will note that TUG was set up to 'identify, develop, operate, fund, support, promote and encourage charitable, educational and scientific programs and projects which will stimulate those who have an interest in systems for typesetting technical text and font design'. The german-speaking group, DANTE, addressed the topic of a future T_EX at their Hamburg conference (reported in this issue of *TUGboat* by Phil Taylor), and Rainer Schöpf has since set up an electronic discussion list. There is a paradox here of course: those who do want to change T_EX are more likely to participate than those who don't. It will be useful and instructive to see what shakes out of these discussions. There has already been a wide range of opinions expressed, from creeping featurism through to the adoption of new paradigms.

Of course, the choices are not simple, or exclusive. Improvements will take place in the user interface; at the same time, some brave souls will modify the underlying code. If changes are not generally available, and are restricted for proprietary or platform reasons, they are unlikely to be adopted by the present user base: if there is insufficient upwards compatibility, the inertial mass of existing documents may also discourage adoption; the prospect of change is ambiguous—it excites some and depresses others. Consider two examples of the diffusion of changes in the T_EX world: the change from Almost Modern to the Computer Modern typeface took an age, perhaps because the changes did not seem noticeable (so much for quality!); the change to T_EX3 appears to have been very swift—the lure of 8-bit input and the enthusiasm of the non-English speaking users seems to have been a major driving force here. Interesting times.

◊ Malcolm Clark
 Information Resource Services
 Polytechnic of Central London
 115 New Cavendish Street
 London W1M 8JS, England
 UK
 Janet: malcolmc@uk.ac.pcl.mole

Editorial Comments

Barbara Beeton

Another honor for Donald Knuth

During a ceremony held in the Stockholm City Hall on November 15th 1991, Donald Knuth was appointed Honorary Doctor of Technology by the School of Computer Science and Engineering, KTH, Stockholm. The appointment was accompanied by this citation.

Professor Donald Knuth is very well known to us, not only in Computer Science, but also in the fields of Mathematics and Typography. He has through his creative research and his monumental work *The Art of Computer Programming* made major contributions to the modern research area of mathematical analysis of algorithms and their complexity (performance), as well as given the virgin computer science a firm mathematical structure of great importance to undergraduate and graduate studies.

Roswitha Graham, head of the Nordic T_EX User Group, has provided the following report.

"Professor Knuth has for a long time had close contacts with researchers within the School of Computer Science and Engineering at KTH, and he is also present daily through his advanced computer tool T_EX for production of technical and



Donald Knuth receiving his KTH degree

mathematical reports with high quality typography, used by many students and researchers at KTH. The Nordic TeX User Group annual meeting 1991, was held on the 18th of November at KTH, with Professor Donald Knuth invited as guest of Honour and participants from four of the Nordic countries.

"The programme focused on needs and solutions for TeX quality typesetting of European languages as well as the LATEX user interface. Specially invited guest speakers were Frank Mittelbach, Germany, and Yannis Haralambous, France, who together with Jan Michael Rynning, KTH, Sweden (Swedish hyphenation for TeX), gave background to problems and pointed to solutions. Other speakers were Leif Andersson (A POSTSCRIPT font family for TeX), Niels Mortensen (Math and natural science typesetting—a LATEX report format), Peter Busk Laursen (XLATEX—extensions to LATEX and TeX at UNIC while we wait for LATEX 3.0) and Steen Larsen (Tailored database publishing with TeX)."

The session with Don was recorded, as was a later, less formal conversation between him and Roswitha. Among the topics discussed were Don's goals for his own work (which are centered around *The Art of Computer Programming* and do *not* include more work on TeX) and his impressions of the current contributors to TeX development. We hope to have excerpts from these sessions ready for publication in the fall issue.

Journals accepting TeX input

A topic that keeps appearing in the electronic TeX discussion lists is, "What journals accept manuscripts prepared in TeX?" My attention to this topic has been sharpened recently with the appearance of two items from unrelated sources.

The first item is a new journal that will be prepared with TeX: the *Journal of Computer Security*, published by IOS Press in Amsterdam. The director of the Press, Dr. Einar H. Fredriksson, sent me a copy for information, with the following comment:

As publisher I feel the TeX developments and potential have reached a point where we may have to re-evaluate the journal publishing system—and make all authors part of your Group.

This hardly sounds like an organization reluctant to embrace a new technology. A statement in the journal acknowledges the use of TeX and states that in general, the author's TeX files will be able to be used for articles accepted for publication; "[i]f possible, the authors are requested to use the

publisher's macros for the journal." (This is a legitimate request, as by so doing, authors will assure that their submissions conform to the production requirements of the journal, and thereby reduce the length of time between acceptance and publication by decreasing the technical demands on the journal staff.) TUG members interested in the research area covered by this new journal can find out more about the journal from one of the editors-in-chief: Prof. Sushil Jajodia of George Mason University (jajodia@gmuvox2.gmu.edu) or Dr. Jonathan Millen of the MITRE Corporation (jkm@mbunix.mitre.org); anyone without net access who requires a postal address can obtain this information from the TUG office.

The second item that caught my attention was a letter in the weekly *Science News* from the editor/publisher of *Solstice: An Electronic Journal of Geography and Mathematics*, Sandra Lach Arlinghaus, Director of the Institute of Mathematical Geography, Ann Arbor, Michigan. This letter states, in part,

Solstice is typeset using TeX, and it is the TeX file that is transmitted, complete with typeset tables as well as complicated mathematical notation. Indeed, *Solstice* has even run (in addition to scientific tables) an occasional crossword or word search puzzle simply to suggest this perhaps unexpected capability. It also transmits some figures—any that can be set using TeX. *Solstice* does claim to disseminate scientific results in an electronic form, and not only does it claim to do so, it does so.¹

The core of this item stresses a fact that has been obvious for years to TeX users: that TeX is not only a tool for communicating on paper, but can also be a means of structuring information for an electronic audience.

An incomplete list of journals, paper and electronic, that accept submissions in TeX form is given in the file `texjourn.bib`; this file and other TeX-related bibliographic information can be found at the archive `math.utah.edu` in the directory `/pub/tex/bib`. The contents of two of the files in that area—`texbook1.bib` and `texbook2.bib`—were published in the 1991 TUG Resource Directory; many additions have been made since then. Further additions to this bibliography are solicited (please check the current listings first); send them, preferably in BIBTeX form, to Nelson Beebe (beebe@math.utah.edu).

¹ Reprinted with the permission of the author.

The new membership list

Accompanying this issue of *TUGboat* is this year's membership list. You may notice that some information that was present last year isn't there in this edition, namely information about the hardware each member is using and the separate listings of members by computer and output device. The reason for this is the reduction of the staff in the TUG office, reported in *TEX* and *TUG News* earlier this year.

All address information has been posted in full, and appears in the same form as before. Regardless of staff availability, address updates must be kept current if TUG is to avoid delivery problems, and thus additional postage costs, on account of out-of-date information.

Hardware information is particularly susceptible to variation, and the time required to normalize and enter it into the database proved to be more than the reduced TUG office staff could handle. Actually, it has not been clear for some time how useful publication of this information is to the "average" member — a page and a half of names under the heading "IBM PC" doesn't seem to represent the same value as the same number of pages of a listing by geographical location. But your opinion is what counts. If you feel this information is essential, and especially if you have suggestions on how to streamline the process and make the presentation more useful, please send your suggestions to the TUG office, marked for the attention of the Membership Committee.

TUGboat production notes

The production of a publication such as this one is somewhat more complex than I believe most readers are aware. Some idea of the technical complexity can be gained by reading the production notes that appear in each issue. There are other facets to this as well, that I don't usually make a fuss about, but feel it's important to let the readers know why it takes so long to put each issue together.

Beginning with volume 12, no. 2, every article published in a regular issue has been subjected to a technical review by a volunteer referee, and the same is being done with this year's annual meeting proceedings. This review is not as intense as those for, say, the *Transactions of the AMS* or *The New England Journal of Medicine*, but it has resulted in numerous changes, and I think has improved the quality of the individual articles. The intent is not to decrease the number of articles published, but to make the articles that are published as accurate and informative as possible. This review takes time,

as does the interaction between editorial board and authors, to make sure the suggested changes are understood and properly installed. I would like to take this opportunity to thank the referees, who shall continue to remain nameless. There is more work than there are referees at present, and if you wish to perform a useful service (and see some interesting material prior to publication), you are invited to send your name in to the TUG office, care of the Editor, or via e-mail to tugboat@math.ams.com; please include a reliable address (e-mail if possible) and a description of your particular strengths and interests as well as identification of areas you wish *not* to cover.

The number of items is an important factor in determining how much time it takes to produce an issue. Most items are represented by two files — one for the publishable item, and one containing various auxiliary information concerning its receipt, review, and progress from submission to publication. But some items are much more complicated, requiring additional files of macros, examples, figures, and the like. For example, the archive for issue 13 no. 1 contains 197 files for 34 items listed in the contents. Ignoring the files of correspondence and other administrivia, that still comes to more than a hundred files. However worthy an idea, electronic distribution of *tugboat* is a concept whose time has not yet come, at least under the present staffing limitations.

A technical complication is the variety of forms in which articles may be submitted. There are essentially no restrictions, and submissions have been received on paper, on DOS or Mac diskettes, as coded (usually by *uuencode* or *atob*, but we haven't had much success with the latter) or unencoded files with or without compression, by e-mail or placed in directories for *ftp* access, as source (sometimes without *TEX* markup, but mostly using the *TUGboat* plain or *LATEX* macros, or, less often, some other scheme that must be translated to *TUGboat* style) or *.dvi* files. Usually, an article or two per issue requires font work — if the author agrees, we will generate fonts for one of the available typesetters (so that the quality of the camera copy is uniform) as well as for the local laser printers that are used to generate proof output. A growing number of submissions incorporate *POSTSCRIPT* inclusions, and the behavior of the (encapsulated) *POSTSCRIPT* code depends highly on the output device driver being used. Even files received in standard *.dvi* form aren't immune from problems; one such article in this year's first issue "broke" three typesetter drivers before we succeeded in finding one that would actually print it (three different

laser printer drivers and two previewers had produced satisfactory output, and the sudden failure at typesetter stage was a big surprise); the situation was dicey for a while, and we weren't sure that we wouldn't have to publish from laser output, but finally an acceptable combination was found and we got our typesetter output. By such little disasters we continue to learn and improve. But all this takes time, and attention from someone with substantial experience. Not a job for beginners.

Another facet of the scheduling problem is my own availability. Since 1986 I have been a member of national and international standards working groups developing a font standard for the International Organization for Standardization (ISO). These working groups meet at least 7 times per year, for a week or sometimes two at a time. Several times in the past few years these meetings have coincided with critical points in the *TUGboat* production schedule. Also, since the beginning of 1991, I have had no production assistance. This affects only regular issues, as proceedings issues are edited by other volunteers, with production in the TUG office. All these complications are magnified by the fact that editing *TUGboat* is a "hobby"; the job that pays the bills is in the Technical Support Group at the American Mathematical Society. What this means is that there is quite frequently nobody home to carry on the necessary correspondence, and delays result. This problem is being worked on, and I am hoping that assistance will once again become a reality later in the year. During this difficult period, thanks for your understanding.

I wouldn't like to leave the impression that the *TUGboat* experience is typical for publishers accepting articles or books in *TeX*. By its nature, *TUGboat* is expected to contain items that stretch the boundaries of what is possible with *TeX*. Most "ordinary" publishers don't want, don't expect, and aren't prepared to deal with such complications. They have, by and large, adopted *TeX* because of pressure from authors. But they still have a bottom line to look out for, and this doesn't allow much experimentation. So the successful publishers create macros that will help an author produce exactly what they are looking for, and instructions in using those macros. And smart authors follow the instructions.

- ◊ Barbara Beeton
American Mathematical Society
P. O. Box 6248
Providence, RI 02940
USA
bnb@Math.AMS.com

TUG Seeks Executive Director

The individual selected for this position will oversee the business and information dissemination activities of TUG; direct the promotional program to develop membership and TUG activities; develop a program of volunteer efforts for TUG activities; manage a small office staff with clerical, technical, and bookkeeping functions; and interact with TUG members and others in fields of interest to TUG. The Executive Director will report to TUG Board of Directors.

The following criteria will be considered as applicants are evaluated:

- experience in managing a business;
- skill in managing the retrieval, organization and dissemination of information;
- experience with the program *TeX* and related programs;
- computer experience and capability of understanding technical questions regarding *TeX* and related programs;
- good writing and speaking skills;
- good interpersonal skills;
- knowledge of considerations in managing a professional, non-profit association.

Applicants for this position should send indication of their interest and copies of their *curricula vitae* to:

Search Committee
TeX Users Group
 P. O. Box 9506
 Providence, RI 02940 USA

The *TeX* Users Group is an Equal Opportunity Employer.

TeX: The Next Generation

Philip Taylor

The notice of the annual DANTE meeting included an announcement that DANTE had decided to set up a working group to co-ordinate the further development of *TeX*. This announcement was circulated widely via the electronic lists, and all were invited to contribute ideas and work towards this goal. The first session of the working group took place at the end of March in Hamburg.

The meeting was opened by Joachim Lammarsch, who announced the creation of a group to co-ordinate an investigation into the future of *TeX* (the name will not be *TeX*, but no specific new name was proposed). The group will be led by Rainer Schöpf, and will include Peter Abbott, Peter Breitenlohner, Frank Mittelbach, Philip Taylor, Joachim Schrod, and Norbert Schwarz.

An e-mail list with open membership will be set up (*NTS-L@VM.URZ.Uni-Heidelberg.de*); it will be the primary source of suggestions to the group. The name 'NTS' is short for 'New Typesetting System', and is intended to be no more than an interim, working, non-contentious, name for the project.¹

Joachim then passed the meeting over to Rainer, who took as his starting point a paper that I had previously written in which were outlined five apparent options. These are:

- 1) Leave *TeX* as it is now. If Don is sufficiently happy with *TeX* that he is prepared to leave it for posterity in its present form, then I think we should certainly consider that as an option for the *TeX* world as a whole.
- 2) Extend *TeX* by just enough that those who really understand its deficiencies agree that the extensions are not only justified but *essential*: i.e. there are some 'simple' typesetting tasks with which *TeX_π* cannot deal correctly, but with which an Extended *TeX* could.
- 3) Extend *TeX* to incorporate the combined wish-lists of the major *TeX* practitioners, while retaining *TeX*'s present 'look-and-feel'.
- 4) Extend *TeX* as in (3), also taking the opportunity to reconsider *TeX*'s 'look-and-feel' and to implement major changes in that area if it is felt beneficial.
- 5) Design a typesetting system for the 21st Century, using whatever elements of *TeX* are felt to continue to represent the state of the typesetting art.

¹ Look at the last three words of the very first line in *The TeXbook*; thanks to Kresten Krab Thorup for pointing this out.

Rainer felt that (1) was not an option, and we then discussed in open forum options (2)–(5), all of which were felt by some present to have some merit. No firm decision was made as to which to pursue, and it was pointed out that the options were not necessarily exclusive—one could, for example, select (2) as a short time scale immediate action plan, while investigating (5).

The meeting then went on to take suggestions from the floor as to possible improvements, enhancements or suggestions. These included:

- Support for graphics;
- Support for colour;
- Support for rotated text;
- Support for grid alignment;
- Improved algorithms for page layout;
- Elimination of hard-coded constants and conventions;
- Improvement of *TeX* as a programming language;
- Extension of the 'boxes & glue' paradigm;
- Support for line numbering;
- Improvement of orthogonality and completeness;
- 'Make *TeX* object-oriented';
- Remove 'superfluous features'.

One speaker presented his own list:

- Who will (or should) use *TeX*?
- What is the user interface to be?
- Is backward compatibility essential, and if so, at the DVI or *TeX* level?
- The legal status;
- An implementation environment.

The meeting attempted to categorise these suggestions in terms of the five proposed options, but it became obvious that this was neither the time nor the place to do so. The meeting was then drawn to a close.

Addendum. The discussion list has now become active. Anyone interested in contributing to this activity can subscribe by sending the message

SUBSCRIBE NTS-L *<given name> <surname>*
to LISTSERV@VM.URZ.Uni-Heidelberg.de. The
NTS-L discussion is archived on

ftp.th-darmstadt.de [130.83.55.75]
directory pub/tex/documentation/nts-l

Files in this directory are bundled by month, named *yy-mm*, where *yy* is the year and *mm* is the month when the mail arrived. Access to this archive is via anonymous ftp. Access via mail-server will be made available later this year.

◊ Philip Taylor
Royal Holloway and Bedford New College
"The University of London at Windsor"
P.Taylor@vax.rhbnc.ac.uk

Software

Knuth's Profiler Adapted to the VMS Operating System

R.M.Damerell

Abstract

This article describes a Pascal profiler originally written by D.E. Knuth. In principle, this should be portable to any machine. In practice it required a lot of work to adapt it to VMS. We believe that the modified profiler can now support the whole of Standard Pascal and many non-Standard parts of VMS Pascal; and that it should be more easily portable than the original. We also provide a companion utility for generating execution count files.

Introduction

This article is about a Pascal profiler which was written by D.E. Knuth and distributed with the Stanford TeX software. As there seems to be no published description, we begin by explaining how it works.

Suppose you have a program—let's call it *Snail*—that runs unbearably slowly. A *profiler* is a supplementary utility that determines how much time the *Snail* is spending in executing different portions of its code. What usually happens is that a typical *Snail* will spend nearly all of its time executing a small subset of itself. Such a subset is usually stigmatised by such names as “bottleneck”, “critical section”, “innermost loop”, etc. Any serious attempt to speed up a *Snail* must concentrate on this “critical” section, either by actually rewriting it to run faster or by rewriting the higher-level code to make it run less often, or maybe adopting an entirely new algorithm. Nothing else is likely to make any significant difference. Thus a profiler is an essential tool for any programmer who is concerned about the execution speed of his or her programs.

Most profilers work by making some special calls to the operating system, asking it to monitor the behaviour of the *Snail* in some way as it crawls. Some typical examples are given in [1,2,6]. Knuth's profiler (called “*Profile*”) works on an entirely different principle. It reads the source code of *Snail*, making a table of the time consumed by

each statement. For each statement in the code, *Profile* estimates the time to be $w * f$ where:

- w , the *weight*, is the time taken to execute the statement once,
- f , the *frequency*, is the number of times the statement was executed in a run of the *Snail* program.

Profile then prints a listing of the *Snail* program with weight and frequency data added. The weight of each statement is estimated by parsing the statement, making reasonable assumptions about how it might be executed on a typical machine. If m and n are integers, the Pascal statement: $x:=2.1*(m+n);$ would probably be executed as: fetch m and n ; add; convert to real; multiply; deposit result in x . The costs of all these primitive operations are stored as constants in *Profile*. *Profile* adds them all together to get the weight, then multiplies by the frequency to get the total cost. The frequency is read in from a supplementary file called a *count* file. This contains a long list of numbers; essentially it lists the number of times every statement in *Snail* was executed in a trial run.

Thus it appears that *Profile* does not require any special help from the local operating system; so in principle it should be runnable on any machine. In practice it is a very different story. The main obstructions to running *Profile* on a new machine are:

1. No mechanism is provided for generating the *count* file.
2. All Pascal compilers implement different languages—of the same name!

We have been working on the problem of installing *Profile* on the VMS operating system, with the ulterior aim of eventually producing a portable version of *Profile*. This article describes the progress made so far. In order to avoid confusion, we call the altered program “VMS-*Profile*”, reserving “*Profile*” for Knuth's original.

Generating the Count File

Profile was originally written for the KL-10 machine at Stanford, on which D.R. Fuchs altered the system debugger to make it generate a *count* file. This is obviously not a practical option for other users. Hardly any manufacturers provide the source of their software and few site managers would allow ordinary users to alter it. Even if we could alter the VMS debugger, it could not be distributed as this would be a breach of copyright.

We have therefore written a completely separate utility called **Preprofile** for generating count files. **Preprofile** reads the source code of the **Snail** program and generates a new program file with a name like **SNAIL_COUNT.PAS**. If all goes well, this will be a valid Pascal program, which does everything that **Snail** does and also writes a count file, called **SNAIL.COU**. This can then be fed into **VMS-Profile** along with the original **SNAIL.PAS** file.

So in order to profile a program on VMS, you need to do the following: compile and link **VMS-Profile** and **Preprofile**; define commands to run them; run **Preprofile** on **Snail**; compile and link **Snail_count**; define further commands to run **Snail_count** instead of **Snail**. Then put the **SNAIL.COU** file into the same directory as the source of **Snail** and run **VMS-Profile** on **Snail** and (with luck) you get a profiled file called **SNAIL.PRO**.

The basic algorithm of **Preprofile** is fairly obvious. At each place in the **Snail** program file where **Profile** will need to see a count, **Preprofile** inserts a piece of code to advance a counter. Roughly speaking:

```
while <condition> do <statement>
becomes
while <condition> do begin
    count[i] := count[i]+1 ;
    <statement> end;
```

In the outermost block of **Snail**, **Preprofile** must declare all the extra variables. At the start of the statement part of the **Snail** program, **Preprofile** inserts code to set all the counters to zero. At the end, it inserts code to open the count file, write all the accumulated counts, then close it.

This mechanism now seems to be working, on all the **Snail** programs that we have tried. The most obvious disadvantage is that the **Snail_count** program will clearly run even more slowly than the original **Snail** did. The extra time is not itself all that important, because with luck you never need to run **Snail_count** more than once. The real disadvantage of the extra time is that **Snail_count** will never produce any useful information unless it can be run to completion. Another problem is that all the extra variables that **Preprofile** inserts into the **Snail** program must have names different from all the variables that were there previously. We have not managed to solve this problem; the best we can do is to give the extra variables unpronounceable names like "ZQRWHZ3XX" which do not figure prominently in most programmers' code.

Preprofile is a much simpler program than **Profile**. **Profile** has to parse the **Snail** program in great detail, but **Preprofile** is interested only in those syntax words of Pascal that affect the flow of control in **Snail**. It turned out that many of the most complicated parts of **Profile** could be replaced by a routine that merely copies parts of the text to the output file.

Improved Output

We have made several changes to **VMS-Profile** to try to improve the usefulness of its output. First consider the index of module names. **Profile** is designed to work with the Stanford WEB system. (We assume that everybody is familiar with WEB; see [4] if not.) As **TANGLE** assembles a WEB program, it inserts markers into its output like {123:}...{:123}, indicating the start and end of the replacement text of each module. If **Profile** sees these markers, it assumes that **Snail** was originally a WEB program and generates an index. For each module that contains executable code, **Profile** calculates the total cost of all the statements in that module. It also calculates the cost of each module as a percentage of the total cost of the whole program.

This index of modules is essential. The output of **Profile** is inevitably bulky, and without an index it would be a hopeless task to wade through an enormous listing in search of the critical sections. But **Profile** only makes an index if it sees WEB-style module markers. Therefore we have altered **VMS-Profile** to make it build an index of functions, in addition to **Profile**'s index of modules. (From now on "function" will include "procedure".) **VMS-Profile** calculates the cost of each function both as an absolute amount and as a percentage of the total cost.

We have made minor changes to the format of the index, to improve its signal-to-noise ratio. Since the percentage costs calculated by **VMS-Profile** are inevitably inaccurate, we see no point in giving them to 6 decimal places. Also we list only those modules or functions that score at least 2% of the total cost.

We have also altered the way **VMS-Profile** lays out the **Snail** program. The main output of **Profile** is the whole of the **Snail** program, with weight and frequency data attached. This is arranged in columns like this:

<statement> <weight> <frequency>

In **VMS-Profile**, we moved the weight and frequency columns to the left hand side. This change seems ridiculously trivial, but is actually important.

The original layout has the disadvantage that the statement has to fit into a fixed width. When the statement is indented, the width is further reduced. The effect is that **Profile** imposes a limit of $62 - k$ on the length of quoted strings in the **Snail** program, where k is the current amount of indentation. This is illogical because **Profile** is supposed to work with **TANGLE** and **TANGLE**'s limit is 69. If this limit is violated, **Profile** stops with a fatal error. VMS-**Profile** allows a much larger limit; if it sees an over-long string it merely splits it, so the output is essentially undamaged. The new layout means that a statement can now spread out to any width; the output file is much shorter because it does not need so much padding; we can add a column for *weight * frequency* (which is the data the user actually needs).

One of the methods that Knuth used for debugging **TeX** was the TRIP test [5,3]. This is a special input file containing many unusual constructions, intended to exercise the entire **TeX** program. He found that this is a powerful device for revealing obscure bugs in a program, after the obvious bugs have been fixed and the program seems to be working. In order to help with this method of debugging, VMS-**Profile** prints a list of the line numbers of executable statements that did not get executed in the trial run.

The Many-Languages Problem

This problem is compounded by the fact that **Profile** uses a rather simple-minded top-down parsing algorithm. It is well known that such parsing methods do not work well on programs that have syntax errors. In theory this should not matter because **Snail** has to be working before it makes any sense to try to profile it. In practice, **Profile** runs into trouble as soon as you try to move it to another machine, say from Machine A to Machine B. Every construction in B-Pascal that is not in A-Pascal is seen by **Profile** as a syntax error. The usual result is that after a little while, **Profile** becomes totally confused and loses track of the boundaries between statements in the **Snail** program. It is therefore essential to adapt **Profile** to read B-Pascal before it can be used on the new machine.

Of course we can always try to make ad-hoc changes to **Profile** to support this or that feature of the new language, but this approach produces masses of bugs. Even with a **debug_help** procedure (based on the one in **TeX**) it is a difficult business to adapt **Profile** to a new system. We believe that

we have managed to make VMS-**Profile** support the whole of Standard Pascal and many of the more accessible features of VMS Pascal. But we can never be sure that we have succeeded. There is always the danger that some unexpected (but perfectly valid) combination of Pascal syntax will reveal another bug. We believe that it will require a great deal of effort to produce a satisfactory solution of the many-languages problem. Meanwhile, neither **Profile** nor VMS-**Profile** can be regarded as portable.

The following examples will show some of the difficulty. Consider what happens when **Profile** reads a variable declaration, say

```
var horse, dog, goat: real;
```

Profile scans the list of names, then the type, then it sets up structures in its memory so that it will in future recognise a "horse" when it sees one. Now suppose that **horse** was previously declared in an outer block. Then **Profile** again does the obvious thing: it saves the previous definition of **horse** on a stack. When the current block is exited the previous definition will be un-saved. Now suppose the definition came in a procedure header, say:

```
procedure hunt(horse, fox: integer; yak: real);
```

Then **Profile** again knows what to do: it first defines the parameters **horse**, **fox**, and **yak**: then it defines the procedure itself.

All this is quite straightforward in principle; the details are not necessary here. Now consider: what must **Profile** do when it reads the word "forward"? If **horse** was defined in an outer block, that definition must be recovered from the stack. But also the new definition of **horse** as a parameter of **hunt** must be saved somewhere so that **Profile** will know what to do with **horse** when scanning the definition of **hunt**. This definition cannot be saved on the stack because the current stack frame will be erased by the time we reach the definition of **hunt**. It follows that we have to assemble an entirely new structure to represent a procedure header in order to handle forward declared procedures before they have been defined.

In VMS Pascal the formal parameters of functions can have default values. In the previous example, suppose that **horse** and **fox** had been declared with default values. Then when the function is called you can omit any parameters with defaults and pass the others by explicit assignment, as in: **hunt(yak:=4)**. The library procedures of VMS Pascal make extensive use of this feature.

The VMS versions of *T_EX* and METAFONT use parts of the VMS system library. In order to handle these programs VMS-Profile must read the library header file (called "starlet.pas"). This file is a monster, nearly three times as large as *TEX.PAS*. We had to increase the size of all the arrays in VMS-Profile to accommodate all the data; in turn this forced us to use long numbers to address these arrays because 16-bit numbers are not large enough.

It is clear that adapting Profile to another machine is not just a simple matter of adapting its system-dependent procedures to the eccentricities of a new compiler. Many of the internal structures have to be redesigned. These structures are represented by linked lists. It is terribly easy for list-processing programs to become messy, and messy programming is utterly abhorrent to the spirit of the WEB language. It is an accepted convention that any respectable WEB program must contain a clear explanation of how it is supposed to work. There seem to be two main difficulties that must be overcome in order to write a clean program that does list-processing. First, it is impossible to specify the structure of a complicated list in words. We need an easily-readable notation. We have therefore included in VMS-Profile the beginnings of a suite of *T_EX* macros for this purpose. These macros are no use for complicated lists; even so, they make a valuable contribution to the clarity of VMS-Profile. The Appendix at the end shows some examples.

The second main difficulty of list-processing is that Pascal has no suitable primitives; so every operation needs half-a-dozen statements. We have therefore written a set of WEB macros for simple list operations.

Although the many-languages problem is unsolved, we have managed to solve a small part of it. VMS Pascal provides a great many non-Standard predeclared functions. Some of these have weird syntax. The most extreme example is the *open* procedure, which links a disk file to a Pascal file variable. This procedure is both complicated and important; it is difficult to imagine how any serious programmer in VMS Pascal could avoid using it. Its declaration is something like this:

```
procedure open(file_variable:file;
  file_name:S_typ:='';
  history:H_typ:=new;
  record_length:integer:=132;
  access_method:A_typ:=sequential;
  record_type:R_typ:=variable;
  carriage_control:C_typ:=list;
```

```
organization:O_typ:=sequential;
disposition:D_typ:=save;
file_sharing:W_typ:=none;
function user_action:integer:=none;
default:S_typ:='';
error:E_typ:=message); extern ;
```

where *S_typ* can be any character string type and *H_typ*, etc., are enumerated types whose values are here immaterial. (The true definition of *open* is even more complicated than this simplified paraphrase suggests; it seems to be impossible to express this in Pascal.) In order to handle these predeclared functions, VMS-Profile must assemble suitable structures in memory to represent their headers. It would be an unbearably long and error-prone task to do this by hand. The only tolerable method is to write the declarations of these functions into a file and make VMS-Profile read them before it reads the Snail program itself.

For this purpose, we use the pool file mechanism of the WEB language. This is a most valuable feature of WEB which deserves to be far more widely used than it is at present. We have yet to see any large Pascal program that could not be improved by judicious use of this mechanism. It was invented by Knuth to circumvent the difficulty that Standard Pascal has no satisfactory mechanism for handling character strings. When TANGLE is assembling a WEB program, if it reads a string in double quotes, it writes that string into a supplementary file called a *pool* file. The idea is that the Tangled program can then read all these strings from its pool file into its memory. So we insert all the predeclarations into the *VMS_PROFILE.WEB* file. When VMS-Profile starts up it reads the pool file before it reads the Snail file. Here are some sample definitions:

```
declare("const true=1;false=0;",
  "maxint=2147483647;minint=-maxint;",
  "minchar=0;maxchar=255;",
  "type boolean=false..true;",
  "integer=minint..maxint;",
  "char=minchar..maxchar;",
  "text=file of char;")
```

The declarations are written in the usual Pascal form; they may extend over several lines and each line must be enclosed in double quotes. Then *declare* must be called on these lines. Several lines may be declared at once; then they must be separated by commas to keep TANGLE happy. Procedures and functions must have just the header, followed by "extern". For comparison, here is part of the equivalent code from Profile:

```

char_loc:=get_avail;
info(char_loc):=char_type;
int_loc:=get_avail;
info(int_loc):=int_type;
p:=get_avail; link(int_loc):=p;
q:=get_avail; val(q):=-max_int;
info(p):=q; q:=get_avail;
val(q):=max_int; link(p):=q;
bool_loc:=get_avail;
info(bool_loc):=int_type;
p:=get_avail; link(bool_loc):=p;
zero_loc:=get_avail; val(zero_loc):=0;
info(p):=zero_loc; one_loc:=get_avail;
val(one_loc):=1; link(p):=one_loc;
id5("f")("a")("l")("s")("e")
  (bool_const)(0);
id4("t")("r")("u")("e") (bool_const)(1);
p:=get_avail; val(p):=max_int;
id6("m")("a")("x")("i")("n")("t")
  (int_const)(p);
id7("i")("n")("t")("e")("g")("e")("r")
  (defined_type)(int_loc);
id7("b")("o")("o")("l")("e")("a")("n")
  (defined_type)(bool_loc);

```

And here is how it all works. `Declare` is a WEB macro with no replacement text. When TANGLE reads a `declare`, it first evaluates the argument. As this is a string in double quotes, it copies the string into the pool file. Then it evaluates the `declare` and solemnly puts nothing into the Pascal file. Then VMS-Profile reads the pool file and parses the declarations as if they were part of the Snail file itself. Where functions use non-standard syntax (like `write` and `open` above) we have made some ad-hoc changes to VMS-Profile's parsing routines to support these.

We believe that this mechanism is much cleaner than the previous one, as you can actually read the declarations. It does have one unfortunate consequence; in order for VMS-Profile to allow for the execution time of these predeclared functions, we must specify these times. This is done using Profile's "change-weight" mechanism. Recall that the weight of a statement is the estimated time to run it once. If Profile gets this wrong, you can rectify this by adding a so called "change-weight" comment, which looks like this: `{+100}`. This means "add 100 units to the cost of the current statement". In VMS-Profile you can add change-weight comments to external declarations, thus:

```

declare("function sin(x:real):real;",
       "extern{+100};")

```

If this comment is seen, then the number is assumed to be the cost of the function. In this respect VMS-Profile is inferior to Profile, because in Profile all the costs are tidily collected together in one place. We think the improvement in clarity outweighs the price.

Future Developments

The current version of VMS-Profile contains several problems besides those mentioned above. The first concerns the accuracy of the calculated profile. Ideally, when moving Profile to a new machine, one ought to calibrate it by measuring the time taken to do all the primitive operations and writing these times into the table of costs in the program. This would be a tremendously long and messy job, and probably not worth doing. Given that VMS-Profile works by examining the source of Snail, it cannot possibly have as close a contact with reality as a profiler that actually monitors the crawling Snail. On the other hand an accurate profile is neither needed nor possible. Any modern operating system is doing several jobs at once; so the time taken for a given task will vary according to the burden of other tasks. If a profiler gives a useful result, that result will be that "function X is using 10 times as much time as everything else". Since the truth is inevitably fuzzy, we believe that any calculated profile within a factor of 2 is probably good enough for practical purposes.

When discussing Profile's index, we said that Profile produces a list of all the modules in Snail and their total costs, and slurred over the question of how this is actually done. There are two ways of doing this. If M is a module, then its *explicit* cost is defined as the total cost of the statements actually contained in M. But WEB modules may be nested to any depth. So we can also define the *implicit* cost of a module. The implicit cost of module M is the total cost of the statements in M and also all modules directly or indirectly included in M. Roughly speaking, the explicit cost of a module is the amount of time you might save by rewriting its code to run infinitely fast; the implicit cost is what you might save if you could bypass that module altogether. Profile lists all the modules in Snail, giving both their explicit and implicit costs.

When VMS-Profile calculates its index of functions, it can only calculate explicit costs. The explicit cost of a function F is the (estimated) amount of time used by the code that is actually part of F, ignoring any time used by functions called by F. The only way we could estimate an

implicit cost of F would be by assuming that every invocation of every function uses the same time. (This assumption is clearly false, but VMS-Profile has no way to get more accurate information.)

Suppose that function F calls function G q times. Then we must add $q * c/n$ to the implicit cost of F, where c is the cost of G and n is the total number of times G has been called. This simple-minded approach fails when functions call one another recursively. In order to find implicit costs, VMS-Profile would have to solve a set of linear equations. It is easy to prove that the matrix of coefficients is nonsingular but ill conditioned. We have not tackled the problems of assembling these equations or of finding a suitable method for solving them.

In conclusion, we believe that Knuth's profiler is potentially a useful program, but it cannot realise its full potential until it is made portable. Copies of

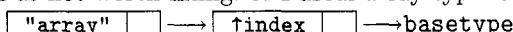
VMS-Profile and Preprofile have been submitted to the archives at Aston, with a suggested directory name “[tex-archive.utils.vms-profile]”. They may be freely copied, “as is”, on condition that no warranty is expressed or implied.

References

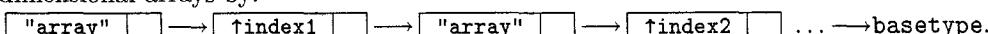
1. M. Bishop, *Profiling under UNIX by patching*, Softw. Pract. Exp., **17**, 729–739, (1987).
 2. T. Cargill and B. Locanthi, *Cheap hardware support for software debugging and profiling*, Computing Architecture News, **15** (5), 82–83, (1987).
 3. D.E. Knuth, *The Errors of TeX*, Softw. Pract. Exp., **19**, 607–686, (1989).
 4. D.E. Knuth, *Literate Programming*, Comput. J., **27**, 97–111, (1984).
 5. D.E. Knuth, *A torture test for TeX*, Stanford Comput. Sci. Report STAN-CS-1027, Nov. 1984.
 6. B. Plattner and J. Nievergelt, *Monitoring program execution: a survey*, Computer, **14**, 76–93, (1981).

Appendix 1

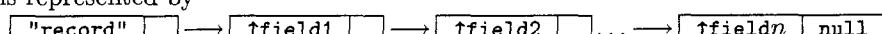
Here we give some examples of the linked-list macros mentioned earlier. There are some errors of mis-alignment, which we regard as not worth fixing. A Pascal array type is represented by the structure:



and multi-dimensional arrays by:

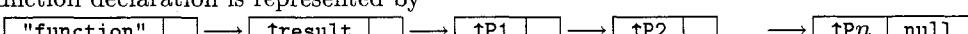


A record type is represented by

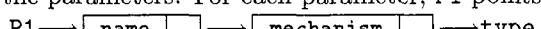


where each pointer field_i points to a table row.

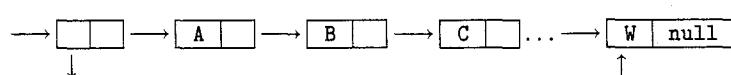
Finally, a function declaration is represented by:



where P1, etc. correspond to the parameters. For each parameter P1 points to:



While a list is being built, it looks like this:



This structure is non-intuitive, but it works. The chief booby-trap is that you must remember to remove or bypass the leading cell before starting to extract data from the list.

Appendix 2

This is the source for Appendix 1, with most of the plain text deleted. First, the underlying macros:

% This one puts a box around its argument; based on the
% 'control sequence token' macro in TeXbook

```
\def\cstok#1{\leavevmode\thinspace\hbox{\vrule\vtop{\vbox{\hrule
    \hbox{\vphantom{\char124}\thinspace\ninett#1}\thinspace}}}
\hrule\vrule}\thinspace}
```

```
% Partitioned boxes for linked lists

\def\abox#1{\cstok{\hskip0.5em#1\hskip0.5em}}
\def\dbox#1#2{\cstok{\hskip0.5em#1\hskip0.5em$char124$\hskip0.5em#2\hskip0.5em}}
\def\leftbox#1{\dbox{#1}{}}
```

```
\def\T0{$\longrightarrow$}
\def\m{\hskip0.5em&\hskip-0.5em}
\def\~{\char'013}
\def\d{$\downarrow$}
\def\u{$\uparrow$}
```

% Pascal arrays:

```
\centerline{\leftbox{"array"}\T0\leftbox{\^index}\T0 {\tt basetype}}
```

```
\centerline{\leftbox {"array"}\T0\leftbox {\^index1}\T0
\leftbox {"array"}\T0 \leftbox {\^index2} \dots\T0 {\tt basetype}.}
```

% Record type:

```
\centerline{\leftbox {"record"}\T0\leftbox {\^field1}\T0
\leftbox {\^field2}\dots\T0\dbox {\^field$n$\}{null}}
\noindent where each pointer {\tt field$i$} points to \dbox {name}{type}.
```

\noindent Finally, a function declaration is represented by

```
\centerline{\leftbox {"function"}\T0\leftbox{\^result}\T0
\leftbox{\^P1}\T0\leftbox{\^P2} \dots\T0\dbox{\^P$n$\}{null}}
```

For each parameter, {\tt P1} points to:

```
\centerline{{\tt P1}\T0 \leftbox{name}\T0 \leftbox{mechanism}\T0 {\tt type}.}
```

% List structure:

```
$$\vbox{ \baselineskip=12pt
\+ \T0\m\leftbox{} \T0\leftbox {A}\T0\leftbox {B}\T0\leftbox {C}\dots\T0
\m\dbox{W}{null}\cr \+\&\d&\u\cr \vskip-9pt \+\&\hrulefill&\cr }$$
```

◊ R.M.Damerell
 Maths Dept,
 Royal Holloway & Bedford New College
 Egham, Surrey, U.K.
 Janet: uhah208@uk.ac.ulcc.pluto

Fonts

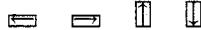
Arrows for Technical Drawings

David Salomon

Introduction

A general note: Square brackets are used throughout this article to refer to *The T_EXbook*. Thus [437] refers to page 437.

Arrows, both vertical and horizontal, are common in technical diagrams. Unfortunately, the arrows available in T_EX (`\rightarrow`, `\Rightarrow`, `\longrightarrow` [437], & `\rightarrowfill` [226]) are of limited use. The arrowheads are available in only one size and style, they already have short rules attached and, as the diagrams below show, they are inconveniently positioned in their bounding boxes.



The left and right arrows are shorter than their bounding boxes, and are not vertically centered in the boxes. The up and down arrows are narrower than their boxes, and have depths (of approximately 1.944pt). As a result, the simple construct

```
\def\vrulefill{\leaders\vrule\vfill}
\vbox to25pt{\offinterlineskip
\hbox{$\uparrow$}\vrulefill
\hbox{$\downarrow$}}
```

creates



and something more complicated is needed to align the rule with the arrows. Also the vertical size of the construction above is 26.944pt, not 25pt, because of the depth of the downarrow. Similarly, the result of

```
\hbox to25pt{$\leftarrow$%
\hrulefill$\rightarrow$}
```

is



Also, since each arrow is about 8pt long, very short double arrows are impossible to create. Something such as

```
\hbox to10pt{$\leftarrow$%
\hrulefill$\rightarrow$}
```

causes an ‘overfull box’.

L^AT_EX offers more arrows in its `line` font. They can point in quite a few directions, but there is only one style.

Description of the arrowheads

To satisfy my personal needs, I decided to develop a font for arrowheads that will be well documented and easy to use, yet general enough to produce arrowheads of many shapes. An important requirement was that the arrowheads be easy to place at the tips of rules. Since T_EX does not have diagonal rules, only horizontal and vertical arrowheads were developed. The methods used here, however, can easily be extended for diagonal arrowheads.

The discussion below assumes a right-pointing arrowhead, but the results can easily be applied to the three other directions. A general arrowhead (see Figure 1) is defined by five points, $z_1 \dots z_5$, of which z_4 is the origin, and z_5 is a reflection of z_2 (about the horizontal line at height $.5ruledim$). The two front edges are curved, the two back ones are straight, and there is a flat area at the base, to attach a rule. The exact shape of the arrowhead depends on the following parameters:

- wd is the distance from the tip to the base of the arrowhead. The bounding box has width wd .

- $tail$ is the distance from the base to the ends of the wings. The total width of the arrowhead is, therefore, $wd+tail$, but only wd units are included in the bounding box; the rest sticks out of it. Negative values of $tail$ produce arrowheads shaped like \blacktriangleleft , and large positive values ($>wd$) create arrowheads shaped like \triangleright . $tail$, therefore, should normally vary in the narrow range $0 \dots wd$.

- ht is the (approximate) total height of the arrowhead. The bounding box has height $.5ht$ (and zero depth). Very tall arrowheads, such as \blacktriangleright , are rarely used, so ht should normally be less than the total width of the arrowhead. Because of the special way arrows are used (see below), the bounding box has no depth. As a result, the left- and right-pointing arrowheads (and, normally, the upward one as well) stick below their boxes.

- The height of a standard `\rule` is 0.4pt, so it makes sense to center the arrowhead 0.2pt above the baseline. However, to allow for any size rule, there is a parameter, `ruledim`, whose value should be the height of the rule to which the arrowhead is going to be attached. The arrowhead is centered $.5ruledim$ above the baseline.

