TUGboat

Volume 17, Number 4 / December 1996

TeX Users Group

Memberships and Subscriptions

TUGboat (ISSN 0896-3207) is published quarterly by the TeX Users Group, Flood Building, 870 Market Street, #801; San Francisco, CA 94102, U.S.A.

1997 dues for individual members are as follows:

- Ordinary members: $60; $55 if payment received before February 28, 1997.
- Students: $40; $35 if payment received before February 28, 1997.

Membership in the TeX Users Group is for the calendar year, and includes all issues of *TUGboat* for the year in which membership begins or is renewed. Individual membership is open only to named individuals, and carries with it such rights and responsibilities as voting in the annual election. A membership form is provided on page 410.

TUGboat subscriptions are available to organizations and others wishing to receive *TUGboat* in a name other than that of an individual. Subscription rates: $70 a year, including air mail delivery.

Periodical-class postage paid at San Francisco, CA, and additional mailing offices. Postmaster: Send address changes to *TUGboat*, TeX Users Group, 1850 Union Street, #1637, San Francisco, CA 94123, U.S.A.

Institutional Membership

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

Addresses

All correspondence, payments, parcels, etc.
TeX Users Group
1850 Union Street, #1637
San Francisco, CA 94123 USA

If you are visiting:
TeX Users Group
Flood Building
870 Market Street, #801
San Francisco, CA 94102, USA

Telephone

+1 415 982-8449

Fax

+1 415 982-8559

Electronic Mail

(Internet)
General correspondence:
`TUG@tug.org`
Submissions to *TUGboat*:
`TUGboat@ams.org`
World Wide Web:
`http://www.tug.org/`

[R]ight-hand justification ... came to the office just when both psychologists and typographers were discovering that uneven right-hand edges are more legible and even more attractive because they permit the most consistent spacing.

Edward Tenner
Why Things Bite Back (1996)

TUGBOAT

COMMUNICATIONS OF THE TEX USERS GROUP

EDITOR BARBARA BEETON

VOLUME 17, NUMBER 4 • DECEMBER 1996
SAN FRANCISCO • CALIFORNIA • U.S.A.

TUGboat

During 1997, the communications of the TeX Users Group will be published in four issues. One issue, still to be determined, will contain the Proceedings of the 1997 TUG Annual Meeting. One issue, also not yet assigned, will be a theme issue.

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

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

Submitting Items for Publication

The next regular issue will be Vol. 18, No. 1; deadlines are January 14, 1997 for technical items, and March 4 for reports and similar items. Mailing is scheduled for March. Deadlines for other future issues are listed in the Calendar, page 403.

Manuscripts should be submitted to a member of the *TUGboat* Editorial Board. Articles of general interest, those not covered by any of the editorial departments listed, and all items submitted on magnetic media or as camera-ready copy should be addressed to the Editor, Barbara Beeton (see address on p. 335).

Contributions in electronic form are encouraged, via electronic mail, on diskette, or made available for the Editor to retrieve by anonymous FTP; 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 network FTP access, they will be sent on request; please specify which is preferred. Write or call the TUG office, or send e-mail to `TUGboat@ams.org`.

This is also the preferred address for submitting contributions via electronic mail.

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@ams.org` or to the Editor, Barbara Beeton (see address on p. 335).

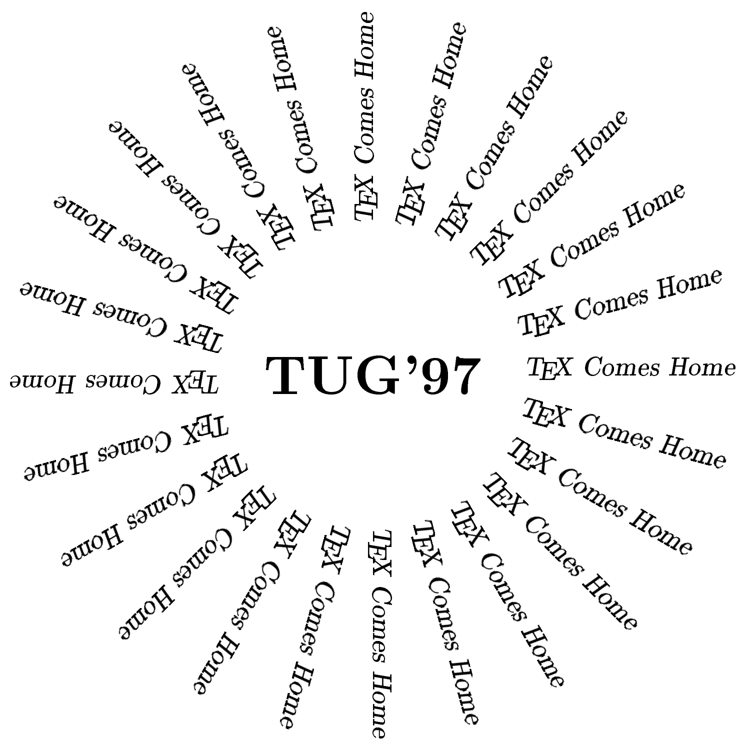
Other TUG Publications

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

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

TUGboat Advertising and Mailing Lists

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



July 28th through August 1st

Lone Mountain Conference Center

San Francisco, California

We would like to extend an invitation to TEX users around the world to join us in one of the most beautiful and exciting cities in the world. The conference will be held at The University of San Francisco Lone Mountain Conference Center, located on a 55-acre hilltop refuge with views of the Pacific Ocean, San Francisco Bay and the dramatic downtown skyline, and is a short walk from Golden Gate Park. Economical guest housing on campus is available to conference attendees.

The facility is about a 15-20 minute bus ride from the center of San Francisco, between town and the ocean beach. A single bus will take passengers from the ferry terminal building on the Embarcadero through town, past the University and to the beach.

The center is very well equipped, with several lecture halls and dining facilities. Shuttle transportation is available to take conference attendees who are staying on campus to the center. Computer accounts will be given out to all conference attendees.

Please forward your abstracts or questions to `tug97@mail.tug.org`. If you are interested in attending but not giving a paper, and wish to be kept up to date with the latest conference information including progam, cost, etc., please send email to `tug@tug.org`, or visit our WWW page at `http://www.tug.org/tug97/`.

Deadlines — Revised

Submission of Abstracts	March 14, 1997
Notification of acceptance to authors	March 21, 1997
Preliminary Papers Due	May 2, 1997
Preprint Deadline	May 16, 1997
Camera Ready Copy Deadline	June 20, 1997

General Delivery

The State of TUG

Michel Goossens

This is the beginning of a new year, and my first thoughts go to you and your families. I wish you all the very best for 1997. I hope that the coming months will bring you happiness, success, and above all good health. May all your long-lasting dreams come true!

Are those computers helping us?

Fewer than three years separate us from the day when the first digit of the year becomes a two, and thousands of pages have already been written about the disaster that will descend upon all file systems when the clock strikes midnight on December 31st 1999. It is feared that algorithms for storing date and time for files and other information will break down at that very moment. I am convinced that, with still more than a thousand days to go, a solution will be found. Yet the bill to fix this "trivial" problem will certainly run into the billions of dollars, and users and system managers all over the world will have to patch or upgrade their software.

The reason I mention this is to show the vulnerability of our modern lifestyle to quite small problems in the functioning of the software or hardware of computers that are controlling almost every aspect of human society. Digital telephone exchanges, satellite channels, fly-by-wire planes and steer-by-wire ships, alarm systems used by police and fire brigades, life-survival systems in hospitals, surveillance systems, computers for predicting the weather, handling data of the stock-market, controlling mobile telephones and other huge databases..., the list goes on and on. How much time, money and human lives have been lost when a single element in one of these electronic chains malfunctions. In many cases the main problem is a mismatch between man and machine: planes have fallen out of the sky and nuclear power plants have gone awry, putting the lives of hundreds of people in jeopardy.

It has become clear that computer tools and programs have not lived up to their expectations. Both developers and users of these systems overestimated their capabilities in the field of integration and streamlining human tasks. At the same time they underestimated the need for training and planning to benefit optimally from these complex tools.

Everybody goes global

Over the last few years we have observed an exponential increase in the number of people connected to the Internet, and every school, administration, indeed every "self-respecting" body in every country now has its Web server. Tens of millions of surfers have spent hours, if not the equivalent of days, in front of their computer screens, jumping in hyperspace from the text of the first act of Macbeth, to a `.gif` image of the Queen, an MPEG movie of the pictures sent back by the Hubble Space telescope, a quick tour of the Caribbean to plan the next holiday — or just to dream — before turning to the weather forecast, the timetable of trains or plains, a map of the next town you want to visit, the list of software tools you need to solve your son's school assignment, or the offerings of the universities in California where your daughter wants to attend college next term. All this information is there, at your fingertips, one or two clicks away. Of course, you also want to be part of it, to be there where the action is. So you decide to be present on the Net with your CV, complemented with a list of publications, a few of them online in PostScript, or even HTML, plus a picture of Family — Wife, Kids, Labrador Jessy, and Tiger, the cat — it is now a must, part of our cyber-culture. Everybody with a phone line can connect to the Global Village at almost no cost, and Internet providers are queueing at your (virtual) door to offer their services.

Billion of bytes at your fingertips

In parallel with the spread of cheap computer connections we have witnessed an equally significant development in the form of the ubiquity of the CD-ROM. These days, almost no computer is sold without this little technological marvel included. Each of these little 12 cm plates weighing a mere 18 grams can contain about 700 million characters, enough to run most of today's applications. The Britannica or Microsoft's Encarta are but two titles amongst several dozen encyclopaedia which give us instantaneous access to an enormous wealth of data and pictures wherever we are. Almost every conceivable form of information has been or is being made available on CD-ROM, and complete distributions of programs or data, which would take hours to download via a MODEM, can be sent for the price of a letter to all those who need them. Moreover, CD-ROM recorders have become affordable (today you can buy a good model for about $500, and prices continue to fall) so that everyone can now store on such a CD 700 Mbytes of data: programs, your family photo album in JPEG, your holiday movie in MPEG, or, perhaps more practically, a backup of your hard-disk.

Yet, this is only the beginning. Computers are now on sale which contain custom chips for a direct interpretation of MPEG (a spin-off of high definition television), and in a couple of years it will be possible to use the new-generation high-definition television set for all kinds of tasks, from browsing the Internet over optical cable at speeds of several Mbytes/sec, to playing complete films with sound in quadraphony. You will be able to enjoy an opera as though you were sitting in the Metropolitan. The Digital Versatile Disk (DVD), a new generation CD-ROM, is able to store between 4.5 and some 18 Gbytes of data, thanks to three technological breakthroughs: shorter laser wavelengths, two-face recording, and writing multiple vertical layers. This allows for a three-hour feature film, and also opens up a whole series of new possibilities in the field of multimedia.

TEX and the Global Village

But what about TEX and TUG in all this? If you look at the number of members of our Organization over the years (table on page 405), you observe that after a build-up in the mid-eighties, the membership stabilized in the range of 3000 – 4000 members, but has fallen by about 10 – 20%/year since 1991. The rise of TEX about ten years ago coincides with the time when more and more people began to be exposed to the personal computer, while the mainframe was still very important. For the first time scientific secretaries, professors, and students had easy access to a computer and were able to typeset scientific documents themselves, obtaining at the same time an appreciable improvement of the generated printed text. TEX became the de-facto standard in the world of mathematics at about that time, and even today many mathematicians send TEX code via email to their colleagues when discussing their work.

But since then local and wide-area networks have diminished the dependence on centralized systems, and most of what a TEX user needs can be transferred conveniently from CTAN, or copied from one of the TEX CD-ROMs now on sale. Today, publishers prefer commercial systems, such as Word, FrameMaker, PageMaker, or Quark Express, since they provide good control on the layout of the final page, or use SGML-based systems, since they allow easy integration of the information in databases, and its re-use in various forms. While TEX is still the text-processing engine in some WYSIWYG products like Scientific Word, other tools, like Mathematica, claim to have developed a better presentation engine for showing text and mathematics on screen, and they provide various output formats, such as TEX, HTML, and PostScript.

Donald Knuth has often wondered why developers have not taken his work in the TEX arena further and come up with extensions in those areas where TEX seemed to have shortcomings. It is only recently that ε-TEX and Ω have started to address some of these problems, but a lot of progress still remains to be made to adapt TEX fully to the needs of the "Windows Generation", where everything is controlled via a visual user interface. The batch-like behavior of TEX confuses many people and, although the quality of the output generated by TEX is in many cases still far better than that obtained by other text processors, its non-intuitive interface, its steep learning curve and, above all, the intricacies to control the micro-layout, in particular to place material precisely on the page, turn off most new and younger users. They prefer the easier-to-use and more intuitive WYSIWYG systems, which, moreover, are better integrated with the system environment, and allow easy cut and paste between various office applications.