Points z_3 , z_4 guarantee that, regardless of the shape of the arrowhead, there will be a flat area of size `ruledim` at the base of the arrowhead, so it can be smoothly connected to the rule. A close look at the code shows that the height of the arrowhead ($z_2 - z_5$) is $ht - ruledim$ so, in order to end up with something that looks like an arrowhead, ht should

be greater than `ruledim`. (In rare cases, such as the 'pacman' arrowhead below, a large negative value of `curv` can increase the height of the arrowhead above this value.)

- The `curv` parameter can be used to curve the front edge of the arrowhead. Its value (in degrees) is added to the direction of the top front edge, and subtracted from that of the bottom front edge. Thus positive values of `curv` result in arrowheads looking like , and negative values, in arrowheads like . As a result, negative values of `curv` would rarely be used. The maximum value of `curv` (see discussion below) depends on the size and shape of the arrowhead, and is typically between 20° and 30°.

- `outlin` is a boolean parameter. If it is *true*, the arrowhead is drawn as an outline , using the procedure suggested in [Ex. 13.23]; otherwise, the arrowhead is solid. For high resolution output devices, Doug Henderson's methods (ref. 1) create better results.

The source code

In a complete arrowhead font, all the characters are arrowheads, differing only in orientation and parameters. It is therefore natural to define the arrowheads in terms of procedures. I have found it convenient to use two procedures, one for left/right arrowheads and the other for up/down ones. The METAFONT code of the procedures is as follows.

```

path outerr;
def outlne = % Outlining, see Ex. 13.23
cull currentpicture keeping (1,infinity);
picture v; v:=currentpicture;
cull currentpicture keeping (1,1)
withweight 3;
addto currentpicture also v
- v shifted right
- v shifted left
- v shifted up
- v shifted down;
cull currentpicture keeping (1,4)
enddef;

% procedure for right left arrowheads
def lr_head(text lr) =
R:=floor ruledim; if not odd R: R:=R+1; fi;
z1=(w,.5R); z2=(-tail,h);
z3=(0,2y1); z4=origin; z5=(x2,R-y2);
sAngle:=angle(z2-z1)+curv;
eAngle:=angle(z1-z5)-curv;
outerr:=z1{dir sAngle}..z2--z3--
z4--z5..{dir eAngle}cycle;
if lr="r":
fill outerr; if outlin: outlne; fi
elseif lr="l":
fill outerr
reflectedabout((0,0),(0,1))
shifted(w,0);
if outlin: outlne; fi

```

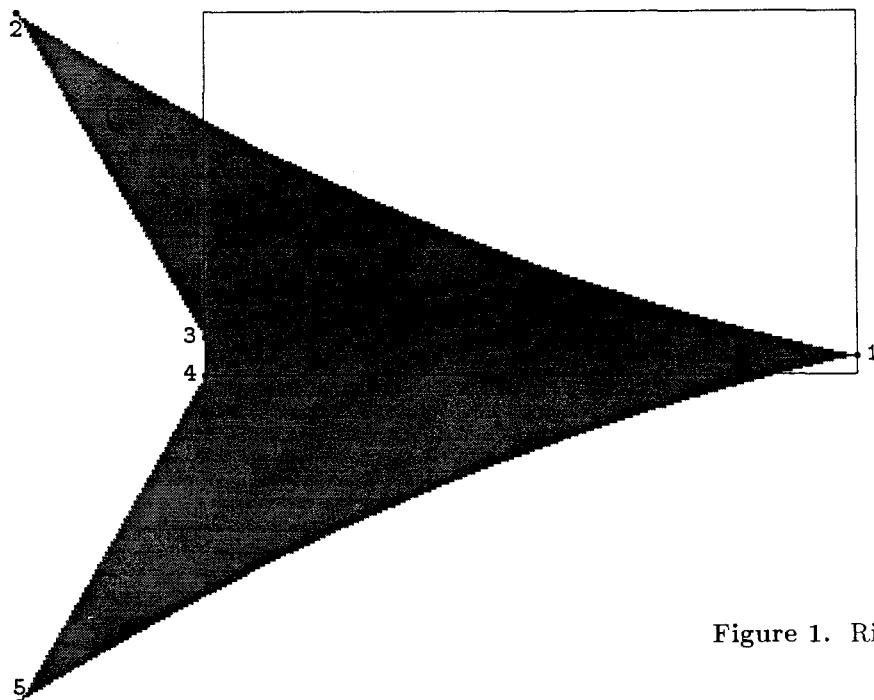


Figure 1. Right arrowhead

```

else: errmessage("wrong parameter,
    should be 'l' or 'r'");
fi
enddef;

% procedure for up down arrowheads
def ud_head(text ud) =
R:=floor ruledim; if not odd R: R:=R+1; fi;
z1=(.5R,h); z2=(w,-tail);
z3=(2x1,0); z4=(0,0); z5=(R-x2,y2);
sAngle:=angle(z2-z1)-curv;
eAngle:=angle(z1-z5)+curv;
outerr:=z1{dir sAngle}..z2--z3--
z4--z5.{dir eAngle}cycle;
if ud="u":
    fill outerr; if outlin: outline; fi
elseif ud="d":
    fill outerr
    reflectedabout((0,0),(1,0))
    shifted(0,h);
    if outlin: outline; fi
else: errmessage("! Wrong parameter,
    should be 'u' or 'd'");
fi
%

```

Following this, arrowheads can be created and placed in the font by, e.g.:

```

ruledim#:=.4pt#; outlin:=false;
ht#:=8pt#; wd#:=7pt#; tail#:=-2pt#; curv:=0;
define_pixels(ht,wd,tail,ruledim);

beginchar("R",wd#,5ht#,0); "right";
lr_head("r");
endchar;

beginchar("L",wd#,5ht#,0); "left";
lr_head("l");
endchar;

beginchar("U",.5ht#,wd#,0); "up";
ud_head("u");
endchar;

beginchar("D",.5ht#, wd#,0); "down";
ud_head("d");
endchar;

```

Note that it is also possible to create a hollow arrowhead by:

1. Drawing it with a 2-pixel wide pen. This may give better results in low resolution output devices.

```

if outlin: pickup pensquare scaled 2;
draw outerr; fi

```

2. After creating the arrowhead, a smaller arrowhead is erased inside. By changing the scale and shift amounts, special shapes can be created.

```

path iner
fill outerr
iner:=outerr;
if outlin: erase fill iner scaled .8
shifted(.1x1,.2y1); fi

```

An an example, the values

```

ht#:=8pt#; wd#:=5pt#; tail#:=2pt#;
curv:=9; ruledim#:=.4pt#;
produce ➤ when outlin:=false; and ➤ when
outlin:=true;;

```

Improving the digitization

The only subtle point about the procedures above is the equation for z_1 . Originally this equation was ' $z_1=(w,.5ruledim)$ '; but this resulted in arrowheads with flat tips, two pixels tall. To get a sharp, one-pixel tip, the y_1 coordinate should be an integer plus $1/2$ (see Ex. 24.7 in *The METAFONTbook*). This was obtained by computing ' $R:=\text{floor ruledim}; \text{if not odd } R: R:=R+1; \text{fi;}$ ' and setting ' $z1=(w,.5R)$ '. (R is the odd integer closest to ruledim , so $.5R$ is an integer plus $1/2$.) To end up with a symmetric arrowhead, the equation for z_5 was also changed from ' $z5=(x2,\text{ruledim}-y2)$ ' to ' $z5=(x2,R-y2)$ '. Notice that the base of the arrowhead (the distance between points z_3 , z_4) is now R and not ruledim , but the difference is at most one pixel, and is not noticeable. Notice also that the whole thing may not be necessary in a high-resolution output device, but in a 300dpi laser printer it significantly improves the appearance of the arrowhead (see Fig. 2).

Special cases

The top front edge of the arrowhead should go in the general direction of the top left point. If that direction is changed too much (by a large value of curv), funny results—and, sometimes, error messages—are obtained. As an example, the following set of parameters

```

ht#:=10pt#; wd#:=10pt#; tail#:=10pt#;
curv:=20; ruledim#:=.4pt#;
produces ➤

```

Some combinations of the parameters create interesting (and, possibly, even useful) shapes, even though they don't look like arrows. A pacman  is a left arrowhead created by:

```
ruledim#:=0pt#; outlin:=false; ht#:=2pt#;
wd#:=4pt#; tail#:=4pt#; curv:=-85;
```

A square diamond \blacklozenge , is created by: ht#:=10pt#;
wd#:=10pt#; tail#:=-5pt#; curv:=0;

A circular wedge \blacktriangleleft , is the result of: ht#:=10pt#;
wd#:=10pt#; tail#:=-7pt#; curv:=-30;

The examples shown here make it clear that the arrowheads are not meant to be used with text. Specifically, they don't have any depth, which interferes with the normal interline spacing, and they stick out of their boxes, which messes up the interword spacing.

Using the arrowheads

The arrowheads are meant to be used in diagrams, stuck at the ends of rules. A horizontal double arrow of size .5in \longleftrightarrow is obtained by \hbox to .5in{\ar l\hrulefill r}. A vertical arrow is also easy to create by means of vertical leaders. The ones shown here:



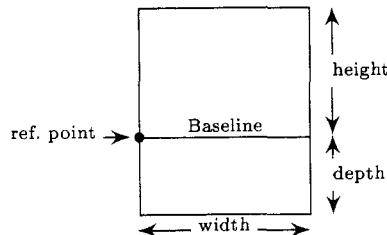
have been produced by

```
\def\vrulefill{\leaders\vrule\vfill}
\vbox to30pt{\offinterlineskip
\hbox{\ar u}\vrulefill\hbox{\ar d}}
\def\vRulefill{\leaders\vrule width1pt\vfill}
```

```
\vbox to30pt{\offinterlineskip
\hbox{\ar U}\vRulefill\hbox{\ar D}}
```

where positions 'u', 'd' of font \ar are occupied by up and down arrowheads with a base 0.4pt wide, and positions 'U', 'D' have similar arrowheads with 1pt wide bases.

As an example, the well known diagram [63] is duplicated here, using our arrowheads.



Bibliography

1. Henderson, D. *Outline fonts with METAFONT*, TUGboat 10, no. 1, 36–38, April 1989.

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

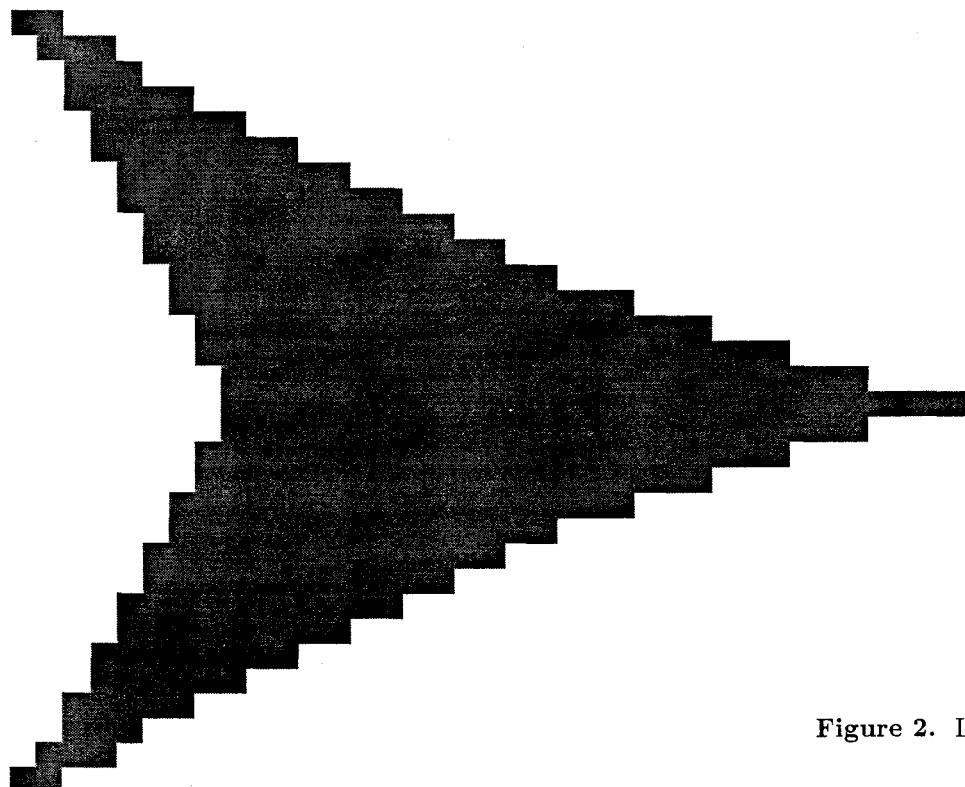


Figure 2. Lowres simulation

Graphics

A Solution to the Color Separation Problem

Daniel Levin

Color is one of the hallmarks of modern publishing, yet *TeX* makes no provisions for it. This is unfortunate, although understandable. Until the age of desktop publishing, colors were not part of the domain of typesetters. If a print job involved colors, the help of others was required: graphic artists to do the paste-up, photographers to do the separations, press operators to mix and apply the ink.

While *TeX* is an exceptional computerized typesetter, it has definitely not kept up with other programs in terms of color capabilities. In the *TeX* community, there is much concern that the program is losing its position as the premier typesetter for books and journals. I believe this process will not only continue, but accelerate, unless *TeX* adapts to the changes that have taken place on desktops and in printshops around the world. The changes are centered around new technology, such as color scanners and printers. But that is not all: the greatest change may be in people's expectations of what a typeset document should look like.

Perhaps some will argue that color is not an important issue, since the needs are different between academic presses, where *TeX* has the upper hand, and commercial operations, where desktop publishing is taking over. The differences are not so great, however. The argument may be based on the view that color is just an embellishment and is best left to graphic artists and photographers. In my opinion, such a view patronizes the arts and constrains the written word.

Color is helpful, if not essential, in many publications. It can highlight parts of a text, clarify certain points, and distinguish phrases and examples. Who has not seen a bible annotated or a textbook printed with answers in red? Who has not written a paper and wished that tables and graphs could have multiple colors? Who has not read a book or magazine and been drawn in by colorful headlines and diagrams, not to mention advertisements? Color printing has arrived and its role is widening, to bulletins, newspapers, even scholarly journals.

This article does not deal with all of the issues surrounding color. Instead, it focuses on just one: how to separate colors—or more specifically, how

to apply different colors to characters and rules. Also, while the title reads "A Solution...", the article is more of a proposed solution. It mentions a superset of *TeX* called Vector *TeX* (or *VTeX*), which uses scalable font technology.¹ So as to avoid the debate about names, we should think of *VTeX*'s newer capabilities as proposed extensions to *TeX*.²

Another focal-point of this article is on a certain type of publication: mathematics textbooks. The discussion is limited not only because of space but because of the author's experience in that field. Obviously there are other important uses of color, and those deserve careful attention before one can really claim to have solved the color separation problem. In particular, there needs to be an agreed-upon model of color that is convenient for both authors and publishers. The model must work with different tints or saturation levels; it should accept both process and spot colors; and it should allow colors to be layered (printed on top of one another) or "cancel out" those underneath. These are interesting topics, but they will have to be dealt with at another time.³

1. The traditional approach

As mentioned at the outset, *TeX* does not make any special provisions for color. You can say that it is a beautiful and flexible system, but when you get right down to it, everything is done in black and white. Nevertheless, there are two possibilities open to someone who wants to create a document with more than one color. They are not very satisfactory methods, but they do work in certain situations. The first method involves 'hboxes' and 'vboxes' and makes use of *TeX*'s \phantom command; the second relies on the METAFONT program to create invisible fonts.

a. Boxes. This method involves putting small amounts of text in boxes, and then keeping or discarding the boxes, depending on whether they match a given color. For instance, suppose you wanted to print a headline in blue and a short paragraph in black. That would require putting the headline in an 'hbox' and the paragraph in a 'vbox'. Then, to separate colors, you process the document twice, each time hiding one of the objects.

Let's say you wanted to print the headline and hide the paragraph. That is accomplished by typing \phantom{\vbox{paragraph material}}. To do the reverse, you remove the \phantom command from the paragraph (but keep the paragraph in its 'vbox') and type \phantom{\hbox{headline}}. In both cases, the \phantom command measures the enclosed box, discards it, and puts an empty,

similarly-sized box in its place. *All of the boxes are necessary.* They assure that the document will be laid out the same way each time it is processed.

The box method works reasonably well if it is done in vertical mode and with short amounts of text (or in horizontal mode with single words or characters). It does, however, have serious limitations. First, since all text is boxed, there is no way for page breaks to occur in the middle of a paragraph. Second, any phrase inside an ‘hbox’ is typeset at its natural width; no glue is allowed to stretch or shrink. Third, **TEX** spends extra time building and sizing boxes. Fourth, the method gets increasingly complicated with each additional color.

A fifth problem relates to the design and proofreading stages: there is no way to approximate how colors look next to each other. Your choices are to print colors one at a time or altogether (presumably in black). Referring to the previous example, it would be nice if you could print the paragraph normally and the headline at, say, 50% gray—in other words, give the headline a rough approximation to blue. (It would be even nicer, if one has a specially-equipped printer, to see a document’s true colors.) Not being able to see colors, or at least approximate them, makes page layout more difficult and color assignment a lot of guesswork.

b. METAFONT. Instead of putting text in a box and then discarding it, you can use the METAFONT program to create ‘invisible’ fonts. These fonts have the same dimensions as visible ones, but they have neither an outline nor a fill—which is to say, they do not show up in print. By substituting invisible for visible fonts, you can hide any amount of text. This method overcomes most of the problems of the box method; glue can stretch and shrink, and **TEX** can calculate optimal breakpoints. But it still leaves you with an ‘all or nothing’ proposition: either things are printed in black or they are completely hidden.

Invisible fonts are often included in **LATEX** and **SUTEX** packages. They are easy to spot, because they have an ‘I’ as the first character of their names (e.g., the invisible roman font is **I**CMR10). If you can obtain invisible fonts, you may find they are not a complete set. Fortunately, the method of creating them is simple. You take a METAFONT file like CMEX10.MF and make a copy with the name ICMEX10.MF. Then, just before the last line of the file, type

```
extra_endchar:=  
    extra_endchar&"clearit";
```

(so that what you type becomes the second-to-last line). Then run the file through the METAFONT program. Everything is done as usual, except that METAFONT erases all character representations; TFM files and “empty” font files are still created.

When dealing with ordinary text, invisible fonts are a reasonably effective way to separate colors. But they do not offer a complete solution. Surprisingly, they work least well with mathematics, because they do not hide rules. You see, rules are not characters, so there is no way to make them invisible using METAFONT. Even if all fonts are made invisible, rules still show up as horizontal bars in fractions and radicals, and as ‘overlines’ and ‘underlines’.

Perhaps the best thing to do with standard implementations of **TEX** is to combine invisible fonts with the box method (above). If something involves a horizontal or vertical rule, you simply put it inside a box and, when the time comes, hide it using the \phantom command. Fractions, radicals, underlined words, etc., are always typeset at their natural width, regardless of whether they are boxed. So the major problems with boxes do not apply here.

With a good font scheme and judicious use of the \phantom command, you can effectively separate colors. You still run into the proofing and design problem mentioned earlier, since you cannot see colors next to each other. Also, you may run into memory problems, because you need to load most fonts twice (a visible and invisible version). And, of course, you must keep many more TFM and PK files on your system. If you are not bothered by such things, it is possible—though still difficult—to print colors one at a time.

2. A newer approach

The VTEX program offers the best method for separating colors that this author has seen. As stated above, VTEX uses scalable font technology, which has all kinds of advantages over traditional bitmapped fonts (not the least of which is a significant reduction in the number of files). In addition to offering limitless font sizes, VTEX can fill characters and rules with a variety of patterns and grayscales. This gives one the capability of separating colors and simulating them in black and white.⁴

VTEX’s method is straightforward. Using a \special command, you specify a ‘fillpattern’ for both characters and rules. The syntax is \special{F#}, where # is a number corresponding to a particular pattern. VTEX has 22 built-in

patterns and supplies an editor for creating new ones. Of special interest are patterns 0 (black), 8 (about 50% gray) and 22 (white). With those three patterns, you can readily design and print a two-color document. (To simulate additional colors, you need to select more patterns.)

a. Implementation. One can think of various schemes for assigning colors with the `\special` command. A simple approach, which also resembles TeX's `\font` command, uses the following macros:

```
\def\color#1{\toks@={#1}
 \afterassignment\color@\count@}
\def\color@{\expandafter\edef
 \the\toks@{\special{F\the\count@}}}
```

Briefly, the `\color` command matches up a color with a particular fillpattern. The syntax is `\color{name}=#`, where the name is written as a control-sequence and the number corresponds to a fillpattern.⁵ For example, if you wanted to assign fillpatterns to two colors, blue and black, you could type `\color\blue=8` and `\color\black=0`. Then, to typeset characters and rules normally, you give the command `\black`. To approximate the other color, you give the command `\blue`. Thus the line

```
\black Something old. \blue
Something new. \black
```

yields

Something old. Something new.

Note: The second `\black` command is necessary. Color changes cannot be confined to a group because the `\special` command is inherently global (it sends messages directly to the device driver).

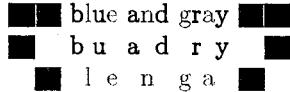
The previous example illustrates colors side by side. By now the reader has probably figured out how to separate them. The trick is to re-define colors. To be specific, any color that should not be printed is assigned fillpattern 22 (white). Following this approach, the lines

```
\def\civilwar{{%
 \blue \square \gray \square \
 \blue b\gray l\blue u\gray e
 \blue a\gray n\blue d
 \gray g\blue r\gray a\blue y\
 \gray \square \blue \square \par}}
\def\square{\vrule width1em}
```



```
\color\blue=0 \color\gray=8 \civilwar
\color\blue=0 \color\gray=22 \civilwar
\color\blue=22 \color\gray=8 \civilwar
```

result in



- | | |
|--|--|
| <ol style="list-style-type: none"> Simplify: $(3xy^{-2})(\frac{1}{6}x^3y^2) = \frac{1}{2}x^4$ Explain why the sum of two odd numbers is always even.
If the first number is $2a + 1$ and the second is $2b + 1$, then the sum is $2(a+b) + 2$ or $2(a+b+1)$, which is even. | <ol style="list-style-type: none"> Solve using the quadratic formula:
$2a^2 - 4a - 5 = 0$
$\frac{2 \pm \sqrt{14}}{2}$ Write an equation for the parabola whose x-intercepts are -3 and 3 and which contains the point $(0,6)$.
$3y + x^2 - 18 = 0$ |
|--|--|

Figure 1. Black = fillpattern 0 ; Red = fillpattern 0.

- | | |
|--|--|
| <ol style="list-style-type: none"> Simplify: $(3xy^{-2})(\frac{1}{6}x^3y^2) = \frac{1}{2}x^4$ Explain why the sum of two odd numbers is always even.
If the first number is $2a + 1$ and the second is $2b + 1$, then the sum is $2(a+b) + 2$ or $2(a+b+1)$, which is even. | <ol style="list-style-type: none"> Solve using the quadratic formula:
$2a^2 - 4a - 5 = 0$
$\frac{2 \pm \sqrt{14}}{2}$ Write an equation for the parabola whose x-intercepts are -3 and 3 and which contains the point $(0,6)$.
$3y + x^2 - 18 = 0$ |
|--|--|

Figure 2. Black = fillpattern 0 ; Red = fillpattern 8.

b. Textbook Example. Another, more realistic example of color separation has to do with school textbooks. It is customary with teachers' editions of textbooks to print answers next to problems (exercises). Usually the answers are printed in a color other than black, in order to distinguish them from problems. This is an especially common practice with mathematics textbooks, and it makes up a good case study for **TEX**. While **TEX** is superb at arranging problems and answers, it does require extensions to handle the color separation.

Let us now consider the situation in a little more detail. Interestingly, color separation is not the toughest obstacle to overcome. With VTEX's **\special** command, as described above, one can easily assign different colors to problems and answers. The real task is in positioning answers next to problems. Ideally, one would like to automate the process, and be economical about it—that is to say, put answers as close to problems as possible. This requires some ingenuity. One needs to figure out how many lines are in a problem, how wide is the last line, and whether an answer will fit in the available space.

In the appendix to this article are some macros which make reasonably intelligent decisions about the placement of answers. There are four primary commands, called **\beginproblem**, **\beginanswer**,

\endanswer, and **\endproblem**. They are used in the following manner.

```
\beginproblem
general text (problem)
\beginanswer
general text (answer)
\endanswer
\endproblem
```

The 'begin' and 'end' commands form boundaries around problems and answers. Incidentally, the **\beginanswer... \endanswer** field is optional.

The **\beginproblem** command serves several purposes: it assigns a color (normally black), it numbers problems automatically, and it puts problems inside 'vboxes' so that they can appear next to each other. The **\beginanswer** command assigns a color (red) and looks at current conditions. If **TEX** is in vertical mode, the answer simply begins a new paragraph. Otherwise, the answer is put inside an 'hbox' and the last line of the paragraph is looked at more closely.

What happens next depends on several things: the 'hsize' and number of lines in the current paragraph (*h* and *n*), the width of the last line (*l*), and the width of the answer box (*a*). Roughly speaking, if *a* < *h* − *l*, the answer is placed at the end of the last line of the problem. Otherwise, the answer is 'unboxed' and begins a new paragraph.⁶

- | | |
|--|---|
| <ol style="list-style-type: none"> 1. Simplify: $(3xy^{-2})(\frac{1}{6}x^3y^2)$
 3. Explain why the sum of two odd numbers is always even. | <ol style="list-style-type: none"> 2. Solve using the quadratic formula:
$2a^2 - 4a - 5 = 0$
 4. Write an equation for the parabola whose <i>x</i>-intercepts are -3 and 3 and which contains the point $(0,6)$. |
|--|---|

Figure 3. Black = fillpattern 0 ; Red = fillpattern 22.

$$\frac{1}{2}x^4$$

$$\frac{2 \pm \sqrt{14}}{2}$$

If the first number is $2a + 1$ and the second is $2b + 1$, then the sum is $2(a + b) + 2$ or $2(a + b + 1)$, which is even.

$$3y + x^2 - 18 = 0$$

Figure 4. Black = fillpattern 22 ; Red = fillpattern 0.

There is just one more determination: if the answer fits on the last line of the problem, and if $n > 1$, the answer box is lowered a little bit. This helps to distinguish the answer from any text which appears above it.

The effect of the macros can be seen in figure 1. Notice that problems and answers are given fillpattern 0 (black). In figures 2–4, the problems and answers are given different fillpatterns; in other words, the commands `\black` and `\red` are re-defined. No other changes are made to the input lines. Finally, notice that all text appears in exactly the same position, regardless of its color (or more precisely, its fillpattern). The color separation is achieved without any extra boxes or font assignments, and with very little fuss!

Conclusion

The importance of color in modern publishing has already been emphasized. The author hopes that the textbook example shows how useful is a good color separation scheme. The macros presented in this article are by no means a complete solution, but they enable *TeX*nicians to perform many important tasks. Also, they introduce some ideas about color that warrant further discussion. Here are three more points that ought to be debated:

1. The separation of colors is best handled by device drivers; *TeX* does not need to be concerned with the fillpattern of characters and rules. The only situation where *TeX* may need to be actively involved is in the layering of colors. For example, a perfectly adaptable system should allow yellow letters to be printed on a cyan background, without the cyan showing through.⁷

2. A color assignment scheme should be part of plain *TeX* (or *LATeX*), so that full-color documents can be shared between various platforms and implementations. This can be done in many different ways, perhaps with a font-like command (as in this article) or else with a numbering system (as in `\newtoks`, `\newfam`, etc.).

3. Not all device drivers need to be color capable, but they need to be color *aware*. Even if a device is monochrome, it can still simulate colors with grayscales and other patterns. In short, no driver should be tripped up by color; it should be able to turn hues into black and white.

Finally, it is hoped that members of the *TeX* community will join together in developing new standards for the program, especially in regard to color. Extensions should not be dismissed out of hand, as if they were unneeded or contrary to the

goals of *TeX*. Rather, they should be looked at real innovations, or as springboards for improving the program, or at the very least as a reflection of the needs of publishers. Sooner or later those needs will be met by a computer system. Wouldn't it be great if that system were *TeX*?

Appendix

The textbook macros are relatively easy to follow, except for `\measurelastline`. This performs some measurements which are used in the positioning of an answer (`\placeanswerbox`). The macro is derived from an example given by Frank Mittelbach.⁸ As he indicates, the only way to measure a line is to go into display mode temporarily. *TeX* then sets `\predisplaysize` equal to the width of the last line (plus two ems). Simultaneously, `\prevgraph` is set to the number of completed lines in the paragraph.

Once the measurements are made, it is necessary to undo the skips and penalties associated with display mode. This sends *TeX* back to the end of the last line, where horizontal (paragraph) mode can begin again, virtually uninterrupted. There is one major complication: when *TeX* begins display mode and then reverses its steps (does an ‘*unskip*’), the depth of the last line is lost. That is why a ‘*vrule*’ is added to the list. It acts like a ‘*strut*’—in other words, it guarantees that the last line has the proper depth.

Note: Another fix is needed if an answer is unusually tall and is placed on the first line of a problem. In that case, a correction must be made to the height of the ‘*vbox*’ which contains the problem and answer. That is not difficult to do and is left to the reader as an exercise.

```

\newdimen\probwidth \probwidth=12pc
\newdimen\gutterwidth \gutterwidth=2pc
\newdimen\answerskip \answerskip=1em
\newdimen\numberwidth \numberwidth=1.5em
\newcount\probnumber \probnumber=1
\newdimen\lastlinewidth
\newdimen\lastlinedepth
\newcount\lastlinenumber
\newbox\answerbox

\def\beginproblem{\black
  \ifodd\probnumber \allowbreak
    \else \hskip\gutterwidth \fi
  \hbox to \numberwidth{%
    \the\probnumber.\hfil}\nobreak
  \vtop\bgroup \hsize=\probwidth
  \noindent}

```

```

\def\beginanswer{\red
\ifvmode
  \setbox\answerbox=\box\voldb@x
  \else \measurelastline
  \setbox\answerbox=\hbox \fi
\begin{group}

\def\endanswer{\egroup
\ifvoid\answerbox
  \else \placeanswerbox \fi}

\def\endproblem{\par
\vskip2\parskip \egroup
\advance\probnumber by 1}

\def\placeanswerbox{%
\dimen@=\hsize
\advance\dimen@-\lastlinewidth
\advance\dimen@-\answerskip
\ifdim\wd\answerbox<\dimen@
  \hskip\answerskip
  \ifnum\lastlinenumber>1
    \lower\ht\answerbox \fi
  \box\answerbox
  \else \par \normalbaselines
  \noindent \unhbox\answerbox \fi}

\def\measurelastline{%
$$\nodisplaylineskip
\global\lastlinenumber=\prevgraf
\dimen@=\predisplaysize
\advance\dimen@-2em
\global\lastlinewidth=\dimen@ $$%
\par \unskip \unpenalty
\setbox0=\lastbox
\lastlinedepth=-\lastskip
\vskip-\parskip
\baselineskip=\z@
\lineskiplimit=-\maxdimen
\noindent \hskip\lastlinewidth
\hrule width\z@ height\z@
  depth\lastlinedepth}

\def\nodisplaylineskip{%
\abovedisplayskip=\z@
\belowdisplayskip=\z@
\abovedisplayshortskip=\z@
\belowdisplayshortskip=\z@
\baselineskip=\z@
\lineskiplimit=-\maxdimen}

```

Notes/References

1. VTEX is a trademark of MicroPress, Inc. See the listing in TUG Resource Directory, *TUGboat* 12, no. 2, supplement, p. 129.
2. A similar point was made by Nelson Beebe in *TUGboat* 11, no. 3, p. 335. He noted that MicroPress "has done very interesting things with extensions to *TeX*," and these should be studied as "a pilot implementation of some ideas for *TeX*'s evolution."
3. Color graphics are something else to consider. Donald Knuth and others have suggested using a "grayscale font" to create halftoned pictures. This is one way to separate colors, as pointed out by Adrian Clark in *TUGboat* 12, no. 1, pp. 157–165. The method, however, requires a lot of processing power and is device-specific. Also, it does not help someone who wants to apply a shade of gray to a regular font.
4. For a review of VTEX, see *AMS Notices*, 38(2), February 1991, pp. 105–109. The program is also discussed by Al Cameron in *Personal Workstation*, June 1990.
5. This is something like the method suggested by Robert Adams, in *TUGboat* 11, no. 3, pp. 405–406. He suggests using commands such as \black and \red, and tying them into the PostScript operator setgray.
6. This approach is not without complications. For an interesting discussion of line-breaks in 'unboxed' text, refer to an article by Michael Downes, *TUGboat* 11, no. 4.
7. Knuth implies as much about device drivers: with the \special command, one can take advantage of any equipment that might be available—e.g., "for printing documents in glorious *TeXnicolor*" (*TeXbook*, p. 229). But he also says the \special command leads to incompatibilities. This author agrees completely, and thinks that is the best argument for extending *TeX*. \special commands—or at least the explicit use of \special—can be avoided only if there emerges a new set of standards for color printing.
8. Refer to Mittelbach's article, "ETEX: Guidelines for Future *TeX* Extensions," *TUGboat* 11, no. 3, p. 344.

A style option for rotated objects in L^AT_EX

Sebastian Rahtz and Leonor Barroca

Contents

1 History	156
2 Usage	156
3 Driver-specific macros	156
4 Rotation environments	159
4.1 Sideways	159
4.2 Rotate	159
4.3 Turn	160
5 Rotated tables and figures	163
5.1 Rotated captions only	165
6 Trigonometry macros	166
7 Examples	169

List of Figures

1 Working out the position of a box by considering x, y coordinates of five vertices	161
2 Rotation of paragraphs between 0 and -320°	171
3 Rotation of paragraphs between 0 and 320°	172
4 Turned, normal, and sideways, pictures within a figure	176
5 Figures rotated with ‘psfig’	176
6 A pathetically squashed rotated pussycat	180

List of Tables

1 This is a narrow table, which should be centred vertically on the final page.	177
2 Grooved Ware and Beaker Features, their Finds and Radiocarbon Dates .	178
3 Minimum number of individuals; effect of rotating table and caption separately	179

Abstract

This article documents a L^AT_EX style option, ‘rotating.sty’, which perform all the different sorts of rotation one might like, including complete figures, within the context of a PostScript driver.

1 History

Sebastian Rahtz first wrote rotation macros in 1988, and has been fighting with them since. The orig-

inal trigonometry macros came from Jim Walker (Dept Mathematics, University of South Carolina); we later borrowed the trigonometry macros in psfig 1.8. This set of macros is a complete overhaul of the practice of rotated L^AT_EX boxes destined for a PostScript driver.

We finally decided to clean these macros up and document them to bare-bones ‘doc’ standard in order to avoid doing some real work in January 1992. We must thank Frank Mittelbach and Rainer Schöpf for promoting this style of literate macro writing, and inspiring the rest of us to patch up our sorry efforts. We apologize for the fact that we have not attempted to make these macros compatible with ‘plain’. Life is just too short.

A modification was supplied 9/2/92 by A. Mason to handle the *Textures* driver, chosen with the \rotdriver{TEXTURES} option. The ‘sidewaysfigure’ environment was fixed on 17/3/92 after suggestions by Rainer Schöpf.

2 Usage

This style option provides three L^AT_EX environments:

sideways prints the contents turned through 90 degrees counterclockwise;

turn prints the contents turned through an arbitrary angle;

rotate prints the contents turned through an arbitrary angle; but does *not* leave any space for the result.

A full set of examples are given in section 7. But now we present the documented code.

3 Driver-specific macros

We try to make this style driver-independent (!) by isolating all the usage of \special into one case statement later, so first we declare dummy values for the two macros which vary according to the driver, in case \rotdriver is never called, or produces no results.

```
\def\rot@start{}  
\def\rot@end{}}
```

This style option (potentially!) supports a variety of dvi drivers; the user must declare the one to be used.

\rotdriver The user can select the specials that should be used by calling the command \rotdriver{*drivername*}. Possible choices are:

- DVIToLN03
- DVIToPS
- DVIPs
- emTeX
- *Textures*

This command can only be used in the preamble of the document. The list of drivers was created for compatibility with the ‘changebar’ macros (version 3.0 of November 1991 by Johannes Braams), but the code only exists in this style option for ‘dvips’ and ‘dvitops’.

The argument should be case-insensitive, so it is turned into a string containing all uppercase characters. To keep some definitions local, everything is done within a group.

```
\def\rotdriver#1{%
  \bgroup\edef\next{\def\noexpand\tempa{#1}}%
  \uppercase\expandafter{\next}%
  \def\LN{DVITOLN03}%
  \def\DVIToPS{DVITOPS}%
  \def\DVIPs{DVIPs}%
  \def\emTeX{EMTEX}%
  \def\Textures{TEXTURES}%
}
```

The choice has to be communicated to the macros later on that will be called from within \document. For this purpose the control sequence \rot@driversetup is used. It receives a numeric value using \chardef.

```
\global\chardef\rot@driversetup=0
\ifx\tempa\LN
  \global\chardef\rot@driversetup=0\fi
\ifx\tempa\DVIToPS
  \global\chardef\rot@driversetup=1\fi
\ifx\tempa\DVIPs
  \global\chardef\rot@driversetup=2\fi
\ifx\tempa\emTeX
  \global\chardef\rot@driversetup=3\fi
\ifx\tempa\Textures
  \global\chardef\rot@driversetup=4\fi
\egroup
```

We use a case statement to define the macros appropriate for each driver. We will define two commands, \rot@start and \rot@end, which assume that the macro \rot@angle produces the angle of rotation.

```
\ifcase\rot@driversetup
```

The first case (0) is for ‘dvitoln03’, for compatibility with ‘changebar.sty’; we don’t have access to this, so pass by on the other side.

```
% case 0
\typeout{WARNING! ****
no specials for LN03  rotation}
\or
```

First real case, James Clark’s ‘dvitops’. This has not been well tested with dvitops; the figures of rotated paragraphs come out oddly. Dvitops has some unusual ways of

dealing with PostScript \special commands; they are kept in a list and dealt with all together. Each time you use an effect, you number it as a block.

```
\rot@count=1
\def\rot@start{\special{dvitops: origin
    rot\the\rot@count}%
\special{dvitops: begin rot\the\rot@count}}%
\def\rot@end{\special{dvitops: end}%
\special{dvitops: rotate rot\the\rot@count \space
    \the\rot@angle}%
\global\advance\rot@count by1}%
\or
```

Case 2, Rokicki's dvips (this code works with version 5.47). We simply emit some literal PostScript (code copied from Rokicki's 'rotate.sty').

```
\def\rot@start{\special{ps:gsave currentpoint
currentpoint translate \the\rot@angle\space
rotate neg exch neg exch translate}%
\def\rot@end{\special{ps:currentpoint
grestore moveto}%
\or % case 3, emTeX
\typeout{WARNING! ***
emTeX does no rotation at this time}
```

To be consistent, lets allow for emTeX one day performing here as well.

```
\or % case 3, emTeX
\typeout{WARNING! ***
emTeX does no rotation at this time}
```

Lastly sofar, one for a Mac TeX. The *Textures* PostScript code has been modified from code provided by:

Charles Karney	Phone: +1 609 243 2607
Plasma Physics Laboratory	Fax: +1 609 243 2662
Princeton University	MFEnet: Karney@PPC.MFEnet
PO Box 451	Internet: Karney@Princeton.edu
Princeton, NJ 08543-0451	Bitnet: KarneyPPC.MFEnet@ANLVMS.Bitnet

The following assumptions are made about the PostScript that *Textures* generates:

1. A single transform is used for all *Textures* output.
2. The PostScript \special is bracketed by gsave ... grestore.
3. Immediately after the gsave, the coordinate system is translated so the origin is at the current point; and the y axis is flipped. (The y-axis isn't flipped any more... rotations are clockwise. A.M.)
4. *Textures* doesn't leave anything on the stack for long periods. (This simplifies restoring the default coordinate system.)

```
\or
\typeout{Textures rotation}
\def\rot@start{\special{postscript
    0 0 transform          % Get current location in device
                           % coordinates.
    grestore               % Undoes Textures gsave.
    matrix currentmatrix   % Save current transform on stack for use
                           % by \unrotate.
    3 1 roll                % Put transform at back of current location.
    itransform              % Current location in Textures coords
    dup 3 -1 roll           % Duplicate the location; x y ==> x y x y
    dup 4 1 roll exch
    translate               % Translate origin to current location
    % 1 -1 scale             % Flip y coordinate
    \the\rot@angle\space rotate % Rotate by \rotation}
```

```
% 1 -1 scale           % Unflip y coordinate
neg exch neg exch translate % Translate origin back
gsave}}
\def\rot@end{\special{postscript}
  grestore               % Undoes Textures gsave
  setmatrix               % Set current transform to value saved on
                         % stack. (Hopefully, it's still there.)
  gsave}}                 % To match grestore
\else
\typeout{WARNING! ***
unknown driver - no rotation}
\fi
}
```

Finally, we will need boxes to take copies of what we are rotating, and will need some registers to store sizes and angles.

```
\newsavebox{\rot@box}%
\newsavebox{\rot@tempbox}%
\newdimen\rot@temp
\newdimen\rot@width
\newdimen\rot@height
\newdimen\rot@depth
\newdimen\rot@right
\newdimen\rot@left
\newcount\rot@angle
\newcount\rot@count
\newcount\rot@circle
```

4 Rotation environments

The basic idea is to put the contents of the environment into a box, change the depth, width and height of that box (as known to T_EX) if necessary, and then rotate it.

4.1 Sideways

The ‘sideways’ environment simply turns the box through 90°, so no trigonometry is necessary.

```
\sideways \def\sideways{\savebox{\rot@box}\bgroup%
\endsideways \def\endsideways{\egroup%
\rot@angle-90
\rot@width\ht\rot@box
\advance\rot@width by\dp\rot@box
\rot@height\wd\rot@box
\wd\rot@box0pt%
\dp\rot@box0pt%
\ht\rot@box0pt%
\rule{\rot@width}{0pt}%
\rlap{\rule{0pt}{\rot@height}}%
\rot@start\usebox{\rot@box}\rot@end%
}
```

4.2 Rotate

In the case of the rotate environment, we are just going to turn the box without working out the space for it, so again no trigonometry.

```
\rotate \def\rotate#1{%
\global\rot@angle=#1
\savebox{\rot@box}\bgroup}
```

```
\endrotate \def\endrotate{\egroup%
\rot@start%
\dp\rot@box=0pt\wd\rot@box=0pt\ht\rot@box=0pt%
\usebox{\rot@box}%
\rot@end%
}\%
```

4.3 Turn

This is the tricky one. We rotate the box, and work out how much space to leave for it on the page. We deal with the box as a whole, i.e. both depth and height are joined to make a single height. After working out the space taken up this box after rotation, we can worry about placing it correctly in relation to the baseline.

The original philosophy was that given a box with width W and height H , then its height after rotation by R is $W \times \sin(R) + H \times \cos(R)$, and it extends $W \times \cos(R)$ to the right and $H \times \sin(R)$ to the left of the original bottom left corner (formula courtesy of Nico Poppelier). This works fine in the ‘top right’ quadrant, but causes problems in the other quadrants, so we adopted a rather more brute-force scheme. We consider three vertices of the original unrotated box (A , B and C in Figure 1), and calculate their x, y co-ordinates after rotation by R degrees. This deals with the *top half* of the box only, that which comes above the baseline; for the lower half (below the baseline), we deal with vertices D and E . Given original dimensions of the box as width W height H , and depth D , the formulae for calculating new positions are:

$$\begin{aligned} Ax &= W \times \cos(R) \\ Ay &= W \times \sin(R) \\ Bx &= (W \times \cos(R)) - (H \times \sin(R)) \\ By &= (W \times \sin(R)) + (H \times \cos(R)) \\ Cx &= H \times \cos(R + 90) \\ Cy &= H \times \sin(R + 90) \\ Dx &= D \times \cos(R + 270) \\ Dy &= D \times \sin(R + 270) \\ Ex &= (D \times \cos(R + 270)) - (W \times \sin(R + 270)) \\ Ey &= (D \times \sin(R + 270)) + (W \times \cos(R + 270)) \end{aligned}$$

We could work out how far the rotated box extends to the right of the ‘starting point’ (S in Figure 1) by taking the largest of (Bx, Cx, Dx, Ex) ; how far it extends to the left by taking the smallest of (Bx, Cx, Dx, Ex) ; how far above the baseline with the largest of (By, Cy, Dy, Ey) ; and how far below the baseline with the smallest of (By, Cy, Dy, Ey) . But that would be a bit slow, so we simplify matters by working out first which quadrant we are in, and then picking just the right values.

```
\turn \def\turn#1{%
\global\rot@angle=#1
\savebox{\rot@box}\bgroup%
}

\endturn \def\endturn{%
\egroup%
```

Because PostScript works clockwise, and because we conceptualize the trigonometry in a counter-clockwise way, we temporarily reverse the direction of the angle:

```
\multiply\rot@angle by-1
```

We are going to need to know the sines and cosines of three angles: R , $R + 90$ and $R + 270$. Simplest to calculate all these now; in fact we can work it out from just two calculations.

```
\Sine{\rot@angle}%
\let\sineA\sine
```

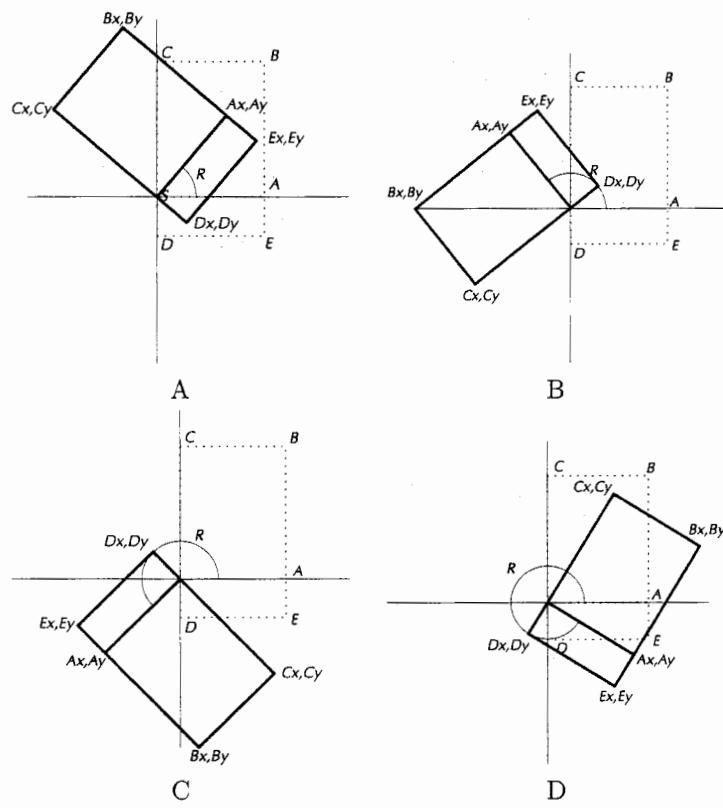


Figure 1: Working out the position of a box by considering x, y coordinates of five vertices

```
\advance\rot@angle by90%
\Sine{\rot@angle}%
\let\sineB\sine
\def\sineC{-\sine}%
\def\cosineA{-\sineC}%
\def\cosineB{-\sineA}%
\let\cosineC\sinA
\advance\rot@angle by-90%
```

Now we can calculate the co-ordinates of the relevant vertices. To make the coding easier, we define the formulae given above as TeX macros (just the ones we ever use).

```
\def\rot@Bx{\rot@temp\cosineA\wd\rot@box
\advance\rot@temp by -\sineA\ht\rot@box}%
\def\rot@By{\rot@temp\sineA\wd\rot@box
\advance\rot@temp by \cosineA\ht\rot@box}%
\def\rot@Cx{\rot@temp\cosineB\ht\rot@box}%
\def\rot@Cy{\rot@temp\sineB\ht\rot@box}%
\def\rot@Dx{\rot@temp\cosineC\dp\rot@box}%
\def\rot@Dy{\rot@temp\sineC\dp\rot@box}%
\def\rot@Ex{\rot@temp\cosineC\dp\rot@box
\advance\rot@temp by -\sineC\wd\rot@box}%
\def\rot@Ey{\rot@temp\sineC\dp\rot@box
\advance\rot@temp by \cosineC\wd\rot@box}%
```

Now a straightforward ‘if’ condition to see which quadrant we are operating in; but if the angle is negative, first add 360.

```
\rot@circle\rot@angle
\ifnum\rot@circle<0
\advance\rot@circle by 360
\fi
\ifnum\rot@circle<90
```

First quadrant: Height = By , Right = Ex , Left = Cx , Depth = Dy

```
\rot@By\rot@height\rot@temp
\rot@Ex\rot@right\rot@temp
\rot@Cx\rot@left\rot@temp
\rot@Dy\rot@depth\rot@temp
\else
\ifnum\rot@circle<180
```

Second quadrant: Height = Ey , Right = Dx , Left = Bx , Depth = Cy

```
\rot@Ey\rot@height\rot@temp
\rot@Dx\rot@right\rot@temp
\rot@Bx\rot@left\rot@temp
\rot@Cy\rot@depth\rot@temp
\else
\ifnum\rot@circle<270
```

Third quadrant: Height = Dy , Right = Cx , Left = Ex , Depth = By

```
\rot@Dy\rot@height\rot@temp
\rot@Cx\rot@right\rot@temp
\rot@Ex\rot@left\rot@temp
\rot@By\rot@depth\rot@temp
\else
```

Fourth quadrant: Height = Cy , Right = Bx , Left = Dx , Depth = Ey

```
\rot@Cy\rot@height\rot@temp
\rot@Bx\rot@right\rot@temp
\rot@Dx\rot@left\rot@temp
```

```

    \rot@Ey\rot@depth\rot@temp
    \fi
    \fi
    \fi

```

Put the angle back to what it was before, to pass to PostScript

```
\multiply\rot@angle by -1
```

At the end of all that nonsense, `\rot@height` contains the amount *above* the baseline we need to leave for the rotated box we are dealing with, and `\rot@depth` the amount *below* the baseline. `\rot@left` and `\rot@right` are offsets to left and right which we need to take into account. We are going to set the size of the box we are dealing with to 0 all round, and put in some struts to force `TEX` to leave space. We will position ourselves at the point where the bottom left-hand corner of the top half of box *would* have been, then swing the box round by that corner. Thinking about this drives you mad.

```
\wd\rot@box=0pt\ht\rot@box=0pt%
```

The left adjustment comes out negative, so be careful:

```
\ifdim\rot@left<0pt
  \rule{-\rot@left}{0pt}%
\fi
```

Put in struts (not advancing forward at all), for the height and depth.

```
\ifdim\rot@height>0pt
  \rlap{\rule{0pt}{\rot@height}}%
\fi
\ifdim\rot@depth<0pt
  \rlap{\rule[\rot@depth]{0pt}{-\rot@depth}}%
\fi
```

Finally emit the PostScript code to start rotation, output the box, end the rotation, and leave some space at the right if needed.

```
\rot@start\usebox{\rot@box}\rot@end%
\ifdim\rot@right>0pt
  \rule{\rot@right}{0pt}%
\fi
}%
```

5 Rotated tables and figures

Now we present some macros adapted from those by James Dolter (`j.dolter@sawtooth.eecs.umich.edu`) which provide a complete environment for sideways figures and tables. We define two environments `sidewaysfigure` and `sidewaystable` that fit in with the normal table and figure floats. These are ‘fixed’ environments that just do 90 degree rotation, but it would be easy to parameterize this to do other rotations if needed (the mind boggles...).

First a generalised ‘`rotfloat`’ environment. We have to take a copy of `LATEX`’s float macros, in order to change the assumed width of a float being `\columnwidth`. We want it to work on a width of `\textheight` so that when we rotate the float, it comes out the right height. This is not actually very satisfactory, since what we *really* want is for rotated floats to occupy the space they actually *use*. The captions are a problem — since they can precede the figure or table, we cannot set them in a box of the right width (i.e. the *height* of the forthcoming object), because it has not happened yet. The result of these difficulties is that rotated figures always end up as full page figures.

```
\def\@rotfloat#1{\@ifnextchar[{\@xrotfloat{#1}}{%
\edef\@tempa{\noexpand\@xrotfloat{#1}[\csname fsp@\#1\endcsname]\@tempa}}
```

```
\def\@xrotfloat#1[#2]{\ifhmode \@bsphack\@floatpenalty -\@Mii\else
  \@floatpenalty-\@Mii\fi\def\@capttype{#1}\ifinner
    \@parmoderr\@floatpenalty\z@\else\@next\@currbox\@freelist{\@tempcnta\csname ftype\#1\endcsname
      \multiply\@tempcnta\@xxxii\advance\@tempcnta\sixt@n
      \@tfor \@tempa :=#2\do
        {\if\@tempa h\advance\@tempcnta \@ne\fi
         \if\@tempa t\advance\@tempcnta \tw@\fi
         \if\@tempa b\advance\@tempcnta 4\relax\fi
         \if\@tempa p\advance\@tempcnta 8\relax\fi
       }\global\count\@currbox\@tempcnta\@fltovf\fi
     \global\setbox\@currbox\vbox\bgroup
```

The only part changed is the setting of `\hsize` within the `\vbox` to be `\textheight` instead of `\columnwidth`.

```
  \hsize\textheight \parboxrestore
}
```

We copy the '`\end@float`' macro and emend it to rotate the box we produce in a float.

```
\def\end@rotfloat{\par\vskip\z@\egroup%
\ifnum\@floatpenalty <\z@
  \global\setbox\rot@tempbox\box\@currbox
  \global\setbox\@currbox\vbox{\centerline{\begin{turn}{-90}}%
  \box\rot@tempbox\end{turn}}\%
  \@cons\@currlist\@currbox
  \typeout{Adding sideways figure to list,
  \the\ht\@currbox\space by \the\wd\@currbox}%
\ifdim \ht\@currbox >\textheight
  \warning{Float larger than \string\textheight}%
  \ht\@currbox\textheight\fi
\ifnum\@floatpenalty <-\@Mii
  \penalty -\@Miv
  \tempdima\prevdepth
  \vbox{}\%
  \prevdepth \tempdima
  \penalty\@floatpenalty
\else \vadjust{\penalty -\@Miv
  \vbox{}\penalty\@floatpenalty}\@esphack
\fi\fi}
```

The following definitions set up two environments, `sideways table` and `sidewaysfigure`, which uses this type of float. Naturally, users may need to change these to suit their local style. Both contribute to the normal lists of figures and tables.

```
\let\l@sidewaysfigure\l@figure
\let\c@sidewaysfigure\c@figure
\let\p@sidewaysfigure\p@figure
\let\thesidewaysfigure\thefigure
\def\fps@sidewaysfigure{tbp}
\def\fitype@sidewaysfigure{1}
\def\ext@sidewaysfigure{lof}
\def\fnum@sidewaysfigure{Figure \thefigure}
\def\sidewaysfigure{\let\make@caption\make@rcaption
\@rotfloat{sidewaysfigure}}
%
\let\endsidewaysfigure\end@rotfloat
%
\let\l@sidewaystable\l@table
\let\c@sidewaystable\c@table
```

```
\let\p@sidewaystable\p@table
\let\thesidewaystable\thetable
\def\fps@sidewaystable{tbp}
\def\ftype@sidewaystable{2}
\def\ext@sidewaystable{lot}
\def\fnum@sidewaystable{Table \thetable}
\def\sidewaystable{\let\make@caption\make@rcaption
\@rotfloat{sidewaystable}}
\let\endsidewaystable\end@rotfloat
```

We need to copy a standard `\@makecaption` to set the caption on a width of the *height* of the float (i.e. the text height). This macro is normally defined in L^AT_EX style files, so style file writers who change that will also need to redefine `\r@caption`.

```
\long\def\@makercaption#1#2{%
  \vskip 10\p@
  \setbox\@tempboxa\hbox{#1: #2}%
  \ifdim \wd\@tempboxa >\vsize
    #1: #2\par
  \else
    \hbox to\vsize{\hfil\box\@tempboxa\hfil}%
  \fi}%
}
```

5.1 Rotated captions only

Sometimes you may find that the rotation of complete figures does not give quite the right result, since they always take up the whole page. You may prefer to rotate the caption and the float contents separately within a conventional figure. Here we offer a suggestion for a `\rotcaption` command, which inserts the caption rotated by 90°. It is essentially a copy of the normal captioning code. Styles which define the `\@makecaption` command may also need to define `\@makerotcaption`.

```
\def\rotcaption{\refstepcounter{\capttype}\@dblarg{\@rotcaption{\capttype}}}
\long\def\@rotcaption#1[#2]#3{%
\addcontentsline{\csname ext@#1\endcsname}{#1}{%
\protect\numberline{\csname the#1\endcsname}{\ignorespaces #2}}%
\par
\begingroup
\@parboxrestore
\normalsize
\@makerotcaption{\csname fnum@#1\endcsname}{#3}%
\endgroup}
\long\def\@makerotcaption#1#2{%
\setbox\@tempboxa\hbox{#1: #2}%
\ifdim \wd\@tempboxa > .8\vsize
\begin{rotate}{-90}%
\begin{minipage}[b]{.8\textheight}#1 #2\end{minipage}%
\end{rotate}\par
\else%
\mbox{\begin{rotate}{-90}\box\@tempboxa\end{rotate}}%
\fi
\hspace{12pt}%
}
```

While we are doing useful new environments, why not add landscape slides?

```
\newenvironment{landscapeslide}[1]{\begin{sideways}%
\ vbox\begin{group}\begin{slide}{#1}\end{slide}%
\end{group}\end{sideways}}
```

6 Trigonometry macros

Now the trigonometry macros which are borrowed from psfig1.8; the original author is not credited there, so we cannot do so either. All we have done is remove some spurious spaces which were creeping into my output (and causing havoc!), and put the comments in ‘doc’ style.

```
\chardef\letter = 11
\chardef\other = 12
```

Turn me on to see TeX hard at work ...

```
\newif\ifdebug%
```

don’t need to compute some values

```
\newif\ifc@mpute%
```

but assume that we do

```
\c@mputetrue%
\let\then =\relax
\def\r@dian{pt }%
\let\r@dians =\r@dian
\let\dimensionless@nit =\r@dian
\let\dimensionless@nits =\dimensionless@nit
\def\internal@nit{sp }%
\let\internal@nits =\internal@nit
\newif\ifstillc@nverging
\def\Mess@ge #1{\ifdebug\then\message {#1}\fi}%
```

Things that need abnormal catcodes

```
{%
\catcode `@ =\letter
\gdef\nodimen {\expandafter\n@dimen\the\dimen}%
\gdef\term #1 #2 #3%
```

freeze parameter 1 (count, by value)

```
{\edef\t@ {\the #1}%

```

freeze parameter 2 (dimen, by value)

```
\edef\t@0 {\expandafter\n@dimen\the #2\r@dian}%
\t@rm {\t@} {\t@0} {#3}%
}%
\gdef\t@rm #1 #2 #3%
{%
\count 0 = 0
\dimen 0 = 1\dimensionless@nit
\dimen 2 = #2\relax
\Mess@ge {Calculating term #1 of\nodimen 2}%
\loop
\ifnum\count 0 < #1
\then\advance\count 0 by 1
\Mess@ge {Iteration\the\count 0\space}%
\Multiply\dimen 0 by {\dimen 2}%
\Mess@ge {After multiplication, term =\nodimen 0}%
\Divide\dimen 0 by {\count 0}%
\Mess@ge {After division, term =\nodimen 0}%
\repeat
\Mess@ge {Final value for term #1 of
\nodimen 2\space is\nodimen 0}%
\xdef\Term {#3 =\nodimen 0\r@dians}%
\aftergroup\Term
```

```

} } %
\catcode `p =\other
\catcode `t =\other
throw away the "pt"
\gdef\n@dimen #1pt{#1}%
} %

just a synonym
\def\Divide #1by #2{\divide #1 by #2}%
allows division of a dimen by a dimen
\def\Multiply #1by #2%
should really freeze parameter 2 (dimen, passed by value)
{{%
\count 0 = #1\relax
\count 2 = #2\relax
\count 4 = 65536
\message {Before scaling, count 0 =\the\count 0\space and
count 2 =\the\count 2}%
do our best to avoid overflow
\ifnum\count 0 > 32767%
\then\divide\count 0 by 4
\divide\count 4 by 4
\else\ifnum\count 0 < -32767%
\then\divide\count 0 by 4
\divide\count 4 by 4
\else
\fi
\fi
while retaining reasonable accuracy
\ifnum\count 2 > 32767%
\then\divide\count 2 by 4
\divide\count 4 by 4
\else\ifnum\count 2 < -32767%
\then\divide\count 2 by 4
\divide\count 4 by 4
\else
\fi
\fi
\multiply\count 0 by\count 2
\divide\count 0 by\count 4
\xdef\product {\#1 =\the\count 0\internal@nits}%
\aftergroup\product
} } %

 $\sin(x + 90) = \sin(180 - x)$ 
\def\r@duce{\ifdim\dimen0 > 90\r@dian\then%
\multiply\dimen0 by -1
\advance\dimen0 by 180\r@dian
\r@duce
 $\sin(-x) = \sin(360 + x)$ 
\else\ifdim\dimen0 < -90\r@dian\then%
\advance\dimen0 by 360\r@dian
\r@duce

```

```

\fi
\fi}%
\def\Sine#1%
{{%
    \dimen0 = #1\r@dian
    \r@duce
    \ifdim\dimen0 = -90\r@dian\then
        \dimen4 = -1\r@dian
        \c@mputefalse
    \fi
    \ifdim\dimen0 = 90\r@dian\then
        \dimen4 = 1\r@dian
        \c@mputefalse
    \fi
    \ifdim\dimen0 = 0\r@dian\then
        \dimen4 = 0\r@dian
        \c@mputefalse
    \fi
}
%
```

convert degrees to radians

```

%
\divide\dimen0 by 180
\dimen0=3.141592654\dimen0
%
```

a well-known constant

```
\dimen2 = 3.1415926535897963\r@dian%
```

we only deal with $-\pi/2 : \pi/2$

```

\divide\dimen2 by 2%
\Mess@ge {Sin: calculating Sin of\nodimen0}%

```

see power-series expansion for sine

```

\count0 = 1%
\dimen2 = 1\r@dian%
\dimen4 = 0\r@dian%
\loop
```

then we've done

```

\ifnum\dimen2 = 0%
\then\stillc@nvergingfalse
\else\stillc@nvergingtrue
\fi
\ifstillc@nverging%
```

then calculate next term

```

\then\term {\count0} {\dimen0} {\dimen2}%
\advance\count0 by 2
\count2 =\count0
\divide\count2 by 2
```

signs alternate

```

\ifodd\count2%
\then\advance\dimen4 by\dimen2
\else\advance\dimen4 by -\dimen2
\fi
\repeat
```

```
\fi
\xdef\sine {\nodimen 4}%
}%
}
```

Now the Cosine can be calculated easily by calling \Sine

```
% 
\def\Cosine#1{\ifx\sine\UnDefined\edef\Savesine{\relax}\else
\edef\Savesine{\sine}\fi
{\dimen0=#1\r@dian\advance\dimen0 by 90\r@dian
\Sine{\nodimen 0}%
\xdef\cosine{\sine}%
\xdef\sine{\Savesine}}}
% end of trig stuff
```

And that's the end of the trigonometry macros. Finally, we'll set up a default for the driver:

```
\rotdriver{dvips}
```

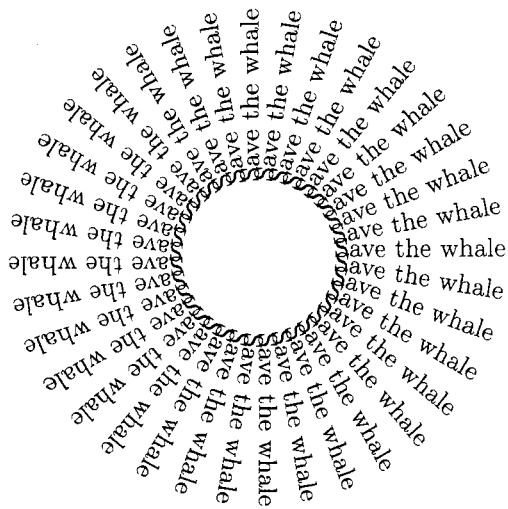
7 Examples

'Rotate' provides a generalised rotation environment, where the text will be rotated (clockwise, as is normal in PostScript) by the number of degrees specified as a parameter to the environment, but no special arrangement is made to find space for the result. Note the % at the end of \begin{rotate}{56} — this is vital to prevent a space getting into the rotated text.

Start here End here
Save whales

Start here
\begin{rotate}{56}%
Save whales
\end{rotate}
End here

A complete example of rotating text without leaving space would be the 'Save the whale' text written at 10 degree intervals round the compass. We use 'rlap' to ensure that all the texts are printed at the same point. Just to show that T_EX can handle PostScript muckings-about properly...



```
\newcount\wang
\newsavebox{\wangtext}
\newdimen\wangspace
\def\wheel#1{\savebox{\wangtext}{#1}%
\wangspace\wd\wangtext
\advance\wangspace by 1cm%
\centerline{%
\rule{0pt}{\wangspace}%
\rule[-\wangspace]{0pt}{\wangspace}%
\wang=-180\loop\ifnum\wang<180
\rlap{\begin{rotate}{\the\wang}}%
\rule{1cm}{0pt}\#1\end{rotate}}%
\advance\wang by 10\repeat}
\wheel{Save the whale}
```

If the user desires L^AT_EX to leave space for the rotated box, then ‘turn’ is used:

Save the whale

Start here *Save the whale*
end here

```
Start here \begin{turn}{-56}%
  Save the whale
\end{turn} end here
```

The environment ‘Sideways’ is a special case, setting the rotation to -90, and leaving the correct space for the rotated box.

Save the whale

Start here *Save the whale*

```
Start here
\begin{sideways}%
  Save the whale
\end{sideways}
End here
```

Start here *Save the whale* End here

If you deal with whole paragraphs of text, you realize that T_EX boxes are not as simple as they sometimes look: they have a height *and* a depth. So when you rotate, you rotate about the point on the left-hand edge of the box that meets the baseline. The results can be unexpected, as shown in the full set of paragraph rotations in Figures 2 and 3. If you really want to turn a paragraph so that it appears to rotate about the *real* bottom of the T_EX box, you have to adjust the box in the normal L^AT_EX way:

Save the whales
Save the whale
Save the whale
Save the whale
Save the whale
Save the whale

Start End

```
\newsavebox{\foo}
\savebox{\foo}{\parbox{1in}{Save
the whales Save the whale
Save the whale
Save the whale}}%
Start
\begin{turn}{-45}\usebox{\foo}\end{turn}
End
```

Save the whales
Save the whale
Save the whale
Save the whale
Save the whale
Save the whale

Start End

```
\savebox{\foo}{\parbox[b]{1in}{Save
the whales Save the whale
Save the whale
Save the whale}}%
Start
\begin{turn}{-45}\usebox{\foo}\end{turn}
End
```

We can set tabular material in this way; at the same time, we demonstrate that the rotation can be nested:

Word	Occurrences
hello	33
goodbye	34

```
\begin{sideways}
\rule{1in}{0pt}
\begin{tabular}{|l r|}
\em Word & \begin{rotate}{-90}%
Occurrences\end{rotate}\\
\hline
hello & 33\\
goodbye & 34\\
\hline
\end{tabular}
\end{sideways}
```

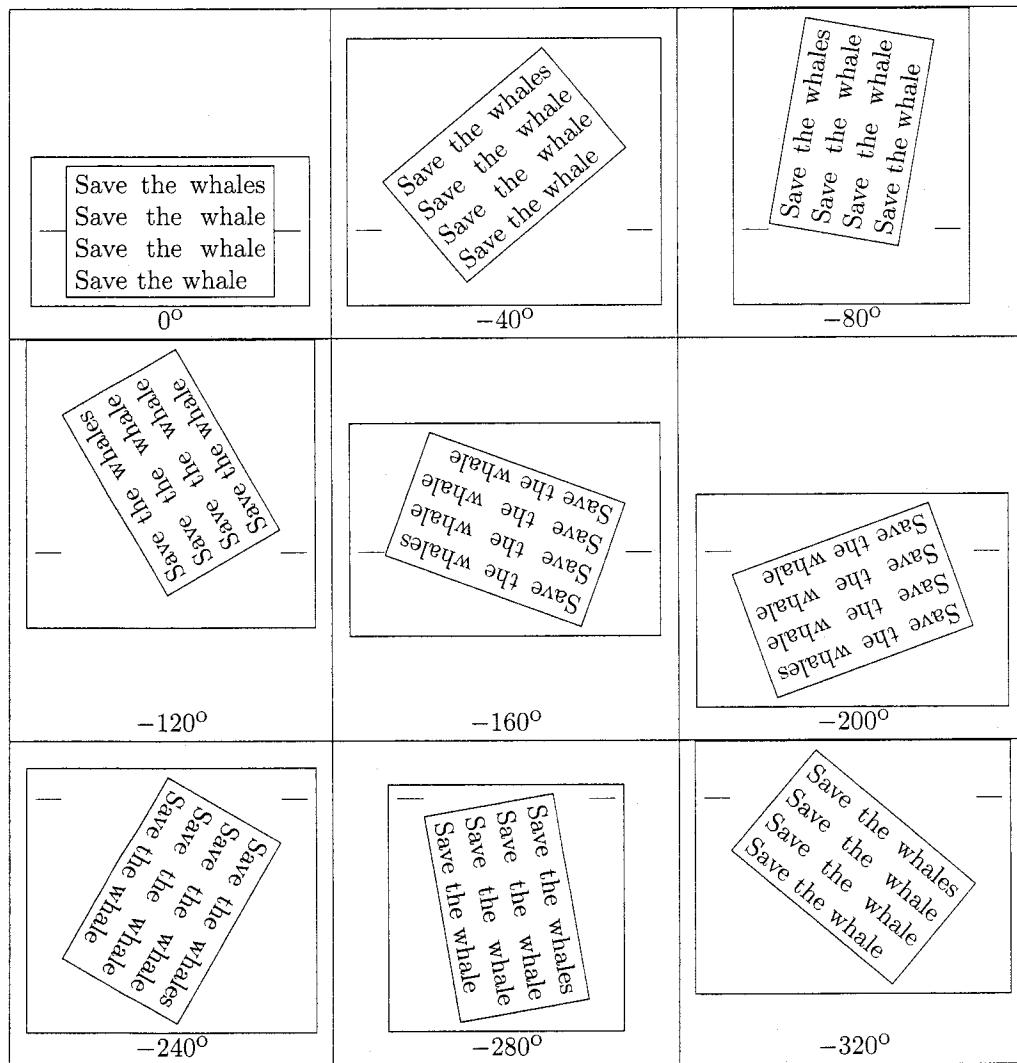


Figure 2: Rotation of paragraphs between 0 and -320°

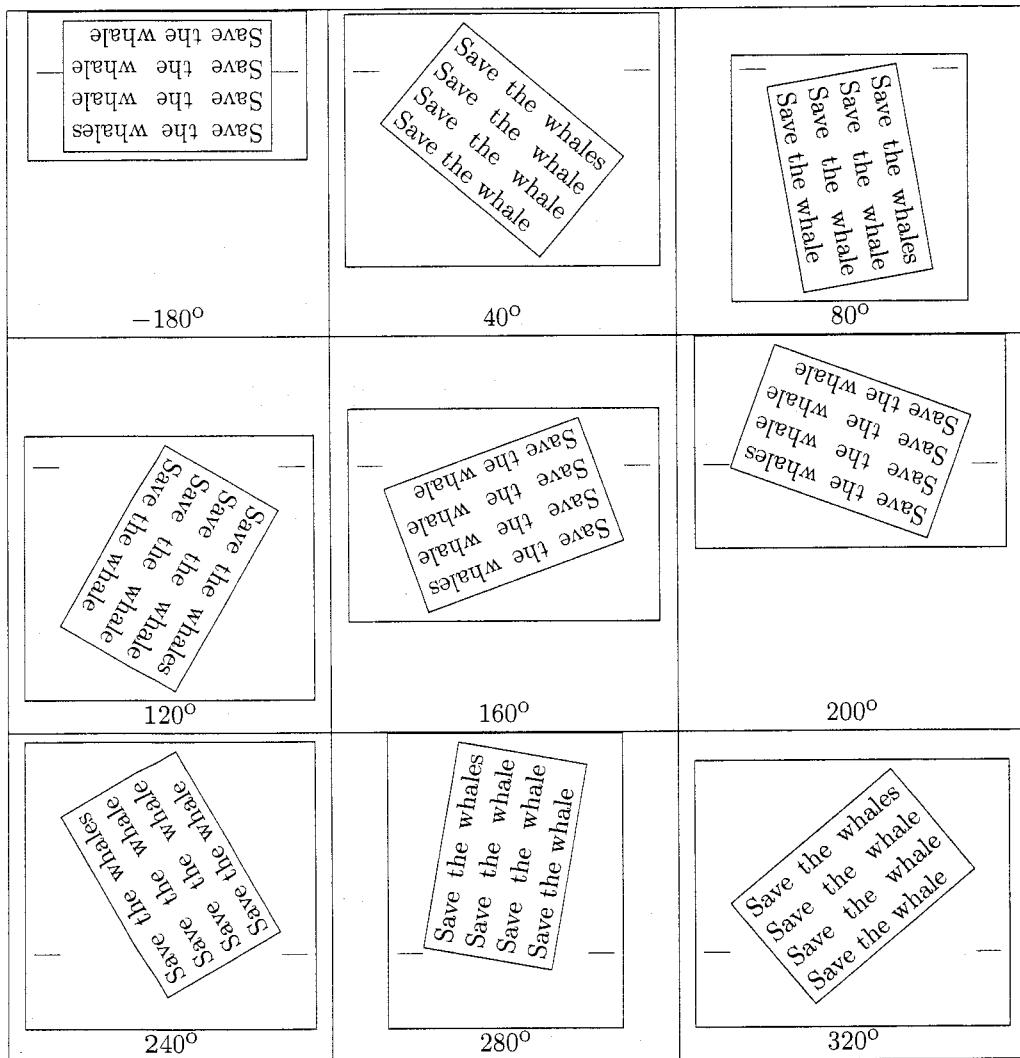


Figure 3: Rotation of paragraphs between 0 and 320°

Column 1
Column 2
Column 3

1	2	3
4	5	6
7	8	9

Column 1	Column 2	Column 3
1	2	3
4	5	6
7	8	9

Column 1
Column 2
Column 3

1	2	3
4	5	6
7	8	9

```
\begin{quote}
\rule{0pt}{1.5in}\begin{tabular}{rrr}
\begin{rotate}{-45}Column 1\end{rotate}&
\begin{rotate}{-45}Column 2\end{rotate}&
\begin{rotate}{-45}Column 3\end{rotate}\\
\hline
1& 2& 3\\
4& 5& 6\\
7& 8& 9\\
\hline
\end{tabular}
\end{quote}

\begin{quote}
\begin{tabular}{rrr}
\begin{turn}{-45}Column 1\end{turn}&
\begin{turn}{-45}Column 2\end{turn}&
\begin{turn}{-45}Column 3\end{turn}\\
\hline
1& 2& 3\\
4& 5& 6\\
7& 8& 9\\
\hline
\end{tabular}
\end{quote}

\begin{quote}
\rule{0pt}{1.5in}\begin{tabular}{rrr}
\begin{rotate}{-45}Column 1\end{rotate}&
\begin{rotate}{-45}Column 2\end{rotate}&
\begin{rotate}{-45}Column 3\end{rotate}\\
\hline
1& 2& 3\\
4& 5& 6\\
7& 8& 9\\
\hline
\end{tabular}
\end{quote}
```

STUDY AREA	NUMBER OF SITES				HYPOTH	
	IN BOUNDARY ZONE		EXPECTED	TO		
	TOT	OBS				
FULL SAMPLE	41	31	10.3	27.0	REJECT	
SAMPLE AREA 1	23	16	4.3	16.7	ACCEPT	
SAMPLE AREA 2	18	15	2.8	13.7	REJECT	
RUSHEN	13	9	1.2	10.4	ACCEPT	
ARBORY	10	7	0.6	8.8	ACCEPT	
MAROWN	10	8	0.4	8.6	ACCEPT	
SANTON	8	7	0.0	7.3	ACCEPT	

```

\begin{sideways}
\begin{tabular}{|l|c|c|c|c|c|p{1in}|}
\hline
&&\multicolumn{4}{c}{NUMBER OF SITES}\vline &ACCEPT or \\
\cline{3-6} &STUDY AREA&&\multicolumn{3}{c}{\% \\
IN BOUNDARY ZONE}\vline&REJECT\\
\cline{4-6}&&&\multicolumn{2}{c}{EXPECTED} \\
\vline&NULL\\
\cline{5-6}&&TOT&OBS&FROM&TO&HYPOTH\\
\cline{2-7}
&FULL SAMPLE&41&31&10.3&27.0&REJECT\\
&SAMPLE AREA 1&23&16&4.3&16.7&ACCEPT\\
&SAMPLE AREA 2&18&15&2.8&13.7&REJECT\\
&RUSHEN&13&9&1.2&10.4&ACCEPT\\
&ARBORY&10&7&0.6&8.8&ACCEPT\\
&MAROWN&10&8&0.4&8.6&ACCEPT\\
\rule{0.5cm}{0pt}\\
\begin{rotate}{-90}PRIMARY UNITS%\end{rotate}\\
\end{rotate}\rule{0.5cm}{0pt}\\
&SANTON&8&7&0.0&7.3&ACCEPT\\
\hline
\end{tabular}
\end{sideways}

```

If you are interested in setting rotated material in tables or figures, this presents no problem. Figure 4 shows how PostScript files which are being incorporated using `psfig` can be rotated at will, while Figure 5 shows, in contrast, how `psfig` itself handles rotation. It is also possible to rotate the whole of the figure environment, including caption, by using the `sidewaysfigure` and `sidewaystable` environments in place of `figure` and `table`. The code used to produce figures 1–6 is as follows:

Figure 1 \begin{sidewaystable}

```

\centering
\caption{This is a narrow table, which should be centred vertically
on the final page.\label{rotfloat1}}
\begin{tabular}{|l l|}
\hline
a & b \\
c & d \\
e & f \\
g & h \\
i & j \\
\hline

```

Figure 2 \begin{sidewaystable}

```
\begin{sidewaysbasic}
\centering
\begin{tabular}{|l l l l l l l l p{1in}lp{1in}|}
```

```

\hline
Context &Length &Breadth/ &Depth &Profile &Pottery &Flint
&Animal &Stone &Other &C14 Dates \\
& &Diameter & & & &
&Bones&&\ \
\hline
&&&&&&&\ \
\multicolumn{10}{|l}{\bf Grooved Ware}\ \
784 &--- &0.9m &0.18m &Sloping U &P1 &$$\times$46
& $ \times $8 && $ \times $2 bone& 2150$ \pm $ 100 BC \\
785 &--- &1.00m &0.12 &Sloping U &P2--4 &$$\times$23
& $ \times $21 & Hammerstone &---&--- \\
962 &--- &1.37m &0.20m &Sloping U &P5--6 &$$\times$48
& $ \times $57* & ---& ---&1990 $ \pm $ 80 BC (Layer 4) 1870 $ \pm $90 BC (Layer 1) \\
983 &0.83m &0.73m &0.25m &Stepped U &--- &$$\times$18
& $ \times $8 & ---& Fired clay&--- \\
&&&&&&&\ \
\multicolumn{10}{|l}{\bf Beaker}\ \
552 &--- &0.68m &0.12m &Saucer &P7--14 &---
& --- & --- &--- &--- \\
790 &--- &0.60m &0.25m &U &P15 &$$\times$12
& --- & Quartzite-lump&--- &--- \\
794 &2.89m &0.75m &0.25m &Irreg. &P16 &$$\times$3
& --- & --- &--- &--- \\
\hline
\end{tabular}

\caption[Grooved Ware and Beaker Features, their Finds and Radiocarbon Dates]{Grooved Ware and Beaker Features, their Finds and Radiocarbon Dates; For a breakdown of the Pottery Assemblages see Tables I and III; for the Flints see Tables II and IV; for the Animal Bones see Table V.}\label{rotfloat2}
\end{sidewaystable}

```

Figure 3 \begin{table}

```

\centering
\rotcaption{Minimum number of individuals; effect of rotating table and caption separately}\label{rotfloat3}%
\begin{sideways}
\begin{tabular}[b]{ccccccccc}
\hline
Phase&Total&Cattle&Sheep&Pig&Red Deer&Horse&Dog&Goat&Other \\
\hline
&1121&&54&12&32&1&1&1&1 polecat\\
3&8255&&58&6&35&1&1&1&1 roe deer, 1 hare, 1 cat, 1 otter\\
4&543&45&6&45&4&1&1&---&--- \\
\hline
&9919&&157&24&112&6&3&3&2&5 \\
\hline
\end{tabular}
\end{sideways}
\end{table}

```

Figure 6 \begin{sidewaysfigure}

```

\centerline{\psfig{figure=cat.eps,width=.8\textheight,height=.4\textwidth}}
\caption{A pathetically squashed rotated pussycat}\label{rotfloat4}
\end{sidewaysfigure}

```

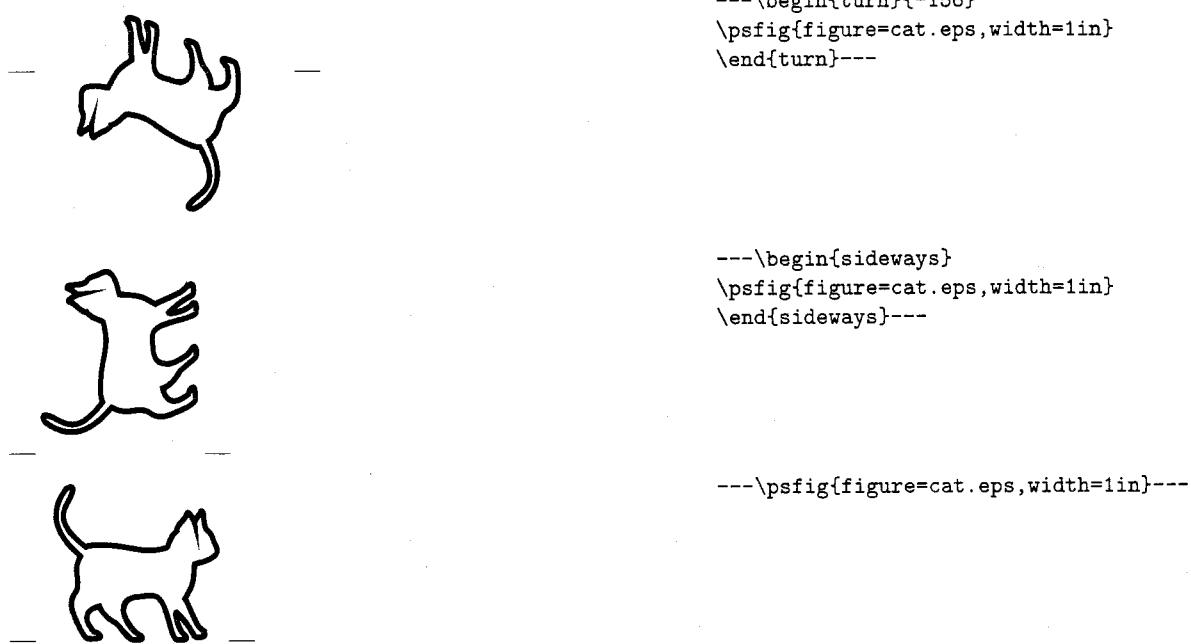


Figure 4: Turned, normal, and sideways, pictures within a figure

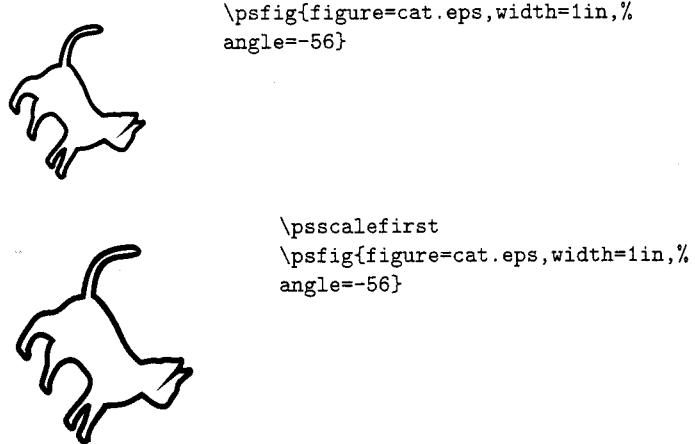


Figure 5: Figures rotated with 'psfig'

Table 1: This is a narrow table, which should be centred vertically on the final page.

a	b
c	d
e	f
g	h
i	j

Context	Length	Breadth/ Diameter	Depth	Profile	Pottery	Flint	Animal Bones	Stone	Other	C14 Dates
Grooved Ware										
784	—	0.9m	0.18m	Sloping U	P1	×46	×8	—	×2 bone	2150± 100 BC
785	—	1.00m	0.12	Sloping U	P2-4	×23	×21	Hammerstone	—	—
962	—	1.37m	0.20m	Sloping U	P5-6	×48	×57*	—	—	1990 ± 80 BC (Layer 4) 1870 ±90 BC (Layer 1)
983	0.83m	0.73m	0.25m	Stepped U	—	—	×18	×8	—	Fired clay —
Beaker										
552	—	0.68m	0.12m	Saucer	P7-14	—	—	—	—	—
790	—	0.60m	0.25m	U	P15	×12	—	—	—	—
794	2.89m	0.75m	0.25m	Irrig.	P16	×3	—	Quartzite-lump	—	—

Table 2: Grooved Ware and Beaker Features, their Finds and Radiocarbon Dates; For a breakdown of the Pottery Assemblages see Tables I and III; for the Flints see Tables II and IV; for the Animal Bones see Table V.

Table 3: Minimum number of individuals; effect of rotating table and caption separately

Phase	Total	Cattle	Sheep	Pig	Red Deer	Horse	Dog	Goat	Other
	1121	54	12	32	1	1	1	1	1
3	8255	58	6	35	1	1	1	1	1 polecat roe deer, 1 hare, 1 cat, 1 otter
4	543	45	6	45	4	1	1	—	—
	9919	157	24	112	6	3	3	2	5



Figure 6: A pathetically squashed rotated pussycat

◊ Sebastian Rahtz
ArchaeoInformatica
12 Cygnet Street
York Y02 1AG
spqr@uk.ac.york.minster

◊ Leonor Barroca
Department of Computer Science
University of York
Heslington
York Y01 5DD
lmb@uk.ac.york.minster

Resources

Book Review: An Italian guide to LATEX

Marisa Luvisetto and Massimo Calvani

Claudio Beccari. *LATEX — Guida a un sistema di editoria elettronica*. Milano: Editore Ulrico Hoepli Milano, 1991. ix + 399 pp. ISBN 88-203-1931-4

When we were asked by the *TUGboat* editor to write a report on a book in Italian on LATEX, we thought: "Gosh, it's a rat". So, it was with a lot of curiosity that we opened the parcel with the book and started going through it. Despite our prejudices,¹ we must admit that our first reaction was very positive. Then we carefully read the book and, to say it briefly, our opinion is that it is a very good book, perhaps not recommended for a novice (we would rather suggest *An introduction to LATEX* by Michael Urban), but a book where all LATEX commands are described in detail and, what is very important, with a lot of examples. The main drawback for wide diffusion of the book is that it is written in italian, but at the same time it is a pleasure to finally have a book in italiano on LATEX.

LATEX, Guida a un sistema di editoria elettronica by Claudio Beccari is a complete guide to the use of LATEX for most scientific users and a good starting point for professional typesetting.

The book has a very good logical layout, not in the style of a school book but as a manual for a demanding user. Chapter 1 contains a nontrivial description of the typesetting process and an introduction to TEX and LATEX. A confrontation between TEX and LATEX philosophy is also included.

Chapter 2 is a tutorial description of LATEX and it is sufficient for non-professional users who produce short documents and articles. What we feel is missing is a suggestion at the very beginning of the book for the novice to go directly to Chapter 2 and to read it thoroughly before trying any test.