Fewer members, bigger tasks

So, what can we do? I am convinced that the majority of the present TUG members have been with us for a long time. We seem to have lost members because they have changed activities and no longer need TEX in their professional life, or are convinced that other tools can do a text processing job as good as TEX. Another fact is that in the early nineties, when membership numbers began to decline, "local" TEX user groups were set up in France (GUTenberg), Germany (DANTE), the Netherlands (NTG), the Nordic countries (NTUG), and the United Kingdom (UKTUG). It goes without saying that such groups can serve their local user communities better (and more cheaply), so that we expect that a fraction of our members transferred to those groups instead. Finally, economic considerations have also played a role, and we must accept that many may find it more difficult these days to write that $55 cheque, since other priorities have overtaken the importance of TUG membership.

For the past few years we have witnessed a drop in the number of people attending TEX conferences. It is also proving ever more difficult to find volunteers for participating in working groups or other TEX-related activities. If you look at the present members of the TUG Board, you will recognize well-known names from the TEX world. However, no young new blood has been found in the last three or four years to take part in running TUG or to work on *TUGboat*. Last year we had only one candidate for six open Board positions. This year we need at least five people to step forward to serve for a four-year term on the Board together with the five remaining members. And there is also a new President to be chosen. Too many good and experienced experts have left the active TEX world, and very few newcomers have replaced them. We need enthusiasm, expertise, willingness to serve the TEX community, and this is a plea to you, dear reader, to make your voice heard and to show your active support of the TEX community.

The TUG Board by itself cannot perform miracles. Our respective employers demand an ever-increasing portion of our time, and at work TEX is rapidly losing ground compared with HTML editors, which turn electronic texts directly into globally accessible information, with the poor quality of the printed output considered as a tolerable evil (who wants to print, anyway...). If we do not want to disappear into oblivion, we shall have to adapt to the Internet paradigm and accept that other text processors can also put words on the page or screen. We must try and live in symbiosis with these systems, and see how we can help TEX users in those areas of the world where publicly available tools are still important (although freely available HTML editors are already commonplace on most computer platforms). TUG should emphasize Internet access (via ftp, WWW), produce TEX CD-ROMs, and continue to publish *TUGboat*, which should become a journal not only for the experienced TEX user, but also for the novice. We should also open up its columns more and more to what is happening in the field of document handling at large.

Actions have been taken

As the financial records on page 405show, TUG's income depends to a large extent on membership contributions. Recently, we have observed a decline in membership, and during the same period the office staff has been reduced. At first we were working with an Executive Director plus two clerical staff, and, since the end of 1995, with just the ED. But even this expense TUG can no longer afford, since the cost of renting office space, telephone, computer equipment, and the payroll of one staff member is too large compared to income. Considering the fact that printing costs for *TUGboat* and, above all, postage (both domestic and overseas) have increased by almost 30%, we just do not have enough money in the bank to go on like this. We can no longer solely rely on donations by individuals of other user groups to carry out our tasks. TUG must have proper funds to allow it to buy computer hardware and pay for Internet services. Therefore, as of January 1st 1997 the TUG Office will be only staffed half time.

In order to optimize our resources I am pleased to announce that TUG Executive Committee has nominated Ms. Mary R. (Mimi) Burbank, working at the Supercomputer Computations Research Institute (SCRI), Florida State University in Tallahassee, Florida) as TUG's Office Management Consultant. Beginning in April, Mimi will begin the process of coordinating the transfer of operations from California to Tallahassee, Florida, with Ms. Patricia Monohon, TUG's outgoing Executive Director, whose contract has come to an end. We hope that the entire process is complete by TUG'97. I would like to thank Ms. Monohon for her work during the five years she has run the TUG office, and I wish her all the best for the future. TUG appreciates it very much that she has agreed to act as contact person for the upcoming TUG'97 Conference in San Francisco in the Summer.

Since we must optimize our human resources, we must prioritize the office duties by concentrating first on essential administrative tasks. Questions directly related to TUG (membership status, *TUGboat* subscriptions, etc.) should preferably be sent via electronic mail to the address `tug@tug.org`. Only those without easy email access should try to send a fax or phone. Technical support questions should be addressed to the vendor or developer of your TEX software (if commercial) or put to the newsgroup `comp.text.tex` on Usenet. TUG can act as a backup on behalf of users with commercial vendors regarding software if there are problems of a general nature.

We have done well, but with your help we have to move on

TEX and TUG have come of age, and they are adults now. TEX can live a life by itself, and nowadays every user should be able to get TEX running without needing the help of a "TEX guru" to install the system. Commercial and free plug-and-play TEX installations are now available from several sources, and, as outlined above, it is clear that TEX users no longer need to handle "nuts and bolts" of the TEX engine but only need to worry about choosing the color of the LATEX Cadillac amongst a palette as varied as a rainbow. Today's TEX users should appreciate the work of countless pioneers, who have each dedicated many person-months to make TEX run on (almost) every computer hardware and operating system in the world. We should also not forget all the TEX-gurus, who have written from scratch the hundreds of (LA)TEX packages, which make life so much easier for us today. Up to the late eighties TUG played an important role in the distribution of TEX, acting as "the" central and official TEX-related distribution point (courses, books, distributions, official information, conferences, ...). Since then the Internet, regional TEX User Groups, and the advent of efficient distribution media have taken over many of the tasks which TUG used to provide. Therefore, as I explained above, TUG must concentrate its efforts in a few essential areas, namely serve as the official clearinghouse for TEX-related information (this includes, of course, publishing *TUGboat*) and act as a liaison with other TEX User Groups.

I am sure that with your continued support TUG will be able to play a significant role in the future. We count on you!

A few words of thanks

I do not want to finish before thanking Karl Berry for his ongoing work in maintaining the `web2c` sources, which form the basis of (almost) every publicly available port of TEX — the latest version (7.0) should be out by the time you read this — as well as for donating his PC to run the TUG WWW server. Another machine installed at UMB, the University where Karl is working, is a Sun Sparc 10 workstation with a 4 Gbytes external hard disk. This machine, which was the former DANTE CTAN node, will soon be the official North American CTAN host (a welcome fact after the demise of SHSU some time ago). Thanks once again to DANTE for this generous donation to the TEX community. And, of course, "un grand merci" to all my colleagues of the TUG Board (present and past) and the *TUGboat* Production Team, who have been working real hard to keep TUG and *TUGboat* going.

Last but not least, let us also have a thought for the ongoing work of TEX developers worldwide: the LATEX3, ε-TEX, and Ω teams, of course, but also those who continue to adapt and optimize the use of TEX for their national language or for their particular application field. The future of TEX lies to a great extent in their hands, and TUG wholeheartedly endorses all these initiatives, which are the best guarantee that TEX is alive and well.

◇ Michel Goossens
CERN, Geneva, Switzerland
`goossens@cern.ch`

Editorial Comments

Barbara Beeton

A historical perspective

Owing to circumstances beyond anyone's control, the planned topical issue on "TEX in the humanities" has not materialized. Instead, we bring you the transcripts of two question and answer sessions with Don Knuth — one in Prague following the awarding of (yet another) honorary doctorate by the Faculty of Informatics of Masaryk University in Brno, Czech Republic, and the second following a talk in celebration of the 50th anniversary of CWI (the Centrum voor Wiskunde en Informatica) in Amsterdam.

Every time I attend a DEK Q&A, or read the transcript after the fact, I learn more about what made this system on which my livelihood depends the special thing it is. Sometimes the message is that, in hindsight, another approach might have been preferable, but more often, it seems that the path taken was prescient, and the techniques have — so far — withstood the test of time. There are also some delightful sidelights, such as the identity of the model for the TEX lion, or why a lion represents TEX (for the latter, you'll have to go all the way back to the "coming out party" for Knuth's *Computers & Typesetting* series in 1986, but maybe I'll be kind and give the answer before signing off).

The Kyoto Prize for Knuth

As announced last June by Dr. Kazuo Inamori, founder and president of the Inamori Foundation, Donald Knuth was one of three scientists awarded the 1996 Kyoto Prize, Japan's equivalent of the Nobel Prize and the country's highest award for lifetime achievement. The award comprises a diploma, a gold medal, and a cash gift of 50 million yen; it was reported that the Knuths had decided to donate the money to charity.

The Kyoto prize is awarded each year in three categories: advanced technology, basic sciences, and creative arts. Knuth won in advanced technology.

In addition to *The Art of Computer Programming*, "the Bible and Encyclopedia for Computer Science", Knuth is widely known for TEX and METAFONT. According to *Stanford Today*, "These programs have been called the single most important achievement in publishing since the invention of the printing press. Rather than copyrighting and licensing the programs, Knuth put them in the public domain."

The formal presentation of the prize took place November 9–12 in Kyoto, Japan.

Recommended reading

Anticipating that this issue was to be a topical one, compiled by guest editors, I spent some spare time reading. One book that I found very interesting (although a bit hard going in places, as it does not "tell a story") is *Why Things Bite Back* by Edward Tenner. This is a compendium of connections between cause and (often unanticipated) effect, in many areas of technology. The computer is not spared, nor is the "do-it-yourself" approach that has displaced many skilled clerical and technical workers.

Let the author speak for himself.[1]

– – * – –

We have all seen the sign "The Difficult We Do Immediately; the Impossible Takes Time." Computerization turns this manifesto on its ancient head. Software can devour highly complex tasks with ease if they fit well into its existing categories. But even a simple change illustrates the revenge effect of recomplicating. The scientific typesetting program TeX, developed by the computer scientist Donald S. [*sic*] Knuth and now the standard in many branches of physics and mathematics, makes short work of the most fearsomely complex equations that once cost publishers up to $60 per page to typeset. An author proficient in TeX — and I have had the good fortune to work with several of them — can prepare camera-ready copy that stands up to most commercially available systems. But making small changes, alterations that might require dropping in a metal slug or pasting in a new line in traditional systems, can sometimes take costly programmers' time. A hairline rule can take more time and money than pages of author-formatted proofs brimming with integration signs, sigmas, deltas, and epsilons.

Minor incompatibilities between authors' TeX programs and publishers' typesetting equipment can delay book-length manuscripts for weeks and run up costs well beyond those of conventional typesetting. Worse still from the publisher's point of view, some inexperienced and unskilled TeX-using authors — including distinguished scientists — blame the publisher and typesetter when their work is held up.

As editors of conventional manuscripts, my colleagues and I could identify problems early and request changes before texts went into production. Even experienced electronic manuscript specialists cannot evaluate a TeX manuscript reliably just by looking at the author's laser-printed version. Messy or nonstandard coding may fail to reproduce the same beautiful output when fed into professional typesetting equipment. Consequently, there are real hidden productivity costs associated with an "inexpensive" TeX manuscript; it may require open-heart surgery rather than a haircut. Publishers and typesetters discovering such insurmountable glitches have been known quietly to set the author's electronic manuscript aside and dispatch the hard copy to Asian compositors for conventional keyboarding. This may be speedier than waiting for the author to learn the fine points of TeX, but it inevitably delays production, embarrasses author and publisher alike, and introduces new errors. What computerization offers — or simplifies — with its right hand it can withdraw — or recomplicate — with its left.

TeX demonstrates the additional burdens of vigilance that advanced technology imposes. It may slash production times and costs for a scientific or engineering publisher, but only if either (1) the whole burden of typesetting is shifted to the author, who then has to be knowledgeable and vigilant about levels of detail that copy editors and typesetters otherwise would supervise, or (2) the author's editor is prepared to spend hours learning the fine points of TeX, adding technical support to his or her job description.

– – * – –

Other staples of the modern computer environment come in for similar scrutiny — icons, Windows, ill-advised use of color, replacement of expensive mainframe hardware by low-cost (but high-upkeep) networked workstations and microcomputers, ...

There's much food for thought in this book, and I recommend it to anyone who wonders where is all that spare time that was promised to us by the proponents of modern technology.

Answer to the question

And why does a lion represent TEX? Here are Don's comments on the subject to the guests at the grand bash held at the Computer Museum in Boston in honor of the publication of the *Computers & Typesetting* series, on May 21, 1986.[2]

> One final note: People often ask me why TEX and METAFONT are symbolized in these books by a lion and a lioness. When Duane Bibby first came up with the lion idea, I instinctively felt that it was right, but I never understood exactly why this was, until about

[1] *Why Things Bite Back: Technology and the Revenge of Unintended Consequences*, by Edward Tenner. New York: Alfred A. Knopf, 1996, ISBN 0-679-42563-2; pp. 192-193, quoted with permission.

[2] For the complete text of all the remarks at that fest, see *TUGboat* **2** #2 (1986), pp. 93–98.

a month ago when I was in the Boston Public Library. I passed by the magnificent stone lions on the library's grand staircase, and I thought: "That's it! TEX and METAFONT try to be like these lions, fixtures that support a great library.[3] I love books, and lions represent books!" No wonder I'm so happy when I realize that TEX and METAFONT have already contributed to the making of several dozen books of fine quality; it makes me extremely pleased to think that this research will probably contribute to the making of many more fine books in years to come.

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

[3] (Editor's note.) One is also reminded of the lions that grandly guard the entrance to the New York Public Library, which celebrated its 75th anniversary during [that] same week.

Amsterdam, 13 March 1996//Knuth meets NTG members*

Abstract

On January 6th 1996, Kees van der Laan informed the NTG that Donald Knuth would be in Holland in March. Knuth was invited by the *Mathematisch Centrum* (MC, nowadays called *Centrum voor Wiskunde en Informatica*, CWI) because of CWI's 50th anniversary. Both Knuth and Mandelbrot were invited as speakers at the celebration.

The NTG noticed that this was an exceptional occasion to organize a special meeting with Knuth for all Dutch TEX and METAFONT users who would like to meet the Grand Wizard himself.

Fortunately Knuth accepted the NTG invitation and so a meeting was organized in 'De Rode Hoed' in Amsterdam on March 13th. About 35 people from all over the country and even from Belgium joined to meet Knuth.

Everything was recorded on both video and audio tape by Gerard van Nes. Christina Thiele volunteered to write this transcript.

Erik Frambach: Welcome, everyone. This is a very special meeting on the occasion that Mr. Donald Knuth is in Holland. The NTG thought it would be a good idea to take the opportunity and ask him if he would be willing to answer our questions about TEX, METAFONT, and anything else connected to the things we do with TEX. Luckily, he has agreed. So we are very happy to welcome Mr. Donald Knuth here — thank you for coming.

Tonight we have time to ask him any questions that we have long been waiting to pose to him. [laughter] I'm sure that all of you have many, many questions that you would like the Grand Wizard's opinion about. So, we could start now with questions.

Donald E. Knuth: I get to ask questions too! [laughter] Last Saturday I was in Prague and the Czechoslovak TEX users had a session something like this and you'll be glad to know that I saw quite a few copies of *4TEX* CD-ROMs at that meeting.

It's not my first time in Amsterdam. I was in Amsterdam in 1961, so it's only been 35 years, and probably less than 35 years till the next time. I guess they're tape recording these questions-and-answers to try to keep me honest, because they also did that

* This transcript was first published in *MAPS* **16**, 96.1, pages 33–44, and appears here by permission; *MAPS* is the journal of NTG, the Dutch language oriented TEX Users Group.

in Prague. So in case the same question comes up, you'll have to take the average of the two answers. [laughter]

Wietse Dol: Did you know that Barbara Beeton does that? She mails you and says "Tape everything."

Knuth: Yes, that's what they said in Prague too! [laughter] I think she's desperate for things to do, or maybe she just has a lot of questions. But before I open questions, let me say that one of the most interesting questions asked me in Prague was after the session. And I wish it would get into [the record]. The question was: how did I meet Duane Bibby, who did the illustrations for *The TEXbook* and *The METAFONTbook*? I always wanted people to know about that somehow.

Here's the story. I had the idea that after writing math books for many years, I wanted to have a book that had more weird — well, anyway, different — illustrations in it. Here I was writing a book about books — books have illustrations, so why shouldn't I have illustrations too. So, I wrote to an artist called Edward Gorey. Does anyone know . . .

Frans Goddijn: Yup. *Amphigorey*. Beautiful.

Knuth: Yes, Edward Gorey. *Amphigorey*. He makes very morbid drawings but with a wonderful sense of humor. I had used several of his books with my children. I thought he would be a natural person. I wrote him two letters but he never responded. Then I wrote to a Japanese artist called Anno, Matsumasa Anno, who is really the logical successor to Escher. [. . .] Anno does what Escher did but in color, so I asked him if he would do it. He sent me back a nice letter, saying "I'm sorry I don't have time because I have so many other commitments, but here are five of my books full of pictures and if you want to use any of those, go ahead." I wanted personalized pictures.

Then I went to a party at Stanford where there was a lady who worked for a publisher. She'd just met a brilliant young artist who she'd just worked with. I invited him to come to my house, and we spent some time together and he's a wonderful person. Duane lives now up in northern California, about 4 hours' drive from my house, so I only went up there once to see him. He sometimes comes down to the San Francisco area on business. First we discussed the book and then he sent me a bunch of drawings and all kinds of sketches that he had. Originally, TEX was going to be a Roman, and he drew this man in a toga with olive branches in his head — which is why the lion has the olive branches now. But all of a sudden he started doing sketches of his cat, which really seemed to click, and pretty soon he had a draft of all 35 or whatever drawings, using a lion. Most of those eventually become the drawings in the book, and we adjusted half a dozen of the others. When I went to visit up at his house, I got to meet TEX the cat. He looks very much like the one you see in the book. So that's the story about Duane Bibby.

Erik: Thank you. Who would like to start with the first question? Please identify yourself when you ask one.

Piet van Oostrum: My name is Piet van Oostrum. You have this wonderful lion on *The TEXbook* the lionness on *The METAFONTbook*. What about baby lions?

Knuth: Oh, I see . . . [laughter] Duane still does illustrations for special occasions. He's made illustrations for the Japanese translation of both *The TEXbook* and *The METAFONTbook*. He has TEX and META both dressed up in Japanese costumes. So now, if there happens to be some kind of an offspring that would come out of some other user, then, I imagine he would glad to help do it. But it would probably be a little bit of an illegitimate child, from my point of view. [laughter] I mean, I wouldn't take responsibility for anything those characters do. [laughter]

Piet: So what are your ideas about the offspring of TEX and METAFONT?

Knuth: Well, I think that no matter what system you have, there will be a way to improve it. If somebody wants to take the time to do a good, careful job with it, then as we learn more about typesetting, it will happen that something else will come along. I personally hope that I won't have to take time to learn a new system, because I have enough for my own needs. But I certainly never intended that my system would be the only tool that anybody would ever need for typesetting. I tried to make it as general as I could with a reasonably small program, and with what we knew and understood about typesetting at the time. So these other projects — I don't consider that they're a threat to me or anything. I hope that there will be some compatibility so that — I mean, I'd like to be immortal — so that the books I've written now could still be typeset 50 years from now without having to go through the files and edit stuff. I like the archival and machine-independent aspects of TEX especially, and I tried to set a model, a minimum standard of excellence for other people to follow.

Hans Hagen: But when you look in the future, ... you consider today's programming by a lot of people as an art, well a lot of art takes hundreds of years to be recognized as art. In about a hundred years there will be pretty different computers, the programming languages will be changed, the media on which we put all those things will be changed. Real programs and everything related to them, will they ever have a chance to become immortal, as you see it?

Knuth: Did you state your name? [laughter]

Hans Hagen: I'm Hans Hagen.

Knuth: You're saying that it's pretty arrogant of us to assume that what we do now will last at all. Technology is changing so fast that we have absolutely no idea what people are going to think of next. One hundred years ago, physicists were saying there was nothing more to do in physics, except to get another decimal — a fifth decimal place for the fundamental constants — and then that would wrap up physics. So, there is no way to know about these things. But I do believe that once we have things in electronic form and we have mirror sites of them, there is a fair degree of immortality — whereas paper burns.

Do you know anything about this project called 'The Clock', being developed by Stewart Brand and his colleagues? He's the one who published the *Whole Earth Catalogue*. They have a bunch of people that are considering if they could build something that would last for a thousand years ... I don't want to go on too much more about that. I do hope that the stability of TEX will make it possible to reproduce the things we're doing now, later. And since it's fairly easy to do that, I think it will happen — unless there's a nuclear holocaust. Some mathematicians have this debate about the Platonic view ... does everything in mathematics exist and we're just discovering it, or are we actually creating mathematics? In some sense, once something gets put into bits, it's mathematics and therefore it exists forever, even if the human race dies out — it's there, but so what?

Erik: Who's next?

Marc van Leeuwen: If I could extend a bit on the previous questions. The stability of TEX itself, I could imagine, might be a stumbling block for development of new things exactly *because* it's so stable and everybody's already using it. So if something comes along that is just a bit better, then people will not tend to use that because it's not available everywhere, and there are all kinds of reasons to keep on using the old thing.

Knuth: I guess I said in Florida that people are still trying to use the old fonts that I'm still trying to stamp out from the world. Four years ago I redesigned the Greek lowercase delta and I made the arrowheads darker. I didn't change anything in the way TEX operates — all the dimensions and the characters' heights and widths stay exactly the same. But I did tune up a lot of the characters. Still I see lots of math journals are still using the old ones from four years ago, and I get letters and preprints from people with the old-style delta. I changed it because I just couldn't stand the old versions. [laughter] Now I've got home pages — if I ever have some errata to TEX or something I put them there: `http://www-cs-faculty.stanford.edu/~knuth`. This gets to my home page, and there's a reference saying, 'Important notice for all users of TEX', and that page says 'Look at the lowercase delta and if you have the wrong one, you die!' [laughter]

I understand that people have a reluctance to change from something that they've become accustomed to. I know of two main successors to TEX: one is ε-TEX and the other is NTS. ε-TEX is going to be apparently 100% compatible with TEX, so if somebody doesn't switch over to incompatible features, then they have a system that still works with old things. That will allow a gradual change-over. It'll take more space on a computer, of course, but that's not a big deal these days. The people who work on ε-TEX always sent me very reliable comments about TEX when they caught errors in my stuff, so I imagine they're going to be doing a careful job. So it'll be one of these things where you walk into a random installation of UNIX or whatever and you'll find ε-TeX there as the default, and you'll still have TEX. Then you also have certain other features that might be really important to you for your special applications.

Johannes Braams: You mentioned ε-TEX and NTS. But are you also aware of the Omega project?

Knuth: Oh, the Omega project? Yes, I'm hoping to use that myself for the authors' names in my *The Art of Computer Programming*. I've been collecting the names of Chinese, Japanese, Indian, Hebrew, Greek, Russian, Arabic authors and I want to typeset their names properly [laughter], not just in transliteration. I have some rudimentary software that will do this for proofing purposes, for getting my database going and for writing to people and saying, 'Is this your name?' With the Omega system, I'm hoping that it'll be accompanied by good fonts that will make it possible for me to do this without a whole pile of work. Right now, to get the Arabic names, I have to use ArabTEX, to get the Hebrew names ... I had a terrible time trying to get Hebrew fonts on CTAN two weeks ago — I can tell you that whole story if you want to know ... I kept clicking on the different things and they would refer to files that didn't exist and README files that were four years out of date and inconsistent, so I couldn't find any Hebrew fonts. Maybe you have it on your CD ...

Johannes: I could certainly point you to someone who could help you with the Hebrew font — I know someone in Israel who's trying to do Hebrew support within the Babel system. And they do do typesetting in Israel with TEX.

Knuth: My own typesetting friend in Israel is Dan Berry, who unfortunately is fairly committed to troff. [laughter] I'm sure that I can get good Hebrew through Yannis and Omega. I sure hope UNICODE is going to arrive sooner rather than later; it's much better than the alternatives for much the reasons that Marc [van Leeuwen] mentioned. I haven't found a great enthusiasm in Japan for UNICODE, because they have a system that seems to work pretty well for them, so why change. Everytime I ask a Japanese for his name in UNICODE, he'll say, 'what's UNICODE? Here's my JIS name'. But the JIS characters don't include the Chinese codes, and in fact, my own name — I have a Chinese name — and my name in JIS isn't quite the same. There are two different unicode characters, one for the Japanese version and one for the Chinese.

In the back? Kees?

Kees van der Laan: I have a lot of questions of course. But I would like to start with some questions about METAFONT. The first one is: how come macro writing in TEX and METAFONT is so different?

Knuth: Why are macros in TEX and METAFONT so different? I didn't dare make TEX as extreme as METAFONT. These languages are of completely different design. METAFONT is in some ways an incredible programming language — it's object-oriented macros. You have macros in the middle of record structures.

The way I designed these languages is fairly simple to describe. Let's take TEX. I wrote down one night what I thought would be a good source file for *The Art of Computer Programming*. I took a look at Vol. 2, which I had to typeset. I started out on the first page, and when I got to any copy that looked very much like something I had already done I skipped that. Finally I had examples of all the different kinds of typesetting conventions that occur in Vol. 2. It totalled 5 printed pages — and you can even see these pages — exactly what my original test program was — in a paper by David Fuchs and myself, where we talked about optimum font caching.[1] In there, we gave an example and we show these 5 pages, which would illustrate what I wanted TEX to be able to do. I wrote out what I thought I would like to type — how my electronic file should look. And then, I said, OK, that's my input, and here's my output — how do I get from input to output? And for this, well, it looks like I need macros. [laughter]

Same thing for METAFONT. I went through my first draft of all the fonts that later became Computer Modern. I wrote actually in SAIL, an Algol-like compiler language, but SAIL had a macro ability, so I developed a few primitive macros in which I could say, 'pick up the pen', 'draw from point 1 to point 2', and things like that. These macros were compiled by the SAIL compiler into machine language, which would then draw the letters. I went though the entire alphabet, and by the end of the year, I had some 300 little programs, each one drawing a letter. Then I realized what kind of a language I would want to write in, to describe the letters. So one day, on a family camping trip — I was in the Grand Canyon with my wife and kids — I took an hour off, sat under a tree and wrote out the program for the letter A, in a language that I thought would be a good algebraic language, reflecting at a high level what I had been doing with pretty primitive low-level instructions in my SAIL programs. I did the letter B, too. Capital A and B, and then went back to the camping trip. These sheets of paper where I have my original programs are now in Stanford's archive — the program for the letter B was published in a Stanford library publication called *Imprint* last year. The woman who's in charge of rare books and manuscript collections at Stanford is quite interested in METAFONT so she wrote a little article about what they have.