Chapter 2 is very well organized and reflects the structure of the whole book. In fact each subsection of Chapter 2 is analyzed in deeper detail as a stand-alone chapter with nearly the same title as the subsection. Here we find that subsections of Chapter 2 and chapter titles exactly the same would more closely show the logical connection of the material.

¹ Editor's note: The authors work mainly with macros of their own devising.

The book gives valuable information on the typesetting process (e.g. a complete description of the usage of the period as a punctuation mark) and the inner working of LATEX, with exhaustive lists of parameters, usage, values, suggested values related to document type, and the like.

Chapters 3 and 4 contain a detailed description of commands for text and maths. They end with notes about composition rules and conventions, but we feel that such notes should be placed more effectively by themselves eventually in an appendix at the end of the book.

Chapter 5 contains a detailed description of all LATEX environments. Command descriptions are followed by examples to clarify the concepts especially in complex cases such as boxes. Warnings and advice on good composition and the way to avoid errors are presented at difficult points throughout the book. More real life examples are probably needed in the array and tabular environments.

Chapter 6 on figures and Chapter 7 on macro definitions are very complex and surely not for the novice. They are a good starting point for the expert, but obviously this is a very difficult part of the typesetting process and would need a book on its own.

Chapter 8 contains a detailed description of LATEX document styles. It is a valuable source of information on LATEX parameters and usage especially for book composition, for which important descriptions on styles, page settings, etc., are given.

Chapter 9 and Appendix C give complete information on fonts, size, types, file names, and the like. A table of magnification values (i.e., correspondence between mag and resolutions like 1000 == 300, 1200 == 360, etc.) is missing. This information would be very useful together with a list of which fonts are stored in the local installation, thus making it possible to choose from the provided magnifications. Also error messages and debugging tools are probably treated too briefly, but this is a problem with all TEX/LATEX books we have used.

Appendix D contains a brief description of italiano grammar rules that are used to create italiano hyphenation patterns.

The book contains a good bibliography and a fair cross-reference index. The index has the usual shortcomings we found in most books, i.e., sometimes the page written in the index does not contain the specified item, but in general the information is useful and well organized. What we find strange, however, is the author's decision not to include the full mathematical command set in the index.

A last remark regards the book as a whole. Typing errors are frequent even if neither disturbing nor misleading. This problem arises because the usual editorial step was skipped by the publisher, as the author has explained, probably due to a relatively informal policy related to electronic publishing in Italy.

To sum up, the global impression is very positive. This is a basic book for *L^AT_EX* not only because it is the only one in Italian but also for its deep insight into *L^AT_EX* and the complete explanation of many complex mechanisms in *T_EX* and *L^AT_EX* and the lot of examples; it is a book that should not be lacking in the library of any more than trivial *L^AT_EX* user and one that surely deserves an English translation. Hopefully this book will fill a gap in the literature of electronic publishing in Italy and will give rise to a series of such books in our schools and Universities.

- ◊ Marisa Luvisetto
Istituto Nazionale di Fisica
Nucleare
Viale Ercolani 8
40138, Bologna, Italy
Internet: *Luvisetto@CNAF.INFN.IT*
- ◊ Massimo Calvani
Osservatorio Astronomico
Vicolo dell'Osservatorio
35122 Padova, Italy

Book reviews

Nico Poppelier

L^AT_EX for Everyone, Jane Hahn, first edition, Personal *T_EX* Inc. 1991, softbound, 346 pages

Writing a book is hard work. It can also be rewarding work — if the readers are satisfied with the book. In comparison, writing a review about a book is easy: in a few paragraphs you criticize what it took years to write. Nevertheless, the readers deserve an honest review, so I won't hide the fact that in my opinion the first book reviewed here is less than what it could have been. This book, *L^AT_EX for Everyone* by Jane Hahn, is published by Personal *T_EX*, Inc. (PTI), and will replace *L^AT_EX: A Document Preparation System* by Leslie Lamport in the PC-*T_EX* packages that PTI sells.

Surely, Lamport's book leaves a lot to be desired as an introductory book. For this purpose, you need a book with a clear expository style, a sufficient number of examples and well designed exercises. On the surface, it looks as if *L^AT_EX for Everyone* could have been such a book, since it has a clear 'if you want this, do that' way of explaining, it has summaries at the end of all sectional units, and lots of exercises.¹ Unfortunately it falls short of being a good introduction: it shows structural flaws, it contains a substantial number of mistakes, and it explains several parts of *L^AT_EX* confusingly or not at all.

Structure

Chapter 2 introduces the basic commands of *L^AT_EX*, and it also tells you how to adjust line spacing, margins, paragraph indentation, and footnote spacing; I will come back to this in a minute.

Chapter 3 is an odd mixture of things: it explains about document styles, typefaces and typeface sizes, sectioning commands, symbolic references, hyphenation, lists, formulas, accents, and headers and footers.

Chapter 4 deals with mathematics, but the environments for displayed equations were treated in chapter 3. Chapter 5, *Rows and Columns*, discusses *tabbing*, *tabular*, *array* and *eqnarray*. There are two problems with this arrangement of material.

1. The information on mathematical formulas is spread over three chapters.
2. *array* is used in chapter 4 on pages 93 and 99, but is not explained until later on, on page 128.

¹ I should add that the answers to the exercises are given in small print below the questions.

Chapter 6, *Customization*, treats page and line breaks, centering, vertical and horizontal space, lengths and boxes. This is followed by a chapter on floating objects and one on preparing large documents. In my view, chapter 6 should have been put after chapters 7 and 8, and combined with parts from chapter 3 in a chapter on influencing the layout.

Chapter 7 contains a lot of useful information about floating tables and figures, but it could have been written more concisely I think. And, like other authors of books on *LATEX*—see some of my earlier reviews—Ms. Hahn does not clarify what **table** and **figure** are, namely ‘envelopes’ for floating figures and tables.

Furthermore, the book contains seven appendices. Appendices A and B, on user-defined commands and counters respectively, contain lots of useful information with instructive examples. Appendix C, on style parameters, is also a nice appendix, but it lacks the page-layout and list-layout diagrams, which are by now familiar to most *LATEX* users.

Appendix D treats the *picture* environment. Appendix E, *Errors*, is a particularly good appendix, with lots of examples. I missed one thing in this appendix: what happens when you forget the required argument of `\begin{thebibliography}`?

Appendix F gives examples in the form of question and answer, and is one of the best parts of the book!

Appendix G ‘discusses’ *SLiTeX* in twelve (sic!) lines. The page on which it is printed can just as easily be torn out of the book, since all it tells the reader is that *SLiTeX* is a program similar to *LATEX*, designed for creating slides, and with commands different from those of *LATEX*. If the reader wants to know more, he or she is advised to print and read *slides.tex* and *local.tex*.

Finally, the index is awkward to work with: all environments must be looked up under the main entry ‘environment’, and all commands under the main entry ‘commands’. Strangely, the entry ‘commands’ is followed by ‘captions’, ‘center’, ‘comment’, ...² My preference would be to list, e.g., ‘picture’ environment between ‘picture’ and ‘placement’, as in the *LATEX* User’s Guide, or to have a separate command index.

My main criticism is that the structure of *LATEX for Everyone* does not reflect the philosophy behind

² Probably because the index was generated as explained on pages 194–197 of the book—see further on.

LATEX—like most other books on *LATEX* unfortunately. Chapters 2–4 of *LATEX, a Document Preparation System* by *LATEX*’s creator Leslie Lamport mostly explain about those features of *LATEX* that are related to logical structure of a document. Only in chapter 5 does he discuss those features that are more related to the visual structure of a document.

By contrast, Ms. Hahn continually mixes structure commands with layout commands.

An example: in almost every chapter Ms. Hahn introduces a command that accepts the `\`` command, and every time she explains what `\`[...]` does. If she had moved this to a separate appendix on layout changes, this would reflect the philosophy of *LATEX*, and it would make the exposition much clearer.

Another one: in section 3.10.1 she gives this example

```
\begin{itemize}
\item [\$\heartsuit$] potatoes
\item [\$\heartsuit$] celery
\item [\$\heartsuit$] frying chicken
\item [\$\heartsuit$] milk
\end{itemize}
```

immediately after she has introduced the *itemize* environment. First of all, this can be done much simpler with a `\renewcommand` of `\labelitemi`. Secondly, this sort of example really belongs in a separate chapter on layout changes.

Errors

This review column does not provide the space required for an extensive summary of all errors in *LATEX for Everyone*. Instead, I will mention a few interesting ones.

1. The author confuses the document style *book* with the abstract class of documents that can be called ‘book’. Furthermore, to confuse the reader she introduces a new term, ‘style guide’, as a synonym for ‘document style’. She also confuses *LATEX* with its standard document styles (pages 69–70).
2. On page 42 she calls *TeX*’s ‘usual’ typeface, Computer Modern, Times Roman.
3. On page 88: ‘A super- or subscript that is an English word should be set in roman type’. Is this not the case for mathematical texts in French or Dutch?
4. An explanation of `*{n}{cols}` is missing in all places where `tabular` is treated (pages 127 and 288).
5. ‘You should get into the habit of typing names as follows: ... J.^~S.^~Bach’ (page 142). Not

- true, since it depends on the particular typographical convention one uses: in common usage the space between 'J.' and 'S.' is omitted.
6. A table in section 6.6 suggests that *LATeX* does not understand the following units of length: *dd*, *cc*, *bp* and *sp*, which the basic *TeX* program, and therefore *LATeX*, an extension, understands.
 7. In section 6.7, the author uses *\makebox* to get an alignment!
 8. On page 164, Ms. Hahn writes that

```
\oddsidemargin=0in
\textwidth=6.5in
```

results in a right margin of 1 inch. This happens sometimes, but only if you use American letter size paper!

 9. The 'default order of preference' for figure placement is [*bthp*]', according to the author, which is wrong, since this default is given by the document style, for example [*tbp*] in *article*.
 10. On pages 194–197 Ms. Hahn suggests producing an index by sorting the entries in the *.idx* file in your editor, manually changing the *\indexentry* commands into *\item* and so forth, and then combining multiple entries into one. I find this appalling advice, with index programs such as *MakeIndex* available.
 11. Similarly, in section 8.5 there is no mention of *BIBTeX*.
 12. The command *\setlength* is discussed in the main text, whereas *\newcommand*, *\renewcommand* and *\newenvironment* are treated in the appendices. In my view, the latter are more important, because they make typing easier or can clarify the structure of a document. A separate appendix on layout changes would be an appropriate place to discuss *\setlength*.
 13. A discussion of *\newtheorem* is completely missing.

Besides this, Ms. Hahn sometimes suggests bad typography. For example a tall formula, an integral in display style, in text. Shouldn't authors of books on *TeX* keep traditional typographical rules of thumb in mind?

Conclusion

On the whole, *LATeX for Everyone* is an unsatisfactory book. It has the potential of becoming a good book, in a revised edition, if the structural flaws are solved and all the errors are removed.

The author considers math *LATeX*'s strongest feature, a position I disagree with strongly: its main merit is document structuring. Math is a *TeX* feature, and *LATeX* does not add new math capabilities: it presents them in a structured and sometimes more user-friendly way. If Ms. Hahn had recognized the key role of document structuring in *LATeX*, she would probably have written a different book.

A final remark: the publisher chose to have the book produced from 2000 dpi camera-ready copy, which is the high quality output a book on *TeX*, made by *TeX* deserves. Unfortunately, the typeface Computer Modern was used, and the layout is the standard *book* style. That *TeX* can produce 'masterpieces of the publishing art',³ using other fine typefaces and a layout created by a professional designer, is shown too rarely—an exception is Victor Eijkhout's recent book *TeX by Topic*.

Practical SGML, Eric van Herwijnen, first edition, Kluwer Academic Publishers 1990, softbound, 307 pages

'A review of a book on SGML in the columns of *TUGboat*?' some of you may wonder. What does SGML have to do with *TeX*? Well, nothing, but since the term SGML has surfaced often in *TUGboat* and on the TUG conferences the past years,⁴ I thought a review of an SGML book could be worthwhile.

Practical SGML is one of the best books on SGML currently available. To be absolutely honest, there are not many books on SGML—yet—but this book is the only one so far with 'many helpful hints and ideas on developing SGML, applications and discussions of the current software written to be conforming to the ISO standard', as is written in the foreword of the book. This is indeed a book about practical SGML!

The book is divided into three parts. Part I, *Getting started with SGML*, is an introduction to SGML. It explains what a document type definition or 'DTD' is, what the role of the DTD in the processing of the document is, and what steps are necessary to create and process an SGML document.

Part II is intended for document managers or programmers, and explains SGML in more depth.

³ See the last line of the last chapter of *The TeXbook*.

⁴ See for example the proceedings of the 1991 TUG conference.

Some of the topics discussed in this part are: formal aspects of the language SGML, distinguishing data characters from markup, and the reference concrete syntax.

Part III is about SGML implementations and should be read by everyone who has to install and maintain an SGML software system. Mr. van Herwijnen discusses what components are usually found in such a system, how to create SGML documents, how to convert SGML documents into documents that can be processed, for instance to get output on paper, or in order to store information in a database. He also gives some examples of SGML parsers.

The book also contains five appendices. Appendix A contains the answers to the exercises in the book. In appendix B Mr. van Herwijnen tells how he wrote *Practical SGML* using SGML, and in appendix C he even gives the complete document type definition for his book.

Appendix D is a short appendix, in which the author gives common SGML definitions for use with TeX. Finally, appendix E contains useful advice on how to read the ISO standard (8879) in which SGML is defined.

At the end of the book we find a glossary and an index, and throughout the book the author gives lots of valuable references to existing literature on SGML and related topics.

Mr. van Herwijnen is leader of the text processing section at CERN, the European Laboratory for Particle Physics in Geneva, Switzerland. SGML is one of the important tools in the text processing section at CERN, which probably explains the high quality of *Practical SGML*: it was written by someone who has extensively used SGML in practice. Since no prior knowledge of text processing or publishing is required to understand what is written in *Practical SGML*, I can highly recommend it to anyone who is interested in this subject.

◊ Nico Poppelier
 Elsevier Science Publishers BV
 Academic Publishing Division
 R&D Department
 Sara Burgerhartstraat 25
 1055 KV Amsterdam,
 The Netherlands
 Internet:
 n.poppelier@elsevier.nl

Book review: *TeX by Topic*¹

Philip Taylor

Perhaps I have been unlucky, but my experience of 'alternative'³ TeX books so far has been rather depressing; in general, they have been badly designed, poorly typeset, and overburdened with errors. It was, therefore, with some trepidation that I agreed to review Victor Eijkhout's *TeX by Topic*. Let me say straight away that on the most significant of these factors—the number of errors—*TeX by Topic* is way ahead of the crowd. I will return to the design and typesetting later in this review.

TeX by Topic is a reference manual to the TeX language, arranged as its title suggests by topic. It makes no pretence to being an introduction to TeX, plunging straight in to the four-level hierarchy ('eyes', 'mouth', 'stomach' and 'bowels') of the TeX processor on page 1. By far the majority of the book is concerned with an explanation of each and every TeX control sequence—primarily those present in *Initex*, but also including those which Victor regards as forming a part of the core of the Plain format (and which are therefore present in the majority of other formats, such as *LATEX*, as well). Unlike *The TeXbook*, the index by command makes no differentiation between true TeX primitives and those provided only by the Plain format; there is, however, a glossary of true TeX primitives. There is also an index by topic, and a comprehensive bibliography composed of some fifty entries. The structured nature of the text becomes apparent on a closer inspection of the indexes, where single references outweigh multiple by approximately 100 : 1.

Each chapter of the book deals with one particular TeX topic: fonts, boxes, modes, numbers and so on; in some cases, a topic is split across several chapters: for example, paragraphs are treated as composed of a start, an end and a shape, each being afforded a chapter of its own. This treatment is highly beneficial: the TeX aficionado will be able to tell just from the table of contents in which

¹ *TeX by Topic: A TEXnician's Reference*: Eijkhout, V; 1991. Published by Addison-Wesley at £24.95² (U.K.), ISBN 0-201-56882-9. 307pp, two indexes. Midway between Foolscap 4to and Super Royal 8vo.

² Addison-Wesley (U.K) refused to quote an American price, despite being told this information was required for a book review

³ By 'alternative', I mean other than from the hand of the Master...

chapter any given aspect of *T_EX* is most likely to be treated. The less practised reader may choose instead to consult the index by topic, or even the index by command if the exact meaning of one or more commands is sought.

The format of almost every chapter is the same: each chapter commences with a very brief discussion of the topic treated (essentially an abstract), and then lists the control sequences relevant to that topic. Then follows an introduction to the topic, followed by a detailed discussion of each aspect of the topic in individual sections. Each control sequence listed at the beginning is discussed at some point in the text, although there is no attempt to force the book into the format of an encyclopaedia: an individual control sequence *may* form a section or subsection in its own right, or it may be discussed in a wider context. This treatment makes the book more readable (at least, for those like myself who actually *enjoy* reading deeply technical matter while lying in bed late at night), although admittedly at some expense to its functionality as a pure work of reference. The compromise is a happy one, and few will have cause to berate the author for lack of consistency. References to the topic which have appeared in other publications are usually deferred to the end of a chapter, and one notes that the author is not averse to self-citation (however, the self-citations form only ten per cent of the bibliography, so other authors need not feel slighted; Knuth, by comparison, forms just over twenty per cent).

In assessing the accuracy of such a highly technical work, one has two choices: either read the entire text like a hawk, searching for infelicities, no matter how small, wherever they occur, or use certain well-known features of the subject which have historically caused the greatest number of errors in previous texts. In assessing *T_EX by Topic*, I have attempted to use both techniques. So far as I can tell, the book is *almost* error-free: the treatment of (for example) \afterassignment is excellent, and makes it quite plain that only one token can be saved in this way; a subsequent use while the first is still pending will override the first. Similarly the treatment of \aftergroup makes it plain that *its* effect is cumulative. In dealing with \futurelet, Victor emphasises that it causes \catcode staticisation of the ‘peeked-at’ token: this point is so poorly understood, and the cause of so many problems in attempts at the advanced use of \futurelet, that documenting this ‘feature’ is essential; I am very pleased to see that it is afforded a paragraph in its own right.

Perhaps one might criticize the fact that Victor does not point out that \afterassignment *transcends* the group structure — i.e. an \afterassignment performed within an inner group, and not ‘used’ within that group by an assignment, will be used whenever the next assignment *does* take place, even if the token which has been saved has gone out of scope. This is, however, nit-picking: the technical accuracy is excellent. (I think I found one serious flaw in the whole book, and a few lesser infelicities; for example, on page 70, Victor asserts that \hfilneg (\vfilneg) is equivalent to \hskip (\vskip) 0 cm minus 1 fil; I would assert that it is equivalent to \hskip (\vskip) 0 cm plus -1 fil, which is entirely different.)

The treatment of terminal-# in a parameter list follows the party line, in stating that the open brace which follows forms a part both of the parameter list and of the replacement text; this explanation, which is essentially the same as that given in *The T_EXbook*, has never seemed entirely satisfactory to me (even though it is factually true), and I would have preferred to see the simple statement that a terminal-# in a macro parameter list requires that the macro and its parameters, when used, *be followed by a brace-delimited token list*. The statement is inaccurate, but leads, I believe, to a more rapid understanding of the whole *raison d'être* of terminal-#. One can always go on to explain that, of course, *T_EX* can't check for the matching close-brace at that point, but at least the opening brace is required and checked for. The example which Victor has chosen makes use of this very feature.

The content is well-chosen: Victor takes as examples for discussion many of the more obscure features of the Plain format (for example, \newif), and gives a very lucid explanation both of their implementation and of their inner workings; I remember only too well asking the combined readers of *T_EXhax* for just such a lucid explanation of \newif during my exploratory years with *T_EX*, and getting no response... The treatment of spaces is admirably comprehensive, with clear differentiation between [*one*] optional space/s] and *T_EX*'s being in state S ('skipping spaces').

The proof-reading is to a very high standard; there are again a few infelicities (for example, on page 296 the column headed \mathcode should actually read \delcode), but these do not detract from the usefulness of the text, nor are they sufficiently numerous to perturb the eagle-eyed reader. The hexadecimal values given in the tables which appear at the end of the text should be treated with a degree of scepticism: Knuth himself has been known

to vacillate about the 'correct' value for some of the more arcane maths delimiters, and they may continue to fluctuate for a while. For reasons which are not at all clear, Victor collates "2200 after "2203 on page 295; I suspect this was a rare oversight.

The grammar and usage are unexceptionable; there is a strange ambiguity as to whether the book is written for an American or a British audience, with 'centre' invariably spelled in accordance with <Br.E> usage, whilst 'mathematics' is invariably abbreviated to 'math' (<Am.E>), where <Br.E> would have 'maths'. (I still can't pronounce the former of these variants; it always sounds to me as if I'm lisping!). Victor concurs with the authors of the *Algol-68 Report*⁴ in treating the plural of 'formula' as 'formulas' rather than 'formulæ'. There is a rather strange usage of 'treat' in the opening paragraphs of the preface, leading the reader to expect the archaic 'treat of', but instead leading to no preposition at all.

It is perhaps unfortunate that reviewers of books on typography and typesetting can no longer allow themselves the luxury of commenting solely on the content—it is almost *de rigueur* to pass judgement on the design and typography of the text as well, even though this may well have been without the control of the author; in the present work, for example, the typographic design is attributed to Merry Obrecht.

Whilst from a content point of view the book can hardly be faulted, the design and typesetting do not, in the opinion of the present reviewer, do it justice. Such criticisms are, of course, highly subjective, unlike those of the accuracy or otherwise of the text; book design is by its very nature a highly personal and individual art-form, and it would be a foolish reviewer indeed who insisted that any particular element of a design was categorically right or wrong. None the less, the design cannot be completely ignored, and the following remarks are therefore offered as one person's view, rather than as facts cast in stone...

Ignoring the received wisdom that underlining is an artifact of typewritten text, and has no place in typeset material, section headings and subsection numbers have both been underlined; this obsession with printed lines also manifests itself in the design of each opening chapter page, where a vertical and horizontal rule (forming an enormous,

horizontally-elongated, letter 'L') serve to set off the title of the chapter from the other material on the page. The title and half-title pages echo this design, but duplicate and offset a second copy of both rules to form two nested 'L's. In the running heads, white space and a forward slash separate the section number from the current section name.

The placement of page numbers is rather less than felicitous on the opening chapter pages; on the first such page, for example, a black rule about 8 pt high and 1.5 pt wide appears at the bottom of the left margin parallel to the last line of the page, and for a long time I thought this was a change bar reflecting some improvement from an earlier edition. Only after several readings did I notice that this was a first edition... In all, the design is rather too fussy and *avant garde* for this reviewer.

The book is set in Baskerville and Gill Sans, and the general impression of the main text is that it is under-inked. I have to hand an issue of *Baskerville*⁵ typeset in Baskerville at 1270 dpi; the visual density of the type is significantly greater, and one wonders the printers were perhaps a little parsimonious in their use of ink (but see below). Whilst the main text just holds together, the slightly smaller font used on the title page and occasionally elsewhere breaks up badly: there are two distinct discontinuities in both lower- and upper-case 'O', and one in upper-case 'C', although, rather intriguingly, there is a single line of nine-point text in the colophon, which is otherwise entirely in ten point, which reads: *Printed in Great Britain by Mackays of Chatham plc*: in this line, the lower-case 'o' does not break up. One wonders if (a) Mackays added this line to the plate themselves, and (b) whether, in fact, the binders were to blame for the apparent under-inking rather than the printers... The back cover suffers from the classic under-kerning of the *TEX* logo which seems to occur whenever professional typesetters are entrusted with the task of reproducing it.

The typesetting conventions of the main text may cause the aware reader some hesitation: en-dashes, set off by the space of the line, have been used where em-dashes might otherwise be expected. When a *TeX* control sequence occurs as a part of a section heading, the necessity to drop temporarily into the lower-case characters of a teletype-like font interrupts the continuity of an otherwise entirely upper-case Gill Sans heading; the interruption is less disturbing in the subsection headings, which are themselves in mixed-case, but there appears to have been no attempt to match for visual density.

⁴ *Report on the Algorithmic Language ALGOL 68*: Wijngaarden, A. van, Mailloux, B.J., Peck, J.E.L. & Koster, C.H.A.; 1969. Offprinted from *Numerische Mathematik*, 14, 79–218 by Springer-Verlag.

⁵ The Annals of the U.K. *TeX* Users' Group

The best point of the typesetting/design is its consistency: first paragraphs are *never* indented, and *almost* all pages are exactly full, even at the expense of an occasional widow or orphan (<Am.E> 'club-line'). The use of Baskerville ensures a highly legible text.

In summary, I have absolutely no hesitation in recommending this book, not only for the *cognoscenti*, but also for the more casual TeX user who finds that the rather less formal but more didactic nature of *The TeXbook* renders it somewhat less than ideal as a work of reference. It seems unlikely that many would choose to learn TeX solely by a study of *TeX by Topic* (after all, even the *Algol-68 Report*, which must rank as one of the most comprehensive language definitions ever written, is accompanied by the less formal but infinitely more readable *Informal Introduction*), but once past the initial learning stage, few would fail to derive benefit from easy access to a copy of *TeX by Topic*. Its accuracy puts most of its competitors (well, to be honest, it doesn't have any *real* competitors) to shame, and its usefulness is without doubt. It will join *Computers & Typesetting: Vols. A-E*⁶ and *Another Look at TeX*⁷ as essential reference material on my TeX shelf. I am reliably informed that *Un petit livre de TeX*⁸ should join these three, but I haven't yet had the opportunity to see a copy, and I've just received *TeX by Example*¹⁰ but am not yet in a position to pass judgement...

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

⁶ *Computers & Typesetting: Vols. A-E*: Knuth, D.E; 1984-. Published by Addison-Wesley in both case-bound (A-E) and soft-bound (A & C) editions. The canon.

⁷ *Another Look at TeX*: Bechtolsheim, Stephan von; 1987. Pre-print copy. Rumoured to be appearing as a multi-volume work by a real publisher (and under another title) 'real soon now'.

⁸ *Un petit livre de TeX*: Seroul, Raymond; 1989. Published by InterEditions, Paris. ISBN 2-7296-0233-X.

⁹ Published in translation as *A Beginner's Book of TeX*: Seroul, Raymond & Levy, Silvio; 1991. Published by Springer-Verlag. ISBN 0-387-97562-4.

¹⁰ *TeX by Example*: Borde, Arvind; 1992. Published by Academic Press at £13-00 (U.K.), \$19-95 (U.S.). ISBN 0-12-117650-9 (A.P. were much more helpful in quoting American prices...)

A TeX Macro Index

David M. Jones

The TeX community is blessed with a plethora of publicly-available macros; a decade's worth of experience is available from a series of archives throughout the world. The hitch, of course, is that there is no systematic catalogue of these macros, so the vast majority of TeX users remain unaware of their existence. Frequently, the only recourse a user has is to cast a message upon the electronic waves and hope that some useful information makes its way back from the depths. For TeX users without access to such electronic forums, the situation is even bleaker.

With this in mind, I decided to compile an index of TeX macros. The scope of the Index includes all macros that are available via anonymous ftp or mail-server or some similar mechanism. Priority is given to the major archives (Aston, Stuttgart, SHSU and ymir). The Index covers a variety of packages, including plain TeX, eplain, LATEX, AMS-TEX, AMS-LATEX, LAMBERT-TEX, YTEX and TEXT1. Commercial packages are included only if the information is supplied to me by the vendor.

A minimal useful index entry consists of the following fields:

Name The name of the macro package, usually the name of the file containing it.

Description A short (1-3 line) description of what the package does.

Keywords A list of keywords to facilitate searching for special-purpose macros, as well as to help describe the macros. A glossary of keywords is included.

Archives A list of archives where the package can be found. Whenever possible, the home location of the package is identified and marked with an asterisk.

Whenever possible or appropriate, the following information is also included:

Author The name and address (preferably electronic) of the author of the package.

Latest Version The date and/or version number of the latest release of the package.

Supported Whether or not the package is officially supported, that is, whether the author wants to receive bug reports and/or comments on the package.

See also A list of other packages with similar features.

Note Any additional information which seems pertinent.

As examples, here are two representative entries from the draft of the Index.

Name: btxmac.tex

Description: Provides support for using BiBTeX with plain TeX.

Keywords: plain TeX, BiBTeX, bibliography

Author: Karl Berry and Oren Patashnik
(opbibtex@cs.stanford.edu)

Supported: yes

Latest Version: v0.99j, 14 Mar 1992

Archives: labrea*, ymir

Name: longtable.sty

Description: LATEX style option defining a multi-page version of tabular.

Keywords: LATEX, array, tabular, page

Author: David Carlisle
(carlisle@cs.man.ac.uk)

Supported: yes

Latest Version: v3.1, 6 Apr 1992

Archives: shsu*

Note: Documentation requires Mittelbach's doc.sty.

See Also: supertab.sty

The current draft of the Index (dated June 1, 1992) has approximately 600 entries. I hope to increase that to 1000 by the end of June, when I plan to release the Index to the general public by making it available by anonymous ftp and mail server. Beginning in July at the Annual TeX Users Group Meeting, the Index will also be distributed through TUG. In the meantime, I'll be contacting the authors of macro packages and requesting their help in verifying the information I have. If you have written a macro package that you think should be mentioned in the Index, please contact me (preferably by electronic mail) at the address below.

- ◊ David M. Jones
MIT Laboratory for Computer
Science
Room NE43-316
545 Technology Square
Cambridge, MA 02139
Internet:
dmjones@theory.lcs.mit.edu

Tutorial

Names of control sequences

Victor Eijkhout

1 Introduction

In the 'Lollipop' format that I wrote, first to typeset my ph.d. thesis, then to set my book 'TeX by Topic' (Addison-Wesley 1992), I try to move away a bit from the ordinary TeX syntax. For instance, declaring a \newskip register, and setting the value of it are done using only one command, with the syntax

```
\Distance:UnitQuad=12pt
\Distance:parindent=UnitQuad
```

The first command here declares a skip register \UnitQuad, and initializes it to 12pt; the second takes the csparindent and sets it to the value of \UnitIndent.

In order to perform these actions correctly, we should be able to distinguish

1. whether a control sequence is already defined (\parindent) or not (\UnitQuad), and
2. whether a string is the name of a control sequence (UnitQuad) or a literal string (12pt).

Both problems are really the same, as we shall see below.

2 Messing with \csname

The matched pair of control sequences \csname and \endcsname can be used to construct control sequences out of arbitrary characters. Ordinarily, names of control sequences are limited to letters only (or, to be more precise, to characters of category 11), but in between these two commands any character can appear. Macros and other expandable commands are also allowed, as long as they will ultimately expand to characters.

For instance

```
\csname a:b\endcsname
```

expands to a control sequence with a colon in the name, and

```
\csname \ifhmode h\else v\fi skip\endcsname
```

expands to either \hskip or \vskip.

A useful property of \csname is that if you form the name of a control sequence that has no definition (that is, it is no primitive, register, macro, or otherwise defined), the result is equivalent to \relax. Thus

```
\csname probably:not-defined!\endcsname
is with a high likelihood equivalent to \relax. We
can use this property to test whether a control
sequence has been defined: if it hasn't it is equivalent
to \relax.
```

For this test we use \ifx which tests equality
of control sequence definitions¹. For instance

```
\let\HorizontalContainer=\hbox
\ifx\hbox\HorizontalContainer % is true
\def\{<>\} \def\b{<>}
\ifx\{a\}b % is true
```

In order to see if a control sequence has been defined,
we have to compare it to \relax.

Suppose we want to have a macro that can be
called

```
\ifUndefined{maybe:macro} .. \else .. \fi
```

We can define this as

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

The \expandafter activates the \csname to form
the control sequence name, and \ifx then compares
it to \relax. Note that we have actually defined
a macro that tests whether a control sequence is
undefined.

We can now start assembling the macro
\Distance.

```
\def\Distance:#1=#2
{\ifUndefined{#1}\MakeNewSkip{#1}{#2}
 \else \SetOldSkip{#1}{#2}
 \fi}
```

(Note that with this definition the second parameter
is a string delimited by a space, for instance the
space resulting from the line end.) So far we have
glossed over one point: the value that is assigned
(parameter 2) can be either a value or again the
name of a control sequence. It makes sense then to
define

```
\def\ValueOf#1{\ifUndefined{#1}#1
 \else \csname#1\endcsname \fi}
```

which takes the argument itself if it is not the name
of a control sequence, and otherwise forms that
control sequence.

Now \SetOldSkip is easy:

```
\def\SetOldSkip#1#2{%
 \csname #1\endcsname=\ValueOf{#2}}
```

For \MakeNewSkip we first need to allocate a new
skip:

```
\def\MakeNewSkip#1#2{%
```

```
\expandafter\newskip
\csname#1\endcsname
\SetOldSkip{#1}{#2}}
```

Just a small remark here: the \newskip macro of
plain TeX has been declared with the prefix \outer,
so it cannot be used the way it was done above.
In order to write the above code, the definition of
\newskip has to be copied from plain TeX, but without
\outer.

3 And now what?

The macro \Distance explained above is not ex-
actly spectacular, but I hope that the readers have
learned some new tricks about control sequences.
Furthermore, I will be using the techniques ex-
plained here in forthcoming articles about certain
parts of my Lollipop format.

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

Puzzle

Where does this character come from?

Frank Mittelbach

Puzzle:

If some complex macro defined by you
produces funny extra characters like
“Ω” or “œ” in the output, what kind of
mistake could be the reason?

◊ Frank Mittelbach
 Electronic Data Systems (Deutschland)
 GmbH
 Eisenstraße 56 (N15)
 D-6090 Rüsselsheim
 Federal Republic of Germany
 Mittelbach@mzdmza.zdv.Uni-Mainz.de

¹ It can also be used to test characters, but that's
not relevant here.

Macros

The bag of tricks

Victor Eijkhout

Yo! Home boys and girls.

Another installment of the bag of tricks, this time with some stuff about hyphenation.

Sometimes you want to prevent hyphenation at a hyphen. Inserting a \nobreak doesn't work. Reader Sonja Maus alerted me to this, and gave as a solution \hbox{-}.

The New York Times uses a typesetting system that produces with disturbing regularity the hyphenation "do-n't". (Question for wizards: why doesn't *TEX* produce this hyphenation?) The following macro provides a solution to this problem: with

```
\def\n{`\discretionary{}{not}{n't}}
typing
do\n'
```

gets hyphenated as if it were written "do not". Try for instance the following input:

```
\spaceskip=3.3pt plus 1.2pt
\setbox0\hbox{I'm perturbed seeing
  words that do}
\hsize\wd0 \parindent0pt
I'm perturbed seeing words
that do\n' hyphenate correctly\par

\setbox0\hbox{I'm perturbed seeing
  words that don't}
\hsize\wd0
I'm perturbed seeing words
that do\n' hyphenate correctly\par
```

The third item in this Bag of Tricks is a homework project. *TEX*'s hyphenation has been giving people trouble for ages, and clever solutions have been known for some time. Here's a way of dealing with problems that was used in German [3] and Dutch [1] extensions to *LATeX*, and that can be adapted for many more applications.

If *TEX* finds discretionary hyphens \- or explicit hyphens - in a word, no other hyphenation positions will be considered. This can be awkward. People have solved this by redefining the double quote as an active character, so that you write

the Zielknijper"-Plrwtskofsky theory

and either of the long names will still be considered for hyphenation. By defining combination of the

double quote with other characters you can achieve other effects. Here are some possibilities, but all of this is subject to taste and to particular applications.

"a (or with any other vowel) gives a disappearing syllable break, which occurs in Dutch and German and older English texts: co"ordinate hyphenates as co-ordinate. (Even more cute, in Dutch be"inken looks like 'beïnnen' and hyphenates as be-inken.)

"" gives a break position that will hyphenate without a hyphen. I use this in bibliographies to enable a break in expressions such as '123(1988)'.

"! will give a double quote when you need that character, for instance in *TEX*'s hexadecimal notation.

"... " can be implemented as language-specific opening and closing quotes. The implementation below is for old-style English.

Here are the macros.

```
\gdef\allowhyphens
  {\penalty\CM \hskip\z@\relax}
{\catcode`"=12 \gdef\hex{"}
\catcode`\'=13
\def "#1{\ifx#1%
  \discretionary{-}{\i}{\{\accent'177 \i\}}%
  \else\ifx#1-%
  \allowhyphens-\penalty\z@\allowhyphens
  \else\ifx#1`%"%
  \else\ifx#1'%"%
  \else\ifx#1"\hskip\z@\relax
  \else\ifx#1!\hex
  \else\ifx#1|\allowhyphens
  \else \discretionary{-}{#1}%
    {\accent'177 #1}%
  \fi\fi\fi\fi\fi\fi}
```

More strange phenomena with hyphenation can be found in [2].

Until next time. See you backstage at the next TUG meeting!

References

- [1] Johannes Braams. Babel, a language option for *LATeX*. *TUGboat*, 12:291–301, 1991.
- [2] Michael J. Downes. Line breaking in \unboxed text. *TUGboat*, 11:605–612.
- [3] Hubert Partl. German *TEX*. *TUGboat*, 9:70–72, 1988.

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

Over the multi-column

Péter Huszár

With a good multi-column environment you can handle almost every problem you are faced with. But sometimes you may want to produce more complicated pages: text around a picture or real newspaper pages which are sets of articles (boxes) rather than sequences of columns. The environment presented in this article gives you a convenient way to describe such pages.

The idea

When you use plain *TEX*, you needn't care about page-setting. *TEX* has a powerful algorithm for making lines into a single-column page. This format is so simple (even if it depends on a dozen or so parameters) that such a page can be produced by a constant output routine provided in *plain*. Thus you only have to type your text and *TEX* builds up this format. However, if you want to produce a more complex page, first you should describe its structure, i.e., how many parts does a page contain and how are they connected logically and physically. For example, a multi-column page consists of as many parts as the number of columns; there is an order on the parts (the sequence of the columns); and they are put next to each other. But this is still a restricted form of a general page which looks as follows:

A page is built up of logical areas. I will use the phrase 'logical area' for the logical units of the page (articles, pictures, etc.). Each area is a list of boxes (you may want to divide an article in two or more parts). For example let's consider the page of Figure 1.

There are six areas on the page:

- the TITLE
- a PICTURE
- article 1, which consists of three boxes:
BOX 1.1, BOX 1.2 and BOX 1.3
- article 2, which consists of just one box:
BOX 2.1
- article 3, which consists of two boxes:
BOX 3.1 and BOX 3.2
- an ADVERTISEMENT

As you can see it is rather difficult to describe this page in a multi-column format which is just one area with restricted placement of its boxes on the page. The main problem is that there is no proper ordering on the boxes (if you represent each box as a point then this is the same problem as ordering on complex numbers), so you can't make one list

from the boxes. I will now give an environment to describe such a page.

How to describe (plan) the page

The strategy of planning is that first we describe the areas and the boxes on the page, then we 'fill' the boxes with their contents (text, picture, etc.). This means that the dimensions (width, height and depth) of a box are independent from its contents, contrary to *TEX*'s boxes where you can prescribe only one of the three dimensions and the other two will be known just after putting the contents into them. (The problem of how much space a text needs is rather general, I guess.)

The description part of the macro package should allow you

1. to describe as many areas as you want to;
2. to describe the list of boxes of each area;
3. to specify vertical and horizontal sizes of each box;
4. to specify the position of each box on the page.

First let's examine these problems from the point of syntax. You should specify boxes and areas as a list of boxes. An element in the list has a data part and a pointer to the next element; this pointer points to 'null' (i.e. nowhere) at the end of the list. It means each box (I'll use the name *planbox* for these boxes to differentiate them from *TEX*'s boxes) should have the following parameters:

- a. a name (we want to handle it as an ordinary allocated \box),

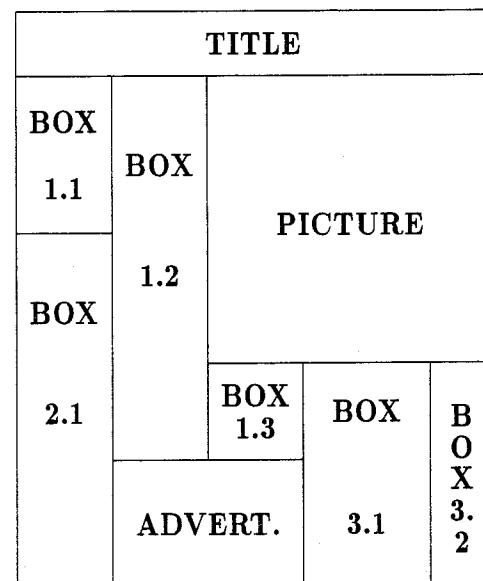


Figure 1.

- b. horizontal and vertical sizes (width and depth),
- c. position on the page specified as horizontal and vertical distance of the left upper corner of the box from the left upper corner of the page,
- d. a pointer to the next planbox (the best way is to give the name of the next planbox);

in other words, something like this:

```
<name>,<width>,<depth>,<hdis>,<vdis>,<next>;
```

where `<next>` is either a `<name>` or the constant `\null`. The semicolon is redundant but you can read the source code easier with it. In this format we can describe the example page above with the `\plan` macro as follows (the parameter of `\plan` is the total number of planboxes on the page):

```
\plan 9; % total number of planboxes
TITLE ,\hsize, 2cm, Opt, Opt,\null;
\ArtIone , 3cm,10cm, Opt, 2cm,\ArtItwo;
\ArtItwo , 3cm,13cm, 3cm, 2cm,\ArtIthree;
\ArtIthree, 4cm, 2cm, 6cm,13cm,\null;
\ArtIIone , 3cm, 6cm, Opt,12cm,\null;
\ArtIIIone, 4cm, 5cm,10cm,13cm,\ArtIIItwo;
\ArtIIItwo, 2cm, 5cm,14cm,13cm,\null;
\AD , 7cm, 3cm, 3cm,15cm,\null;
\PICTURE , 10cm,11cm, 6cm, 2cm,\null;
```

This format also specifies the areas. Each area starts after the end of the previous area, i.e., after a `\null` pointer. You'll see below that this information is enough to handle the areas.

Until now I've written about whole pages, but you aren't restricted to plan the whole page every time. If you plan just a part of the page, the origin for the positions of planboxes is the left upper corner of the planned part (i.e., relative positions, so you can shift the plan on the page without changing them) and the planned part will be put at the current position when `\bpage` (see below) is performed.

Filling the page

The plan is ready, `TEX` knows the structure of the page (or the planned part of it), and we can start putting the text into planboxes. The procedure my macro offers you is the following:

- You should choose an area you want to deal with.
- You can type in your text.
- At any point you are allowed to choose another area.
- At any point you are allowed to switch to the next box within the area (like an `\reject`).

- If you don't want to switch by hand, the macro automatically switches to the next box when the current one is full. After the last box in the area it gives you a warning.

Let's consider the actions step by step.

To start the page (the planned part) and to select an area you simply type:

```
\bpage<area>;
```

where `<area>` is the `<name>` of the first planbox in an area. (You can choose not just the first but any box in the area; however, you can't switch back to the previous boxes.) The chosen planbox becomes an individual page (with its own width as `\hsize` and height as `\vsize`). This is a page from the view of the algorithm but it hasn't got `\headline`, `\footline`, `\footnote` or floating insertions.

Now you can type your text for this area. Having finished, you can choose another area by saying:

```
\nextarea<area>;
```

It is quite simple, isn't it? You can fill the areas one by one in any order you want (there is no restriction). Indeed you can go back to a previous area but if you do so the previous 'value' of the area will be overridden.

Inside an area you are allowed to switch to the next box with the command:

```
\nextbox;
```

There is no need to specify the next box by giving its name. With the pointer technique the macro figures it out.

At `<nextbox>` and `<nextarea>` you have a choice: you can put a `\vfill` at the end of the current planbox with `\fillON` or you can omit it with `\fillOFF`. The macro package sets `\fillOFF` at the beginning of every box.

At the end simply say:

```
\epage
```

to finish the page. All four commands can be used immediately after a paragraph, but not inside one!

Hyphenation. The macro package tries to avoid any hyphenation (for the reasons see below). But sometimes (especially in narrow planboxes) it gives very poor output (underfull hboxes). You can enable hyphenations with `\hyphensON`. However, this way the package may produce hyphenations in the midst of some lines. You can correct mistake by saying `\hbox{word}` to enclose the hyphenated word.