That program again implied that I wanted some macros to go with it. But these needed to be much more structured than the macros of TEX. It had to be that when I said, 'z 1 prime', this would actually be equivalent to '(x 1 prime, y 1 prime)' and I wanted to be able to write, 'z 1 prime' without any delimiters. It turned out that in order to have a high-level language that would feel natural to

[1] *ACM Transactions on Programming Languages and Systems* **7** (1985), 74.

me writing the program, it had to look completely different from TEX. So TEX and METAFONT share a common format for error messages and certain other data structures inside, but otherwise, they're quite different systems because, in order to have a good high-level language, I don't want to have to waste time writing parentheses, brackets, commas, and other delimiters.

Kees: It's a nice introduction to my second question: [laughter] For the future of MetaPost, which allows mark-up of pictures, with `.eps` as the result, what is your attitude to 2.5d for MetaPost and METAFONT? For example, adding a triple as an analogy of the paired data structure?

Knuth: MetaPost already has a data structure for triples because of color. So RGB are actually triples of numbers.

Kees: Yes, but the triple as a data point in space?

Knuth: Ah, I see. I did write METAFONT in a way that has hooks in it so that it can be easily extended; [for example], if you want to draw 3-dimensional pictures, for perspective and projective geometry instead of affine geometry. The program itself for METAFONT was written so that it could easily be changed by people who wanted to have a system that goes beyond the basics. I always wanted the systems that I would make widely available would be be able to handle 99% of all applications that I knew. But I always felt there were going to be special applications where the easiest thing would be to change the program, and not write a macro.

I tried to make the programs so that they would have logical structure and it would be easy to throw in new features. This hasn't happened anywhere near as often as I thought because people were more interested, I think, in inter-changeability of what they do; once you have your own program, then other people don't have it. Still, if I were a large publisher, and I were to get special projects — some encyclopaedia, some new edition of the Bible, things like that — I would certainly think that the right thing to do would be to hire a good programmer and make a special computer system just for this project. At least, that was my idea about the way people would do it. It seems that hasn't happened very much, although in Brno I met a student who is well along on producing Acrobat format directly in TEX, by changing the code. And the Omega system that you mentioned, that's 150,000 lines of change files. [laughter] I built in hooks so that every time TEX outputs a page, it could come to a *whatsit* node and a *whatsit* node could be something that was completely different in each version of TEX. So, when the program sees a *whatsit* node, it calls a special routine saying, 'how do I typeset this *whatsit* node?' It'll look at the sub-type and the sub-type might be another sub-type put in as a demo or it might be a brand-new sub-type.

Similar hooks are in the METAFONT program. If people have extra time when they're not browsing the Web [laughter], I recommend as a great recreation to read the program for METAFONT. Some parts of it are pretty rough going and I hope that nobody ever finds a bug there because I'd hate to have to look at them. [laughter] But those are the rasterization routines, the things that actually fill in the pixels. There are many other things in that program — the linear equation solver that it has and the data structure abilities ... lots of beautiful algorithms are in there — to take square roots in fixed point, and the intersection of two curves, and so on. METAFONT is full of little programs that were great fun to write and that I think are useful and interesting in their own right. I think when John Hobby wrote MetaPost, he enjoyed it, because he could add his own nice little programs to the ones that are already there.

I'm a big fan of MetaPost for technical illustrations. I don't know anything that's near as good, so I'm doing all the illustrations of *The Art of Computer Programming* in MetaPost. Also, the technical papers I've written are going to be published in a series of eight volumes by Cambridge University Press, and all the illustrations, except the photographs, are going to be MetaPosted. The first volume of these eight was the book *Literate Programming*; the second volume is going to come out this summer and is going to be called *Selected Papers in Computer Science*. It reprints a dozen or 15 papers that I wrote for general audiences, not for specialists in computer science, but in *Scientific American* or *Science* magazine and things like that. The third volume will be about digital typography, and it'll reprint all my articles in *TUGboat* and things about TEX. What do you think, by the way — should I publish in that third volume the memo that I wrote to myself the first night, when I designed TEX? I put it in a computer file and it's in the archives, but I've never shown it to anyone. [round of "of course!" and "sure" and laughter from the audience] Maybe it'd sell more books [more laughter].

Frans Goddijn: You need to put it on your home page and we can then decide —

Knuth: No, no. That way we'd never sell the books. [laughter] Not that I'm a mercenary type of person, of course. It's in a file called TEX — well 'teks', actually. I have to admit I pronounced it 'teks' for a month or two — I was thinking of 'technical texts', though. `tex.one` was the name of the file and it would make interesting reading probably, someday.

And your name is?

Jan Kardan: In this company I will probably ask a very heretic question, but a little heresy makes a lot of fun — talking about METAFONT. There are probably many type foundries now [that] crank out lots of good-quality fonts and kerning tables. It's not clear whether PostScript or True Type will survive. Do you think that METAFONT will survive text fonts? Not talking about the math fonts.

Knuth: I don't think the extra capabilities of METAFONT have proved to be necessary for good-quality type fonts, although I think that you can still make better-quality type fonts with it. Designers find it difficult to think as a computer person does, in the sense that when people in the computer business automate something, trying to make the computer do something, it's natural for us to have parameters and say that we're going to try to solve more than one problem. We try to solve a whole variety of problems based on the parameters that people set. But it's much easier if people gave us only a single problem with a single parameter, then we could have the computers do exactly the prescribed thing. Computer scientists have become accustomed to thinking of how we would change behavior as conditions change, but designers aren't at all accustomed to this. They are much happier if the boss says one month, "Give me a roman font," and the next month, "Give me a bold font." It's much more difficult to say, "Show me how you would draw something no matter how heavy I want the letters to be." METAFONT provides a way to solve that problem and to draw characters with parameters, but it's a rare designer who's comfortable with that notion. They can do multiple master fonts by making multiple drawings and then matching up points between the drawings and having the computer interpolate. The multiple master fonts in PostScript allow up to four parameters, and almost all of them have only one or two parameters. The most I know of is two; probably others have gone all the way to four. But then they have to provide drawings for all the extreme points of these parameters.

In spite of this limited use of parameters, what's available commercially is quite beautiful, as far as readability is concerned, although it doesn't really provide the quality that you guys had in the Netherlands in the 17th century. What's the man's name, the great punch cutter at Enschedé — he made 4.5, 5pt up to 16pt, and each letter was designed for its size, and fonts had a nice uniform appearance. This wouldn't have happened at all with the Type1 fonts. There were two guys who did most of the punch cutting for Enschedé and others in the 18th century: One of them, Fleischman, was a genius for really beautiful letters; the other, Rosart, was just good at making lots and lots of letters. [laughter][2] [...] They were fun. Rosart would make all kinds of highly decorated alphabets and things like that. I have a big coffee-table book that gives examples of all the fonts from Enschedé, which was translated into English by Matthew Carter's father. Anyway, in this book, *Typefoundries in the Netherlands*, you can look at these typefaces and weep.[3]

Still, on a laser printer, we get pretty good fonts now, and therefore it looks like there won't be that many professional type designers using METAFONT. Pandora was a good design by a genuine graphic artist. METAFONT has turned out to be wonderful for making ordered designs and special-purpose things for geometry. There's now this really neat system in Poland where they have TEX and METAFONT in a closed loop — TEX outputs something and then METAFONT draws a character and if that doesn't fit, TEX says, 'go back and try it again'. Jackowski and Ryćko understand TEX and METAFONT, and the programs are well documented and can do these things. So METAFONT isn't going to disappear for that reason; but it's never going to be taught in high school.

Frans Goddijn: My name is Frans Goddijn and I have one question with some sub-questions [laughter]; I'd like to ask the sub-questions first. What I'm wondering — and this may have been asked often before — is whether you would consider, in retrospect, what you have created [to be] an art or a tool? And the reason I ask is — when I hear you speak with so much passion for type fonts and the beautiful algorithms that you put into METAFONT that you would like to point people to and the recognition that you get from people who understand that — but, there is a vast majority of users who just got

[2] Johann Michael Fleischman, 1701–1768; Jacques-François Rosart, 1714–1777.

[3] *Typefoundries in the Netherlands from the Fifteenth to the Nineteenth Centuries*, by Charles Enschedé, translated by Harry Carter (Haarlem: Stichting Museum Enschedé, 1978), 477 pp. This magnificent book was composed by hand and printed by letterpress to commemorate the 275th anniversary of Joh. Enschedé en Zonen.

TEX from some server, never realized who created it, and use it to typeset not always very pretty documents. [laughter] They do that in a very crude way and don't care less. You froze TEX at a certain point, allowing other people to build around it. I was wondering how such a thing would feel to a father — are you father of a piece of art that other people use as a tool, or is it a child that you have frozen in its development, that will never grow up ... there are so many questions ... if you just go back to the art vs. tool idea, and your feelings about that.

Knuth: Obviously, if I write something that has a lot of power to do many different things, it'll be possible to make it do awful things. I just came from the *Rijksmuseum*, where they have an exhibit called "The Age of Ugliness". It was a whole bunch of fancy silver bowls from the late 19th century .... When you say an art, I'm not sure I understand exactly what you mean. To me, art is used in two quite different senses, most often nowadays in the sense of fine art, while art, originally, *Kunst*, was anything that was not natural — so we have the word *artificial*, something that is made by people instead of by nature. The Greek word is *techne*. [laughter] But then you refer to a tool as something that is maybe just a device that is the fastest way to get from here to there but maybe you don't care about elegance ... But what I think you mean when you talk about art is the aesthetics — something about beauty and something with a little bit of love in it. With TEX, my idea was to make it possible to produce works that you are proud of; I assumed that people can enjoy actually spending a little extra time making the results better. I didn't expect that the whole world would be doing this. [laughter]

Incidentally, I can't understand the mentality of a person who writes graffiti on a beautiful building although I can see why drawing is fun. Why would you want to scrawl something — some kind of animal instinct of territory might account for it, I suppose, but it's really impossible for me to conceive of such actions.

When it comes to matters of aesthetics, you can't dictate taste. You can't say that your idea of beauty is going to match anyone else's idea of beauty. But I did want to have a tool where we could reach the highest levels of beauty according to our own tastes. I didn't allow people to have letterspacing very easily, but I tried to make everything else easy. [laughter] .... Of course, I originally designed TEX just for myself, for *The Art of Computer Programming*; I thought my secretary and I were going to be the only users. And it wasn't until later that I was convinced that I should make it more general and so on. But I did want a tool for myself by which I could produce books that would make me feel good after spending almost all my life writing those books.

I started writing *The Art of Computer Programming* when I was 24 years old and I still have 20 years of work to do on it. That's a lot of time. I don't want to write those books if they're going to come out looking awful. I wanted a way to make it possible [to produce good-looking books]. Originally, when computers started out, they knew only numbers, digits. The 19th-century computers could print tables. Then we had computers that could do numbers and letters, but only on a teletype machine; so you had some capital letters and a 32-character set. But then, after I graduated from college, we got ... let me see, I was probably ten years out of college before we could do lowercase letters on a computer. You know, the PASCAL language, when it came out, it used all uppercase letters — there was never any consideration that there would be more than 64 characters in a computer's repertoire. Finally, we were beginning to see in the middle 70s that computers could actually do lowercase letters, and produce something that looked a little bit readable, a little bit like books. Wow! [laughter]

Then there was this development of typographic software starting at MIT in 1960 and going through 4 or 5 generations, leading to troff and EQN, where there was even mathematics being typeset. In 1977 I therefore knew an existence theorem: It was possible to typeset something that looked almost like good mathematics. EQN was being used in physics journals and experience showed that secretaries could learn how to do it. So I thought, "Why not go all the way to the end, to convergence?" What I wanted to do with TEX was not to be a little refinement over troff and the other things, but I was saying now, "Let me try to go to the best typography that's ever been achieved by mankind" Except for the illuminated gold leaf type of lettering, I wanted to at least — when it came to black and white printing — I wanted to match the best conventions that had been achieved. Computer typesetting had gone through this lengthy development, getting a little better and a little better. It was time to say, "Well, let's jump to the end now." Of course, I didn't think this would be an activity that everybody would want to do. But there were enough people that would care about trying to get as much quality as possible, that they could be — well, that's why I finally made TEX more available. The American Math Society were the first people, nearly the first people who convinced me that I should make the system do more than I originally intended.

Andries Lenstra: Why didn't you start from troff? It was completely inappropriate?

Knuth: Yes, yes. You see, troff was patched on top of ... I mean, there was a whole system, it was a fifth generation, each of which was a patch on another one. So it was time to scrap it and start all over again: "Here's what the language should be, so let's design some good data structures for it." Not "Let's try to be compatible." I had the advantage that I was not at Bell Labs, where I wouldn't be hurting anybody's feelings by saying, "Let's throw it all away." [laughter] It was impossible for the people at Bell Labs to do such a thing — it wouldn't be nice. But it occurred to me that now that we had proof that this goal was possible, I should start over, and rethink how I could get from input to output, so the program could be much more unified, much smaller, and would also work. I mean, troff was collapsing all the time. A lot of the earliest users of TEX had been frustrated by troff breaking over and over again, so it had gotten unwieldy. But it had also proved that there was light at the end of the tunnel.

I also had to scrap TEX, you know, and start over again; after five years, I decided that it would be best to go back and re-do the program. But it would have been very hard to do that if my friend in the next office had done it. [laughter] So, I just have this philosophy that there will be always some people who are more interested in quality than others, and I wanted to make TEX good for them. I don't see any good way to make it impossible to make a bad document, unless you have only a system with a small menu of options; that's good for a large class of users, to make a system that's so simple that you can't possible do anything ugly in it.

Erik: I think it's time for a coffee break now — we'll take five or ten minutes.

Knuth: Johannes, you had a question that you had to ask, so let's get that over with. [laughter]

Johannes Braams: It's about typesetting. What is your opinion about the skyline model of typesetting? In TEX, you talk about boxes: each letter is inside a box, and we glue boxes together to a line, and the line itself is inside a box, and each line is viewed as a box and the boxes are fitted together to form a paragraph. The skyline model tries to go a little bit further than the rigid box and line, and tries to take into account that some of the descenders in the upper line and the high parts in the lower line don't overlap, so that you could actually have lines much tighter together — especially in math typesetting, that could be an advantage.

Knuth: Hmmm, I guess you're talking about general principles of computer graphics where you have rectangles inside a picture, instead of having the rectangles grouped only inside of a rectangle. ... This certainly would be a major change in all the data structures of TEX. You could go to a quad tree structure or something like that. All the things that people use to solve hidden-line problems and do rendering, to find out what's in front of something else, and all the algorithms they use to make movies like *Toy Story*. It would be most valuable, I imagine, for catching unusual cases in math formulas.

I have two feelings about these things. One is that I like to see people extending the things that computers can do automatically. People learn a lot when they try to do this. The whole field of artificial intelligence has been one of the areas that has had greatest spin-offs to computer science because they've tried to solve very hard problems. Especially in the early days, they came up with methods that turned out to be useful in many other parts of computer science. So, it's my feeling that when people are working on more ambitious goals, they develop powerful techniques that often have very relevant spin-offs. Even so, after they've solved that problem, they're going to think of something else which will be another refinement and so on — they'll never have a situation where they're going to automatically create the most beautiful document. There's going to be a time when you can look at the output and see that you can still improve it. Designers of the most automatic systems would be well advised to at least still leave a chance for somebody to move something up and down and fake out their automatic algorithm.

The philosophy that I had when I did TEX was that I would try to have a system that did 99% of everything automatically, and then I would look at what remained and I would kludge the rest. But kludging it is one way to say it; another way of saying it is "Tidy up the rest," or "Dot the i's and cross the t's." My feeling is that this non-automatic part gives me a little extra pride that I have put the spit and polish on the final product, that I know I did it. If it occurs a lot, then it's a nuisance and I'm wasting time. But if I can really limit this to 1% — if I've spent 30 hours writing a paper and it takes me only another 15 minutes to clean up, then I'm happy to do another 15 minutes at the end. It's a small little extra that gives me a chance to celebrate the fact that I've finished the paper.

The spacing that TeX does worst right now, in my experience, is with respect to square root signs being a little too tight, with the operand either too close to the radical sign or too close to the bar line or both; I find that I'm most often fiddling with that. I've adopted in the book *Concrete Math* and also in *The Art of Computer Programming* now, the convention where in the math formula I put an @-sign where I want one math unit of extra space. The @-sign is then defined to have a math code of hexadecimal 8000, which means that this will invoke in math mode and the @-sign will be regarded as a macro that adds one math unit of space. So I'll type 'square root of' '@-sign' 'log of n' [laughter], because otherwise the space before 'el' is a little bit too tight. Now maybe even this skyline model wouldn't know that 'el' was too tight, maybe it would. But it's cases like that ...

The most common case really is where I have something like 'x squared over 3', where you have a simple superscript and then a slash, and then the denominator. There's almost always too much space before the slash. And this is true, I find, in all the books that I used to think were typeset perfectly by hand [laughter], but now I'm sensitive to this. Now I go through, typically with emacs, and look for all occurrences of something with a one-character exponent followed by a slash, and most of those look better with a negative thinspace before the slash. It would be nicer if I didn't have to do that. But still, it's a small thing for me.

Would the skyline model help me much? Sometimes I run into cases where I'll add another word to the answer to an exercise in order to avoid a clash between lines. Here, the lines are actually not getting spread apart too far, but they're so close together that the 'subscript k less-than-or-equal to n' will clash with a left parenthesis in the next line. And I don't want the type to be quite so close together there. Now, if I had been smarter, I would have designed my $\leq$-sign to have a diagonal stroke under the $<$ instead of a horizontal bar, and I wouldn't have had those clashes — too late for that now. [laughter]

Kees?

Kees: May I ask you a question about your attitude to mark-up in general? And let me illustrate it by first telling a story. When we started with using TeX etc., we mean actually we start with LaTeX — I mean, that is the effect in Holland. And then I looked at the products of the mark-up and I did not like it. And then I was wondering, what is your attitude to that? I'm sorry to say so, I paged through *The TeXbook* file `texbook.tex` and I looked at all the things in there and then I thought, "Well, I have some idea of what your ideas are of mark-up." And when you explained about METAFONT and all those things not in there, which you have implicit — am I wrong if I summarize this, that you adhere to something like minimal mark-up?

Knuth: Yes. For example, when I am reading Edsger Dijkstra's books, every time I get to a section where it says 'End of Comment', it strikes me as redundant. And I always think, "Oh, yes, this is Edsger's style." When I wrote a paper for his 60th birthday, I said at the end, "Acknowledgment, I want to thank Edsger for such-and-such," and 'End of Acknowledgment'. [laughter][4] But that's the only time in my life I'll ever do that. Maybe I'm an illogical person, but apparently half the people using `html` now type only the `p` at the beginning of a paragraph, and the other half type only a `/p` at the end of a paragraph. [laughter] Hardly anybody uses both, according to what my spies tell me. And I don't know what the heck these systems actually do with the unbracketed material. When I write `html`, I'm scrupulous with my mark-up. If you look at my home pages — I'll pay you $2.56 if you find any case where I started something and didn't close it with the right tag. I tried to be very careful in that, and to indent everything very well, and so on. But I found it a terrible nuisance, because it's not the way I think.

I think a high-level language, to me, is something that should reflect its structure in some visual way but not necessarily explicitly; so that, when I know the conventions, we can suppress some things. Parentheses are one such convention and mathematics got a lot better when people invented other notations like operator precedence that we can see structure without spelling it out in too much detail. A mathematician spends a lot of time choosing notations for things, and one of the things we try to avoid in mathematics is double subscripts. I read one French PhD thesis where the author had five levels of subscripts [laughter] — he kept painting himself into a trap. He started out with a set x_1 through x_n, so then when he talked of a subset, it had to be x sub-i_1 through x sub-i_k, and then he wanted to talk of a subset of this, so then he had a theorem that says, let a sub-b sub-c ... and so on. [laughter]

[4] *Beauty is Our Business* (Springer, 1990), 242.

I try to choose notations that give me the economy of thought at a high level.

That's why I probably didn't believe in a great deal of mark-up in *The TEXbook*; I would begin typewriter type and end typewriter type for sections by saing `\begintt` and `\endtt`. I would also delimit the lines and when I'm presenting parts of the plain TEX macros, `\beginlines` and `\endlines` — those macros are in the file, since it's very important to me to see how that works. But in other cases, I've left [things] as simple as possible, for me to see visually the beginning and end of stuff. It's something also like problem solving — sometimes, if I've solved a problem and I'm not worried about it anymore, I forget to tell anybody else the solution. I was always a very bad committee chairman because I'm not very good at finishing that last ending line, I guess. Still, with `html`, the document was short and I decided that my home pages were going to be used by many different kinds of browsing software so I had better be very rigorous.

While I was developing TEX, I attended one of the meetings of the committee that designed SGML and had a very good discussion with Charlie Goldfarb and the other people on the committee — we only had that one meeting near Stanford. Certainly I appreciate the fact that this structure makes it possible to build other kinds of programs around what you have. The more structure you have in a document, the easier it is to make a database that includes things about it, and knows what's going on. I never objected to it; I just always felt that in order to maximize my efficiency, I didn't want to mess around with full mark-up unless I had to.

X: SGML allows minimizations; that's why the end-paragraph is not necessary. So that's one of the reasons why it's so difficult sometimes. You have a formalization to minimize.

Knuth: But LATEX doesn't allow it.

Johannes: But we do have some books, however, permitting omitted end-tags in LATEX3, but that's not far enough along.

Knuth: Well, talk to him. [laughter] I don't need a special editor for `html` — people are hyping fancy things where you can click on a tool and it'll put in the start and end tag together — but when I wrote my files, I did make up a simple emacs macro that would take whatever tag I just typed and create the end-tag. All it had to do was search back till it found a less-than sign and then copy that string twice and put a slash in front of it, so I used that all the time — it was easy.

Johannes: Quite different type of question now, from someone who'd like to ask here: literally, he writes, "Why is the height of the minus sign in the cm symbol font the same as the height of the cmr plus sign?"

Knuth: Ah. A lot of people are wondering about that one. Where you have 'a minus c' or you say 'x sub minus' or something, why is it that the height and depth are greater than the actual shape of the minus sign?. In fact, it's not just the plus and minus, it's also the $+$, $-$, $\pm$, *MetaPost*, $\oplus$, $\ominus$, $\otimes$, $\oslash$, $\times$ and $\div$ — if you look at the code for these, there is a **beginarithchar** macro that begins all of the arithmetic characters in the font, guaranteeing that they will have the same size.

Johannes: But it doesn't say why.

Knuth: That's right — it doesn't say why. And the reason is that early on, I wanted certain things to line up the same. For example, if you had

$$\sqrt{x+y}+\sqrt{x-y},$$

I wanted the square root signs to be place in the same way. Otherwise, you would get

$$\sqrt{x+y}+\sqrt{x-y}.$$

And so there are many other cases where you have formulas where there's a plus sign in one part of a formula and a minus sign in the other part, and for consistency of spacing, it ought to look symmetrical. There are other cases, I readily admit, where you have only a minus sign — you never have a similar thing with a plus sign, and you wonder why there's extra space left there. So I say `\smash minus` [laughter] in those cases.

Johannes: The particular application, why this question was asked — Michael Downes from the AMS —

Knuth: Yes, Michael Downes, he has more experience than any of us in this room; he's the chief typesetter of most of the mathematics in the world.

Johannes: He has a problem properly attaching a superscript on top of the `\rightarrowfill` ...

Knuth: The `\rightarrowfill`? OK ... The `\rightarrowfill` is this thing that makes a right arrow of any desired length, and then he wants to put a superscript on this. What's the macro for building that up? I haven't used that page in a long ... [laughter][5] The `\rightarrowfill` is made up of minus signs and so probably if I had known Michael ... known about that in the old days, I would have changed the plain TEX macros so that

[5] Knuth was trying to remember `\buildrel`; see *The TEXbook*, p. 437.

it would not use the height of the minus sign in the `\rightarrowfill` operator [...][6] Anyway, I've now told you the reason why it's there for the other ones.

Johannes: Another question, which is about multiple languages. There's a problem when you have one paragraph where you have different languages.

Knuth: Yes, the `\lccode` changes. This is the ...

Johannes: And I've been told that inside one paragraph you can only use one hyphenation table, which is the one which is active at the end of the paragraph. So, switching hyphenation tables inside paragraphs. Suppose, for example, you have a paragraph with English text, with a German quote inside it, the German quote being several lines long.

Knuth: I know that TEX will properly keep track of which hyphenation table to use. The glitch, the mistake, that I didn't anticipate is if the two languages have different `\lccode` mappings — so that each has a different idea of which characters are lowercase. When you hyphenate, you need to hyphenate an uppercase word the same as an lowercase word, so TEX uses the `\lccode` of a character to convert every letter into the lowercase code of that letter. I didn't anticipate that people might, for different languages, have a different mapping from uppercase to lowercase. And so it's that mapping that, at the end of a paragraph, applies to all the languages in the paragraph. But otherwise, TEX is careful to keep track of what language you have.