Automatic switch

The main advantage of the macro package is the automatic switch. If a planbox is full of text the macro package automatically switches to the next box in the area. With this feature you can e.g. have text to flow around a picture (see An example below), or plan a page like the one above.

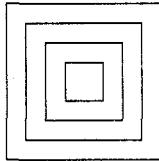
The default way to fill the page is the automatic switch. You can disable it with `\automaticOFF` and enable it again with `\automaticON`.

If there are no more boxes in an area the macro package produces an error message (a warning), and calculates and writes to the logfile the space which the rest of the text needs.

An example

This is a very simple example but it can be used often in everyday **TEX**ing. The problem is a picture which is in the middle of the page and which is narrower or wider than a column or the page:

Here is the text above a picture of width (5pc) less than one third of the column's (18.75pc):



and the text continues here below the picture. The empty space around the picture is about 6.25pc × 13pc. With this environment you can have the text flow around the picture. The parameters are as above (picture height=6.1pc):

```
\let\bs=\baselineskip
\let\hs=\hsize \let\vs=\vsize
\newdimen\pwd \pwd=6.25pc % picturewd
\newdimen\pht \pht=6.1pc % pictureht
\newdimen\ptop \ptop=2\bs
\newdimen\pbtop \pbtop=\ptop
\advance\pbtop \pht
\def\boxit#1{\vbox{\hrule\hbox{\vrule
  \kern7pt\vbox{\kern7pt#1\kern7pt}%
  \kern7pt\vrule}\hrule}}
\plan 4;
\above, \hs, \ptop, Opt, Opt, \near;
\near, 11pc, \pht, Opt, \ptop, \below;
\below, \hs, 2\bs, Opt, \pbtop, \null;
\pict, \pwd, \pht, 12pc, \ptop, \null;
Just a little text outside of the plan to
show the possibility of planning a part
of the page.
\bpage\above;
\automaticON \hyphensON
```

Here are two lines of text above the picture. When these two lines are full the text continues at the side of the picture. The picture is placed on the right side of the column as you can see. When this box is full of text (somewhere here at this point) the text flows automatically to the next box which is actually a little space below the picture `\fillON` for the rest of the paragraph.

```
\nextarea\pict;
\fillON
\leftline{}\nointerlineskip
\vfill
\centerline{%
\boxit{\boxit{\boxit{\boxit{}}}}}}%
}

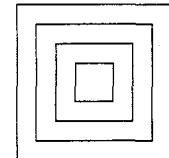
\epage
And now we're outside of the planned part
again.
```

This code yields (notice that the whole plan is in the middle of a multicolumn environment):

Just a little text outside of the plan to show the possibility of planning a part of the page.

Here are two lines of text above the picture. When these two lines are full the text continues at the side of the picture. The picture is placed on the right side of the column as you can see. When this box is full of text (somewhere here at this point) the text flows automatically to the next box which is actually a little space below the picture for the rest of the paragraph.

And now we're outside of the planned part again.



TEXnicalities

The rest of the paper looks behind the screens and explains how this package works.

Planning the page

When the user invokes `\plan` the package should store all the information about the structure of the page. This includes the attributes of the planboxes and some supplementary information used by the package. The attributes require registers for their values while the system information supplies some notes on these registers. First of all, `\plan` stores

the number of planboxes on the page in `\planb@xno` and the current insertion number in `\maxplanb@xno` (the reason for this will be explained later).

```
\newcount\planb@xno
\newcount\maxplanb@xno
\def\plan#1; {\bgroup \global\planb@xno #1
  \global\maxplanb@xno\insc@unt
  \makeplanb@x}
```

Information about each planbox is stored by the recursive macro `\makeplanb@x`. This is done in two steps: first allocating the necessary registers (`\planboxall@c`), then setting their values (`\setsiz@s`):

```
% creating a planbox
% #1 name, #2 width, #3 depth
% #4 hdis, #5 vdis, #6 next
\def\makeplanb@x#1,#2,#3,#4,#5,#6; {%
  \planboxall@c#1#6
  \setsiz@s#2,#3,#4,#5;
  \advance\planb@xno \m@ne
  \ifnum\planb@xno>\z@
    \let\next\makeplanb@x
  \else \let\next\plan@rest \fi \next}
```

Allocation. We need six registers for a planbox: a box register named `<name>` for the contents; four dimension registers for `<width>`, `<depth>`, `<hdis>` and `<vdis>`; and a token list register for `<next>` which contains the `<name>` of the next planbox. The package should figure out from the `<name>` the other registers for this planbox. Using an insertion, we get just a box register, but no token list register and instead of four just one dimension register. The token list register can be ‘appended’ to the insertion but we have to allocate the dimension registers separately from the other registers and use a ‘pointer’ to these dimensions. This pointer comes from the count register of the insertion. It always points to the highest of the four consecutive dimension registers. The first version of this would look like this:

```
\def\planboxall@c#1#2{%
  \newdimen#1      % the names
  \newdimen#1      % of these
  \newdimen#1      % registers are
  \newdimen#1      % unimportant
  \newinsert#1
  \global\toksdef#2=\allocationnumber
  \global\toks\allocationnumber{#2}
    % the token list register
  \global\count#1=\count11
    % the pointer to the
    % highest allocated dimen
}
```

Unfortunately there are some problems with this construction. First of all the `\new...` commands cannot be used inside a macro because they are defined to be `\outer`. Redefinition would be a good solution for this but not the other problems. Namely in this case you would find in the log file many messages about insertion and dimension allocations but no information about planbox allocation. Furthermore this is not an efficient construction and its form is quite different from the other kinds of allocation in `plain`. Yet planbox is something like an insertion: not just a set of certain registers but also logical connections among them. For this reasons my solution for the allocation is the following, which is a simple merge of a `\newinsert`, a `\newtoks` and four `\newdimen` commands with only one `\ch@ck` for each kind of register and with the appropriate message to the log file. I also set the value of the token list register here because I want to keep the assignments connected with `\allocationnumber` together:

```
% allocation for width,depth,
%                               hdis,vdis,name,next
\def\planboxall@c#1#2{%
  \global\advance\insc@unt by\m@ne
  \global\advance\count11 4
  \ch@ck0\insc@unt\count
  \ch@ck1\insc@unt\dimen
  \ch@ck4\insc@unt\box
  \ch@ck5\insc@unt\toks
  \allocationnumber=\insc@unt
  \global\chardef#1=\allocationnumber
  \global\toksdef#2=\allocationnumber
  \global\count#1=\count11
  {\advance\count#1 -3
  \wlog{\string#1=\string\insert
    \the\allocationnumber;
    \dimen\the\count#1...
    \dimen\the\count11 .}}
  \global\toks\allocationnumber{#2}}
```

For storing the values, it would be better to use the pointer of the planbox, but instead I use the ‘dirty information’ that right after the allocation `\count11` points to the same register as the pointer of the planbox would do (this way I save an indirection):

```
% storing width, depth, hdis and vdis
% in the appropriate registers
\def\setsiz@s#1,#2,#3,#4; {%
  {\global\dimen\count11 #4
    \advance\count11 \m@ne
  \global\dimen\count11 #3
    \advance\count11 \m@ne
  \global\dimen\count11 #2}
```

```
\advance\count11 \m@ne
\global\dimen\count11 #1}}
```

After getting the attributes of the planboxes, `\pl@nrest` is invoked. It stores the current insertion number in `\minplanb@xno`. This register and `\maxplanb@xno` point out the place of the planboxes; this information will be used at the end of each use of the package to release the occupied registers. I also allocate a planbox for `\null`. It's a trick and seems to be useless here but you'll see its importance below. Nevertheless, notice that `\null` is not a real null pointer but a planbox!

```
\newcount\minplanb@xno
\def\pl@nrest{%
\global\minplanb@xno\insc@unt
\planboxall@c\null\zero
\setsiz@s\hsize,\maxdimen,\z@,\z@;
\egroup}
```

Filling the page

The structure is ready, all the registers have been allocated and all the logical connections are set; we can start to fill the planboxes with their contents.

Let's consider the actions step by step:

The first macro invoked is `\bpage`:

```
\def\bpage#1; {\bgroup \s@vepagesofar
\tolerance=10000
\showboxbreadth1 \showboxdepth1
% there are many Underfull hboxes
% while processing
\advance\baselineskip Opt
plus .3pt minus .1pt
\wlog{Beginning of Page.}
\def\par{\endgraf\egroup\pl@npar}
\output{\fullb@xoutput}\topskip\z@
\bb@x#1; }
```

Its main task is some preparation and initialization. Before any action, it saves the part of the page which is ready at this time in `\s@vepagesofar`:

```
\newbox\p@gesofarbox
\def\s@vepagesofar{\output{%
\global\setbox\p@gesofarbox\vbox{%
\unvbox255}}}\eject}
```

Afterwards, it sets `\tolerance=10000` to avoid overfull hboxes in the planned page. The package produces many underfull boxes without any visible reason. Thus `\showboxbreadth` and `\showboxdepth` are set to their minimal values. The little stretchability and shrinkability of `\baselineskip` is needed because of the relatively small height of the planboxes. After the message to the log file comes the essential part of the macro.

The algorithm of the process. The idea is that the package proceeds through the entire text paragraph by paragraph. Each paragraph is put in a vbox. If this paragraph has room in the current planbox, then it is simply added to the material so far; otherwise the paragraph has to be split up into two parts. The first part goes to the current planbox and the second one to the next planbox. At the same time we should finish the current planbox and switch to the next one. Then the next paragraph is processed.

Redefinition of `\par` is the essence of this idea. Namely, it finishes the vbox by `\egroup` and does the necessary actions through `\pl@npar` (see Accumulating the paragraphs below). The last of these actions is starting a new paragraph and also a new vbox. The output routine is also redefined (this will be explained later) and `\topskip` is set to zero because we're making not a whole page, just a part of it. Believe it or not, no more preparation is needed; we can start the current planbox (in `\bb@x`):

```
\newcount\curplanb@xno
\newdimen\curplanb@xsofar
\newif\iffill@
\def\bb@x#1; {\initb@x#1; \fillOFF \st@rtpar}
\def\newsiz@s#1{%
\advance\count#1 -3
\hsize\dimen\count#1
\advance\count#1 \one
\vsiz@\dimen\count#1
\advance\count#1 \tw@}
\def\initb@x#1; {%
\wlog{The next planbox is #1.}
\global\curplanb@xno#1
\curplanb@xsofar\z@
\newsiz@s\curplanb@xno}
```

Again, first some initialization for the box (`\initb@x`). This means a message to the log file, a note on the current planbox to `\curplanb@xno`, resetting the height of the material in the planbox so far to zero (there is no material at all) and setting `\hsize` and `\vsiz@` to the `<width>` and `<depth>` of the planbox. The last step of the initialization (`\fillOFF`) is the decision that at the end of the planbox we don't want to fill the rest space with `\vfil` (detailed explanation will come below). Finishing the initialization we can start to process the first paragraph:

```
\def\st@rtpar{%
\advance\curplanb@xsofar \parskip
\vskip\parskip
\setbox\c@rrpar\vbox\bgroup}
```

The reason for putting `\parskip` into the vertical list by hand is that with our macros TeX sees only vboxes and not paragraphs, since we put every paragraph into `\c@rrpar`, which is the box finished in `\par`.

The normal way of processing is simply to accumulate the paragraphs. Two things may happen which can break this accumulation. The first occurs when the planbox is full, and the second when a user command is encountered. The former causes the automatic switch to be invoked (see the next section). The user commands can be divided in two classes: the first class contains the commands related to the parameters of the package. The second is formed by `\nextbox`, `\nextarea` and `\epage`. As I mentioned before, the commands of the second class can be performed only between two paragraphs, but not inside one.

Switching by hand

The commands `\nextbox` and `\nextarea` have much the same code:

```
\def\nextarea#1; { \endplanb@x
    \wlog{New area.}\bb@x#1; }
\def\nextbox{\endplanb@x
    \bb@x\the\toks\curplanb@xno; }
\def\endplanb@x{\iffill@ \vfill \fi \break}
```

Both of them should finish the current planbox (`\endplanb@x`) and start the next one (`\bb@x`). At the time `\endplanb@x` is invoked we are in vertical mode (between two paragraphs), hence `\break` causes a page break, i.e., it causes the output routine to be invoked (still see below). But before this we should decide if a `\vfill` is needed at the bottom of the current planbox. In `\bb@x` the decision is no (`\fillOFF`) but you can change it (the ultimate explanation will come soon).

To start the new planbox in `\nextbox`, `\bb@x` uses the `<next>` attribute of the planbox while `\nextarea` works with its parameter, the `<name>` of another planbox. Let me remind you that this `<name>` can be the `<name>` of *any* planbox with its successors (see above).

The third command of the second class is `\epage`:

```
\def\epage{\endplanb@x\egroup
    \box\p@gesofarbox
    \nointerlineskip
    \vskip\aboveplanskip
    \hbox{%
        \loop
            \advance\maxplanb@xno \m@ne
            \ifvoid\maxplanb@xno
```

```
\else \dimen@\wd\maxplanb@xno
    \advance\count
        \maxplanb@xno \m@ne
    \kern\dimen\count \maxplanb@xno}%
        % kern <hdis>
    \lower\dimen\count \maxplanb@xno
        \hbox{\box\maxplanb@xno}%
        % lower <vdis>
    \advance\count
        \maxplanb@xno \m@ne
    \kern-\dimen\count \maxplanb@xno}%
        % kern -<hdis>
    \kern-\dimen@
        % kern -<width>
    \fi
\ifnum\maxplanb@xno>\minplanb@xno
    \repeat\wlog{End of Page.}%
\rele@seplan}
\def\rele@seplan{%
    \global\insc@unt\maxplanb@xno
    \advance\maxplanb@xno \m@ne
    \advance\count\maxplanb@xno -4
    \global\count11\count\maxplanb@xno}
```

It has a more difficult job to do. After finishing the last planbox, `\epage` should construct the whole page, i.e., put each planbox in its place on the page. But first it leaves the group started in `\bpage` and puts back the part which was ready before the planned part. The variable `\aboveplanskip` has the same function as `\topskip` for whole pages. After putting it on the vertical list, an `\hbox` is started in order to keep the planned part together, and separately from the other material.

Inside the `\hbox` a `\loop` goes through all the planboxes. Apart from empty planboxes, placing a planbox means a horizontal kerning for `<hdis>`, a vertical kerning for `<vdis>`, putting the box at the point reached, and afterwards coming back to the origin. The horizontal kerning is done by a real `\kern` where `\hdis` comes from the indirection through the `\count` register of the planbox. Vertical kerning and placing are done by a `\lower` command. Again, `\vdis` is found with the same indirection. After performing `\lower` the ‘cursor’ of the page goes back automatically to its original place so we have to ‘undo’ only the horizontal kerning. This also includes the width of the planbox.

And finally we should release all the planboxes (`\rele@seplan`). This feature is missing from plain TeX so `\rele@seplan` has to do it explicitly. Resetting `\insc@unt` is simple because `\bpage` has stored its value in `\maxplanb@xno`. The dimen allocation register (`\count11`) was originally four less than the value of the `\count` register of the

first planbox, since this count register points to the highest of the four dimension registers related to the planbox.

The output routine. This is an interesting output routine because the major part of it does nothing else but give information:

```
\def\fullb@xoutput{%
  \global\setbox\curplanb@xno
    \vtop to \vsize{%
      \line{\hfil}\nointerlineskip
      \unvbox255}%
  \ifnum\null=\the\curplanb@xno
    \errhelp{I'll forget
      the superfluous text.}
    \errmessage{Current area is full.
      You'll lose a part of your
      text on the output}
    \wlog{There're no more boxes for
      this area, so I forget}
    \wlog{the superfluous text.
      The text needs about:}
    \setbox\null\vtop{\unvbox\null}
    \wlog{vertical : \the\dp\null,
      horizontal : \the\wd\null,}
    \wlog{or any equivalent space.}
  \else
    \wlog{Current planbox is full.}
  \fi}
```

Its task is to save the main vertical list in the box related to the planbox. The modification (`\line{\hfill}\nointerlineskip`) prevents the commands `\vfil`, `\vskip`, ... from getting lost.

And now comes the trick of `\null!` If there is not enough room in the area for the text, then the package switches from the last box to `\null`. Since `\null` has `<depth>=\maxdimen`, the rest of the text goes to `\null`. And at the end of `\null` the output routine is able to give you the information about the amount of the lost text.

Setting the parameters. After the interruption of the output routine let's go back to the first class of the user commands. The algorithm depends on three parameters, which are chosen to be exact. The simplest decision is mentioned twice above: the user should decide if he/she wants to fill out the space at the bottom of a planbox (handled with the `\newif` construction):

```
\newif\iffill@
\def\fillON{\global\fill@true}
\def\fillOFF{\global\fill@false}
```

The second parameter gives a choice about hyphenation (for exact explanation of how `\pretolerance` works see *The T_EXbook*, p. 96):

```
\def\hyphensON{\pretolerance 300 }
\def\hyphensOFF{\pretolerance 10000 }

\newtoks\aut@switch
\def\automaticON{\aut@switch={%
  \ifdim\curplanb@xsofar>\vsize
    \splitit@p \fi}}
\def\automaticOFF{\aut@switch={\relax}}
\def\pl@npar{%
  \advance\curplanb@xsofar \ht\c@rrpar
  \advance\curplanb@xsofar \dp\c@rrpar
  \the\aut@switch
  \unvbox\c@rrpar
  \afterassignment\wh@tnext\let\nextt=}
```

Both definitions is connected to `\pl@npar`. This leads us to the last part of this section:

Accumulating the paragraphs. I hope you remember that I haven't mentioned how to append the current paragraph to the current planbox. All I have written about is how 'to cut out' a paragraph and to put it into a vbox. But after this, `\pl@npar` is invoked in `\par`:

```
% from \bpage
...
\def\par{\endgraf\egroup\pl@npar}
...
```

Behaviour of `\pl@npar` depends on whether `\automaticON` or `\automaticOFF` is active. In both cases it measures the vertical size of the material in the current planbox and appends the current paragraph (`\c@rrpar`) to the vertical list. When `\automaticON` is active, `\pl@npar` checks whether the current planbox is full or not. If it is, then the code for the automatic switch (`\splitit@p`) takes place (see Automatic switch below).

On the other hand, when `\automaticOFF` is active, the material is not handled automatically. Thus the user him/herself should take care of page-setting, i.e., invoking `\nextbox` or `\nextarea` at the necessary points in the text.

You may say that no one will use `\automaticOFF` since it has only disadvantages. But this is not true. If two consecutive planboxes have the same width, then by using `\automaticOFF` the page builder of plain T_EX may be executed to find another (and perhaps better) solution for page breaking than the automatic switch of my package.

And one more thing about `\automaticOFF`: it should do all other things except checking because the user may switch ON again in the same planbox where it was switched OFF (even if there is no reason for doing so).

Let us return to `\pl@npar`. At the end it looks ahead for the next token:

```
\def\wh@tnext{\let\next\nextt
  \ifx\nextbox\next
  \else \ifx\nextarea\next
  \else \ifx\epage\next
  \else \let\next\st@rtpar
    \afterassignment\nextt
  \fi \fi \fi \next}
```

If the first token is one of `\nextbox`, `\nextarea` or `\epage` then that command should be performed, because they can be performed just outside the `vbox` containing the paragraph. As you can see, `\wh@tnext` performs at most one command before starting a new paragraph. Hence, if you want two commands to be performed, leave a blank line between them. It has just one effect: the empty line means an empty paragraph between the two commands. And if there is no command at all, a new paragraph is to be started and the token is to be put back. But this happens just after the new paragraph, i.e., the `vbox` has been started! Fortunately `\afterassignment` puts the saved token back right after starting the `vbox`.

Again, we are at the beginning of a new paragraph.

Automatic switch

Let's pick up the thread at `\splitit@p`:

```
\def\splitit@p{%
  \m@veextra
  \unvbox\c@rrpar \endplanb@x
  \initb@x\the\toks\curplanb@xno;
  \r@typeset
  \global\curplanb@xsofar=\ht\c@rrpar
  \global\advance\curplanb@xsofar
    \dp\c@rrpar
  \ifdim\curplanb@xsofar>\vsize
    \let\next\splitit@p
  \else \let\next\relax \fi \next}
```

This macro is invoked when the current paragraph has no room in the current planbox. First `\m@veextra` reduces the vertical size of the paragraph to the appropriate size by removing the last lines of it. Then the remaining part is appended to the current planbox. This planbox is finished and a new one is initialized (not started!). The removed part of the paragraph is then `retypeset` with the new `\hsize`. If this amount of material is too much for this planbox, then the whole process is repeated.

The macro `\m@veextra` removes the necessary lines one by one with `\rem@velastline`:

```
\def\m@veextra{%
  \global\setbox\c@rrpar\vbox{%
    \unvbox\c@rrpar \rem@velastline}%
  \global\setbox\extrat@xt\vbox{%
    \unvbox\extrat@xt\box\l@stline}%
  \ifdim\curplanb@xsofar>\vsize
    \let\next\m@veextra
  \else \let\next\relax \fi \next}
```

It also accumulates these lines in `\extrat@xt`, and it goes on until the vertical size of the material is less than or equal to `\vsize`. One single line is removed by `\rem@velastline`:

```
\def\rem@velastline{%
  \global\setbox\l@stline\lastbox
  \ifvoid\l@stline
    \global\advance\curplanb@xsofar
      -\lastskip \unskip
    \unpenalty
    \global\advance\curplanb@xsofar
      -\lastkern \unkern
    \let\next\rem@velastline
  \else
    \global\advance\curplanb@xsofar
      -\ht\l@stline
    \global\advance\curplanb@xsofar
      -\dp\l@stline
    \let\next\relax
  \fi \next}
```

The macro works with `TeX`'s `\lastbox` and `\un...` operations. If a box could be removed, the macro returns it in `\l@stline`. Otherwise `\rem@velastline` tries to remove the last item in the vertical list and updates `\curplanb@xsofar`. Unfortunately there is no proper `\un...` command for each type of item, but the commands for the missing types ("whatsit", mark, insertion) are mode-independent, so in general you can avoid their being appended to the vertical list. (I hope this feature of `TeX` won't cause too much trouble for you and for the package.) Moreover there is no opportunity to check whether the last item is glue or not, because there is no `\ifskip` command to distinguish `\z@skip=0pt plus 0pt minus 0pt` from let's say `\parskip=0pt plus 1pt`. Thus brute force is used instead of checking the last item with `\if...` operations.

Notice that `\extrat@xt` contains only lines of text, i.e., neither glue items nor kerns nor penalties, and the lines are placed in reverse order. They will be reversed again in `\r@typeset`:

```
\def\r@typeset{%
  \global\setbox\extrat@xt\vbox{%
    \unvbox\extrat@xt
  \global\setbox\n@xtline\lastbox}%
}
```

```

\setbox\n@xtline\hbox{\unhbox\n@xtline
    \unskip}%
\global\setbox\c@rrpar\vbox{\noindent
    \unhbox\n@xtline}%
{\parskip\z@skip
\loop
\global\setbox\extrat@xt\vbox{%
    \unvbox\extrat@xt
    \global\setbox\n@xtline\lastbox}%
\global\setbox\c@rrpar\vbox{%
    \unvbox\c@rrpar \rem@velastline
    \openhbox\l@stline
    \setbox\n@xtline\hbox{\unhbox\n@xtline
        \unskip}%
    \noindent \unhbox\l@stline\ %
    \unhbox\n@xtline}%
\ifdim\ht\extrat@xt>\z@ {\repeat}%
\def\openhbox#1{\setbox#1 \hbox{\unhbox#1%
    \unskip \unskip \unpenalty}}%

```

Before examining the code, let's go through the idea. It seems to be simple: join the lines again and let the line breaking algorithm form the new paragraph. Unfortunately the task is more difficult. The main problem is that because of the different `\hsize`, the breakpoints in the new paragraph will be at other points than they were in the original paragraph. The line breaking algorithm puts `\rightskip` at the end of every line. So `\rightskip` is to be removed from the original ends of lines.

At the end of a line a hyphenation may occur, too. The trade-off is to check every line end with a number of complicated macros or to leave the task of correcting the bad hyphenations to the user. Because of the easy correction I decided to use the latter option.

On the other hand, joining the lines means that the first couple of lines form the beginning of the new paragraph and the next line is joined to the last line of the partial paragraph. And the last line of a paragraph contains not just `\rightskip` but three more items related to `\par`, namely, `\penalty10000`, `\hskip\parfillskip` and `\penalty-10000` (*The TeXbook*, p. 100). The third item is discarded at the line break but the two other items also should be removed before the join. The macro `\openhbox` removes all three items from its parameter box.

Last but not least, there is no space between the last word of a particular line and the first word of the next line. Hence we should put a space before each line except the first one. The first line differs from the others in another respect: We don't need to apply the joining algorithm just discussed because there is no last line of the empty paragraph.

Let's go back to the code! In the code up to the `\f` before the `\parskip\z@skip` command the first line is retypereset in three steps. First the line is removed from `\extrat@xt` (there is no need to use `\rem@velastline` because `\extrat@xt` only contains the lines). Then `\rightskip` is removed. Finally, the line is put into `\c@rrpar`. Because `\c@rrpar` now contains not a real paragraph but just the second part of it, `\noindent` is inserted before the line.

The other lines will be appended according to the algorithm. First the line is removed from `\extrat@xt` the same way as the first line. Then the last line of the paragraph is removed with `\rem@velastline`. Both lines are 'peeled' and then joined with a space between them. We should insert `\noindent` before `\l@stline` because this line may be the first in the paragraph; so the paragraph may be restarted at this point. This is the reason for setting `\parskip` to `\z@`. When a paragraph starts `\parskip` is automatically inserted. In our case the paragraph may start again but we don't need another `\parskip`.

The whole action is repeated until all the lines have been joined together.

And in the end...

This package is developed to handle pure text. It was created in such a way that it avoids interfering with `\plain TeX` whenever possible. On the other hand the package has to change things deep inside `TeX`. I'm sure these changes mean restrictions of the usage but only experiments can discover all of them. However, I think that independently from the restrictions the package is useful and helps to create documents with a better look.

Already at the moment there is one possible improvement: the multipage version of the package. This needs only technical, and not fundamental changes, but it gives the possibility of making whole newspapers. I hope sooner or later that version will come out.

My only reference was *The TeXbook*. If you find my article does not explain a notion you will find the best possible answer in this book.

At last: I hope you will enjoy this 'pagemaker'.

◊ Péter Huszár
Budapest
Bogdánffy út 10.b.
H-1117 Hungary
h1612hus@ella.hu

The Elementary Particle Entity Notation (PEN) Scheme

Michel Goossens and Eric van Herwijnen

Abstract

In this article an Elementary Particle Entity Notation (PEN) scheme is proposed for use with *TEX* and SGML. This scheme not only assures the typographic correctness of the printed symbols, but also eases the automatic extraction of information about the article by the recognition of the entity names.

1 Typographical rules for scientific texts

In scientific texts the printed form of a symbol often implies a meaning which is not easily captured by generic markup. Therefore authors using some form of generic coding (like *LATEX* or SGML) need to know about typographical conventions. The following is a brief summary of the most important rules for composing scientific texts [1, 2].

1. The most important rule is **consistency**: a symbol should always be the same, whether it appears in a formula or in the text, on the main line or as a superscript or subscript. That is, in *TEX*, once you have used a symbol inside mathematics mode ('\$'), always use it inside mathematics mode. Inside math mode, *TEX* by default prints characters in *italics*.

For scientific work, however, quite a few symbols must be set in **roman** (upright) characters¹. This is the case for the following families of symbols, which represent the names of:

- units, such as g, cm, s, keV. Note that physical constants are usually in *italics*, so units involving constants are mixed roman-*italics*, e.g. GeV/c (where the *c* is italic because it symbolizes the speed of light, a constant);
- particles, for example p, K, q, H. For elementary particles the PEN (Particle Entity Notation) scheme is proposed (see the next section);
- standard mathematical functions (sin, det, cos, tan, Re, Im, etc.). Use the built-in *TEX* functions for these (*\sin*, etc.);
- chemical elements, for example Ne, O, Cu;
- numbers;
- names of waves or states (p-wave) and covariant couplings (A for axial, V for vec-

¹ With *LATEX* roman type in maths mode can be achieved by the *\mbox* or *\mathrm* commands.

tor), names of monopoles (E for electric, M for magnetic);

- abbreviations that are initials or bits of words (exp, for experimental; min, for minimum);
- the 'd' in integrands (e.g. *dp*).

In all cases, following these rules will help the reader understand at first glance what one is talking about. Some instances in which it is important to use the correct symbol, in the correct type, are shown in Table 1.

2. Let your word processor do as much work as it can. Do not try to change your system's defaults too much; this will decrease the portability and maintainability of your documents. *TEX* implements a lot of the rules mentioned above by default in math mode.
3. Do not add blanks at random to make formulae look "nicer".
4. Refrain from using specific page layout commands (like *\break* with *TEX*). You will forget that you put them in your text and later wonder why some text is badly adjusted or starts a new line.

2 Entity definitions for elementary particles

In texts on high energy physics frequently reoccurring strings are the names of elementary particles. For example, the Z^0 particle can be coded in various different ways with *LATEX*: *\mbox{Z}^0*, *\mathrm{Z}^0* and *Z^0* all achieve the same typographical effect, a roman Z with a superscript 0. In the interest of standardization and typing convenience, we propose below an "entity" naming scheme, which will not only relieve the user from having to worry about the correctness of what he types, but also will allow an automatic extraction of the particle names from the input file, so that it will be easy to enter data about an article using this convention into a database of abstracts.

The naming scheme uses a notation which takes the following constraints into consideration:

1. The notation should be able to describe all particles in the particle data summary tables from the "Review of Particle Properties" [3] and any future extension to these.
2. The names should not exceed eight characters. This is the maximum length for entities in the SGML reference concrete syntax [4]. Staying within this limit means that the notation can be used with most SGML applications.

roman type	italic type
A ampere (electric unit)	<i>A</i> atomic number (variable)
e electron (particle name)	<i>e</i> electron charge (constant)
g gluon (particle name)	<i>g</i> gravitational constant
l litre (volume unit)	<i>l</i> length (variable)
m metre (length unit)	<i>m</i> mass (variable)
p proton (particle name)	<i>p</i> momentum (variable)
q quark (particle name)	<i>q</i> electric charge (variable)
s second (time unit)	<i>s</i> c.m. energy squared (variable)
t tonne (weight unit)	<i>t</i> time (variable)
V volt (electric unit)	<i>V</i> volume (variable)
Z Z boson (particle name)	<i>Z</i> atomic charge (variable)

Table 1: Example of differences in meaning of a symbol depending on the type.

3. Common particles such as protons and electrons should have short and simple names.
4. Items that are indicated by superscripts are indicated before items that are indicated by subscripts.

Due to the eight character limitation the mass could not be added to the name. This means that in general an entity on its own is not adequate to unambiguously identify a particle, cf. $\eta(549)$ and $\eta(1300)$ are both referred to as Pgh. Including mass dependences into the names is not a good idea anyway, since the mass can change with time when more precise measurements become available. The ambiguity was solved by adding a letter to the end of the name where a mass appears in the name in the particle data summary tables. Thus $\eta(549)$ is referred to as Pgh while $\eta(1300)$ is referred to as Pgħa. Higher letters correspond to higher masses, in the order given in the tables.

The PEN scheme is independent of any text processing system. We have implemented it in TeX (in such a way that it may be used in all macro packages, e.g. LATEX) and SGML. The TeX implementation will print particle masses, which will be regularly updated according to the Review of Particle Properties publication. It is constructed so that the PEN name can be used in both mathematics and text mode.

2.1 Principles of the Particle Entity Notation (PEN)

Starting at the left, a name is built from the following characters:

1. Start the entity with a recognized string (in the following this was chosen as uppercase P). This is necessary to uniquely identify entities as following the PEN convention.

2. The following letters act as an escape to signal a special interpretation of the string. Present escape sequences are:

- a for anti particle (normally represented visually with a bar over the particle's name)

- b for bottom particle

- c for charmed particle

- g for indicating the subsequent letter is Greek. The correspondence between Latin and Greek letters is based on the notation for mathematical Greek characters used by the AAP mathematical formula application [5]:

```
<!NOTATION greek2 PUBLIC "+//ISBN
1-880124::NISO//NOTATION GREEK-2//EN">
```

This one-letter correspondence is shown in Table 2.

- q for quark particle

- s for strange particle

- S for supersymmetric particle

- t for top particle

3. The one-letter name of the particle

4. Optionally followed by other information

- z for zero, i for one, ii for two, iii for three, iv for four

- m for minus, p for plus, pm for plus/minus

- pr for prime

- st for asterisk (star)

- L for left-handed, R for right-handed

- any one-letter particle name

Greek name	code	Greek name	code
α	alpha	a	A
β	beta	b	B
γ	gamma	g	Γ
δ	delta	d	Δ
ϵ	epsilon	e	E
ζ	zeta	z	Z
η	eta	h	H
θ	theta	q	Θ
ι	iota	i	I
κ	kappa	k	K
λ	lambda	l	Λ
μ	mu	m	M
ν	nu	n	N
ξ	xi	x	Ξ
\o	omicron	o	O
π	pi	p	Π
ρ	rho	r	P
σ	sigma	s	Σ
τ	tau	t	T
υ	upsilon	u	Υ
ϕ	phi	f	Φ
χ	chi	c	X
ψ	psi	y	Ψ
ω	omega	w	Ω

Table 2: The AAP codes for the Greek letters.

2.2 Particle encodings according to the PEN Scheme

In table 3 we show how to encode the particles from the summary tables of particle properties in the “Review of Particle Properties” [3] using the PEN convention. In the rightmost column we give the computer name of the particle, as defined by “A Guide to Experimental Elementary Particle Physics Literature (1985–1989)” [6]. This is the name to be used when searching the Particle Data Group’s databases. Notice that these names cannot be used for either TeX or SGML, as they do not satisfy the constraints of the PEN scheme as defined above. When a name is marked as “not available”, sometimes a charged or neutral version exists (not given in the table).

The TeX implementation is available as a style file `pennames.sty`, which should be input in the usual way at the start of the document for TeX or specified as a minor option on the `\documentstyle` command for LATEX. To obtain the symbol required, prefix the PEN name by a backslash (\).

The SGML implementation exists as a public entity set, that can be included in SGML documents with the following entity definition:

```
<!ENTITY % PEN PUBLIC
  "+//ISBN 92-9083-041-7::CERN//ENTITIES
  Particle Entity Names//EN">
```

Refer to a particle entity by prefixing its name by an ampersand (&) and suffixing it with a semi-colon (;), e.g. &Pgr; would give ρ (770).

3 How to get the files

A file `pennames.sty` with the TeX particle name definitions, `pennames.entities` with the SGML entity names, and `pennames.ps` containing the PostScript source of this document, are available via anonymous ftp as follows (commands to be typed by the user are underlined):

```
ftp cernvm.cern.ch
Trying 128.141.2.4...
220-FTPIBM at cernvm.CERN.CH...
Name (cernvm:goossens): anonymous
230 ANONYMOU logged in with no special a...
Remote system type is VM.
ftp> cd tex.802
250 Working directory is TEX 802 (ReadOnly)
ftp> get pennames.sty
ftp> get pennames.entities
ftp> get pennames.ps
ftp> quit
```

References

- [1] International Union of pure and applied Physics. *Symbols, Units, Nomenclature and Fundamental Constants in Physics*. Physica, 146A:1–67, 1987.
- [2] D.E. Lowe. *A Guide to international recommendations on names and symbols for quantities and on units of measurements*. World Health Organization, Geneva, 1975.
- [3] Particle Data Group. *Review of particle properties*. Physics Letters B, 239:1–516, April 1990.
- [4] E. van Herwijnen. *Practical SGML*. Wolters-Kluwer Academic Publishers, Boston, 1990.
- [5] American National Standards Institute. *American National Standard for Electronic Manuscript Preparation and Markup*. ANSI/NISO Z39.59-1988, 1988.
- [6] Particle Data Group. *A Guide to Experimental Elementary Particle Physics Literature (1985–1989)*. Lawrence Berkeley Laboratory, LBL-90 Revised, UC-414, November 1990.

◊ Michel Goossens
 Eric van Herwijnen
 CERN, CH-1211 Geneva 23,
 Switzerland
 goossens@cernvm.cern.ch
 eric@cernvm.cern.ch

Table 3: PEN names for elementary particles in PDG list

PEN	symbol	conventional name	computer name
Gauge and Higgs bosons			
Pgg	γ	gamma	GAMMA
PW	W	W boson	W
PWp	W^+	W plus	W^+
PWm	W^-	W minus	W^-
PZz	Z^0	Z zero	Z
PHz	H^0	Higgs zero	not available
PHpm	H^\pm	Higgs plus/minus	HIGGS+-
PWR	W_R	right-handed W	not available
PWpr	W'	W prime	WPRIME
PZLR	Z_{LR}	left-right handed Z	not available
PZgc	Z_χ	Z chi	not available
PZgy	Z_ψ	Z psi	not available
PZge	Z_η	Z eta	not available
PZi	Z_1	Z one	not available
PAz	A^0	axion	AXION
Leptons			
Pgne	ν_e	electron neutrino	NUE
Pagne	$\bar{\nu}_e$	anti electron neutrino	NUEBAR
Pgnhm	ν_μ	muon neutrino	NUMU
Pagngm	$\bar{\nu}_\mu$	anti muon neutrino	NUMUBAR
Pgngt	ν_τ	tau neutrino	NUTAU
Pagngt	$\bar{\nu}_\tau$	anti tau neutrino	not available
Pe	e	electron	not available
Pep	e^+	positron	E+
Pem	e^-	e minus	E-
Pgm	μ	muon	not available
Pgmm	μ^-	mu minus	MU-
Pgmp	μ^+	mu plus	MU+
Pgt	τ	tau	not available
PLpm	L^\pm	charged lepton	LEPTON+-
PLz	L^0	stable neutral heavy lepton	not available
PEz	E^0	neutral para- or ortho-lepton	not available
Light Unflavored Mesons (S=C=B=0)			
Pgp	π	pion	PI
Pgpm	π^-	pi minus	PI-
Pgpp	π^+	pi plus	PI+
Pgppm	π^\pm	pi plus/minus	PI+-
Pgpz	π^0	pi zero	PIO
Pgh	η	eta	ETA
Pgr	$\rho(770)$	rho	RHO(770)
Pgo	$\omega(783)$	omega	OMEGA(783)
Pghpr	$\eta'(958)$	eta prime	ETAPRIME(958)
Pfz	$f_0(975)$	f zero	F0(975)
Paz	$a_0(980)$	a zero	A0(980)
Pgf	$\phi(1020)$	phi	PHI(1020)
Phia	$h_1(1170)$	h one	H1(1170)
Pbi	$b_1(1235)$	b one	not available
Pai	$a_1(1260)$	a one	A1(1260)
Pfii	$f_2(1270)$	f two	F2(1270)
Pfi	$f_1(1285)$	f one	F1(1285)
Pgha	$\eta(1295)$	eta 1295	ETA(1295)
Pgpa	$\pi(1300)$	pion 1300	not available
Paii	$a_2(1320)$	a two	A2(1320)
Pgoa	$\omega(1390)$	omega 1390	not available
Pfza	$f_0(1400)$	f zero 1400	F0(1400)
Pfia	$f_1(1390)$	f one 1420	F1(1420)

Table 3: PEN names (continued)

PEN	symbol	conventional name	computer name
Pghb	$\eta(1440)$	eta 1440	ETA(1440)
Pgra	$\rho(1450)$	rho 1450	not available
Pfib	$f_1(1510)$	f one 1510	F1(1510)
Pfiipr	$f'_2(1525)$	f two prime	F2PRIME(1525)
Pfzb	$f_0(1590)$	f zero 1590	F0(1590)
Pgob	$\omega(1600)$	omega 1600	not available
Pgoiii	$\omega_3(1670)$	omega three	OMEGA3(1670)
Pgpii	$\pi_2(1670)$	pi two	PI2(1670)
Pgfa	$\phi(1680)$	phi 1680	PHI(1680)
Pgriii	$\rho_3(1690)$	rho three	not available
Pgrb	$\rho(1700)$	rho 1700	RHO(1700)
Pfia	$f_2(1720)$	f two 1720	F2(1720)
Pgfiii	$\phi_3(1850)$	phi three	PHI3(1850)
Pfiib	$f_2(2010)$	f two 2010	F2(2010)
Pfiv	$f_4(2050)$	f four	F4(2050)
Pfiic	$f_2(2300)$	f two 2300	F2(2300)
Pfiid	$f_2(2340)$	f two 2340	F2(2340)
Strange Mesons (S=±1, C=B=0)			
PK	K	kaon	K
PKpm	K^\bullet	K plus/minus	K+-
PKp	K^+	K plus	K+
PKm	K^-	K minus	K-
PKz	K^0	K zero	K0
PaKz	\bar{K}^0	anti K-zero	KBAR0
PKgmiii	$K_{\mu 3}$	K mu three	not available
PKeiii	$K_{e 3}$	K e three	not available
PKzS	K_S^0	K zero short	not available
PKzL	K_L^0	K zero long	not available
PKzgmiil	$K_{\mu 3}^0$	K zero mu three	not available
PKzeiii	$K_{e 3}^0$	K zero e three	not available
PKst	$K^*(892)$	K star	not available
PKi	$K_1(1270)$	K one	K1(1270)
PKsta	$K^*(1370)$	K star (1370)	not available
PKia	$K_1(1400)$	K one (1400)	not available
PKstz	$K_0^*(1430)$	K star zero (1430)	not available
PKstii	$K_2^*(1430)$	K star two (1430)	not available
PKstb	$K^*(1680)$	K star (1680)	not available
PKii	$K_2(1770)$	K two (1770)	not available
PKstiii	$K_3^*(1780)$	K star three	not available
PKstiv	$K_4^*(2045)$	K star four	not available
Charmed Mesons (C=±1)			
PDpm	D^\pm	D plus/minus	D+-
PDm	D^-	D minus	D-
PDp	D^+	D plus	D+
PDz	D^0	D zero	D0
PaDz	\bar{D}^0	anti D zero	DBAR0
PDstpm	$D^*(2010)^\pm$	D star plus/minus	$D^*(2010)^{+-}$
PDstz	$D^*(2010)^0$	D star zero	$D^*(2010)0$
PDiz	$D_1(2420)^0$	D one zero	$D_1(2420)0$
PDstiiz	$D_2^*(2460)^0$	D star two zero	$D_2^*(2460)0$
Charmed Strange Mesons (C=S=±1)			
PsDp	D_s^+	D s plus	D/S+
PsDm	D_s^-	D s minus	D/S-
PsDst	D_s^*	D s star	D/S*
PsDipm	$D_{s1}(2536)^\pm$	D s one plus/minus	not available
Bottom Mesons (B=±1)			
PB	B	B	B

Table 3: PEN names (continued)