And by the way, there's a file called `tex82.bug`. Go to the CTAN archives, and find subdirectory `/knuth`, and under that `/errata`, and that's where this is. At the end of `tex82.bug` this particular error about `\lccode` is mentioned as being something that's an oversight that's too late to fix.

Marc van Leeuwen: Why is it too late to fix? It would conflict with other things?

Knuth: Yes. So that people are already using these things in lots of documents, and it's very hard to change. In fact, I don't see any way to fix it. [laughter] I would say that when you are faced with a situation where you're doing multiple languages with multiple `\lccode`s, this is a good reason to write your own version of TEX.

Andries Lenstra: Could I ask a question? Happily enough, I'm not the first person to mention LATEX, so I may mention it now. There's a situation that often arises when people try to write a PhD thesis where they want to change LATEX code because they think they know better about things of beauty or typography, and unhappily enough they are not experts on LATEX, so they don't succeed or they succeed badly. In general, people who know about typography can't write beautiful LATEX code or other forms of code, and vice versa — people who know how to write these forms of code, are no experts on typography. What do you think of the endeavors in the past to bring the two worlds together, for instance, as Victor Eijkhout has tried to do with his `lollipop` format, a machine to create other formats. I would have thought that it would have had a big success but the opposite seems to be the truth. What do you think of it?

Knuth: I'm not familiar with the details of `lollipop`. I suppose that was based on a famous quotation from Alan Perlis, who said that, "If somebody tells you that he wants a programming language which will only do the right thing, give him a lollipop."

Andries: Yes.

Knuth: I'm sure that the lollipop effort was instructive and worthwhile, but I don't know the details so I can't answer in great detail on this. Probably the type designers didn't find the language easy to learn. I do think that we're having much more communication now, as every month goes by, between the people that know about type and the people that know about macros. It's just a matter of time as we wait for these waves to continue moving — we're nowhere near a convergent stage, where TEX has reached its natural boundary and the type designers have reached their natural boundary. They're still moving toward each other. I don't think it's like a hyperbolic geometry, where they never will get together.

The main difficulty of course is that TEX is free, and so a lot of people will say, "Well, how could it be any good, if you're not charging money for it?" A lot of the people in the type design community would only work in things where there's money behind it; money proves to them that it's worth talking to people. So it just takes a little while till they see some good examples, which will make them more open for these discussions. And that's happening all the time in different countries.

In the Czech Republic I was quite delighted to learn that the new encyclopaedia in Czech, which is the first one for many years, is being done with TEX. And not only that, it's being done with a very high budget. They made this decision because they tried all the other systems and were disgusted with

[6] In fact, the `\leftarrowfill` and `\rightarrowfill` macros now omit the height and depth of the minus, in `plain.tex` version 3.14159 (March 1995).

them. They had good results with TeX. Many other commercial publishers are using it too because they talk to their friends at the big publishing houses. This will, I think, be solved with time. And products like `lollipop` are very worthwhile in the meanwhile to facilitate this. It takes time to bring different communities together. I think the financial factor is definitive for a lot of people.

Piet van Oostrum: I don't know if you have ever looked into the LaTeX code inside, but if you look into that, you get the impression that TeX is not the most appropriate programming language to design such a large system. Did you ever think of TeX being used to program such large systems and if not, would you think of giving it a better programming language?

Knuth: In some sense I put in many of the programming features kicking and screaming, and I'll try to explain the background. I know how Leslie went about writing LaTeX
Dash first he would write the algorithms out in a high-level programming language, with **while**'s and **if-then**'s and so on, and then he would pretty much mechanically convert this to TeX macros. If I had suspected that such a style was going to be the most common use of TeX, I probably would have worried a lot in those days. Now, computers are so fast that I don't worry so much about the running time, because it still seems to go zip!

In the 70s, I had a negative reaction to systems that tried to be all things to all people. Every system I looked at had its own universal Turing machine built into it somehow, and everybody's was a little different from everybody else's. So I thought, "Well, I'm not going to design a programming language; I wanted to have just a typesetting language." Little by little, I needed more features and so the programming constructs grew. Guy Steele began lobbying for more capabilities early on, and I put many such things into the second version of TeX, TeX82, because of his urging. That made it possible to calculate prime numbers as well as do complicated things with page layout and figure placements. But the reason I didn't introduce programming features at first was because, as a programmer, I was tired of having to learn ten different almost-the-same programming languages for every system I looked at; I was going to try to avoid that. Later, I realised that it was sort of inevitable, but I tried to keep it as close to the paradigm of TeX as a character-by-character macro language as I could. As I said before, I was expecting that the really special applications would be done by changing things in the machine language code. But people didn't do that, they wanted to put low-level things in at a higher level.

Piet: What do you think, for example, of something like building in a programming language which is, from a software engineering point of view, easier to use?

Knuth: It would be nice if there were a well-understood standard for an interpretive programming language inside of an arbitrary application. Take regular expressions — I define UNIX as "30 definitions of regular expressions living under one roof." [laughter] Every part of UNIX has a slightly different regular expression. Now, if there were a universal simple interpretive language that was common to other systems, naturally I would have latched onto that right away.

Piet: The Free Software Foundation is trying to do that and Sun is trying to do it and Microsoft is trying to ...

Knuth: The Free Software Foundation is trying actually to include also the solutions of Sun and Microsoft. In other words, to make all of the conventions work simultaneously as much as possible. And that conflicts with my own style, where I've tried to have unity rather than diversity ... I didn't go for ten ways to do one thing. C++ is similar — when the committee would say, "Well, we could do it this way or this way," they did both. I hadn't gone that route in my system, because it is messy. But I admit that the messy way is the best that can presently be realized in practice.

Marc van Leeuwen: I have a question about literate programming. I know you must be very fond of it, if I understand your interviews —

Knuth: Yes, I'm so fond of it that I could ... well ... OK. [laughter] You know, I'm really so fond of literate programming, it's one of the greatest joys of my life, just doing it.

Marc: My question was that obviously it's not nearly as popular as TeX is, and, what's more, there isn't much coherence in the world of literate programming. There are a dozen different systems being used — some people favor this, some people favor that — and this worries me a bit. I too am very fond of this style of programming, but I would like to see it being used much more.

Knuth: Literate programming is so much better than any other style of programming it's hard to imagine why the world doesn't convert to it. I think that Jon Bentley put his finger on the reason and it was something like this: There aren't that many people in the world who are good programmers and there aren't that many people in the world who are good writers, and here we are expecting them to be both. That overstates the case but it touches the key point. I think that everyone who's looked at literate programming agrees that it's a really good way to go, but they aren't convinced that ordinary students can do it. Some experiments at Texas are proving otherwise, and I've had a smaller-scale experience at Stanford. It's a hypertext way of programming and I imagine that with better hypertext systems that we're seeing now and people becoming so familiar with the Web, we're going to get a variety of new incompatible systems that will support literate programming. Hopefully somebody with time and talent, and taste, will put together a system of literate programming that is so charming it will captivate a lot of people. I believe that the potential is there, and it's just waiting for the right person to make that happen.

Marc: I think one problem might be that if you compare your programs with the average program that people write, there just aren't nearly as many interesting algorithms in the average program, so literate programming doesn't add too much to a program which is very dull by itself.

Knuth: Well, thank you for your comment. But maybe sometimes I make a non-interesting algorithm interesting just by putting in a joke here or there. I've taken programs that I got from Sun Microsystems, for example, and as an exercise, spent the afternoon converting them to a literate form. There weren't any exciting algorithms in there, but still, you could look at the final program and it was better — it had better error diagnostics, better organisation, it corrected a few bugs. I don't have time to go over to Sun and show them this, and say, "Why don't you rewrite your operating system?" [laughter] but I know that it would be much better. So all I ever published was the very simple rewrite of the wc word count routine in UNIX, which is not at all an exciting algorithm, but as a demo of how it could be done.[7]

My approach to literate programming isn't the only one, of course, and in the recent book by a group at Princeton, *A Retargetable C Compiler*, Chris Fraser and Dave Hanson used a variety of literate programming to describe their C compiler. Other books are coming out now that are using some flavors of literate programming. I was talking to someone at Microsoft who said that he thought literate programming was on the rise, and I said, "Does that mean the next version of Windows is going to be all done in literate programming?" "Well, no, not exactly" ... [laughter] The people who've experienced literate programming will never go back, and they'll probably gan influence gradually. The companies that use it are going to sell more products than their competitors, so pretty soon this will happen. I imagine that there are about ten thousand users of literate programming and a million users of TEX, so it's a factor of a hundred.

Marc: Do you think it still has to develop? I get the impression that with so many tools around, that it's not yet mature. The idea is mature, but the implementation still has to ...

Knuth: Yeah, that's true. There's great need for programming environments based on this idea. It's not at all easy to create these environments and to have the power to promote them and maybe the support to do it in a way that wouldn't make it too expensive or too hard for people to install. The most ideal thing would be if the Free Software Foundation were to adopt it, or something like that, or some of the people they work with. Actually, [Richard] Stallman [of the FSF] designed a variant of literate programming that he uses, and he has it well integrated with TEX, in his own style. He hasn't put it in too many of his programs, but he has a version. It's one of these things that needs, as you say, to mature.

Marc: Do you believe it should go in the direction of integrated systems, where you really have all the facilities you need in one system? Because I think the tendency is more towards very minimalistic systems that do not do any pretty printing because that gets you into too much trouble when you're switching programming languages. So it really boils down to something which is very flexible but not very convenient for someone to use.

Knuth: One programming language is good enough for me so I'm not the right person to ask. But then, I guess, for the *The Art of Computer Programming*, for the next twenty years, I'm pretty sure that CWEB is going to be as good as anything I need. I'll write programs for Mathematica and I'll write some programs for MetaPost; I could develop or use literate programming for those programs, but I don't think I will. I don't write so many lines that I would gain a great deal ... although I would get a better program afterwards. Unless somebody already presents me with a good system for it, I won't go ahead with MathWeb or MPWeb. But with

[7] D.E. Knuth, *Literate Programming* (Stanford, 1991), 341–348, based on a prototype by Klaus Guntermann and Joachim Schrod, *TUGboat* **7** (1986), 135–137.

CWEB, I'm going to write an average of five programs a week for the foreseeable future, and there, my productivity is infinitely faster when I do it with literate programming.

One other thought flashed in my mind as I was talking just now ... I wrote a paper last year, I think it was, about mini-indexes for literate programs[8] and here I was trying to show what sort of programming environment would help me. In the listings for TEX the program and METAFONT, and also for the Stanford Graphbase, on the righthand page of each two-page spread, there's an index to all the identifiers used on that page and where they were declared. My paper explains the system I used to get those indexes, and this kind of functionality would also be needed in any hypertext system. These minimalistic systems are attractive primarily because a good programmer can write them in a couple of days, understand them and use them, and get a lot of mileage out of them. Once somebody writes a good hypertext system for literate programming, I think that'll attract a lot of people — a system that doesn't crash, and has a familiar user interface because it's like other hypertext systems that we're already using. The time for that will be ripe in about two years.

Erik: It's half past nine now and I think we'll have to stop here. I want to thank our special guest, Donald Knuth, for the time here. I think we've all learned a lot now. We're very happy that you were able to be here. Thank you very much.

Knuth: I really appreciate all the work you did to get this special room here on rather short notice. [applause]

Erik: Also, thank you to Elsevier Science, who helped, in the person of Simon Pepping; and your English colleague, Sebastian Rahtz, who is not here, although I expected him. But he paid for the coffee and tea, so thanks. There's of course a little present that we have for you. I hope you like it! [He presents a book about Dutch art called '*De Stijl*'.]

Knuth: Yes, ... the Dutch type designer, Gerard Unger, came to Stanford for three weeks, he and his wife Marjan, and they talked about things like this to our type designers. They also related fashion of clothes and furniture and architecture to type styles as well. This is great. This was done with TEX?

Erik: I don't think so ... as we are in Holland now ... [laughter] [He also presents a pair of wooden tulips]

Knuth: A nice gift for my wife.

Erik: And of course a copy of the *EuroTEX'95*. [He presents the proceedings]

Knuth: Oh!! I thought you'd never ... [laughter] Yes, I was looking at this last week in the Czech Republic, so thank you everyone.

Erik: What is your opinion about the fonts we used?

Knuth: I think it's ... oh, you used the Computer Modern Brights. Yes, the only complaint I had was that the kerning in the word 'TEX' itself could be tuned a little bit.[9] It's quite attractive — thank you very much.

[8] *Software Concepts and Tools* **15** (1994), 2–11.

[9] "The TEX logo in various fonts," *TUGboat* **7** (1986), 101.

CSTUG, Charles University, Prague, March 1996 Questions and Answers with Prof. Donald E. Knuth

Karel Horák:
[Introductory remarks in Czech, then English.]