PEN	symbol	conventional name	computer name
PBp	B ⁺	B plus	B+
PBm	B ⁻	B minus	B-
PBpm	B [±]	B plus/minus	B+-
PBz	B ⁰	B zero	B0
Pcgh	$\eta_c(1S)$	eta c	ETA/C(1S)
PJgy	J/ $\psi(1S)$	J psi	J/PSI(1S)
Pcgcz	$\chi_{c0}(1P)$	chi c zero	CHI/CO(1P)
Pcgci	$\chi_{c1}(1P)$	chi c one	CHI/C1(1P)
Pcgci	$\chi_{c2}(1P)$	chi c two	CHI/C2(1P)
Pgy	$\psi(2S)$	psi	PSI(2S)
Pgya	$\psi(3770)$	psi 3770	PSI(3770)
Pgyb	$\psi(4040)$	psi 4040	PSI(4040)
Pgyc	$\psi(4160)$	psi 4160	PSI(4160)
Pgyd	$\psi(4415)$	psi 4415	PSI(4415)
PgU	$\Upsilon(1S)$	Upsilon	not available
Pbgcz	$\chi_{b0}(1P)$	chi b zero	CHI/B0(1P)
Pbgci	$\chi_{b1}(1P)$	chi b one	CHI/B1(1P)
Pbgcii	$\chi_{b2}(1P)$	chi b two	CHI/B2(1P)
PgUa	$\Upsilon(2S)$	Upsilon (2S)	UPSI(2S)
Pbgcza	$\chi_{b0}(2P)$	chi b zero (2P)	CHI/B0(2P)
Pbgcia	$\chi_{b1}(2P)$	chi b one (2P)	CHI/B1(2P)
Pbgciia	$\chi_{b2}(2P)$	chi b two (2P)	CHI/B2(2P)
PgUb	$\Upsilon(3S)$	Upsilon (3S)	UPSI(3S)
PgUc	$\Upsilon(4S)$	Upsilon (4S)	UPSI(4S)
PgUd	$\Upsilon(10860)$	Upsilon (10860)	UPSI(10860)
PgUe	$\Upsilon(11020)$	Upsilon (11020)	UPSI(11020)
N Baryons (S=0, I=1/2)			
Pp	p	proton	p
Pn	n	neutron	n
PNa	N(1440)P ₁₁	N (1440) P 11	N(1440P11)
PNb	N(1520)D ₁₃	N (1520) D 13	not available
PNc	N(1535)S ₁₁	N (1535) S 11	not available
PNd	N(1650)S ₁₁	N (1650) S 11	not available
PNe	N(1675)D ₁₅	N (1675) D 15	not available
PNf	N(1680)F ₁₅	N (1680) F 15	not available
PNg	N(1700)D ₁₃	N (1700) D 13	not available
PNh	N(1710)P ₁₁	N (1710) P 11	not available
PNi	N(1720)P ₁₃	N (1720) P 13	not available
PNj	N(2190)G ₁₇	N (2190) G 17	not available
PNk	N(2220)H ₁₉	N (2220) H 19	not available
PNl	N(2250)G ₁₉	N (2250) G 19	not available
PNm	N(2600)I _{1,11}	N (2600) I 1,11	not available
Δ Baryons (S=0, I=3/2)			
PgDa	$\Delta(1232)P_{33}$	Delta (1232) P 33	DELTA(1232P33)
PgDb	$\Delta(1620)S_{31}$	Delta (1620) S 31	not available
PgDc	$\Delta(1700)D_{33}$	Delta (1700) D 33	not available
PgDd	$\Delta(1900)S_{31}$	Delta (1900) S 31	not available
PgDe	$\Delta(1905)F_{35}$	Delta (1905) F 35	not available
PgDf	$\Delta(1910)P_{31}$	Delta (1910) P 31	not available
PgDh	$\Delta(1920)P_{33}$	Delta (1920) P 33	not available
PgDi	$\Delta(1930)D_{35}$	Delta (1930) D 35	not available
PgDj	$\Delta(1950)F_{37}$	Delta (1950) F 37	not available
PgDk	$\Delta(2420)H_{3,11}$	Delta (2420) H 3,11	not available
Λ Baryons (S=-1, I=0)			
PgL	Λ	Lambda	LAMBDA
PgLa	$\Lambda(1405)S_{01}$	Lambda (1405) S 01	LAMBDA(1405S01)
PgLb	$\Lambda(1520)D_{03}$	Lambda (1520) D 03	LAMBDA(1520D03)
PgLc	$\Lambda(1600)P_{01}$	Lambda (1600) P 01	not available

Table 3: PEN names (continued)

PEN	symbol	conventional name	computer name
PgLd	$\Lambda(1670)S_{01}$	Lambda (1670) S 01	not available
PgLe	$\Lambda(1690)D_{03}$	Lambda (1690) D 03	not available
PgLf	$\Lambda(1800)S_{01}$	Lambda (1800) S 01	not available
PgLg	$\Lambda(1810)P_{01}$	Lambda (1810) P 01	not available
PgLh	$\Lambda(1820)F_{05}$	Lambda (1820) F 05	not available
PgLi	$\Lambda(1830)D_{05}$	Lambda (1830) D 05	not available
PgLj	$\Lambda(1890)P_{03}$	Lambda (1890) P 03	not available
PgLk	$\Lambda(2100)G_{07}$	Lambda (2100) G 07	not available
PgLl	$\Lambda(2110)F_{05}$	Lambda (2110) F 05	not available
PgLm	$\Lambda(2350)H_{09}$	Lambda (2350) H 09	not available
Σ Baryons ($S=-1, I=1$)			
PgSp	Σ^+	Sigma plus	SIGMA+
PgSz	Σ^0	Sigma zero	SIGMA0
PgSm	Σ^-	Sigma minus	SIGMA-
PgSa	$\Sigma(1385)P_{13}$	Sigma (1385) P 13	not available
PgSb	$\Sigma(1660)P_{11}$	Sigma (1660) P 11	not available
PgSc	$\Sigma(1670)D_{13}$	Sigma (1670) D 13	not available
PgSd	$\Sigma(1750)S_{11}$	Sigma (1750) S 11	not available
PgSe	$\Sigma(1775)D_{15}$	Sigma (1775) D 15	not available
PgSf	$\Sigma(1915)F_{15}$	Sigma (1915) F 15	not available
PgSg	$\Sigma(1940)D_{13}$	Sigma (1940) D 13	not available
PgSh	$\Sigma(2030)F_{17}$	Sigma (2030) F 17	not available
PgSi	$\Sigma(2050)$	Sigma (2250)	not available
Ξ Baryons ($S=-2, I=1/2$)			
PgXz	Ξ^0	Xi zero	XIO
PgXm	Ξ^-	Xi minus	XI-
PgXa	$\Xi(1530)P_{13}$	Xi (1530) P 13	not available
PgXb	$\Xi(1690)$	Xi (1690)	not available
PgXc	$\Xi(1820)D_{13}$	Xi (1820) D 13	not available
PgXd	$\Xi(1950)$	Xi (1950)	not available
PgXe	$\Xi(2030)$	Xi (2030)	not available
Ω Baryons ($S=-3, I=0$)			
PgOm	Ω^-	Omega minus	OMEGA-
PgOma	$\Omega(2250)^-$	Omega (2250) minus	OMEGA(2250)-
Charmed Baryons ($C=+1$)			
PcgLp	Λ_c^+	charmed Lambda plus	LAMBDA/C+
PcgXz	Ξ_c^0	charmed Xi zero	not available
PcgXp	Ξ_c^+	charmed Xi plus	not available
PcgS	$\Sigma_c(2455)$	charmed Sigma 2455	not available
Supersymmetric Particles			
PSgg	$\tilde{\gamma}$	photino	PHOTINO
PSgxz	$\tilde{\chi}_1^0$	neutralino	NEUTRALINO
PSZZ	\tilde{Z}^0	supersymmetric Z zero	ZINO
PSHz	\tilde{H}_1^0	Higgsino	HIGGSINO
PSgxp	$\tilde{\chi}_{\pm i}^{\pm}$	chargino	CHARGINO
PSWpm	\tilde{W}^{\pm}	supersymmetric W plus/minus	not available
PSHpm	$\tilde{H}^{\pm j}$	charged Higgsino	not available
PSgn	$\tilde{\nu}$	scalar neutrino	not available
PSe	\tilde{e}	scalar electron	not available
PSgm	$\tilde{\mu}$	scalar muon	not available
PSgt	$\tilde{\tau}$	scalar tau	not available
PSq	\tilde{q}	scalar quark	not available
PSg	\tilde{g}	gluino	GLUINO

LATEX

From TeX to LATEX

Maria Luisa Luvisetto and Enzo Ugolini

1 Introduction

Our Institute is a very old TeX site (since 1982) and users have basic or good knowledge of TeX, but would also like to use LATEX without the need of reading manuals and documentation. For such users we have prepared a fast reference with guidelines for article setup (title, authors, page numbers, etc.), page setup (section, subsections), font selection, mathematics, tabular information (item, subitem, tables, etc.), index and bibliographic reference, comparing TeX with LATEX commands. Tools are provided to help in editing the document and inserting complex elements, such as tables.

2 LATEX Syntax

LATEX defines the *logical design* of a document and provides *environments* for title, author, abstract, and the like. Environments are structures starting with \begin and ending with the corresponding \end statement. Each document unit is enclosed in a structure, the whole document is delimited by \begin{document} and \end{document}, where begin and end have the same function as {...} in TeX. A LATEX input looks like the following:

```
\documentstyle[12pt]{article}
%
% preamble section
% add other options in [...]
% add size and paging options here, if any
% add definitions
%
\title{Any Title}
\author{Any N. Author}
%
\begin{document}
% document structure init
\maketitle
% produce title from definition
%
\begin{abstract}
... abstract text ...
\end{abstract}
%
\section{First}
... section text ...
%
...
...
```

```
% \section{Last}
... section text ...
%
\begin{thebibliography}
\bibitem{bib:one} ... bib text ...
...
\bibitem{bib:end} ... bib text ...
\end{thebibliography}
%
\tableofcontents
%
\end{document}
```

We will limit our description to *articles* and to basic typesetting, but LATEX can refer to any other style your local installation supports. In general, each document is made at least of a style definition command (preamble), the document structure and some text (the document body), an optional abstract, a few sections, reference information and a table of contents, with reference and index at the end of the document. Macro definitions and style changes are declared in the preamble.

LATEX changes font size for titles and authors, automatically centers both title and authors, adds document date, numbers sections and tabular information, etc. For articles, the sectioning commands are: \section \subsection \subsubsection \appendix

The input format is similar to TeX, with the escape character \ and the same special character set (# \$ % & _ ^ \ { }). Basic TeX macros are common to LATEX. Obviously structures (\begin ... \end) and braces ({...}) must be balanced. Environments can have optional parameters that are enclosed in brackets. Options must be specified immediately after the environment call, multiple options are separated by commas, no blank is allowed inside the brackets.

As in TeX, the document can be split into multiple input files; these are included unconditionally using \input{file-name} or conditionally using \include{file-name} to include **only** the files named in the preamble through the command:

```
\includeonly{file-1,file-2,file-3...}
```

If the preamble does not contain an \includeonly command all files are included. If \includeonly has an empty argument list, no file is included.

3 Fonts

Font selection in LATEX is almost the same as in TeX. The default active font is *roman*, the default inactive font is *italic*. Font size is selected at document level; the default is 10 pt.

Furthermore, LATEX provides a useful tool to emphasize text elements, the \em environment. The

\em command switches the default font from the active one to the inactive; thus if the current font is **roman**, the emphasized text is printed in *italic*, and vice versa. For entire sentences or paragraphs, the emphasized mode is declared as a structure, i.e. \begin{em} ... \end{em}. As a consequence of the switching feature, in a long emphasized text typeset in italic, a shorter fragment can be emphasized in roman, as in the following example:

A long emphasized text can include emphasized strings written in roman, inside an italic sentence.

The above fragment is produced by the following source code:

```
\begin{em} A long emphasized text can
include {\em emphasized strings} written
in {\em roman}, inside an italic
sentence.\end{em}
```

Other predefined fonts are:

bf	bold	Bold font
sf	sans serif	Sans Serif font
sl	slanted	Slanted font
sc	small caps	SMALL CAPS FONT
tt	type writer	Typewriter font

Fonts are declared as in TeX (i.e. {\bf text}). Accents and symbols are typeset as in TeX. Other fonts are defined for mathematical use, like greek letters (limited set), calligraphic ones (only uppercase), plus the mathematical italic (\mit) that is the default type for maths and mathematical bold (\boldmath) fonts.

In technical manuals, especially when reporting computer programs, it is required not only to use a non-proportional font as \tt, but also to reproduce the text as it stands, including producing a mark such as \ for required spaces. For this purpose LATEX has four commands:

```
\begin{verbatim} ... \end{verbatim},
\begin{verbatim*} ... \end{verbatim*},
\verb, \verb*
```

The * commands typeset \ for blanks. The **verbatim** environment typesets the text on a new line; thus it is used for long insertions. The \verb command is used for short strings inside the current line. The text is delimited by any pair of identical characters such as ! as in the following example:

To display blanks in computer programs type \verb!int l,k;! , the typeset result will be int\l,k;*

Font size can be changed with the commands \tiny or \small for smaller fonts, or \Large or \Huge for bigger fonts, followed by the font specification if different from roman, so we have \large\bf

for large bold letters. Examples of the \tiny, \Huge and \large\bf follow:

Font Font

Users can define other fonts not provided in the default set, as in TeX, with the command:

```
\newfont{\symbf}{cmsy10 scaled\magstep1}
```

where \symbf is name of the new font that is available in the cmsy10 font description file in an enlarged size. To typeset some text in the new font, just call it as any other font: {\symbf \symbol{26}} to write the symbol with character code 26 (i.e. C).

4 Notes

LATEX provides two types of notes: *footnotes* and *marginal notes*. The syntax for footnotes is similar but not identical to TeX. In TeX footnote numbering is required (number and text enclosed in braces) and number generation is not handled, thus the user must take care of footnote numbering.

In LATEX the number is a positive integer automatically stepped for the next footnote command. The numbering can be changed at user's will as an option. The syntax is: \footnote [num]{text} where num is the optional number for the following footnote text.

Marginal notes are not numbered and are placed in the free paper margin with the first line even with the line of text in which the note is inserted, as happens here. The note is placed according to the style in use: it is placed on the right for one-sided documents, on the outside margin for two-sided printing, in the nearest margin for multi-column style.

The marginal note is produced by \marginpar and different text can be produced for left and right margin as an option. The syntax is:

```
\marginpar[left_text]{right_text}
```

5 Item Lists

LATEX has extensive capabilities to handle tabular information, both in the form of tables (see Section 6) and in the form of aligned text.

LATEX defines a set of structures to format lists of items or quotations. The structures are: **quote**, **quotation**, **itemize**, **enumerate**, **description**. Inside the structure, each entry is declared through the \item command.

The quote environment is used for short quotations, the quotation environment for longer ones. The "quoted" text is indented, as shown here.

LATEX is able to handle quotations.

Each quotation is indented.

note

The above example is produced by:

```
\begin{quote}
\LaTeX{} is able to handle quotations.
```

Each quotation is indented.

```
\end{quote}
```

More useful in technical documents are the `itemize` and `enumerate` environments described in the following example.

- Each item in a list is marked by a *bullet*.
- Item lists can be nested.
 1. Items in enumerated lists are labelled by numerals.
 2. Lists can contain two or more items.
 3. If there is only one item, there is logically no list.
- Blank lines are ignored.
- In the input file, indent lines to show the item list structure.

Item lists can be nested in complex manners, as shown by the above example produced by the following code:

```
\begin{itemize}
\item Each item in a ...
\item Item lists can be ...
\begin{enumerate}
\item Items in ...
\item Lists can ...
\item If there is ...
\end{enumerate}
\item Blank lines are ...
\item In the input file, ...
\end{itemize}
```

Another very useful feature to format item lists is the `declaration` environment. In this environment an item has a name, which is typeset in bold-face, and is followed by its description, which is typeset as itemized text:

X nX delete one or ‘n’ characters starting at cursor position.

dnG delete all lines starting with the current line up to line ‘n’.

The commands for the above example are:

```
\begin{description}
\item[X nX] delete one or 'n' char...
\item[dnG] delete all lines starting ...
\end{description}
```

6 Tables

In `TeX` tables are handled by the “`settabs`” commands, that can either create fixed width columns

or use a template to describe the table fields. `LATEX` has two environments for tables: the `\tabbing` and the `tabular` environments. The first one emulates `TeX` commands with template description, while the second one enables the user to create very complex tables in an easy way. Furthermore `LATEX` code for tables is much more readable than `TeX` one.

The `tabbing` environment handles tables of any length that span across pages. The tab stops can be set either in a prototype or as columns are typed. The tabbing information is a structure started by `\begin{tabbing}` and ended by `\end{tabbing}`. The command `\=` sets the tab stop, `\>` moves to the next stop. The element of the first column has no tab information, and the line is ended by `\\"`. If the line represents a prototype, it ends with `\kill`.

As a tabbing example consider the following table describing how to type some `LATEX` special characters in normal text:

\{	{	open brace
\}	}	close brace
\\$	\$	dollar sign
_>	_	underscore
\%	%	percent

This table makes use of a prototype line and is generated by the following code:

```
\begin{center}
\begin{minipage}{\hsize}
\begin{tabbing}
xxxx \= xxxxxxxx \kill
\verb!\"! \> \{ \> open brace \\
\verb!\"! \> \} \> close brace \\
\verb!\"! \> \$ \> dollar sign \\
\verb!\"! \> _ \> underscore \\
\verb!\"! \> \% \> percent
\end{tabbing}
\end{minipage}
\end{center}
```

The `tabular` environment creates tables that are essentially boxes, that behave like figures and can float around the page but cannot span pages. Frequently these tables are enclosed in drawn rectangles containing vertical and horizontal lines to separate the columns. The tab stops are handled automatically and specified by `&`, the position of the items in the column is defined within the `tabular` environment by one of the characters `l r c` to respectively align on the left, on the right or center the item, and the line is ended by `\\"`.

A vertical line is drawn with `|` declared in the `tabular` specification; the command `\hline` after `\\"` draws a horizontal line across the full width of the

table. The command `\cline{i - j}` draws a horizontal line across columns i through j , inclusive.

When an argument spans multiple columns, it is produced by the `\multicolumn` command with the following syntax:

```
\multicolumn{n}{pos}{item}
```

where n is the number of columns to be spanned, pos defines the position: `l` (for left), `r` (for right), `c` (for centre), and $item$ is the text to be typeset.

As an example consider the following table:

Cray Total Gain (millisec)			
Level	Time	Gain	CDC/Cray
0	33.37	—	1.36
1	29.86	10.5	1.52
2	24.19	19.0	1.87
3	21.92	9.4	2.07

that was produced by the following code:

```
\begin{center}
\begin{tabular}{|c|c|c|c|c|} \hline
\multicolumn{4}{|c|}{Cray ...} \\ \hline
Level & Time & Gain & CDC/Cray \\ \hline
0 & 33.37 & -- & 1.36 \\ \hline
1 & 29.86 & 10.5 & 1.52 \\ \hline
2 & 24.19 & 19.0 & 1.87 \\ \hline
3 & 21.92 & 9.4 & 2.07 \\ \hline
\end{tabular}
\end{center}
```

To get an idea of the easy tabular environment provided by L^AT_EX, note that the T_EX code used to produce the same table is made of 22 lines, each longer and more complex.

7 Mathematics

Mathematical formulas can appear as *in-text* elements (`math` environment) or as displayed formulas (`displaymath` environment). Numbered displayed formulas are produced in the `equation` environment. The commands to select the environments are:

in-text maths: `$...$` or `\(...\)` or

```
\begin{math} ... \end{math}
```

displayed maths: `\[...]` or

```
\begin{displaymath} ... \end{displaymath}
```

equation: `\begin{equation} ... \end{equation}`

Most math elements and symbols are made as in T_EX. The unchanged items are: subscripts and superscripts, greek and calligraphic letters, math spacing, symbols, ellipsis, etc. Most math commands are identical to T_EX, such as `\overline` and `\underline`, `\vec`, etc. The same applies to font selection for math, text and scripts.

Fractions are handled in an easy way by the `\frac` command that has two arguments: numerator and denominator.

$$x = \frac{y+z}{y^2+z^2}$$

```
\[ x = \frac{y+z}{y^2+z^2} \]
```

$$y = \frac{a+b}{1+\frac{a}{a^2+b^2}}$$

```
\[ y = \frac{a+b}{1+\frac{a}{a^2+b^2}} \]
```

Arrays are produced with the `array` environment, which is similar to the `tabular` one. The declaration specifies the size (number of columns) and the alignment of items: `l r c` for left, right or center. New items are begun with `&`, rows are ended with `\\\`

$$A = \begin{pmatrix} x-1 & 1 & 0 \\ 0 & x-1 & 1 \\ 0 & 0 & x-1 \end{pmatrix}$$

The above array is typeset with the following commands. Note that array syntax in L^AT_EX is very similar to T_EX.

```
\[ A = \left( \begin{array}{ccc}
x-1 & 1 & 0 \\
0 & x-1 & 1 \\
0 & 0 & x-1
\end{array} \right) \]
```

The delimiters for arrays are typeset in the same way as T_EX: the commands `\left` or `\right` to specify the left or right delimiter followed by the delimiter itself () [] {}|. The `\left` and `\right` commands must come in matching pairs, but the delimiters do not need to match in any form.

Long or multiple formulas are displayed with the `eqnarray` environment, that enables line numbering and equation splitting. Rows are separated by `\\\`, items by `&`; numbering is disabled by `\nonumber`. The following example shows the use of `eqnarray`.

$$x = a + y - c^2 \quad (1)$$

$$\begin{aligned} y &= a^2 + b^2 - x - \\ &\quad xy + c^3 - p \end{aligned} \quad (2)$$

```
\begin{eqnarray}
x &= a + y - c^2 \\ 
y &= a^2 + b^2 - x - \nonumber \\ 
&\quad xy + c^3 - p
\end{eqnarray}
```

When symbols must be typeset one above another, use the command `\stackrel`:

$$A \xrightarrow{a'} B$$

```
\[ A \stackrel{a'}{\rightarrow} B \]
```

Theorems can receive a name, a label and a number when defined by the `\newtheorem` command that takes two arguments: the name and the label. The theorem text is emphasized. The numbering is automatic and can be computed within the specified sectional unit using the optional argument. The sectional unit can be one predefined by L^AT_EX such as `chapter`, `section`, etc., or the name of a user defined theorem, so that all theorems of the same type are numbered in the same sequence.

```
\newtheorem{guess}{Conjecture}[section]
% define conjecture in section
...
\begin{guess} This is a guess. \end{guess}
Conjecture 7.1 This is a guess.
```

8 Definitions

L^AT_EX provides tools to define new commands (T_EX macros). The new commands are defined by `\newcommand`, followed by the name, the *optional* arguments and finally the definition. The name **must** be prefixed by `\`; when used, arguments must be enclosed in braces. In the following example is shown L^AT_EX syntax to define and call the macro `\abx` and the typeset formula thus generated.

```
\newcommand{\abx}[2]{\$#1x+\#2\$} \abx{5a}{b}
5ax + b
```

In a similar way, to change style, fonts, emphasis, etc., the user can define a new environment with `\newenvironment`.

Commands can be defined anywhere, but the definitions must appear before their use.

9 Graphics – Floating Objects

L^AT_EX has a limited capacity for creating graphic objects and is able to move figures around in a *floating* way to avoid splitting between pages. Figures can receive a caption; in this case a caption number is produced (for an example see Figure 1). Suppose that you must insert a figure 5cm tall in your text; the L^AT_EX code is:

```
\begin{figure}
\vspace{5cm} % leave space for figure
\caption{Fractal image.}
\end{figure}
```

In the above example we leave blank space to insert the picture at a later time with cut and paste methods. For simple graphics L^AT_EX is able to draw axes, lines, circles. The coordinate system is expressed in `\unitlength` with default value 1 point (nearly 1/72 inch).

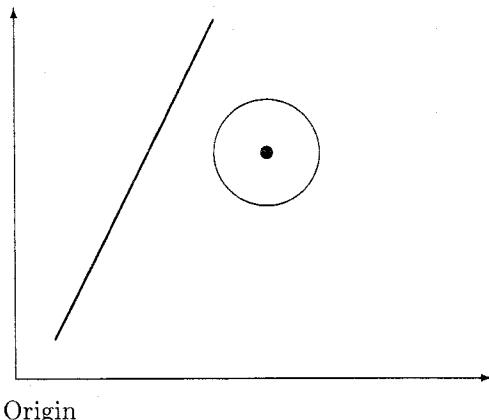


Figure 1: Graphic example

A picture is created with the `picture` environment by specifying the picture's $x - y$ dimensions and, optionally, origin, enclosed in parentheses:

```
\begin{picture}(200,150)(20,10)
```

Any graphic and/or text information is positioned in the picture with the `\put` command, which is followed by the coordinates in parentheses and by the object to put in braces: `\put(0,-10){Origin}`. Objects are ordinary text, straight lines, arrows, circles, ovals.

The line and arrow syntax is the same; the commands are `\line` and `\vector`. The command arguments are slope expressed as $(\Delta x, \Delta y)$ and length. For `\circle` the argument is the diameter. For `\oval` the arguments are width and height plus an optional argument to draw only half or a quarter of the complete oval. Lines can have two standard thicknesses: `\thinlines` (default) and `\thicklines`. Both declarations are used in the example.

The source code example for a simple drawing follows. L^AT_EX output is shown in Figure 1. Note that L^AT_EX requires the use of parentheses () when describing graphic objects.

```
\begin{figure}
\begin{picture}(200,150)
\put(0,-10){Origin}
\put(5,5){\vector(1,0){180}} % x-axis
\put(5,5){\vector(0,1){140}} % y-axis
\thicklines
\put(20,20){\line(1,2){60}} %draw line
\thinlines
\put(100,90){\circle{40}} %draw circle
\put(100,90){\circle*{5}} %fill center
\end{picture}
\caption{Graphic example}\label{fig:ex}
\end{figure}
```

Using the same basic criteria, virtual boxes can be created to split the physical page into sub-areas called *boxes*. Text can be placed inside the boxes at center (default), left (l) or right (r). There are two commands to handle such boxes: `\makebox` and `\framebox`; the second one draws a frame around the box. Both commands have the same syntax: two optional arguments for width and position, and the text to be framed: `\framebox[width][pos]{text}`. Both commands can define box size and text position as optional arguments. The text to be typeset is enclosed in braces. The syntax and typeset results are shown in the following example:

```
framebox      \framebox[2cm]{framebox}
framebox      \framebox[2cm][1]{framebox}
```

Besides the above commands, the `minipage` environment enables the user to split the typeset information into variable size paragraphs of specified width and position typed as multicolumns side by side inside the current environment. The `minipage` environment is used in the above example with the following commands, in which boxes are created in a nested way. The first `minipage` is 2.5cm wide and the second one is wider (4.0cm). Both minipages are typeset with the top line at the current text position ([t]). Note that the two `minipage` environments are typeset side by side as normal text with a `\quad` horizontal space.

```
\noindent
\begin{minipage}[t]{2.5cm}
\framebox[2cm]{framebox}
\framebox[2cm][1]{framebox}
\end{minipage}
\quad % align second minipage
\begin{minipage}[t]{4.0cm}
\verb! \framebox[2cm]{framebox}!
\verb! \framebox[2cm][1]{framebox}!
\end{minipage}
```

Text can be positioned in the center, left or right of the page using the `center`, `flushleft` or `flushright` environments. To start new lines in such environments use `\backslash` as in tabular and array structures.

10 Reference – Index – Bibliography

L^AT_EX provides an easy way to create cross-references linking the various elements of the document, such as figures, equations, sections. Each element can receive a name through the `\label` command. Any string can be assigned to the name, suggested naming conventions are `eq:euler`, `sect:syntax` and the like to create mnemonic

names related to structures and thus more easily identifiable.

Once an element is named, it is referenced with the `\ref` command. The name can be defined in any place in the source code (before or after being referenced), but it should be typed immediately after the referenced item, i.e. if the user wants to label a caption to refer it by figure number, the `label` statement must be typed after the caption title: `\caption{Graphic example}\label{fig:ex}`.

L^AT_EX writes temporary files to handle references that are resolved on the next run, thus it **must** be run twice to typeset the updated reference information. If the temporary files are missing or possibly not up to date, a warning message is written. As an example consider the following fragment:

Equation 3 is very famous.

...

Energy equation is

$$E = mc^2 \quad (3)$$

produced by:

```
Equation \ref{eq:ck} is very famous.
Energy equation is \begin{equation}
E = mc^2 \ \label{eq:ck} \end{equation}
```

References can be set also on any page using `\label` to name the text and the command `\pageref` to get the page number of the named text, the source code looks like the following:

see `page^\pageref{fonts}` for more details.

...

Predefined `\label{fonts}` fonts are:

Keep label definitions to a reasonably short size to avoid *L^AT_EX* problems with internal space. Avoid defining labels that are never used or used too seldom. Keep a list of used labels and their meanings to produce a readable and maintainable input file. More than forty labels can cause problems.

Finally, *L^AT_EX* can produce a table of contents and bibliographic reference with auto-labels. The style of this information is, as always, related to the document style. Its position inside the document is determined by the place in the input file: at the beginning if the command is typed before `\maketitle`, at the end if it is typed before `\end{document}`.

The table of contents is produced by the command `\tableofcontents`; other index information can be produced for tables and pictures using `\listoffigures` and `\listoftables`.

To produce a bibliography, the user defines entries in the `thebibliography` structure, which can have as optional argument the definition of the widest label in the item list. Each item is inserted with `\bibitem`, which has the following arguments:

an optional label that overides the default numbering scheme, the key-name for citations and the entry text. The items are referenced with the \cite command.

An example of bibliographic data is given by the following environment definition:

```
\begin{thebibliography}
\bibitem{bib:la} L. Lamport. \LaTeX:
  {\em User's Guide and Reference ... }
  ...
\bibitem{bib-my} M. L. Luvisetto, ...
{\em Introduzione ... }
\end{thebibliography}
```

and are called in any place as shown:

```
... for more information see
\cite{bib:la,bib:le} and ...
\cite{bib-my}.
```

11 Useful Tools

At our site, many researchers have a workstation, but a language sensitive editor is provided only on some machines. To help our users in typing documents we have created a set of files containing a template of the most common environments. The files are defined at system level: as logicals under VMS, as symbolic links under Unix, so they can be inserted into a document during the editing session.

As most users are not computer professionals, their knowledge of the editors is a basic one. Therefore we have developed the utility program PreLATEX that, when run on a new file, interactively asks for title, authors, etc., and produces a template file similar to the one listed at the beginning of this article. The program asks also for section names and bibliography; thus the user can create a skeleton of the document at his first session.

During the editing session, LATEX environments such as:

```
quotation itemize enumerate
description tabbing tabular
array minipage figure
```

can be inserted in template form by inserting one of the template files.

The file contains the name of the environment, the optional arguments, a title to declare its usage and empty lines to produce the wanted information. By following the template it is easy to create tables, arrays, item lists, etc., even for newcomers. In a few hours, inexperienced users are able to produce simple documents. The file names and usage are described in help files. For people using workstations, it is simple to keep this information in a separate window ready for use.

An example of a template file is given below for the tabbing environment.

```
\begin{tabbing}
% create table, set tab with \=
% recall tab with \> end with \\
% set template fields (f)
% insert fields between tabs
\f1 \= f2 \= f3 \= f4 \= \kill
  \>    \>    \>    \> \\
  \>    \>    \>    \> \\
  .....\\
\end{tabbing}
```

Thus the user needs only to fill in the columns with his own values.

PreLATEX can be run on an existing file. In this case, it produces a list of all sections together with their headings, a list of all ref, pageref and label commands and, at the user's request, a list of all figures, captions, equations and the like. If the bibliography data are present they are listed at the end, together with cite commands. This feature is found very helpful in checking cross-references, especially in the final steps of the document preparation.

PreLATEX is not static. Planned extensions include a reformatting option that would change the columnar alignment in the source of tables, arrays, etc., to more closely resemble the output. A source in this form should be easier to read and maintain.

The above software is free and available from the authors, who can be contacted at the cited e-mail address.

◊ Maria Luisa Luvisetto and Enzo
Ugolini
Istituto Nazionale di Fisica
Nucleare
Viale Ercolani 8
40138, Bologna, Italy
Internet: Luvisetto@CNAF.INFN.IT
Internet: Ugolini@CNAF.INFN.IT

Geometric diagrams in L^AT_EX

Peter J. Cameron

1 Introduction

Teachers of geometry since Euclid have known the value of diagrams to help students in following geometric arguments. Paradoxically, it is necessary that the argument be capable of standing alone, without the diagram—Euclid himself realised the possibility of reaching false conclusions from reasoning depending on a faulty diagram—but an accurately-drawn diagram can be a valuable aid to following the reasoning.

L^AT_EX provides tools (in the `picture` environment) for drawing accurate and clear diagrams, which are a great improvement on what I can produce by hand. How adequate are these tools for complicated diagrams?

An important feature of diagrams for pedagogical purposes is that they should be *generic*. For example, no two lines should be parallel, and no triangle isosceles, unless this is part of the specification of the diagram or a geometric consequence of it. Such unwanted special features could create the misleading impression that they are necessary for the conclusion.

Of course, more powerful (and cumbersome) picture-drawing tools are available; but L^AT_EX has the advantage that it is always there. Also, as we will see, the investigation throws up some interesting problems!

2 Triangles

The `picture` environment enables line segments with any one of 48 different slopes to be drawn. The allowable slopes are 0 (horizontal), ∞ (vertical), any ratio x/y where x and y lie between 1 and 6 inclusive, and the negatives of these.

Three different slopes determine uniquely the shape of a triangle. We are still free to determine the size and position of the triangle by positioning the line segments appropriately.

It is easy to draw an isosceles triangle in which the base is horizontal or vertical (the other two slopes a, b should satisfy $b = -a$) or makes an angle of 45° with the axes (the other slopes satisfy $ab = 1$). Curiously, it turns out that any other isosceles triangle which can be drawn is right-angled (with base angles of 45°). There appears to be no logical reason for this fact, which is established by checking all possibilities. A typical example has slopes of $1/5$ (for the base), $-2/3$ and $3/2$. Up to reflection in the axes, there are twelve such triangles.

Note in passing that it is not possible to draw an equilateral triangle accurately in L^AT_EX, for all allowable slopes are rational numbers, and so the tangents of the angles between them are also rational; but

$$\tan 60^\circ = \sqrt{3}$$

is irrational. Tolerable approximations can be produced; for example, $5/3$ differs from $\sqrt{3}$ by only about 4%.

On the other hand, rectangles and squares can be drawn in many different orientations, since lines with slopes a and $-1/a$ are necessarily perpendicular. The interested reader may like to consider other metric properties of quadrilaterals and higher polygons.

3 Quadrangles

The last section was about *Euclidean geometry*, concerning actual values of lengths and angles. (In practice this means that properties will be destroyed if the horizontal and vertical resolution of the output device are not equal; for example, isosceles triangles will no longer be isosceles). In what follows, I will be concerned with *affine geometry*, which does not have this defect. Only incidence and parallelism, and properties derived from these are significant. For example, we can say that two line-segments have the same direction; if they have the same direction (but not otherwise), it is meaningful to say that they have the same length.

The first problem that arises is to draw a *complete quadrangle*, consisting of four points and all six lines joining them. Here, the slopes of the lines determine the figure up to choice of position and scale; but the six slopes are not independent. If they are a, b, c, d, e, f , where the pairs a and b , c and d , e and f are “opposite”, and a, b and c pass through one of the points, then

$$(b - c)(d - e)(f - a) = (a - d)(b - e)(c - f).$$

Using this formula, f can be expressed in terms of the other five slopes. Of course, not all choices of a, \dots, e will give an allowable value for f . Determining the allowable possibilities is clearly work for computers!

A note on this computation. It is an advantage to do all the calculations in integers. If the non-zero L^AT_EX slopes are multiplied by 60, they become integers, and can be recognized by the fact that they are all the divisors of 3600 between 10 and 360 inclusive, except for 16, 18, 25, 144, 200, 225 and their negatives. Moreover, since our equation is homogeneous, it remains true after this multiplication. The

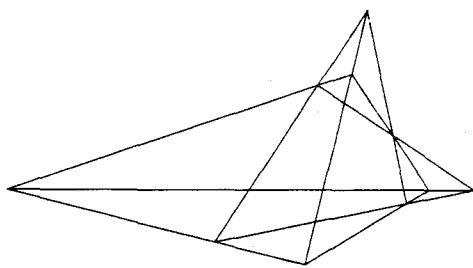


Figure 1: Desargues' Theorem.

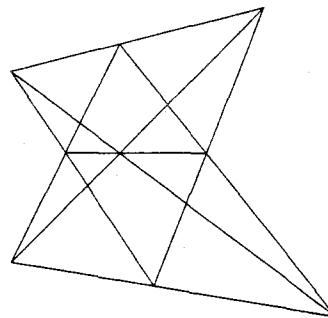


Figure 2: Pappus' theorem.

good news to emerge from the computation is that there are far too many solutions to list here!

4 Desargues and Pappus

Two very important theorems of affine geometry are those of Desargues and Pappus. Each has the property that it is cumbersome and unenlightening to state in words, but simply expressed by a diagram. Desargues' theorem states that if we have the ten points and nine of the ten lines of Figure 1, then the tenth line also occurs (that is, the three corresponding points are collinear). Pappus' theorem has a similar diagram with nine points and nine lines (see below).

If we look at the Desargues configuration, we see five complete quadrangles in it. So the work done in the last section can form the basis of a computer search for suitable parameters: seven slopes will determine the shape of the figure. Before I made this search, I spent some time looking for solutions by trial and error, without success. The surprising result of the exhaustive search was that there are many tens of thousands of solutions. The particular one used to draw Figure 1 is as follows. It is chosen to be reasonably "generic", in the earlier sense.

```
\setlength{\unitlength}{0.3mm}
\begin{picture}(220,130)(-10,-40)
\put(0,0){\line(1,0){207}}
\put(0,0){\line(3,1){153}}
\put(0,0){\line(4,-1){132}}
\put(92,-23){\line(2,3){68}}
\put(92,-23){\line(5,1){115}}
\put(132,-33){\line(5,3){55}}
\put(132,-33){\line(1,4){28}}
\put(177,-6){\line(-1,5){17}}
\put(187,0){\line(-2,3){34}}
\put(207,0){\line(-3,2){69}}
\end{picture}
```

Pappus' theorem is illustrated in Figure 2.

Unlike the last one, this figure contains no complete quadrangles, so we have to start its analysis from scratch.

There is an added difficulty. The equation connecting the slopes of a complete quadrangle has degree 3; so individual terms cannot exceed $360^3 = 46656000$; using 32-bit integers, there is no risk of overflow. However, in the case of Pappus, we have a fifth-degree equation (even after assuming that one line is horizontal), and integer overflow is inevitable. This can be alleviated to some extent by dividing by common factors wherever possible; but there were still many cases when the terms grew too large. So I abandoned the exhaustive search and simply looked for what I could find.

The general picture was the same as for Desargues: there are thousands of acceptable solutions, but it is very difficult to find even one by trial and error.

5 Conclusion

L^AT_EX provides tools for drawing quite complicated diagrams. But there are some things that can't be done, and others where you have to work quite hard (and even resort to computation) to find the right way to draw your figure. Readers are encouraged to try their luck with other geometric configurations. (I have concentrated on some of the figures I had to draw for a volume of lecture notes on geometry.)

Perhaps someone will be inspired to write a program which does all this automatically. The input should be the specification of the diagram, with optionally some comments about desirable or undesirable coincidences of length or slope. The program should either produce L^AT_EX input for an acceptable figure, or tell the operator that it can't be done.

◇ Peter J. Cameron
School of Mathematical Sciences
Queen Mary and Westfield College
Mile End Road
London E1 4NS
U.K.
Janet: pjc@uk.ac.qmw.maths

How to change the layout with L^AT_EX 2.09

Hubert Partl

The L^AT_EX principle

Everything that is printed consists of two components: content and layout. The author provides the contents, the publisher's layout designer provides the layout, the typesetter puts the contents into the desired layout, and the printer puts it onto the paper.

These steps are clearly separated by L^AT_EX:

- The author specifies the contents in the document, i.e. in the `.tex` file.
- The layout designer specifies the layout in the document style, i.e. in one or more `.sty` files.
- L^AT_EX typesets the contents (i.e. everything between `\begin{document}` and `\end{document}`), using the layout that results from the main document style and the document-style options specified in the `\documentstyle` command.
- The device driver prints the results.

Four main styles (`article`, `report`, `book`, `letter`) and several options (`twoside`, `twocolumn`, etc.) are distributed together with L^AT_EX, and many more are available in the style collections that can be found on various servers (Aston, Heidelberg, Stuttgart, SHSU and so on).

Now, let us assume that you are an author who wants to print something in a certain layout.¹ If you find a document style or document-style option that produces just this layout, all you need to do is copying that file to your computer and naming it in the `\documentstyle` command. If no such style exists yet, just write your own, and don't be afraid: it's not so complicated as you may think.

How to proceed

For most cases, I recommend the following way:

1. Find an original L^AT_EX style file that produces a layout which has some similarities with your required layout, and note all the differences between that original layout and your required layout.
2. Find the original definitions of all the features that have to be changed.
3. Write a document-style option that contains the modified versions of these definitions.

¹ Of course, this layout has to be specified by a professional designer — either hire such a professional to design it for you, or follow a professionally designed layout that you have seen somewhere and that suits your needs...

4. Specify the name of the original document style as the argument (i.e. between the curly braces) in the `\documentstyle` command, and add the name of your new document-style option as the last option between the square brackets.

Since L^AT_EX reads first the main document style and then the document-style options, your modified definitions will override the original ones.

Thus, the problem is reduced to the question: where do I find the original layout definitions and an explanation of how to modify them?

Where to find the original definitions

Since Leslie Lamport's L^AT_EX manual is aimed at authors, not at layout designers, the required information is distributed over several places. I recommend the following search order:

1. the L^AT_EX manual,
2. the files `article.doc`, `art10.doc`, etc., which contain the definitions from the corresponding `.sty` files, but with explanations added on comment lines,
3. the file `latex.tex`,
4. Donald Knuth's T_EX book.

For the new version 3.0 of L^AT_EX, Frank Mittelbach und Rainer Schöpf have promised to provide a separate and complete documentation of the style designer interface.

The following hints and examples refer to the current L^AT_EX version 2.09.

An example

Let's look at a simple example: in an article that contains several sections with a large number of mathematical equations, the equations must be numbered separately in each section rather than consecutively throughout the article.

In the L^AT_EX manual, we learn that in the `report` document style something similar is provided: there, equations are numbered separately in each chapter.

In the file `report.doc` we find the following definitions:

```
\@addtoreset{equation}{chapter}
\def\theequation{\thechapter
    .\arabic{equation}}
```

The first line resets the equation counter at each chapter, and the second one defines the equation label to consist of the chapter counter, a full stop and the equation counter.

Therefore, we write the following new definitions to a file that we call `eqpersec.sty`:

```
\@addtoreset{equation}{section}
```

```
\def\theequation{\thesection
    .\arabic{equation}}
```

Of course, the file must contain more than just these two lines: we need comment lines that state the name of the file, its author, the version (date of last change), an explanation of its purpose and usage, and explanations of the definitions. Also, we properly end it with `\endinput`. Figure 1 shows the complete file.

With this file being available, we can change the equation numbering scheme by specifying the new style option name `eqpersec` in

```
\documentstyle[11pt,eqpersec]{article}
```

Now that we have mastered this exercise, let us have a look at some other typical layout problems.

Page dimensions

Changes to the page grid are rather simple: they can be achieved by setting the appropriate length parameters.

From the L^AT_EX manual, we learn the following: vertically, each page consists of the following parts (from top to bottom)

- 1 inch plus `\topmargin` white space on top,
- `\headheight` space for the running head,
- `\headsep` white space below the running head,
- `\textheight` space for the main text (including figures, tables and footnotes),
- `\footskip` space for the running foot (including the white space above it),
- and the rest of the paper height for the white space at the bottom.

Horizontally, normal pages² consist of the following parts (from left to right):

- 1 inch plus `\oddsidemargin` (for odd-numbered pages, or `\evensidemargin` for even-numbered pages) white space on the left,
- `\textwidth` space for the main text (including indentations), which may be split into two columns separated by `\columnsep`,
- `\marginparsep` white space between text and marginal notes,
- `\marginparwidth` space for marginal notes,
- and the rest of the paper width for the white space on the right.

Section headings

Let us assume that we want to change the section headings in two respects: they must be less prominent, i.e. use a smaller typeface and smaller vertical

² i.e. pages where marginal notes are to appear in the right margin.

spacing, and hyphenation must be disabled within the headings.

In the file `art10.doc` we find the following definition:

```
\def\section{\@startsection
    {section}{1}{\z@}%
    {-3.5ex plus -1ex minus -.2ex}%
    {2.3ex plus .2ex}%
    {\Large\bf}}
```

and similar definitions for subsections etc. The absolute values of the fourth and fifth parameters specify the spacing before and after the heading, and the sixth parameter specifies its style. Therefore, we arrive at the following modified definition:

```
\def\section{\@startsection
    {section}{1}{\z@}%
    {-1.75ex plus -0.5ex minus -.1ex}%
    {1.15ex plus .1ex}%
    {\secshape\large\bf}}
```

(and similar ones for subsections etc.), with an extra definition

```
\def\secshape{\rightskip=0pt plus 1fil
    \hyphenpenalty=2000\relax}
```

which sets the text ragged right without hyphenating.