I'm very glad to have such a happy occasion to introduce you, Professor Knuth, to our audience, who are mostly members of CSTUG, the Czech/Slovak TEX User Group, but also some academicians from Prague because this session is organized by CSTUG and the Mathematics Faculty of Charles University. We are very happy to have you here, and I would be happy, on behalf of Charles University, to give you a special medal. [wide applause]

DEK: [surprised] Thank you very much.

Prof. Ivan Netulka: Professor Knuth, dear colleagues, dear friends, ladies and gentlemen. I feel really very much honored having the opportunity to greet Professor Donald Knuth, as well as most of you here sitting in this guildhall, on behalf of the Dean of the Faculty of Mathematics and Physics of Charles University, Professor S.G. Sedwa.

As far as I know, Professor Knuth has come to Prague for the first time. Despite this fact, he has been known here, not only among *all* mathematicians, *all* computer scientists, but also many physicists, and even to people having nothing to do with our subjects. People here are fully aware of the significance of Donald Knuth's [...] treatise, *The Art of Computer Programming*. Many of us have had the opportunity to be pleased by reading the charming booklet devoted to *Surreal Numbers*. We know — and here I am going to fall [stumbled] — that Donald Knuth's favorite way to describe computer science is to say that it is the study of algorithms. We share his opinion that the study of algorithms has opened up a fertile vein of interesting new mathematical problems and that it provides a stimulus for many areas of mathematics which have been suffering from a lack of new ideas.

My personal experience — the personal experience of a mathematician — says that, for every mathematician, there exists a personality who [has] brought an extraordinarily great service to his field. Here we have a rare case where, in that statement, the order of the quantifiers may be reversed, maybe: There exists a personality who [has] brought a great service, an extraordinarily great service, to every mathematician. Here is my one-line proof: Donald Knuth — TEX.

Professor Knuth, in acknowledgement of your achievements in computer science, in mathematics, as well as in computerized typography, which has given the whole of the community an excellent tool for presenting scientific results, the Faculty of Mathematics and Physics of Charles University [has] decided that you be awarded the Faculty's Memorial Medal. I am happy to make that presentation now. [wide prolonged applause]

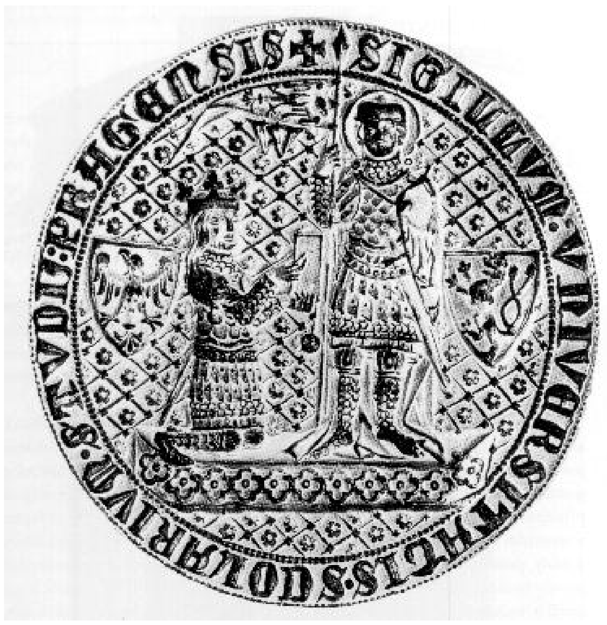


Figure 1: The Seal of Charles University

DEK: Well, this is a quite beautiful medal; I hope you can come and look at it. "Universitas Carolina Pragensis" — so we all speak Latin; maybe I should speak Latin today. [laughter]

I don't know much about the Czech language, but I've tried to learn some of it. On many doors this week I see the word "Sem". [laughter] And then as I came up to this lecture hall today, there were many other signs that said "TEX". [laughter] So I thought we could have an especially powerful version of TEX [writes 'SemTEX' on the blackboard; more laughter] but perhaps it's dangerous; I don't know....

This morning I have no prepared lecture, but I want to say just what you want to hear, so I want to answer your questions. This is a tradition that I maintained in California: The very last session of every class that I taught at Stanford was devoted to questions and answers. I told the students they didn't have to come to that class if they didn't want to, but if they came I would answer any question that they hoped to have answered when they signed up for the class. I actually borrowed this tradition from Professor [Richard] Feynmann at Caltech. And I decided I would do it in my classes, too; it's a wonderful idea that I recommend to all professors — to have open-ended question and answer sessions.

I've recently made some home pages on the World Wide Web that you can get via
`http://www-cs-faculty.stanford.edu/~knuth`
and there on those pages I have the answers to all frequently asked questions. But today, you can ask me the *un*frequently asked questions. [laughter] By the way, I'll tell you one more joke and then we'll get started. Do you know what the home page is of OJ — O.J. Simpson — in the United States? It's "http colon slash slash slash backslash slash escape". [laughter]

Now, please ask me questions. [pause]
Well, if there are no further questions, ... [laughter]. You may ask in Czech, and then someone will translate.

?: Maybe a question to start [with]. I learned TEX carefully, and I had a problem when someone asked me to take the integral with tilde accent. I found that maybe there isn't one with TEX because you can't specify an italic correction to boxes.

DEK: The italic correction is ... With each character there's a limited amount of information that goes in the data structure for each character, and so we have [drawing on blackboard] the height, the depth, the width, and the italic correction. But those are the only numbers that are allowed, and in mathematics mode, the italic correction is used in a different way from outside of mathematics. In mathematics mode, the italic correction is actually used for subscripts; it's the amount by which you would bring the subscript to the left — otherwise, it would typeset "P sub n" (P_n) like this: $P\ _n$.

The italic correction on the integral sign might even be another case because the large operators use the italic correction to cover the spacing between the lower limit and the upper limit. Anyway, there's only one number in there. If you want a special construction that demands many more numbers, the only way I know is to make a special macro for that. I would carry the information somewhere up in the TEX level, not in the inside, not with the character. You would have to build a structure that has this information in it. I don't know how general a solution you need, but certainly if you said the ... I can't even remember the name now ... my goodness, how do you get the ... like the same mechanism by which someone would take an equal sign and then put something over it, like this. It's defined in plain TEX by a macro ...

?: It's something like `\mathord` and upper limits ... [he means `\buildrel`]

DEK: I would build it up out of the primitives, but if you had different integral signs, you would probably have to allow the person who specified the font to ...

?: I have a solution, but it is not a TEX solution: I used METAFONT to produce special characters, which have the [...]

DEK: Yes, using METAFONT would be the ideal way to get the correct artistic effect, but then everyone else has to get your METAFONT code and compile your font. Just by a combination of boxes and glue, you should be able to position the characters that you have. You could just make a `\vbox` [drawing on blackboard] or a `\vcenter` of something or other, and then you build the `\hbox` of ... with a kern and then a tilde or so on. Otherwise, I don't know any simple way of doing exactly that balancing because it's complicated by the visual proportions of the spacing with integral signs — it gets really complicated to handle *all* cases.

My general philosophy with TEX was to try to have a system that covers 99% of all cases easily [laughter]; and I knew there would always be a residual number. But I felt that this residual would only be needed by the people who really care about their papers, and then if they're only spending 1% of the time on this, then they would enjoy feeling that they had contributed something special by adding their little signature, their special character to it. So, I didn't try to do everything automatically. I still believe that it's worthwhile thinking about how to do more automatically, but I don't believe you ever get all the way there.

Karel Horák: I would be very interested in your way of thinking — when you started thinking about making TEX and the typesetting system — when you realized that you also needed to produce some letters, to have not only TEX but also METAFONT. Because — I don't know too much about all types [typefaces] of digital typography — but I think there weren't very many types which you *could* use with TEX, so probably you started thinking about METAFONT, about something like that, from the first?

DEK: Exactly.

I have to erase this beautiful calligraphy [laughter]. It's too late now; well, whoever did it can do it again later, but I need the board. It's gorgeous, although this should really be a different "A". [laughter]

Let's go back to April 1977. [writes on board] I sat down at a computer terminal and started writing a memorandum to myself about what I thought would be a good language for typesetting. And in May 1977, I began working on fonts. This was going to be my sabbatical year, where I would do no teaching through the end of 1977, and the beginning of 1978. I thought that I would write a typesetting system just for myself and my secretary. [laughter] I had no idea that I would ever be seeing TEX on, for example, the tram signs in Brno [laughter] or by the churches of the city, and so on. It was just for my own purposes, and I had one year to do it. And I thought it would be easy. So, in May of 1977, I went to Xerox PARC, the place where the ideas of mouses and windows and interfaces and so on were being worked on, and I knew that they were playing with splines for letterforms. I saw Butler Lampson at a computer terminal, and he was adjusting splines around the edges of letters that he had magnified; so I thought, "good, I'll make an arrangement to work at Xerox PARC during my sabbatical year, and use their cameras and make the type."

I knew from the beginning that I wanted the type to be captured in a purely mathematical form; I wanted to have something that would adapt to technology as it kept changing, so that I would have a permanent mathematical description of the letters. Unfortunately, Xerox said, "Yes, you're welcome to use our equipment, but then we will own the designs, they will be the property of Xerox." I didn't want any of this work to be proprietary; I didn't want people to have to pay to use it. ... A mathematical formula is just numbers — why shouldn't everybody own these numbers?

So instead, I worked only at Stanford, at the Artificial Intelligence Laboratory, with the very primitive equipment there. We did have television cameras, and my publisher, Addison-Wesley, was very helpful — they sent me the original press-printed proofs of my book, from which *The Art of Computer Programming* had been made. The process in the 60s that I wanted to emulate was interesting: They would first print with metal type, Monotype, onto good paper, one copy. They made one copy with the metal, then they photographed that copy and printed from the photograph. They gave me that original copy from which they had made the original photographs. So I could try putting the TV camera on that, and go from the TV camera to a computer screen to copy the letters. At that time, we could connect our display terminals to television and movies on television; people were looking at the titles of movies, and capturing the frames from the movies and then making type. They would keep waiting for more episodes of *Star Trek* or something so that we would have the whole alphabet; eventually we would get a title with the letter "x" in it. That's how we were trying to get type by means of television at the time.

I thought it would be easy, but immediately I noticed that if I turned the brightness control a very little bit, the letters would get much thicker. There was a tremendous variation, so that what I would see on my TV screen had absolutely no consistency between a letter that I did on Monday and a letter that I did on Tuesday, the following day. One letter would be fat and one letter would be thin, but it would be the same letter because the brightness sensitivity was extremely crude. This is still true now: If you look at a scanner and you change the threshold between black and white, a small change in the threshold changes the character of the letter drastically. So I couldn't use TV.

For the next attempt, my wife made photographs of the pages and then we took our projector at home and projected them down a long hallway. On the wall I would try to copy what the letters were. But at that point I realized that the people who had designed these typefaces actually had ideas in their mind when they were doing the design. There was some logic behind the letters. For example, you have the letter 'm', you have the letter 'n', you have an 'i' and an 'l', and I noticed that the 'm' was 15 units, the 'n' was 10 units, and the 'i' was 5 units. Aha! A pattern! The 'l' was 5 units, the 'f' was 5 units, the 'fi' ligature was 10 units. So, if you cut off the tops of these letters, you would see an exact rhythm of 5 units between stems. Great — there were regularities in the design! That's when it occurred to me that maybe I shouldn't just try to copy the letterforms, but I should somehow try to capture the intelligence, the logic, behind those letterforms. And then I could do my bold font with the same logic as the regular font.

The truth therefore is that in May 1977 I didn't know what to do about fonts; June 1977 is when I started to have the idea of METAFONT.

I spent the summer of 1977 in China, and I left my students in California; I told them to implement TeX while I was gone. [laughter] I thought it would be very easy; I would come home and they would have TeX working, and then I could do the fonts. But when I got back, I realized that I had given them an impossible task. They actually had gotten enough of TeX running to typeset one character on one page, and it was a heroic achievement, because my specifications were very vague. I thought the specifications were precise, but nobody understands how imprecise a specification is until they try to explain it to a computer. And write the program.

When I was not in China — in June, the first part of July, and September, October, November — I spent most of my time making fonts. And I had to, because there was no existing way to get a font that would be the same on different equipment. Plenty of good fonts existed, but they were designed specifically for each manufacturer's device. There was no font that would go to two devices. And the people at Xerox PARC — primarily John Warnock — were still developing their ideas; they eventually founded Adobe Systems about 1980 or so. Now, with the help of many great designers, they have many beautiful fonts. But that came later, about two or three years after I had an urgent need for device-independent type.

My lecture to the American Math Society was scheduled for January 1978. The transcript of the lecture that I gave, the Gibbs Lecture to the Society, shows the work that I did with fonts in 1977. It was a much longer task than I ever believed possible. I thought it would be simple to make something that looked good — it was maybe six years before I had anything that I really was satisfied with.