Running headers and footers

From our sources we learn that running headers and footers are specified by page styles, and that page styles are defined by commands of the form `\ps@name`, which can then be selected by the `\pagestyle` command.

Now we want to define a new page style `myfootings` that produces a running footer similar to the running header of the `myheadings` page style, but — to make matters even more complicated — with a horizontal line above.

In file `article.doc`, we find the following definition for the `myheadings` pagestyle:

```
\def\ps@myheadings{\let\@mkboth\@gobbletwo
\def\@oddhead{\hbox{}\sl\rightmark\hfil
    \rm\thepage}%
\def\@oddfoot{}%
\def\@evenhead{\rm\thepage\hfil
    \sl\leftmark\hbox{}}
\def\@evenfoot{}%
\def\@sectionmark##1{}%
\def\@subsectionmark##1{}}
```

The definitions of `\@oddhead`, `\@oddfoot`, `\@evenhead` and `\@evenfoot` specify the running headers and footers of odd and even numbered pages, respectively, either as a line or as a `\parbox` of width `\textwidth`. Within these lines, `\rightmark`

and `\leftmark` are variable texts that can be inserted with the `\markboth` and `\markright` commands, and `\thepage` is the page number. We define

```
\def\ps@myfootings{\let\@mkboth\@gobbletwo
  \def\@oddfoot{\parbox{\textwidth}{%
    {\rule{\textwidth}{0.4pt}}\hbox{%
      \small\sffamily\rightmark\hfill
      \small\sffamily Seite~\thepage}}%
  \def\@oddhead{}%
  \def\@evenfoot{\parbox{\textwidth}{%
    {\rule{\textwidth}{0.4pt}}\hbox{%
      \small\sffamily Seite~\thepage\hfill
      \leftmark\mbox{}}}}%
  \def\@evenhead{}%
  \def\sectionmark##1{%
    \def\subsectionmark##1{}}}
```

and put this definition into a file `myfoot.sty`. To switch on this page style, we have to specify the document-style option `myfoot`, to specify the page style `myfootings` and to fill in the variable texts:

```
\documentstyle[11pt,myfoot]{article}
\pagestyle{myfootings}
\markboth{News of \today}{News of \today}
```

Paragraphs

Let us assume that we want paragraphs to be separated by a certain amount of vertical space and without horizontal indentation (although some people consider this a bad practice).

The obvious way is to set `\parskip` and `\parindent` accordingly. In order to help L^AT_EX to find the best places for page breaks, we give `\parskip` a non-zero stretch component. Thus we arrive at the following:

```
\parskip=0.5\baselineskip
\advance\parskip by 0pt plus 2pt
\parindent=0pt
```

However, this has some undesired side effects:³ the vertical spacing before, after and within lists and other environments depends on the value of `\parskip`, too. From the L^AT_EX manual we learn the following spacing conventions for list-like environments:

- `\parskip` plus `\topsep` before the first item of a list (before the environment),
- `\parskip` plus `\itemsep` between items, and
- `\parsep` only between paragraphs within one item.

³ at least in the current L^AT_EX version 2.09.

If we want all these skips to equal `\parskip`,⁴ then we have to set `\parsep` to `\parskip`, and both `\topsep` and `\itemsep` to zero. However, it is not sufficient to set these values globally. Rather, the initialization macros for the list at the various nesting levels have to be redefined. Figure 2 shows what has to be added to our paragraph shape definitions.

Where to find more information

For all who want to know more about "How to change the layout with L^AT_EX 2.09", I have written a booklet of about 30 pages—in German. It is available freely. The L^AT_EX source files (including their own special document-style options) can be obtained via Bitnet from the server `listserv@dhdurz1` in Heidelberg—GET the file `LAYOUT ZOOUE` and decode and unpack it to obtain the files `layout.tex`, `layout2.tex`, `refman.sty` and `german.sty`. Members of the German-speaking T_EX users group can also obtain it on a PC diskette from the DANTE association in Heidelberg.

At the European T_EX Conference 1990 in Cork,⁵ Frank Mittelbach and Rainer Schöpf announced that version 3.0 of L^AT_EX will contain several significant changes (improvements) to the document-style design interface of L^AT_EX, and that they will provide a complete documentation of this interface in addition to the traditional L^AT_EX user's manual. The completion of the L^AT_EX 3.0 project will be announced via *TUGboat* in due time.

Acknowledgements

I want to take this opportunity to thank Sue Brooks, Paul Stiff, David Rhead, and Nelson Beebe, who were very helpful with guiding my first steps into the miraculous land of L^AT_EX layouts.

◊ Hubert Partl
EDV-Zentrum
Universität für Bodenkultur
Feistmantel-Straße 4
A-1180 Wien, Austria
Bitnet: `z3000pa@awitwu01`

⁴ which, however, leaves no visual distinction between lists within paragraphs from lists at the beginning or end of a paragraph.

⁵ see *TUGboat* 12, no. 1 (1991), pp. 74–79.

Editor's note: An update on the L^AT_EX 3.0 project appeared in *TUGboat* 13, no. 1 (1992), pp. 96–101.

Figure 1: Example of a complete style option file

```
% This is EQPERSEC.STY by H.Partl, TU Wien (Austria)
% Last change: 7 Feb 1990
% Document-style option for LaTeX 2.09,
% to make equations numbered per section,
% to be used only with the document style 'article'.

% Reset equation counter at each section:
\@addtoreset{equation}{section}

% Equation label = section number dot eqation number:
\def\theequation{\thesection .\arabic{equation} }

\endinput
```

Figure 2: Modifications for the list environments

```
\def\@listI{\leftmargin\leftmargini
            \topsep\z@ \parsep\parskip \itemsep\z@}
\let\@listi\@listI
\@listi
\def\@listii{\leftmargin\leftmarginii
              \labelwidth\leftmarginii
              \advance\labelwidth-\labelsep
              \topsep\z@ \parsep\parskip \itemsep\z@}
\def\@listiii{\leftmargin\leftmarginiii
              \labelwidth\leftmarginiii
              \advance\labelwidth-\labelsep
              \topsep\z@ \parsep\parskip \itemsep\z@}
```

SGML

SGML—Questions and Answers

Reinhard Wonneberger and Frank Mittelbach

Abstract

This paper explains SGML fundamentals in a concise Questions & Answers way.

More detailed information can be found in the works that are listed in the bibliography.

Abbreviations and acronyms are explained in a glossary.

Usage of proprietary names in this paper must not be construed to mean that they are free of rights.

1 Functionality

[?] What does SGML mean?

SGML stands for *Standard Generalized Markup Language*. It is the most important standard for document processing [App89].

[?] What is General Markup?

General Markup is a special sort of text that will tell a reader or a program about the logical function of ordinary text. In the L^AT_EX sequence \section{Headline}, the markup \section states that the text Headline is to be considered as a section heading. General Markup is described in more detail in [CRD87].

[?] How can SGML be converted into print?

The proper way is to use a parsing and translating program, which will check that the document corresponds to the Document Type Definition (DTD) and then will translate the document into one of the available document processing languages, e.g. DCF-GML, T_EX, TROFF, or some other system, or into internal code with systems like Interleaf.

There are also some programs that can interpret a subset of SGML-conformant markup, due to the fact that they allow implementation of *General Markup*, see below. This is especially true for DCF and T_EX [Wonng]. Interpreting SGML, however, does not allow checking for conformance with a DTD.

[?] What is the difference between GML and SGML?

SGML Markup will normally be parsed by an independent program to assure conformance with the

Document Type Definition (DTD) and then converted to some processing language, like DCF-GML or L^AT_EX.

GML Markup is one of the processing languages, and is tied to DCF, a proprietary program of IBM.

[?] What other use can be made of SGML?

One main application area is databases. SGML markup can be mapped to database fields and vice versa, and specific information can be extracted from SGML-tagged sources automatically. Another application is hypertext [Det91].

2 Usage

[?] Who are the key users of SGML?

Many producers of large quantities of documents, among them the US Department of Defense (DoD) and many other government authorities in several countries, the European Community, several research institutions, etc.

[?] Is SGML proprietary?

SGML is a Standard, that has been adopted by Standards committees on different levels, and as such it is non-proprietary. The software to process SGML may or may not be proprietary, depending on the software supplier.

[?] Is there a business impact of SGML?

Definitely yes, as SGML has been made an integral part of CALS, which is a bundle of requirements for submitting bids to the already mentioned DoD in electronic form. Due to the influence of the DoD, many vendors are obliged to meet these requirements, and other institutions will follow suit in imposing this standard.

[?] Can SGML documents be exchanged across computer platforms?

Yes, if you have true SGML software. Some programs up to now, however, support only a special predefined set of DTDs.

[?] Is SGML human readable?

That is one of its main advantages over the so-called WYSIWYG systems. You might even read a complete SGML source on the phone!

[?] Does SGML run on my system?

As SGML-tagged documents do not contain hidden characters, you can generate and edit them on any computer system. SGML parsers for many computer

platforms are already available or under development. See also section 4 for some information on available software.

[?] How do I enter an SGML document?

By using your favourite text editor, e.g. ISPF for MVS systems. There are also editors that help you with entering correct SGML syntax (that's the luxury class).

[?] Where can I get more information on SGML?

Just refer to the following books [Gol90, Her90, Bry88] and articles [Bar89] or [Laa91, Pop91].

3 Applications

[?] How can we benefit from SGML?

Our documents if tagged with SGML may have a good chance to survive us, to say nothing of the many benefits of the General Markup approach in general [CRD87] as they are well-known from systems like DCF or L^AT_EX. In addition, SGML gives independence of specific programming environments, providing full portability of sources like T_EX and L^AT_EX. Unlike T_EX and L^AT_EX, however, SGML is a true international standard, not only a *de facto* standard set by usage, and a standard that is backed by powerful institutions. The fundamental advantages of General Markup have been described so often that we don't dare to repeat that here, cf. [Won90].

[?] Are applications confined to document processing?

No. Another important area of application is databases. Because tags can specify the contents of a field, they can also be used to load information into a database. The other way round, information from a database can be output with tags so that it can be formatted automatically into a specific document.

[?] How could SGML interact with a database?

In some databases, there will be 'free-form texts'. In order to get specific items from such fields for reporting, one might use SGML tagging inside free-form text fields, then write out the contents of these fields to a file and process the file with an SGML parser and typesetter. In addition, fields defined in the database might be written out with generated tags. Like this, the benefits of a database can be combined with the benefits of Structured Document Processing.

4 Products

[?] How can SGML be used on a mainframe host?

If an MVS host is implied, there are several possibilities we know about:

- Use an SGML parser from IBM.
- Emulate SGML with DCF, provided your input obeys a few restrictions.
- Use the DAPHNE software from DFN for translation into T_EX, TROFF, or DCF [SC88].
- Interpret SGML with T_EX, which might be an attractive solution; cf. [Won92] for details.

[?] What about workstations?

A well-known product is Interleaf, which is also available from IBM under the name TPS. Another important product is the Publisher from ArborText.

Glossary

AAP American Association of Publishers.

Attribute Modifier of a Markup Tag, used to specify different values.

CALS Computer-Aided Acquisition and Logistics Support.

DCF Document Composition Facility (IBM product), ⇒GML.

DTD Document Type Definition.

General Markup Meta-information specifying the logical function of ordinary text.

GML General Markup Language, part of ⇒DCF.

ISO International Organisation for Standardisation.

ISPF Interactive System Productivity Facility, a user interface for ⇒MVS.

L^AT_EX Lampert's T_EX, markup language and ⇒T_EX macro package by Leslie Lampert.

MVS Multiple Virtual Storage, production-oriented IBM operating system for large mainframes.

Parser Program performing syntax analysis on some source.

SGML Standard Generalized Markup Language.

Tag Explicit label that marks start (and end) of a source entity.

T_EX (from the Greek $\tau\acute{e}x\nu\eta$) typesetting program by Donald E. Knuth.

WYSIWYG 'What you see is what you get'; slogan describing screen-oriented text processing tools, often considered as the opposite approach to ⇒General Markup.

Bibliography

[App89] W[olfgang] Appelt. Normen im Bereich der Dokumentverarbeitung. *Informatik-Spektrum*, 12:321–330, 1989.

[Bar89] David Barron. Why use SGML? *Electronic Publishing (EPodd)*, 2(1):3–24, April 1989.

- [Bry88] Martin Bryan. *SGML: An Author's Guide to the Standard Generalized Markup Language*. Addison-Wesley, Woking, England; Reading, Massachusetts, second edition, 1988.
- [CRD87] James H. Coombs, Allen H. Renear, and Steve J. DeRose. Markup systems and the future of scholarly text processing. *Communications of the ACM*, 30(11):933–947, November 1987.
- [Det91] Christine Detig. *TeX & Hypertext — The future of electronic publishing*. In Guenther [Gue91], pages 8–12. TUGboat 12 (March 1991) Number 1.
- [Gol90] Charles Goldfarb. *The SGML Handbook*. Clarendon Press, Oxford, 1990.
- [Gue91] Mary Guenther, editor. *TeX90 Conference Proceedings; University College; Cork, Ireland, September 10–13, 1990*. Providence, Rhode Island, U.S.A., March 1991. *TeX* Users Group. TUGboat 12 (March 1991) Number 1.
- [Her90] Eric van Herwijnen. *Practical SGML*. Kluwer, Dordrecht, NL, 1990.
- [Laa91] C.G. [Kees] van der Laan. SGML(, *TeX* and ...). In Guenther [Gue91], pages 90–104. TUGboat 12 (March 1991) Number 1.
- [Pop91] N.A.F.M. Popelier. SGML and *TeX* in scientific publishing. In Guenther [Gue91], pages 105–109. TUGboat 12 (March 1991) Number 1.
- [SC88] A. Scheller and C. Smith. *DAPHNE; Document Application Processing in a Heterogeneous Network Environment; Dezentrale Verarbeitung von Dokumenten auf der Basis von SGML; Benutzeranleitung (Version 3.0)*. GMD-FOKUS, Berlin, im Auftrag des Vereins zur Förderung eines deutschen Forschungsnetzes e.V., April 1988.
- [Won90] Reinhard Wonneberger. Structured document processing: the I^AT_EX approach. In J. Nadrlchal, editor, *Man-Machine Interface in the Scientific Environment. Proceedings of the 8th European Summer School on Computing Techniques in Physics. Skalský Dvůr, Czechoslovakia, 19–28 September 1989*, volume 61 of *Computer Physics Communications*, pages 177–189. North Holland Publishing Company; Elsevier Science Publishers B.V., 1990.
- [Won92] Reinhard Wonneberger. Approaching SGML from *TeX*. *TUGboat* 13(3):223 (July 1992).
- [Wonng] Reinhard Wonneberger. *TeX* in an industrial environment. In Anne Brüggemann-Klein, editor, *Proceedings of the 4th European *TeX* Conference, September 11th–13th, 1989*, Karlsruhe, forthcoming.

◊ Reinhard Wonneberger and
Frank Mittelbach
EDS Electronic Data Systems
(Deutschland) GmbH
Eisenstraße 56 (N15)
D-6090 Rüsselsheim
Federal Republic of Germany

Dreamboat

Editor's note: This column heading hasn't appeared for years, but it seemed an appropriate corner in which to collect ideas and suggestions related to the topic "Where do we go from here?" In addition to the following articles, which were written before the formal recognition of interest in future directions, Philip Taylor has reported in this issue (p. 138) on the first meeting of the working group coordinating the discussion.

TeX wish list

Michael Barr

It is the rare user of *TeX* who has not, at some time, felt that *TeX* lacks some feature or other. Since Knuth has announced that *TeX* is now frozen, save for an occasional bug fix, it is up to the *TeX* community to give thought to the kinds of features that we want in any successor to *TeX*.

I do not expect that my wish list will be exhaustive or that the future program will implement every one of my suggestions. I am merely trying to start a dialog on the kind of program we want in the future.

Let me say a few words about what I don't want. I don't expect to see a WYSIWYG program, although a multitasked previewer would be nice. I don't expect to see a page layout program. In fact, I don't want to think about page design at all. Ideally, future *TeX* will take care of all design details itself. It is a *tour de force* to lay out *TV Guide* in *TeX*, but *TeX* is not the tool I would have chosen for the job.

Here are some of the things that I have felt lacking in *TeX*, in no particular order. I divide them into two groups, depending on whether or not they could be made compatible with current device drivers. The reason is that there is basically only one *TeX* program, but as many device drivers, and more, as there are devices. Thus the amount of work that is involved in upgrading the latter is orders of magnitude larger than that which is involved in upgrading *TeX* itself.

Features that could be implemented without changing device drivers

A smart \put. By a smart \put, I mean a procedure similar to the \point defined on page 389

of *The TeXbook*, but one that would set the width of the box properly. If you actually try that procedure, you will find that the box has zero width. The height and depth are set to the actual height and depth, but not the width. No variation I tried was able to do it either.

The reason I consider it important is that I use L^AT_EX's picture mode extensively for commutative (and even non-commutative) diagrams, and you have to tell picture mode exactly what dimensions your picture is. What nonsense! T_EX is smart enough to figure out how large your picture is, isn't it? Well, yes it is, but not at any great speed. I don't know what design consideration caused Leslie Lamport to implement \picture mode as he did, but it is entirely possible that it was the long time it took for a picture to work out its own size. If this were implemented in the program, it would take a fraction of the time. For my own macros, I have reimplemented both \put and \picture. However the compilation of a diagram of any complexity takes a long time. A page with even one complicated diagram takes an appreciable part of a minute (on my 16 Mhz 386SX computer).

Implicit in this point is that there should be a built-in picture mode. It would be faster and more reliable than the L^AT_EX \picture procedure. By the way, although it is not an important point, Lamport erred in having his coordinate system use the mathematician's orientation. T_EX is a typesetting program and to a typesetter the positive *y* direction is down, not up. I find it a real nuisance to think upside down when drawing a complicated diagram.

More reliable program control. This rubric covers so many different things that I hardly know where to start. Take the entire appendix D of *The TeXbook* and ask yourself why most of them should require dirty tricks? Most of them are quite reasonable things and it is a mystery to me why you should have to resort to dirty tricks to do reasonable things. Take the discussion of trying to place `\n` stars on a page, where `\n` is an integer variable. *Why* is this so hard to do? It is, after all, a perfectly reasonable thing to want to do; why shouldn't the language provide a way to do it straightforwardly? I know that Knuth is exceedingly clever, much cleverer than I, but why didn't he design a language that I could program in? The June 1991 issue of *TUGboat* had no fewer than three new implementations of procedures for outputting `\n` asterisks, and each of them was based on some clever trick.

I suspect that one of the problems is that Knuth didn't at first think of T_EX as a programming lan-

guage. This seems even clearer if you look at T_EX78. It was so deficient that you couldn't \advance a numeric variable, only increment or decrement it. Imagine how hard it would be to implement L^AT_EX in that language!

I am getting indigestion hearing about T_EX's digestive tract. The discussion of \expandafter is ludicrous. Both \expandafter and \noexpand ought to be able to take an entire brace-delimited phrase as argument, not just a single control sequence. Moreover a new control sequence \expand ought to be provided, preferably with a second, optional, parameter that tells how many levels of expansion are wanted, since in many cases you want only one level of expansion, not to the very bottom. More generally there ought to be a simple mechanism by which the user can specify when a control sequence should be expanded. For example, \expandafter is what in FORTH would be called an immediate control sequence; it controls compilation. The user should have the ability to define his own "immediate" control sequences as well as ways of overriding this specification (it is often necessary to override the immediate specification when defining a new immediate word).

Better arithmetic. This includes the ability to use numeric expressions as arguments and having real number registers. I have been told that the reason for the lack of the latter is that Knuth didn't want the user to have any access to the underlying floating point. Why should T_EX use floating point arithmetic at all? Wouldn't everything be faster if everything were in fixed point? I thought all distances were in scaled points anyway and a scaled point is smaller than one wavelength of visible light.

As for using expressions as arguments, almost anyone who has ever used a macro has had to write complicated procedures because you couldn't give, say, \hsize-10pt or similar expressions involving counters as arguments. At one time, this lacuna was justified on the grounds that T_EX was to run in as small a memory as possible, but this is no longer a valid reason.

Successive super and subscripts. This seems like a picky point, but in my work it comes up surprisingly often. I refer, in the first instance, to the fact that you cannot say `x_1_2` for `x_{12}`. Why not? They are logically equivalent. To see what pain even Knuth had to go through on this point, see the definition of the \prime operator. I have an operator \op defined as `\{\}^{\op}` and the initial

brace pair is there to avoid running into the “double” superscript error. But it also means that it doesn’t work properly if there is a subscript on the same symbol. Surely a simple parser could interpret double superscripts properly.

More reliable global page procedures. I was recently unable to get marks to work right in the twocolumn environment of either the macros supplied by L^AT_EX or those of Frank Mittelbach. Footnotes are not reliably placed by Mittelbach’s style either. It is not clear if, in the present version of T_EX, it is possible to combine a multicolumn style that allows changing the number of columns in the middle of a page with proper placement of footnotes and marks. I don’t use inserts, but virtually everyone who does complains that they don’t work as expected. Changebars have proved extremely difficult to implement reliably. Someone wrote to T_EXhax several months ago asking if it was possible to leave a 2 by 2 inch box blank in a lower corner of each page. So far as I know, it can’t be done, except perhaps by some sort of cut and try procedure similar to that of the column balancing on page 387 of *The T_EXbook*.

More control over tfm’s. The internal variables pertaining to a whole font can be changed, but not, as far as I am aware, those for single characters. I have occasion to use fairly frequently the notations d^0 and d^1 . I do not know how these would look in the family used to print *TUGboat*, but in the cmmi font the first of these comes out with the top of the *d* running into the 0. Since the 1 is thinner, this doesn’t happen. And of course, as it happens, *d* is one of only three characters in the lowercase Roman alphabet that have an ascender sticking out that far to the right (*l* and *f* being the others). If I understand rules 17 and 18 of page 445 of *The T_EXbook* correctly, an italic correction is added between a character and a superscript. But the italic correction is set globally in a font and it seems clear that a bit more is needed for those three letters when they have a superscript.

A completely different example is provided by my experience in making a minus sign with a dot on it. Try as I might, I could not get the dot low enough. Eventually, I asked T_EXhax and got an answer from Barbara Beeton. For some reason Knuth gave all the standard arithmetic operators the same height as the largest, which is probably the plus. The result is that a dot on the minus comes out at the same height as it would on a plus and, of course,

looks awful. The definition I now uses \smash and then gives the minus sign the (completely arbitrary, as far as I am concerned) height of 0.55ex. My feeling is that what Knuth did was an error in judgment, but that is not my point here. If the user had control over these things, then the height of the minus could have been left at its natural height and defined as being the height of the plus any time that was needed. The reason I think Knuth was in error is that you can make a box containing the minus whose height is that of the plus, but given that the tfm entry for the minus gives it the height of the plus, there is no way of getting its natural height back. You simply have to guess a number like 0.55ex, which is bad for a number of reasons. It might be wrong, it might not be correct in a different sized font, depending on how that size was selected and it might not be right in a different family.

Features that require new device drivers

Diagonal rules. Traditional typesetting didn’t have anything like diagonal rules, but it would be extremely helpful if T_EX went beyond traditional typesetting here. To some extent, the L^AT_EX line fonts compensate for this, but only partly and unsatisfactorily. First off, the number of different slopes is severely limited. Only 26 slopes are allowed (including horizontal and vertical) and arrowheads are available at only 14 of them. This isn’t so limiting; what is more serious is the fact that the shortest segment available at any oblique slope is much too long. I have been trying to implement diagonal dashed lines (and arrows), but the shortest segments available are much too long and it will have to be done with dots. This is inefficient both in time and in memory.

Opaque boxes. It doesn’t come up often, but every once in a while I feel the need to be able to place one box opaquely over another. I don’t even know if this is possible in either HP printer control language or PostScript, but it would be awfully handy if it were. One example of where this could be used would be if one box had an arrow and a second had a label for that arrow in a suitably sized box that you wanted to cover part of the arrow.

Documentation

I find *The T_EXbook* pretty good for the most part, but people unused to programming mostly find it impenetrable. But the story for L^AT_EX is much worse. It has seriously retarded the adoption of L^AT_EX as a standard. Several of my colleagues tell me they won’t use L^AT_EX because ‘using L^AT_EX you

can't do X' . In every case, you can do X often more easily than you can in plain. But it is not documented anywhere. Our office staff mostly use plain \TeX because they find the \LaTeX book so uninformative. As difficult as they find *The \TeXbook*, they feel they can eventually get the information out of it, but it just isn't there in the \LaTeX manual. Of all its deficiencies, the worst is the paucity of examples. The situation is somewhat better in French and German, and one of our secretaries makes good use of Raymond Seroul's book, *Le petit Livre de \TeX* [InterEditions, 1989, ISBN 2-7296-0233-X]. A somewhat expanded version, by Raymond Seroul and Silvio Levy, has now appeared in English: *A Beginner's Book of \TeX* [Springer Verlag, 1991, ISBN 0-387-97562-4]. Leaving all other considerations aside, I consider \LaTeX far superior to plain because it encourages you to think of a document in logical, not page layout terms. The criticisms of the diagram mode and $\backslash\text{put}$ above are precisely because they are such a departure from that ideal. Lamport actually suggests laying your diagrams out on graph paper before entering them. This is absurd. I have coauthored two books using \TeX and they each include several hundred diagrams.

Conclusions

I have successfully used \TeX for books, papers and even routine letters. I find it much easier to use than the standard text processors. Nonetheless, I find it has some deficiencies. Since Knuth has decided that \TeX will remain static, the time has come to think of a possible successor. I have set out above some of the possible directions in which change might come. Some of them might be done by a few modifications to the language that would leave the dvi output format unchanged. These could be accomplished by modifications to the underlying language, but would leave all device drivers and previewers current. However, some of the changes would require new device drivers which would render many of our auxiliary tools obsolete.

When \TeX was written the computing power available to the average user was much less. Freed from such limitations, we can now hope for a language that is a lot more powerful and easier to use. I hope to see a successor to \TeX that is worthy of its predecessor.

Since the first draft of this paper was written, there has been a new development. A formal network, called NTS-L ("New Typesetting System List") has been set up to discuss the question of a successor to \TeX . All issues are up for discussion. Should this new language be an incremental

improvement to \TeX or a new beginning? Should it be upward compatible? Should it be aimed at microcomputers or only for workstations and larger? Even, should it make a pass at being WYSIWYG? The debate is wide-ranging and sometimes heated. Anyone interested should subscribe. Send email to listserv@vm.urz.uni-heidelberg.de with a one line message
subscribe nts-l {Your Name Here}.

◊ Michael Barr
Department of Mathematics and
Statistics
McGill University
Montreal, Quebec, Canada
barr@math.mcgill.ca

Approaching SGML from \TeX

Reinhard Wonneberger

Abstract

The present memorandum intends to encourage discussion on a pragmatic \TeX approach to SGML.

It assumes a basic knowledge about SGML and builds on [WM92], which also contains bibliographic information.

Comments and contributions are welcome.

Situation

§ 1 Concern

Although \TeX has become a *de facto* standard by now, the corresponding General Markup language \LaTeX cannot claim to be a standard.

This implies severe limitations in using \TeX outside the academic world.

Such limitations might be overcome by combining \TeX with an accepted General Markup standard, which seems to be SGML.

§ 2 καιρός (*time of opportunity*)

The present development project of a new \LaTeX gives the unique chance to introduce a new Markup Language instead of staying frozen in upward compatibility.

§ 3 Conclusion

The community of \TeX users, esp. the implementors and other wizards, are encouraged to think about

the far-reaching consequences of the present chance and to actively pursue the project of approaching SGML from *T_EX*. I suggest an active approach to SGML to the *T_EX* Implementors Community, i.e. those colleagues who actively participate in *T_EX* implementation, adaptation, and development.

Suggestions

§ 4 *T_EX-based Implementation*

Rather than following the official approach of using a parser, the first concern should be to implement a *T_EX* format which is capable of interpreting one of the general SGML Document Type Definitions (DTD).

§ 5 *Backing*

This suggestion is based on the assumption that *T_EX* might be a well-suited implementation language. First implementation experiments seem to be encouraging.

§ 6 *Possible Steps*

The project might advance in the following steps:

1. Implement interpretation of a general DTD.
2. Implement document structure validation.
3. Implement definition syntax of SGML.

§ 7 *L_AT_EX*

If the first step could be completed successfully, the SGML general DTD might be offered either as the future *L_AT_EX* user interface or as an additional one.

Benefits

§ 8 *Savings*

The following benefits are anticipated:

1. Elimination of unnecessary parsing software if not required;
2. Elimination of unnecessary parse processing if not required.

§ 9 *Standardization*

SGML processing could inherit most of the advantages of *T_EX* itself, especially

1. vendor independence;
2. portability of the software;

All this could help to avoid a split of user worlds between SGML and *T_EX*.

[WM92] Reinhard Wonneberger and Frank Mittelbach. SGML. Questions and answers. *TUGboat* 13(2):221 (July 1992).

◊ Reinhard Wonneberger
EDS Electronic Data Systems
(Deutschland) GmbH
Eisenstraße 56 (N15)
D-6090 Rüsselsheim
Federal Republic of Germany
wonneberger @
mzdmza.zdv.uni-mainz.de

Abstracts

Les Cahiers GUTenberg Contents of Recent Issues

Numéro 12 - Décembre 1991

B. GAULLE, Éditorial : à propos d'erratum;
pp. 1-2

The President of Gutenberg remarks on the success of the special issues of the *Cahiers* (the proceedings of EuroTEX and GUTenberg'91 and "Premiers pas en L_AT_EX") and corrects some misconceptions regarding the use of *T_EX*, SGML, typographic style, and *T_EX* in Europe.

E. GÖPELT & B. SCHMID, WYSIWYG-*T_EX*-editors on the basis of object-oriented system technology;
pp. 3-12

This paper describes the motivation for and planned implementation of a WYSIWYG editor for the COMPINDAS (Computerized Integrated Data Base Production System) of FIZ Karlsruhe.

Michael SPIVAK, L_AM_S-*T_EX*: A Public Domain Document Preparation System Extended
*A*_M_S-*T_EX*; pp. 13-20

*L_AM_S-*T_EX** provides three basic extensions to *A*_M_S-*T_EX*:

- (1) As the 'L' in the name implies, *L_AM_S-*T_EX** provides the functionality of *I_AT_EX*, including (a) automatic numbering, together with symbolic labelling and cross-referencing, for equation numbers, lists, chapter and section headings, figure captions, theorems, lemmas, etc., etc.; (b) automatic placement of floating figures; (c) automatic table of contents generation and tools for creating an index; (d) literal mode; and (e) bibliographies (including interfacing with *BIBT_EX*, if desired). However the approach is rather different, with syntax that is generally much more concise, and *designed to provide the user with much greater flexibility*.
- (2) There are special macros, and extra fonts, for easily producing complicated commutative diagrams; the results are at least as good as those found in any professional books and journals. There are also special macros for partitioned matrices and "bordered matrices".
- (3) Finally, extensive table macros provide all the special refinements expected from professional typesetters.

Pierre MACKAY, Un regard sur les pixels.

Obtention de fontes de qualité pour imprimantes à laser à 300 dpi grâce à METAFONT; pp. 21–35

Two otherwise identical documents printed at the industry quasi-standard medium resolution of 300 dots/inch on laser printers can appear very different depending on whether a “write-black” or a “write-white” engine was used to print them. Most font-design and font expression systems appear to favor “write-black” technology, and there is some reason to suspect that “write-white” will never be entirely satisfactory. In any case, it is a good idea for the designer who expects to see a great deal of 300 dots/inch output to be aware of the difficulties involved in trying to support both technologies with the same design.

[Editor’s note: This paper is a french translation of “Looking at the Pixels. Quality Control for 300 dpi Laser Printer Fonts, Especially METAFONTS” in *Raster Imaging and Digital Typography II* (R. MORRIS & J. ANDRÉ eds.), Cambridge University Press, 1991, 205–217.]

Disquettes Euro-OzTEX available from Association GUTenberg; Adaptation française: Yannis Haralambous; p. 36

Michel GOOSSENS and Eric VAN HERWIJNEN, Introduction à SGML, DSSSL et SPDL; pp. 37–56

This article provides an introduction to ISO Standard 8879 SGML, the “Standard Generalized Markup Language” and discusses its relation with two other standards being drafted in the area of electronic document description, DSSSL for the page layout and SPDL for the visual presentation.

Jacques ANDRÉ and Philippe LOUARN, Notes en bas de pages : comment les faire en LATEX?; pp. 57–70

Some facilities with LATEX’s footnotes are exhibited, such as how to call footnotes from tabular array or how to refer the same note from different places.

Alexander SAMARIN and Anatoliy URVANTSEV, CyrTUG, le monde TeX en cyrillique; pp. 71–73

This article presents an overview of publishing in the (former) USSR, how TeX fits into this environment, and a report on the CyrTUG organization, its structure and goals.

[Editor’s note: This report was originally presented at EuroTeX’91.]

Hanna KOŁODZIEJSKA, TeX en Pologne; pp. 74–77

This is a report on the history and current use of TeX in Poland, including commercial activities a characterization of the user population.

[Editor’s note: This report was originally presented at EuroTeX’91 under the title *TeX in Poland*.]

Jacques ANDRÉ *et alii*, Lu, vu, ou entendu; pp. 78–83

Short takes on varied topics, with bibliographic information on recent books and other publications: orthography reform; typography; periodicals; publications prepared with, or about (La)TeX; conferences.

Table des matières de 1991; pp. 84–85

Baskerville

Contents of Recent Issues

Volume 2, Number 1, March 1992

Philip Taylor, Colophon; Editorial; p. 1

The editor of this, the second issue of The Annals of the UK TeX Users’ Group, describes its production, and relates how he has handled articles that were received during the extended lapse between the premier issue and the present one.

Peter Abbott, Chairman’s Report; pp. 1–2

The present Chairman of the UK TeX Users’ Group recounts what has happened in the U.K. TeX world since the first issue of *Baskerville* appeared, and directs some words of encouragement to the readers, announcing the editor of the next issue, Sue Brooks.

Malcolm Clark, TUG in Europe and Amerika; pp. 2–7

The present president of TUG and former chairman of the UK TeX Users’ Group gives an account of “the range of ‘organised’ TeX related activities in the world, mainly concentrating on the known national and language groups.” The groups reviewed are those representing the German-speaking, Japanese, French-speaking, Nordic, and Dutch-speaking TeX users groups and those in the United Kingdom, Czechoslovakia, Hungary, Poland, Jugoslavia, Soviet Union, Mexico, and Ireland. This

is followed by remarks on common themes which recur throughout all the groups.

The final section of the article deals with the recent history of TUG and *TeX*, in particular the upheaval which resulted in Malcolm's selection as interim president of TUG, and with his view of the future.

This paper was originally presented at the February 1991 Dante meeting in Vienna.

Malcolm Clark, The Outgoing Chairman's Report; pp. 7–10

This review opens with the statement "The group's second year can be summarised in a very similar way to the first—'a measure of success, leavened with a few disappointments'." A summary of the year's activities begins with short descriptions of the meetings: a very wide range of topics was covered, at one-day meetings that are relatively easy to attend owing to the compact geographical area involved. Various other services are reviewed, both those specific to the group and some offered jointly with other groups. The article ends with comments on the future and some personal observations.

Chris Rowley, Gleanings Past and Present; p. 10

This short article delves into the first issue of *TUGboat* to recover some of Knuth's thoughts on *TeX*'s user interface. It then relates some comments made on and offstage at a recent Monotype Conference in London.

Chris Rowley and Frank Mittelbach,
The *LATEX3* Project; pp. 10–11

This is the text of a proposal to the TUG Board of Directors for support of the *LATEX3* project.

[Editor's note: A slightly modified version appeared in *TeX* and *TUG News*, Vol. 1, No. 1.]

Chris Rowley, The 1990 A.G.M.; pp. 11–12

The official report of the Annual General Meeting of the UK *TeX* Users Group, held at Aston University on Wednesday, 17 October 1990.

Chris Rowley, The 1991 A.G.M.; p. 12

The official report of the Annual General Meeting of the UK *TeX* Users Group, held at Aston University on Wednesday, 17 October 1991.

Philip Taylor, Postscript; p. 12

Final comments on production of the issue, plus the editor's best wishes to Sue Brooks, who assumes the editorship with the next issue.

Late-Breaking News

Production Notes

Barbara Beeton

Input and input processing

Electronic input for articles in this issue was received by mail, on diskette, and was also retrieved from remote sites by anonymous ftp. In addition to text, the input to this issue includes METAFONT source code and several encapsulated PostScript files. For one article, which was based on an extended implementation of *TeX*, several illustrations were received on paper to be pasted in (see the "output" section). Most articles as received were fully tagged for *TUGboat*, using either the plain-based or *LATEX* conventions described in the Authors' Guide (see *TUGboat* 10, no. 3, pages 378–385). Several authors requested copies of the macros (which we were happy to provide); however, the macros have also been installed at labrea.stanford.edu and other good archives, and an author retrieving them from an archive will most likely get faster service. Of course, the TUG office will provide copies of the macros on diskette to authors who have no electronic access.

Font work was required for the article by salomon on arrows (p. 146).

The article by Rahtz and Barroca incorporates several (encapsulated) PostScript images, and was also most reliably processed using the New Font Selection Scheme; camera copy for this article only was output on the Math Society's Compugraphic 9600 Imagesetter.

About 50% of articles and 60% of the pages in this issue were prepared using *LATEX*.

In organizing the issue, attention was given to grouping bunches of plain or *LATEX* articles, to yield the smallest number of separate typesetter runs, and the least amount of handwork pasting together partial pages. This also affected the articles written or tagged by the staff, as the conventions of *tugboat.sty* or *ltugboat.sty* would be chosen depending on what conventions were used in the preceding and following articles; no article was changed from one to the other, however, regardless of convenience.

Test runs of articles were made separately and in groups to determine the arrangement and page numbers (to satisfy any possible cross references). A file containing all starting page numbers, needed

in any case for the table of contents, was compiled before the final run. Final processing was done in 3 runs of *TEX*, 2 of "old" *LATeX*, and 1 of *LATeX* incorporating the NFSS.

The following articles were prepared using the plain-based *tugboat.sty*:

- all articles in General Delivery.
- R.M. Damerell, *Knuth's profiler*, page 139.
- David Salomon, Arrows, page 146.
- Daniel Levin, ... the color separation problem, page 150.
- Philip Taylor, Book review: Victor Eijkhout, *TEX by Topic*, page 185.
- Péter Huszár, Over the multi-column, page 192.
- abstracts of the *Cahiers GUTenberg*, page 227.
- abstracts for *Baskerville*, page 228.
- the TUG calendar, page 231.
- announcement of Euro \TeX 92 in Prague, page 232.
- these Production notes
- "Coming next issue"

Output

The bulk of this issue was prepared at the American Mathematical Society from files installed on a VAX 6320 (VMS) and *TEX*'ed on a server running under Unix on a Solbourne workstation. Most output was typeset on an APS- μ 5 at the AMS using resident CM fonts and additional downloadable fonts for special purposes. The one exception was the article by Rahtz and Barroca mentioned earlier.

One photograph, photographically screened in the traditional manner, appears in the announcement of Knuth's degree (p. 134). The large arrows in the Salomon article (p. 146) are METAFONT proof output printed on an Imagen 5320 laser printer at 300 dpi. The gray-scale illustrations in the article by Levin (p. 150) were provided by the author as 300 dpi laser printer output and pasted in.

The output devices used to prepare the advertisements were not usually identified; anyone interested in determining how a particular ad was prepared should inquire of the advertiser.

Coming Next Issue

Anchored Figures at Either Margin

A figure in a box can be placed in text at one margin or the other, by measuring the box and adjusting the paragraph shape parameters so as to allow room for it. Macros that try to accomplish this automatically must be resourceful enough to decide what to do in a variety of special circumstances; the correctness or appropriateness of each decision depends on the requirements of the user. Daniel Comenetz presents his solution to the problems that arise in mathematics texts.

ZzTeX: A macro package for books

Paul Anagnostopoulos describes the design decisions behind a macro package intended to produce books to varying specifications with a minimum of macro modification. A book is considered as a structure of blocks, each of which may contain independent design specifications as well as specs governing the interaction of adjacent or nested blocks. All the usual features of scientific and scholarly books are supported, including cross-referencing and indexing. [Delayed by technical difficulties.]

A Multimedia Document System Based on *TEX* and DVI Documents

R. A. Vesilo and A. Dunn examine the development of a multimedia document system based on *TEX*. Multimedia document systems involve many complex components including editors, formatters, display systems and components to support the different media. By using *TEX* to do the formatting, using a standard text editor to enter the document text contents and define the document structure, and modifying a DVI previewer to include support for non-text contents, the amount of effort required to develop a multimedia document system is greatly reduced.

XBibTeX and Friends

Support facilities to make *BIBTeX* input more straightforward and reliable are described by Chris Bischof. [Delayed by technical difficulties.]

Calendar

1992

- Jun/Jul ukTeXug: "Design Issues": A visit to the Dept. of Typography, University of Reading.
For information, contact Peter Abbott (pabbott@nsfnet-relay.ac.uk).
-
- TUG'92 Conference, Portland, Oregon**
- Jul 20–24 Intensive Beginning/Intermed. TeX
- Jul 26 TeX for Publishers
- Jul 27–30 **TUG Annual Meeting: "TeX in Context"**, Portland, Oregon.
For information, contact the TUG office. (See page opposite inside back cover.)
- Jul 31 –
Aug 1 Practical SGML and TeX
Graphic Design
- Aug 3–7 Advanced TeX and Macro Writing
Intensive LATEX
-
- Aug 18 **TUGboat Volume 13,**
3rd regular issue:
Deadline for receipt of technical manuscripts.
- Sep 3–4 DANTE e.V.: General Meeting, Clausthal-Zellerfeld, Germany
For information, contact S. J. Šarman (rzsys@ibm.rz.tu-clausthal.de).
- Sep 14–16 EuroTeX 92, Prague, Czechoslovakia.
For information, contact Jiří Veselý (jvesely@cspguk11.bitnet).
(See *TUGboat* 13, no. 1, p. 107.)
- Sep 14–16 Astronomy from Large Databases.
Hagenau, France.
Topics include tools for integration of bibliographic and textual information.
For information, contact Dr. André Heck (Bitnet: Heck@FRCCSC21, or +33-88.35.82.22).
- Sep 15 **TUGboat Volume 13,**
3rd regular issue:
Deadline for receipt of news items, reports.

- Sep 22 UK TeX Users' Group joins the British Computer Society Electronic Publishing Group, Nottingham, England.
Topic: Structured Documents.
Host: Prof. David Brailsford.
For information, contact Carol Hewlett (hewlett@vax.lse.ac.uk).
- Oct 14 UK TeX Users' Group, Annual General Meeting, Aston University, Birmingham, England. Followed by a session on "LATEX, AMS-Tex and LAMS-Tex compared for setting maths". For information, contact Carol Hewlett (hewlett@vax.lse.ac.uk).
- Oct 19–23 Intensive Beginning/Intermed. TeX, Chicago, Illinois.
- Oct 26–30 Intensive LATEX, San Diego, California.
- Nov 2–6 Intensive Beginning/Intermed. TeX, San Diego, California.
- Nov 9–13 Intensive LATEX, Providence, Rhode Island.
- Nov 17 **TUGboat Volume 14,**
1st regular issue:
Deadline for receipt of technical manuscripts (tentative).
- Nov 19 NTG Fall Meeting,
"LATEX, LATEX 3, and font selection", Meppel (near Groningen), The Netherlands. For information, contact Gerard van Nes (vannes@ECN.NL).
- Nov 24 **TUGboat Volume 13,**
3rd regular issue:
Mailing date (tentative).
- Dec 15 **TUGboat Volume 14,**
1st regular issue:
Deadline for receipt of news items, reports (tentative).

1993

Feb	UK <i>TeX</i> Users' Group, London. Topic: Front ends for <i>TeX</i> ; how successful are the WYSIWYG packages for non- <i>TeX</i> users and for wizards? For information, contact Carol Hewlett (<i>hewlett@vax.lse.ac.uk</i>).		Jun 6 - 9	Society for Technical Communication, 40 th Annual Conference, Dallas, Texas. For information, contact the Society headquarters, 901 N. Stuart St., Suite 304, Arlington, VA 22203-1822. (703-522-4114; Fax: 703-522-2075) Proposals for presentations due by August 1, 1992.
Feb 16	<i>TUGboat</i> Volume 14, 2nd regular issue: Deadline for receipt of technical manuscripts (tentative).		Aug 17	<i>TUGboat</i> Volume 14, 3rd regular issue: Deadline for receipt of technical manuscripts (tentative).
Feb 24 - 27	CONCEPPTS 93, The Prepublishing Conference, Orange County Convention Center, Orlando, Florida. "International Conference on Computers and Electronic Publishing and Printing Technologies". For information, phone: 703-264-7200, Fax: 703-620-9187.		Sep 14	<i>TUGboat</i> Volume 14, 3rd regular issue: Deadline for receipt of news items, reports (tentative).
Mar 9	<i>TUGboat</i> Volume 14, 1st regular issue: Mailing date (tentative).		Nov 23	<i>TUGboat</i> Volume 14, 3rd regular issue: Mailing date (tentative).
Mar 9 - 12	DANTE'93 and General Meeting, Chemnitz, Germany. For information, contact Dr. Wolfgang Riedel (<i>wolfgang.riedel@hrz.tu-chemnitz.de</i>).			For additional information on the events listed above, contact the TUG office (401-751-7760, email: <i>tug@math.ams.com</i>) unless otherwise noted.
Mar 16	<i>TUGboat</i> Volume 14, 2nd regular issue: Deadline for receipt of news items, reports (tentative).			
Mar	UK <i>TeX</i> Users' Group, Glasgow, Scotland. (Two days, just before the BCS EPSG meeting; postponed from April 1992.) Topics: METAFONT, theoretical and practical; and font selection schemes, virtual fonts, multiple languages and hyphenation, etc.—everything you need to know to use <i>TeX</i> to typeset foreign languages. For information, contact Carol Hewlett (<i>hewlett@vax.lse.ac.uk</i>).			
May	UK <i>TeX</i> Users' Group, Chichester, England. Visit to John Wiley & Sons Ltd. Host: Geeti Granger. For information, contact Carol Hewlett (<i>hewlett@vax.lse.ac.uk</i>).			
May 25	<i>TUGboat</i> Volume 14, 2nd regular issue: Mailing date (tentative).			

**EuroTeX'92, Praha
14–18 September 1992**

As already announced, EuroTeX'92 is organized by the Czechoslovak *TeX* Users Group in collaboration with Charles University and the Czech Technical University, Prague, under the auspices of both Rectors. It takes place in Prague from September 14 through 18, 1992.

The Programme Committee consists of Peter Abbott, Jacques André, Jana Chlebikova, Yannis Haralambous, Bernard Gaulle, Karel Horak, Joachim Lammarsch, Kees van der Laan, Erich Neuwirth, Petr Novak, Stefan Porubsky, Phillip Taylor, Jiří Veselý and Jiří Zlatuška.

Invited speakers include Yannis Haralambous, John Hobby, Alan Hoenig, Anita Hoover, Chris Rowley, Daniel Taupin, Philip Taylor. They will cover topics including the use of *TeX* for "non-standard" languages, typesetting music, Metafont-like language for PostScript drawings, special effects with METAFONT and *TeX*, the LATEX3 project, aspects of the use of *TeX* and LATEX at University, WEB, and even more ...

Tutorials delivered after the conference by Yannis Haralambous, Phil Taylor and Klaus Thull will cover METAFONT and advanced topics in T_EX, as well as an introduction to WEB.

Any questions concerning the programme can be directed to the Programme Committee chairman, Jiří Zlatuška (zlatuska@cspuni12.bitnet).

Why come to EuroTEX 92?

Besides the privilege of hearing the invited talks of leading specialists you will have the pleasure of listening to other lectures and meeting T_EX friends from many countries. The meeting is the first to offer really extensive contacts with people "from behind the iron curtain". It takes place in the Golden Heart of Europe — Prague, one of the most fascinating capitals in Europe. You can visit it at a surprisingly low cost. Indeed, we would like to make EuroTEX'92 in Prague accessible to the majority of T_EX fans from all over the world. Please note that low prices do not mean a compromise in quality; rather, they take advantage of the favourable exchange rates applicable to the "hard currency countries".

Accommodation is booked in a modern student hostel Kajetanka in double rooms. We plan to arrange transport by bus to Czech Technical University, where the programme will be held (it is a 25 minute walk from Kajetanka). All participants will be provided with a card for all Prague public transport during the period Monday–Friday.

Lunches will be served at the conference site; during breaks some refreshments will be available. Dinners are not included so that you can research Czech restaurants and pubs on your own. An informal welcome party will be held on Monday evening. An organ concert (probably on Wednesday) and some other pleasant surprises can be expected. For those arriving on Sunday or on Monday morning, a sightseeing tour on Monday afternoon is under negotiation. The whole programme from Monday to Friday forms a package (accommodation in double rooms, half pension from Tuesday to Friday, opening party on Monday evening, concert, conference fee, proceedings, tutorials for those who can stay a bit longer) for 330 DM (60 DM extra for a single room). For an additional 35 DM (45 DM for a single room) a day, a limited number of participants may stay one or two days more (until Sunday) either for tutorials or just to enjoy meeting friends and to have good beer in some of the pubs, such as Good Soldier Svejk liked. A special programme will be organized for accompanying persons,

Climate. Since much of Prague's fascination is historical, architectural and cultural, it can be enjoyed at any time. The average maximum temperature in September is 18° C (64° F) and the weather is relatively stable.

Currency. The Czechoslovak Crown is rated approx. 17:1 to DM, or 28:1 to US\$. The present prices of goods are higher, and they are slowly approaching "western standards", but in many respects Czechoslovakia is considered favorable and cheap for western tourists.

Transport. Czechoslovakia is easily accessible by plane. The Prague airport is about 15 km from the city centre (the Kajetanka hostel is even closer). Public transport is relatively cheap, but taxi fares are better agreed on beforehand since prices are not fixed. Roads are relatively good but with only a few motorways which are not so fast as in the West (speed limits: 110 km/h on motorways, 90 on roads, 60 in towns). Parking in Prague is generally difficult. There are places to park in the neighbourhood of the hostel, but not a (guarded) parking. International trains connect Prague with Berlin, Munich, Nurnberg, Wien, Warszawa, Budapest, etc.

Payment. Those deciding to take part are requested to send money via the following account: 34735-021/0100 at KOMERČNÍ BANKA, PRAHA (there is one "hacek" and one "prime" accent in its name: please, do not forget to use them). The address of the bank is

KOMERČNÍ BANKA,
pob. Praha – MĚSTO,
Vaclavske nam. 42
110 00 PRAHA 1
Czechoslovakia

while the name of the account is

Československé sdružení uživatelů TEXu
(our surface address (in Czech): Sokolovská 83, 186 00 Praha 8, Czechoslovakia).

If you are submitting payment, you should also fill out a registration form. You may obtain a form by sending your name and a valid email address to: eurotex@cspguk11.bitnet

(Copies of the form may also be obtained on request from the TUG office, address on page 131.) Please indicate how you have transmitted the payment to the above account, and, if possible, include a xerocopy of the order. (It could help us to trace a lost payment.) We were told that the important thing is to use the **Czech** name of the bank mentioned above since its translation might cause some mistakes.

TEXniques

Publications for the TEX Community

Available now:

1. **VAX Language-Sensitive Editor (LSEDIT)**
Quick Reference Guide for Use with the LATEX Environment and LATEX Style Templates by Kent McPherson
2. **Table Making – the INRSTEX Method** by Michael J. Ferguson
3. **User's Guide to the IdxTEX Program** by R. L. Aurbach
4. **User's Guide to the GloTEX Program** by R. L. Aurbach
5. **Conference Proceedings, TEX Users Group Eighth Annual Meeting, Seattle, August 24–26, 1987, Dean Guenther, Editor**
6. **The PICTEX Manual** by Michael J. Wichura
7. **Conference Proceedings, TEX Users Group Ninth Annual Meeting, Montréal, August 22–24, 1988, Christina Thiele, Editor**
8. **A Users' Guide for TEX** by Frances Huth
9. **An Introduction to LATEX** by Michael Urban
10. **LATEX Command Summary** by L. Botway and C. Biemesderfer
11. **First Grade TEX** by Arthur Samuel
12. **A Gentle Introduction to TEX** by Michael Doob
13. **METAFONTware** by Donald E. Knuth, Tomas G. Rokicki, and Arthur Samuel
14. **A Permuted Index for TEX and LATEX** by Bill Cheswick

In production:

15. **EDMAC: A Plain TEX Format for Critical Editions**
by John Lavagnino and Dominik Wujastyk

**TEX Users Group
P.O. Box 9506
Providence, R.I. 02940, U.S.A.**



Complete and return this form with payment to:

TEX Users Group
 Membership Department
 P.O. Box 594
 Providence, RI 02901 USA
 Telephone: (401) 751-7760
 FAX: (401) 751-1071
 Email: tug@Math.AMS.com

Membership is effective from January 1 to December 31 and includes subscriptions to *TUGboat*, *The Communications of the T_EX Users Group* and the TUG newsletter, *T_EX and TUG News*. Members who join after January 1 will receive all issues published that calendar year.

For more information ...

Whether or not you join TUG now, feel free to return this form to request more information. Be sure to include your name and address in the spaces provided to the right.

Check all items you wish to receive below:

- Institutional membership information
- Course and meeting information
- Advertising rates
- Products/publications catalogue
- Public domain software catalogue
- More information on T_EX

Individual Membership Application

Name _____

Institutional affiliation, if any _____

Position _____

Address (business or home (circle one)) _____

City _____

State or Country _____ Zip _____

Daytime telephone _____ FAX _____

Email addresses (*please specify networks, as well*) _____

I am also a member of the following other T_EX organizations:

Specific applications or reasons for interest in T_EX:

Hardware on which T_EX is used:

Computer and operating system *Output device/printer*

There are two types of TUG members: regular members, who pay annual dues of \$60; and full-time student members, whose annual dues are \$50. Students must include verification of student status with their applications.

Please indicate the type of membership for which you are applying:

- Regular @ \$60 Full-time student @ \$50

Amount enclosed for 1992 membership: \$ _____

(Prepayment in US dollars drawn on a US bank is required)

- Check/money order payable to T_EX Users Group enclosed
 Charge to MasterCard/VISA

Card # _____ Exp. date _____

Signature _____

TEX USERS GROUP

Complete and return this form with payment to:

TEX Users Group
Membership Department
P.O. Box 594
Providence, RI 02901 USA

Membership is effective from January 1 to December 31. Members who join after January 1 will receive all issues of *TUGboat* published that calendar year.

For more information ...

Correspondence

TEX Users Group
653 North Main Street
P.O. Box 9506
Providence, RI 02940
USA

Telephone: (401) 751-7760
Fax: (401) 751-1071
Email: tug@math.ams.com

Whether or not you join TUG now, feel free to return this form to request more information.

Check all items you wish to receive below:

- Course and meeting information
- Products/publications catalogue
- Public domain software catalogue

Institutional Membership Application

Institution or Organization _____

Principal contact _____

Address _____

City _____

State or Country _____ Zip _____

Daytime telephone _____ FAX _____

Email addresses (*please specify networks, as well*) _____

Each Institutional Member entitles the institution to:

- designate a number of individuals to have full status as TUG individual members;
- take advantage of reduced rates for TUG meetings and courses for *all* staff members;
- be acknowledged in every issue of *TUGboat* published during the membership year.

Educational institutions receive a \$100 discount in the membership fee. The three basic categories of Institutional Membership each include a certain number of individual memberships. Additional individual memberships may be obtained at the rates indicated. Fees are as follows:

Category	Rate (educ./non-educ.)	Add'l mem.
A (includes 7 memberships)	\$ 540 / \$ 640	\$50 ea.
B (includes 12 memberships)	\$ 815 / \$ 915	\$50 ea.
C (includes 30 memberships)	\$1710 / \$1810	\$40 ea.

Please indicate the type of membership for which you are applying:

Category _____ + _____ additional individual memberships

Amount enclosed for 1992 membership: \$ _____

Check/money order payable to TEX Users Group enclosed

(payment is required in US dollars drawn on a US bank)

Bank transfer bank _____

ref # _____

Charge to MasterCard/VISA

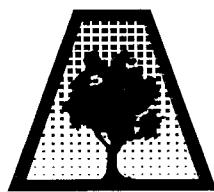
Card # _____ Exp. date _____

Signature _____

Please attach a corresponding list of individuals whom you wish to designate as TUG individual members. Minimally, we require names and addresses so that TUG publications may be sent directly to these individuals, but we would also appreciate receiving the supplemental information regarding phone numbers, email addresses, TEX interests, and hardware configurations as requested on the TUG Individual Membership Application form. For this purpose, the latter application form may be photocopied and mailed with this form.



And you thought that all we knew was TEX !



ARBORTEXT INC.

1000 Victors Way Suite 400 Ann Arbor, MI 48108 FAX (313) 996-3573 Phone (313) 996-3566

CAPTURE

Version 2

CAPTURE is a system of programs for inserting graphics into \TeX , on IBM™ PC systems with LaserJet™ (and PostScript™) printers.



- Acquire graphics from redirected printer output, the screen, or file conversion from TIFF and PCX formats.
- Convert graphics to and from PCX format for image editing.
- Convert graphics to PK/TFM format for inclusion in PostScript™ \TeX documents.



- Tested with PCTEX™, TurboTEX™, $\mu\text{\TeX}^{\text{TM}}$, and $\text{\TeX}^{\text{Plus}}^{\text{TM}}$; DOS™ 2-5, Windows™ 3.0-3.1, LaserJet II and LaserJet III.
- Optionally remove extra white space.
- *Scale graphics to virtually any size.*
- Comes complete with \TeX and \LaTeX insertion macros, printer capture, screen capture, and file conversion utilities.



CAPTURE is designed to work with \TeX . Graphics files are processed to remove all control codes and positioning commands that can disrupt \TeX . Graphics can be manipulated by \TeX and treated as any other font of type. Graphics and text are intermingled gracefully, using the full power of \TeX . Notice how the text wraps around the graphics in this advertisement; it was all done with \TeX . In addition, CAPTURE can convert graphics to the \TeX standard PK/TFM file format. Graphics literally become a \TeX type font and can be used with any \TeX platform. For example, IBM/PC graphics from application programs that support the LaserJet can be inserted into PostScript documents, or used with VMS or UNIX systems.

Version 1 Customers: Wynne-Manley is charging nothing to upgrade. Version 2 upgrade is available from Micro Programs (see below) for a nominal media and handling charge. And we thank you for your support.

Distributors:

Micro Programs, Inc.
251 Jackson Ave.
Syosset, NY 11791-4117
(516) 921-1351

Personal \TeX , Inc.
12 Madrona Ave.
Mill Valley, CA 94941
(415) 388-8853

\TeX Users Group
P.O. Box 9506
Providence, RI 02940
(401) 751-7760

For T_EX Users . . .

New Services and Prices from Computer Composition Corporation

We are pleased to announce the installation of several ***new output services*** now available to T_EX users:

1. High Resolution Laser Imaging (1200 dpi) from Postscript diskette files created on either Mac- or PC-based systems.
2. High Resolution Laser Imaging (960 dpi) from DVI magnetic tape or diskette files using a variety of typefaces in addition to the Computer Modern typeface family.
3. High quality laser page proofs at 480 dpi.
4. **NEW PRICING** for high resolution laser imaging:
 - a. From **Postscript text files** in volumes over 400 pages **\$2.00 per page**
 - b. From **Postscript text files** in volumes between 100 & 400 pages **\$2.25 per page**
 - c. From **Postscript text files** in volumes below 100 pages **\$2.40 per page**
 - d. From **DVI files** in volumes over 400 pages **\$2.15 per page**
 - e. From **DVI files** in volumes between 100 & 400 pages **\$2.30 per page**
 - f. From **DVI files** in volumes below 100 pages **\$2.45 per page**

NOTE: DEDUCT \$1.00 FROM THE ABOVE PRICES FOR HIGH QUALITY LASER PAGE PROOFS.

5. **All jobs shipped within 48 hours.**

Call or write for page samples or send us your file and we will image it on the output unit of your choice.



COMPUTER COMPOSITION CORPORATION

1401 West Girard Avenue • Madison Heights, MI 48071

(313) 545-4330 FAX (313) 544-1611

— Since 1970 —

T_EX Publishing Services



From the Basic:

The American Mathematical Society offers you two basic, low cost T_EX publishing services.

- You provide a DVI file and we will produce typeset pages using an Autologic APS Micro-5 phototypesetter. \$5 per page for the first 100 pages; \$2.50 per page for additional pages.
- You provide a PostScript output file and we will provide typeset pages using an Agfa/Compugraphic 9600 imagesetter. \$7 per page for the first 100 pages; \$3.50 per page for additional pages.

There is a \$30 minimum charge for either service. Quick turnaround is also provided... a manuscript up to 500 pages can be back in your hands in one week or less.

To the Complex:

As a full-service T_EX publisher, you can look to the American Mathematical Society as a single source for any or all your publishing needs.

Macro-Writing	T _E X Problem Solving	Non-CM Fonts	Keyboarding
Art and Pasteup	Camera Work	Printing and Binding	Distribution

For more information or to schedule a job, please contact Regina Girouard, American Mathematical Society, P. O. Box 6248, Providence, RI 02940, or call 401-455-4060.

FOR YOUR T_EX TOOLBOX

CAPTURE

Capture graphics generated by application programs. Make LaserJet images compatible with T_EX. Create pk files from pcl or pcx files. \$135.00

texpic

Use texpic graphics package to integrate simple graphics—boxes, circles, ellipses, lines, arrows—into your T_EX documents. \$79.00

Voyager

T_EX macros to produce viewgraphs—including bar charts—quickly and easily. They provide format, indentation, font, and spacing control. \$25.00

FOR YOUR T_EX BOOKSHELF

T_EX BY EXAMPLE

NEW!

Input and output are shown side-by-side. Quickly see how to obtain desired output. \$19.95

T_EX BY TOPIC

NEW!

Learn to program complicated macros. \$29.25

T_EX FOR THE IMPATIENT

Includes a complete description of T_EX's control sequences. \$29.25

T_EX FOR THE BEGINNER

NEW!

A carefully paced tutorial introduction. \$29.25

BEGINNER'S BOOK OF T_EX

A friendly introduction for beginners and aspiring "wizards." \$29.95



Micro Programs Inc. 251 Jackson Ave. Syosset, NY 11791 (516) 921-1351

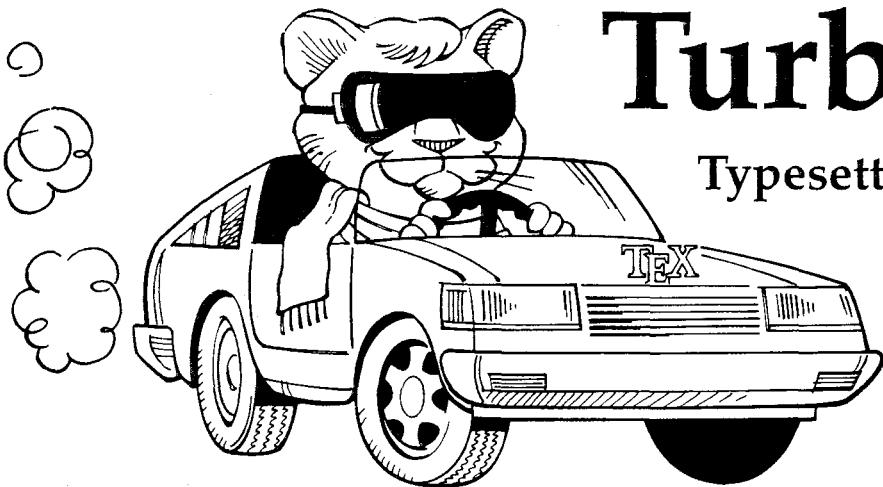
The solution is ETP.

$$\Delta \mathcal{P} = \sum_W \left[Q_{\text{IPR}} \int_{4-1-87}^{\infty} (D_p + D_m + D_s)^T + \epsilon(P_m - I_P) dt \right] \equiv \text{ETP}$$

ETP Services offers solutions to the problems facing the publishers of technical books and journals, with a complete array of composition-related services.

Electronic Technical Publishing Services Company

2906 N.E. Glisan Street
Portland, Oregon 97232
503-234-5522 • FAX: 503-234-5604
mimi@etp.com



TurboTeX

Typesetting Software

Executables \$150
With Source \$300



THE MOST VERSATILE \TeX ever published is breaking new ground in the powerful and convenient graphical environment of Microsoft Windows: TurboTeX Release 3.1E. TurboTeX runs on all the most popular operating systems (Windows, MS-DOS, OS/2, and UNIX) and provides the latest \TeX 3.14 and METAFONT 2.7 standards and certifications: preloaded plain \TeX , \LaTeX , \AmS-\TeX and \AmS-\LaTeX , previewers for PC's and X-servers, METAFONT, Computer Modern and \LaTeX fonts, and printer drivers for HP LaserJet and DeskJet, PostScript, and Epson LQ and FX dot-matrix printers.

■ Best-selling Value: TurboTeX sets the world standard for power and value among \TeX implementations: one price buys a complete, commercially-hardened typesetting system. *Computer* magazine recommended it as "the version of \TeX to have," *IEEE Software* called it "industrial strength," and thousands of satisfied users around the globe agree. TurboTeX gets you started quickly, installing itself automatically under MS-DOS or Microsoft Windows, and compiling itself automatically under UNIX. The 90-page User's Guide includes generous examples and a full index, and leads you step-by-step through installing and using \TeX and METAFONT.

■ Classic \TeX for Windows. Even if you have never used Windows on your PC, the speed and power of TurboTeX will convince you of the benefits. While the \TeX command-line options and $\text{\TeX}book$ interaction work the same, you also can control \TeX using friendly icons, menus, and

dialog boxes. Windows protected mode frees you from MS-DOS limitations like DOS extenders, overlay swapping, and scarce memory. You can run long \TeX formatting or printing jobs in the background while using other programs in the foreground.

■ MS-DOS Power, Too: TurboTeX still includes the plain MS-DOS programs. Virtual memory simulation provides the same sized \TeX that runs on multi-megabyte mainframes, with capacity for large documents, complicated formats, and demanding macro packages.

■ Source Code: The portable C source to TurboTeX consists of over 100,000 lines of generously commented \TeX , TurboTeX, METAFONT, previewer, and printer driver source code, including: our WEB system in C; PASCHAL, our proprietary Pascal-to-C translator; Windows interface; and preloading, virtual memory, and graphics code, all meeting C portability standards like ANSI and K&R.

■ Availability & Requirements: TurboTeX executables for IBM PC's include the User's Guide and require 640K, hard disk, and MS-DOS 3.0 or later. Windows versions run on Microsoft Windows 3.0 or 3.1. Order source code (includes Programmer's Guide) for other machines. On the PC, source compiles with Microsoft C, Watcom C 8.0, or Borland C++ 2.0; other operating systems need a 32-bit C compiler supporting UNIX standard I/O. Specify 5-1/4" or 3-1/2" PC-format floppy disks.

■ Upgrade at Low Cost. If you have TurboTeX Release 3.0, upgrade to the latest version for just \$40 (ex-

ecutables) or \$80 (including source). Or, get either applicable upgrade free when you buy the AP- \TeX fonts (see facing page) for \$200!

■ No-risk trial offer: Examine the documentation and run the PC TurboTeX for 10 days. If you are not satisfied, return it for a 100% refund or credit. (Offer applies to PC executables only.)

■ Free Buyer's Guide: Ask for the free, 70-page Buyer's Guide for details on TurboTeX and dozens of \TeX -related products: previewers, \TeX -to-FAX and \TeX -to-Ventura/Pagemaker translators, optional fonts, graphics editors, public domain \TeX accessory software, books and reports.

Ordering TurboTeX

Ordering TurboTeX is easy and delivery is fast, by phone, FAX, or mail. Terms: Check with order (free media and ground shipping in US), VISA, Mastercard (free media, shipping extra); Net 30 to well-rated firms and public agencies (shipping and media extra). Discounts available for quantities or resale. International orders gladly expedited via Air or Express Mail.

The Kinch Computer Company

PUBLISHERS OF TURBOTEX
501 South Meadow Street
Ithaca, New York 14850 USA
Telephone (607) 273-0222
FAX (607) 273-0484

AP-T_EX Fonts

TEX-compatible Bit-Mapped Fonts Identical to Adobe PostScript Typefaces

If you are hungry for new **TEX** fonts, here is a feast guaranteed to satisfy the biggest appetite! The AP-**TEX** fonts serve you a banquet of gourmet delights: 438 fonts covering 18 sizes of 35 styles, at a total price of \$200. The AP-**TEX** fonts consist of PK and TFM files which are exact **TEX**-compatible equivalents (including "hinted" pixels) to the popular PostScript name-brand fonts shown at the right. Since they are directly compatible with any standard **TEX** implementation (including kerning and ligatures), you don't have to be a **TEX** expert to install or use them.

When ordering, specify resolution of 300 dpi (for laser printers), 180 dpi (for 24-pin dot matrix printers), or 118 dpi (for previewers). Each set is on ten 360 KB 5-1/4" PC floppy disks. The \$200 price applies to the first set you order; order additional sets at other resolutions for \$60 each. A 30-page user's guide fully explains how to install and use the fonts. Sizes included are 5, 6, 7, 8, 9, 10, 11, 12, 14.4, 17.3, 20.7, and 24.9 points; headline styles (equivalent to Times Roman, Helvetica, and Palatino, all in bold) also include sizes 29.9, 35.8, 43.0, 51.6, 61.9, and 74.3 points.

The Kinch Computer Company
PUBLISHERS OF TURBOTEX
501 South Meadow Street
Ithaca, New York 14850
Telephone (607) 273-0222
FAX (607) 273-0484

Helvetica, Palatino, Times, and New Century Schoolbook are trademarks of Allied Linotype Co. ITC Avant Garde, ITC Bookman, ITC Zapf Chancery, and ITC Zapf Dingbats are registered trademarks of International Typeface Corporation. PostScript is a registered trademark of Adobe Systems Incorporated. The owners of these trademarks and Adobe Systems, Inc. are not the authors, publishers, or licensors of the AP-TeX fonts. Kinch Computer Company is the sole author of the AP-TeX fonts, and has operated independently of the trademark owners and Adobe Systems, Inc. in publishing this software. Any reference in the AP-TeX font software or in this advertisement to these trademarks is solely for software compatibility or product comparison. LaserJet and DeskJet are trademarks of Hewlett-Packard Corporation. TeX is a trademark of the American Math Society. TurboTeX and AP-TeX are trademarks of Kinch Computer Company. Prices and specifications subject to change without notice. Revised October 9, 1990.

Avant Garde	Bold
Avant Garde	Bold Oblique
Avant Garde	Demibold
Avant Garde	Demibold Oblique
Bookman	Light
Bookman	Light Italic
Bookman	Demibold
Bookman	Demibold Italic
Courier	
Courier	Oblique
Courier	Bold
Courier	Bold Oblique
Helvetica	
Helvetica	Oblique
Helvetica	Bold
Helvetica	Bold Oblique
Helvetica Narrow	
Helvetica Narrow	Oblique
Helvetica Narrow	Bold
Helvetica Narrow	Bold Oblique
Schoolbook	New Century Roman
Schoolbook	New Century Italic
Schoolbook	New Century Bold
Schoolbook	New Century Bold Italic
Palatino	Roman
Palatino	Italic
Palatino	Bold
Palatino	Bold Italic
Times	Roman
Times	Italic
Times	Bold
Times	Bold Italic
Zapf Chancery	Medium Italic
Symbol ΔΦΓΘΛΠΘ	
Zapf Dingbats	☒☒☒

Publishing Companion® translates

WordPerfect

to

T_EX or L_AT_EX

IN ONE EASY STEP!

With **Publishing Companion**, you can publish documents using T_EX or L_AT_EX with **little or no T_EX knowledge**. Your WordPerfect files are translated into T_EX or L_AT_EX files, so anyone using this simple word processor can immediately begin typesetting their own documents!

Publishing Companion translates EQUATIONS, FOOTNOTES, ENDNOTES, FONT STYLES, and much more!

Retail Price	\$249.00
Academic Discount Price	\$199.00

For more information or to place an order, call or write:



30 West First Ave, Suite 100
Columbus, Ohio 43201
(614)294-3535
FAX (614)294-3704

TYPESET QUALITY WITH THE EASE OF WORD PROCESSING

SPRINGER FOR T_EXNOLOGY

R. Seroul, Université Louis Pasteur,
Strasbourg, France; **S. Levy**, University of
Minnesota, Minneapolis, MN

A Beginner's Book of T_EX

This is a friendly introduction to T_EX, the powerful typesetting system developed by Don Knuth. It is addressed primary to beginners, but contains much information that will be useful to aspiring T_EX wizards. Moreover, the authors kept firmly in mind the diversity of backgrounds that characterize T_EX users: authors in the sciences and the humanities, secretaries, and technical typists. The book contains a wealth of examples and many "tricks" based on the authors' long experience with T_EX.

Contents: What is T_EX? • The Characteristics of T_EX • Groups and Modes • The Fonts T_EX Uses • Spacing, Glue and Springs • Paragraphs • Page Layout • Boxes • Alignments • Tabbing • Typesetting Mathematics • T_EX Programming • Dictionary and Index

1991/283 pp./Softcover \$29.95/ISBN 0-387-97562-4

**SPRINGER-VERLAG
NEW YORK, INC.**



S. v. Bechtolsheim, West Lafayette, IN

T_EX in Practice

A recent surge of good T_EX implementations for PC's has put T_EX on the disks of many people including writers, designers, desktop publishers, and engineers. With such increased interest in T_EX, there is a need for good T_EX books. **T_EX in Practice** is the ideal reference and guide for the T_EX community. The four-volume set is written by an acknowledged expert in the field and addresses the needs of the T_EX novice to the more experienced "T_EXpert." The book provides step-by-step introduction to the various functions of T_EX with many relevant examples.

Volume 1: Basics

1992/359 pp., 9 illus./Hardcover \$49.00
ISBN 0-387-97595-0

Volume 2: Paragraphs, Maths, and Fonts
1992/384 pp., 22 illus./Hardcover \$49.00
ISBN 0-387-97596-9

Volume 3: Tokens, Macros

1992/544 pp., 22 illus./Hardcover \$49.00
ISBN 0-387-97597-7

Volume 4: Output Routines, Tables

1992/300 pp., 10 illus./Hardcover \$49.00
ISBN 0-387-97598-5

FOUR-VOLUME SET

1992/\$169.00/ISBN 0-387-97296-X
Monographs in Visual Communication

To Order: Return this coupon with payment to Springer-Verlag New York, Inc, Attn: J. Jeng, 175 Fifth Avenue, New York, NY 10010, Or Call Toll-Free 1-800-SPRINGER (In NJ, call 201-348-4033).

Yes! Send me the following books:

____ ISBN 0-387-____
____ ISBN 0-387-____

Subtotal ____

* CA, MA, NJ, NY & VT residents add sales tax ____

Shipping \$2.50 (\$1.00 each additional book) ____

TOTAL ____

* In Canada, add 7% GST.

Name _____

Address _____

City _____

State _____ Zip _____

[] Check enclosed.

Charge my: [] VISA [] MC [] AmEx

[] Discover

Card No. _____

Exp. Date _____

Signature _____

**The TEX Users Group
is seeking applicants for the position of
Executive Director**

The individual selected for this position will oversee the business and information dissemination activities of TUG; direct the promotional program to develop membership and TUG activities; develop a program of volunteer efforts for TUG activities; manage a small office staff with clerical, technical, and bookkeeping functions; and interact with TUG members and others in fields of interest to TUG. The Executive Director will report to TUG Board of Directors.

The following criteria will be considered as applicants are evaluated:

- experience in managing a business;
- skill in managing the retrieval, organization and dissemination of information;
- experience with the program TEX and related programs;
- computer experience and capability of understanding technical questions regarding TEX and related programs;
- good writing and speaking skills;
- good interpersonal skills;
- knowledge of considerations in managing a professional, non-profit association.

Applicants for this position should send indication of their interest and copies of their *curricula vitae* to:

Search Committee
TEX Users Group
P. O. Box 9506
Providence, RI 02940 USA

The TEX Users Group is an Equal Opportunity Employer.

Make Your Best Work Look Its Best!

Name	Definition
Gamma	$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt$
Sine	$\sin(x) = \frac{1}{2i}(e^{ix} - e^{-ix})$
Error	$\text{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-z^2} dz$
Bessel	$J_0(z) = \frac{1}{\pi} \int_0^\pi \cos(z \sin \theta) d\theta$
Zeta	$\zeta(s) = \sum_{k=1}^{\infty} k^{-s} \quad (\Re s > 1)$

PCTEX Typesetting Software

For professional publishing and the power to produce high-quality books, technical documents, scientific notation, mathematical formulas, and tables, rely on PCTEX to make your work look its best.

The PCTEX Laser System includes:

- PCTEX and PC T_EX/386
- Our screen previewer, PTI View
- HP LaserJet and PostScript printer drivers
- Computer Modern Fonts at 300dpi
- A_MS-T_EX and L_AT_EX Macro Packages
- The PC T_EX Manual and L_AT_EX for Everyone
- Free Technical Support



12 Madrona Avenue
Mill Valley, California 94941
(415) 388-8853; Fax: (415) 388-8865

**Call for a free catalog and demo disk.
See the best for yourself!**

Index of Advertisers

- | | |
|----------|---------------------------------------|
| 240 | American Mathematical Society |
| 237 | ArborText |
| Cover 3 | Blue Sky Research |
| 239 | Computer Composition |
| 241 | ETP (Electronic Technical Publishing) |
| 244 | K-Talk Communications |
| 242, 243 | Kinch Computer Company |
| 247 | MG Software |
| 240 | Micro Programs, Inc. |
| 247 | Personal T _E X Inc. |
| 245 | Springer-Verlag |
| 234, 246 | T _E X Users Group |
| 238 | Wynne-Manley Software, Inc. |
| 248 | Y&Y |

a torus

$x = (1 + 0.4 \cos v) \cos u$
 $y = (1 + 0.4 \cos v) \sin u$
 $z = 0.4 \sin v$

MG
Mathematical
Graphics
System

- an MS-DOS based system for generating high quality mathematical graphics on screen and for printing in a PostScript environment.
- generates both encapsulated PostScript and special PostScript for including in a T_EX document where T_EX will typeset the labels.

Write or phone for information.

Single CPU: US\$95.00
Shipping: \$5 USA & Canada, \$10 overseas
Demo diskette: \$10 (credit towards purchase)

MG Software
4223 W. 9th Ave.
Vancouver, B.C.
Canada. V6R 2C6
(604)-228-8550

$x = t^3 - t$, $y = \frac{t^2 - 1}{t^2 + 1}$

TeX without Bitmaps

Wouldn't it be nice to be able to preview DVI files at any magnification, not just those for which bitmap fonts have been pre-built? Or to produce truly resolution-independent output that will run on any PostScript device, whether image setter or laser printer?

Perhaps you are looking for an alternative to Computer Modern? Well, there now exist complete outline font sets which include math fonts that are direct replacements for those in CM. Even if you do want to remain faithful to CM, there are distinct advantages to switching to the outline version of the fonts. We supply the tools to do all of this:

DVIWindo — preview DVI files calling for outline fonts

- * Preview at arbitrary magnification
- * Preview in Windows™ — a simple, standardized user interface
- * Print to any printer with a Windows printer driver
- * Show EPSF files with preview on screen — and insert TIFF images.

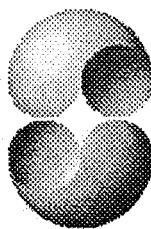
DVIPSONE — partial font downloading for speed and efficiency

- * Avoid running out of memory on the printer
- * Produce truly resolution-independent output
- * Designed from the bottom up for use with outline fonts on the PC

FONTS — available from Y&Y in Adobe Type 1™ form (ATM compatible)

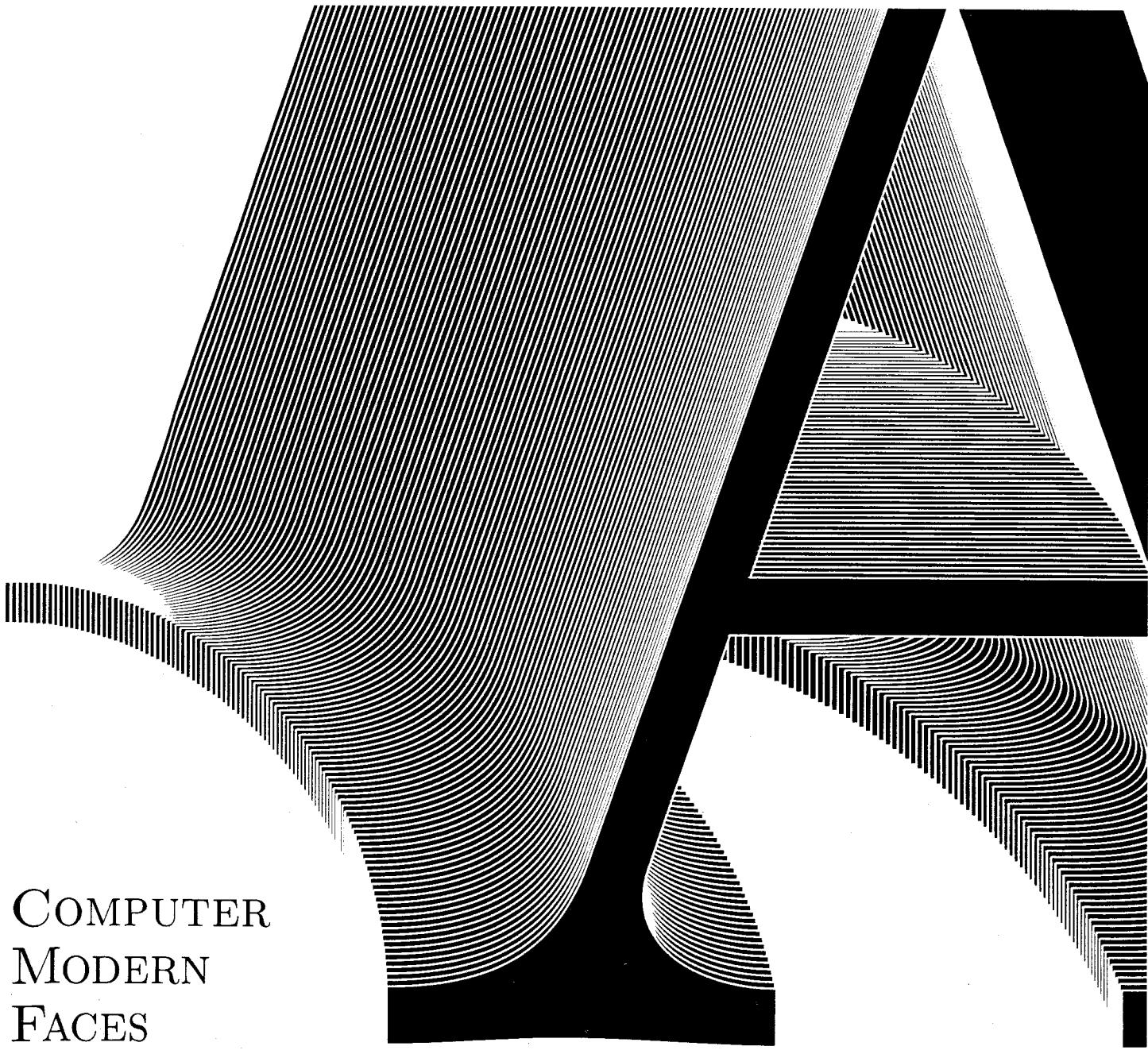
- * BSR Computer Modern fonts — with accented characters built in
- * LATEX + SLATEX fonts in outline form
- * Euler font set — the most popular faces from the AMS font set.
- * Lucida®Bright + LucidaBrightMath — a complete alternative to CM

Resolution-independent PostScript files using outline fonts can be printed by any service bureau, not just those with TeXpertise — and that translates into considerable savings for you. Perhaps it is time to get rid of those huge, complex directories full of bitmap fonts?



Y&Y, 106 Indian Hill, Carlisle, MA 01741 (800) 742-4059 (508) 371-3286 Fax: (508) 371-2004

Lucida is a registered trademark of Bigelow & Holmes Inc. Type 1 is a trademark of Adobe Systems Inc. TeX is a trademark of the American Mathematical Society



COMPUTER MODERN FACES

ADOBE
TYPE 1
POSTSCRIPT
FONTS

BLUE
SKY
RESEARCH

Forty faces of Computer Modern
designed by Donald Knuth
published in Adobe Type 1 format
compatible with
Adobe Type Manager
and all PostScript printers

\$345.00 Educational \$195.00
Macintosh or MS-DOS

Blue Sky Research
534 Southwest Third Avenue
Portland, Oregon 97204 USA
(800) 622-8398, (503) 222-9571
FAX (503) 222-1643

TUGBOAT

Volume 13, Number 2 / July 1992

	131 Addresses
General Delivery	133 Changing T <small>E</small> X? / <i>Malcolm Clark</i> 134 Editorial comments / <i>Barbara Beeton</i> 137 TUG seeks Executive Director 138 T <small>E</small> X: The next generation / <i>Philip Taylor</i>
Software	139 Knuth's profiler adapted to the VMS operating system / <i>R.M. Damerell</i>
Fonts	146 Arrows for Technical Drawings / <i>David Salomon</i>
Graphics	150 A solution to the color separation problem / <i>Daniel Levin</i> 156 A style option for rotated objects in T <small>E</small> X / <i>Sebastian Rahtz and Leonor Barroca</i>
Resources	181 Book review: An Italian guide to L <small>A</small> T <small>E</small> X (by Claudio Beccari) / <i>Marisa Luvisetto and Massimo Calvani</i> 182 Book reviews: Jane Hahn, <i>L<small>A</small>T<small>E</small>X for Everyone</i> ; Eric van Herwijnen, <i>Practical SGML</i> / <i>Nico Poppelier</i> 185 Book review: Victor Eijkhout, <i>T<small>E</small>X by Topic</i> / <i>Philip Taylor</i> 188 A T <small>E</small> X macro index / <i>David M. Jones</i>
Tutorial	189 Names of control sequences / <i>Victor Eijkhout</i>
Puzzle	190 Where does this character come from? / <i>Frank Mittelbach</i>
Macros	191 The bag of tricks / <i>Victor Eijkhout</i> 192 Over the multi-column / <i>Péter Huszár</i> 201 The elementary Particle Entity Notation (PEN) scheme / <i>Michel Goossens and Eric van Herwijnen</i>
L<small>A</small>T<small>E</small>X	208 From T <small>E</small> X to L <small>A</small> T <small>E</small> X / <i>Maria Luisa Luvisetto and Enzo Ugolini</i> 215 Geometric diagrams in L <small>A</small> T <small>E</small> X / <i>Peter J. Cameron</i> 217 How to change the layout with L <small>A</small> T <small>E</small> X 2.09 / <i>Hubert Partl</i>
SGML	221 SGML—Questions and answers / <i>Reinhard Wonneberger and Frank Mittelbach</i>
Dreamboat	223 T <small>E</small> X wish list / <i>Michael Barr</i> 226 Approaching SGML from T <small>E</small> X / <i>Reinhard Wonneberger</i>
Abstracts	227 Cahiers GUTenberg #12 228 Baskerville, Volume 2, Number 1, March 1992
Late-Breaking News	229 Production notes / <i>Barbara Beeton</i> 230 Coming next issue
News & Announcements	231 Calendar 232 EuroT <small>E</small> X 92, Prague, 14–18 September 1992
Forms	235 TUG membership application
Advertisements	247 Index of advertisers
Supplements	TUG Membership List