So, the first big ideas were to get fonts that would be machine independent and work on many different computers, including future ones that had not been invented, by having everything defined in mathematics. The second idea was to try to record the intelligence of the design. I was not simply copying a shape, I also would specify that if part of the shape changes, the other should change in a logical way. My goal was to understand the designer's intention, and not just copy the outcome of the intention.

Well, I didn't have TEX running until May of 1978 — I didn't have TEX — I drew the fonts first. For the article, "Mathematical Typography", my talk to the American Math Society,[1] I made individual letters about 4cm high and I pasted each one on a big sheet of paper and took a photograph of that.

That's a long answer. I hope I answered the question.

Karel Horák: I have another question about this system; it is, when you started to learn typography, you had some knowledge before, or you started in the process, learning more and more? Because my experience with *The TEXbook*, and [that of] others also, is that there is very much about typography. You can learn a lot about typography, much more than some people who are doing typesetting on the professional level, using those windows mouse systems. They never can learn from the books which are supplied with those systems.

DEK: Thank you. So, what was my background before 1977? When I was in secondary school — like gymnasium — I had a part-time job setting type (so-called) on what was known as a mimeograph machine. I'm not sure what would be the equivalent here. On a mimeograph you had a sort of blue gelatinous material. The typewriter typed into it and it made a hole. I would also use a light table, and special pens, and try to make music or designs on the mimeograph stencil. I had a summer job where I would type, and then I would use my stylus to inscribe pictures on the gel. So I knew a little bit about typography. This was not fine printing, of course; it was very amateurish, but at least it gave me some idea that there was a process of printing that I could understand. After making the stencils, I would run the machine, and cut the paper, and so on. I was doing this as a student.

Later, my father had a printing press in the basement of our house, and he did work for the schools of Milwaukee; this was to save money from going to the professional places. He would work for some architects that were friends of ours, to make their specification documents. Also in the schools, there would be a program for a concert, or graduation ceremony, something like that, printing tickets for football games ... he would do this in our basement. He started with a mimeograph machine, then he upgraded to something called a VariTyper, which was marvelous, because it had proportional spacing — some letters were wider than others; the fonts were terrible, but we had this machine, and I learned how to use it.

Still later, I started writing books, *The Art of Computer Programming*. So, by 1977, I had been proofreading thousands of pages of galley proofs. I certainly was looking at type. And you might say I was getting ink in my blood.

But I also knew that engineers often make the mistake of not looking at the traditions of the past. They think that they'll start everything over from scratch, and I knew that that was terrible. So actually, right during April and May of 1977, when I was thinking about starting my sabbatical year of typesetting, I took a trip with the Stanford Library Associates, a group of book lovers from Stanford. We visited places in Sacramento, California, where people had special printing presses. We stopped at a typographic museum, which had a page from a Gutenberg Bible, and so on. Everywhere we went on this tour, I looked intently at all the letters that I saw. And I saw people's collections of what they felt was the finest printing.

At Stanford Library there is a wonderful collection of typographic materials donated by a man named Gunst, who spent a lifetime collecting fine printing. As soon as I got back from the library trip, I knew about the Gunst collection, so I spent May and June reading the works of Goudy and Zapf and everything I could find, back through history. First of all, it was fascinating, it was wonderful, but I also wanted to make sure that I could capture as well as possible the knowledge of past generations in computer form.

The general idea I had at that time was the following. At the beginning, when I was young, we had computers that could deal only with numbers. Then we had computers that knew about numbers *and* capital letters, uppercase letters. So this greatly increased our ability to express ourselves. Even in Volume 1 of *The Art of Computer Programming* when I designed my `MIX` computer, I never expected that computers could do lowercase letters. [laughter] The Pascal language was developed approximately 1968, 1969; Pascal originally used only uppercase letters, and parentheses, commas, digits, altogether 64 characters.

Next, in the early 1970s, we had lowercase letters as well, and computers could make documents that looked almost like a typewriter. And then along came software like the *eqn* system of UNIX, which would make documents that approached printing.

[1] *Bull. (N.S.) Amer. Math. Soc.* **1** (1979), pages 337–372; republished in *TEX and METAFONT: New Directions in Typesetting*, Bedford, MA: Digital Press, 1979.

You probably know that *troff* and the *eqn* system for mathematics were developed at Bell Labs. This was an extension of a program that began at MIT in 1959 or 1960, and it developed through a sequence of about five levels of improvement, finally to *eqn* in 1975.

So I knew that it was possible, all of a sudden, to get better and better documents from computers, looking almost like real books. When contemplating TEX I said, "Oh! Now it's time to go all the way. Let's not try to *approach* the best books, let's march all the way to the end — let's do it!" So my goal was to have a system that would make the best books that had ever been made, except, of course, when handmade additions of gold leaf and such things are added. [laughter] Why not? It was time to seek the standard for the solution to all the problems, to obtain the very best, and not just to approach better and better the real thing. That's why I read all the other works that I could, so that I would not miss any of the ideas. While reading every book I could find in the Gunst collection, to see what they could tell me about typesetting and about letterforms, I tried to say, "Well, how does that apply, how could I teach that to a computer?" Of course, I didn't succeed in everything, but I tried to find the powerful primitives that would support most of the ideas that have grown up over hundreds of years.

Now, of course, we have many more years of experience, so we can see how it is possible to go through even many more subtle refinements that I couldn't possibly have foreseen in 1980. Well, my project took more than one year, and I had more than one user at the end. The subsequent evolution is described in my paper called "The errors of TEX", and the complete story after 1978 is told in that paper.[2]

In 1980, I was fortunate to meet many of the world leaders in typography. They could teach me, could fill in many of the gaps in my knowledge. Artisans and craftsmen usually don't write down what they know. They just do it. And so you can't find everything in books; I had to learn from a different kind of people. And with respect to type, the interesting thing is that there were two levels: There was the type *designer*, who would draw, and then there was the *punchcutter*, who would cut the punches. And the type designer would sometimes write a book, but the punchcutter would not write a book. I learned about optical illusions — what our eye thinks is there is not what's really on the page. And so the punchcutter would not actually follow the drawings perfectly, but the punchcutter would distort the drawings in such a way that after the printing process was done and after you looked at the letter at the right size, what you saw was what the designer drew. But the punchcutter knew the tricks of making the right distortions.

Some of these tricks are not necessary any more on our laser printers. Some of them were only for the old kind of type. But other tricks were important, to avoid blots of ink on the page and things like that. After I had done my first work on METAFONT, I brought Richard Southall to Stanford; he had been working at Reading University with the people who essentially are the punchcutters. He gave me the extra knowledge that I needed to know. For example, when stems are supposed to look exactly the same, some of them are a little bit thinner, like the inside of a 'p' — you don't want it to be quite as thick, you want it to be a little thinner; then, after you have the rest of the letter there, the lightened stem will look like it was correct. Richard taught me that kind of requirement. I learned similar things from Matthew Carter, Hermann Zapf, Chuck Bigelow, Gerard Unger, and others.

But we had very primitive equipment in those days, so that the fonts that we could actually generate at low resolution did not look professional. They were just cheap approximations of the fine type. Stanford could not afford an expensive typesetting machine that would realize our designs at the time. Now I'm so happy that we have machines like the LaserJet 4, which make my type look the way I always wanted it to look, on an inexpensive machine.

?: Now that PostScript is becoming so widely used, do you think it is a good replacement for METAFONT — I mean, good enough? Right now, we can use TEX and PostScript ...

DEK: The question is, is PostScript a good enough replacement for METAFONT?

I believe that the available PostScript fonts are quite excellent quality, even though they don't use all of the refinements in METAFONT. They capture the artwork of top-quality designs. The multiple master fonts have only two or three parameters, while Computer Modern has more than sixty parameters; even with only two or three it's still quite good. The Myriad and Minion fonts are excellent.

I'm working now with people at Adobe, so that we can more easily substitute their multiple master fonts for the fonts of public-domain TEX documents. The goal is to make the PDF files

[2] *Software — Practice and Experience* **19** (1989), pages 607–681. Reprinted with additional material in *Literate Programming* (CSLI Publications, Stanford, 1992, and Cambridge University Press), pages 243–339.

smaller. The Acrobat system has PDF files which are much larger — they're ten times as big as `dvi` files, but if you didn't have to download the fonts, they would only be three times as large as the `dvi` files. PDF formats allow us search commands and quite good electronic documents. So I'm trying to make it easier to substitute the multiple master fonts. They still aren't quite general enough. I certainly like the quality there.

Adobe's font artists, like Carol Twombly and Robert Slimbach, are great; I was just an amateur. My designs as they now appear are good enough for me to use in my own books without embarrassment, but I wouldn't mind using the other ones. Yes, I like very much the fonts that other designers are doing.

Asking an artist to become enough of a mathematician to understand how to write a font with 60 parameters is too much. Computer scientists understand parameters, the rest of the world doesn't. Most people didn't even know the word 'parameters' until five years ago — it's still a mysterious word. To a computer person, the most natural thing when you're automating something is to try to show how you would change your program according to different specifications. But this is not a natural concept to most people. Most people like to work from a given set of specifications and then answer that design problem. They don't want to give an answer to all possible design specifications that they might be given and explain how they would vary their solution to each specification. To a computer scientist, on the other hand, it's easy to understand this kind of correspondence between variation of parameters and variation of programs.

In the back?

Láďa Lhotka: I have a problem for you. [question about structured programming]

DEK: I was talking with Tony Hoare, who was editor of a series of books for Oxford University Press. I had a discussion with him in approximately ... 1980; I'm trying to remember the exact time, maybe 1979, yes, 1979, perhaps when I visited Newcastle? I don't recall exactly the date now. He said to me that I should publish my program for TEX.[3]

As I was writing TEX I was using for the second time in my life ideas called "structured programming", which were revolutionizing the way computer programming was done in the middle 70s. I was teaching classes and I was aware that people were using structured programming, but I hadn't written a large computer program since 1971. In 1976 I wrote my first structured program; it was fairly good sized — maybe, I don't know, 50,000 lines of code, something like that. (That's another story I can tell you about sometime.) This gave me some experience with writing a program that was fairly easy to read. Then when I started writing TEX in this period (I began the implementation of TEX in October of 1977, and I finished it in May 78), it was consciously done with structured programming ideas.

Professor Hoare was looking for examples of fairly good-sized programs that people could read. Well, this was frightening. This was a very scary thing, for a professor of computer science to show someone a large program. At best, a professor might publish very small routines as examples of how to write a program. And we could polish those until ... well, every example in the literature about such programs had bugs in it. Tony Hoare was a great pioneer for proving the correctness of programs. But if you looked at the details ... I discovered from reading some of the articles, you know, I could find three bugs in a program that was proved correct. [laughter] These were *small* programs. Now, he says, take my *large* program and reveal it to the world, with all its compromises. Of course, I developed TEX so that it would try to continue a history of hundreds of years of different ideas. There had to be compromises. So I was frightened with the idea that I would actually be expected to show someone my program. But then I also realized how much need there was for examples of good-sized programs, that could be considered as reasonable models, not just small programs.

I had learned from a Belgian man (I had met him a few years earlier, someone from Liège), and he had a system — it's explained in my paper on literate programming.[4] He sent me a report, which was 150 pages long, about his system — it was inspired by "The spirit in the machine". His 150-page report was very philosophical for the first 99 pages, and on page 100 he started with an example. That example was the key to me for this idea of thinking of a program as hypertext, as we would now say it. He proposed a way of taking a complicated program and breaking it into small parts. Then, to understand the complicated whole, what you

[3] "I looked up the record when I returned home and found that my memory was gravely flawed. Hoare had heard rumors about my work and he wrote to Stanford suggesting that I keep publication in mind. I replied to his letter on 16 November 1977 — much earlier than I remembered." – D. Knuth

[4] Pierre Arnoul de Marneffe, *Holon Programming.* Univ. de Liège, Service d'Informatique (December, 1973).

needed is just to understand the small parts, and to understand the relationship between those parts and their neighbors.

In February of 1979, I developed a system called DOC and UNDOC ... something like the WEB system that came later. DOC was like WEAVE and UNDOC was like TANGLE, essentially. I played with DOC and UNDOC and did a mock-up with a small part of TEX. I didn't use DOC for my own implementation but I took the inner part called *getchar*, which is a fairly complicated part of TEX's input routine, and I converted it to DOC. This gave me a little 20-page program that would show the *getchar* part of TEX written in DOC. And I showed that to Tony Hoare and to several other people, especially Luis Trabb Pardo, and got some feedback from them on the ideas and the format.

Then we had a student at Stanford whose name was Zabala — actually he's from Spain and he has two names — but we call him Iñaki; Ignacio is his name. He took the entire TEX that I'd written in a language called *SAIL* (Stanford Artificial Intelligence Language), and he converted it to Pascal in this DOC format. TEX-in-Pascal was distributed around the world by 1981, I think. Then in 1982 or 1981, when I was writing TEX82, I was able to use his experience and all the feedback he had from users, and I made the system that became WEB. There was a period of two weeks when we were trying different names for DOC and UNDOC, and the winners were TANGLE and WEAVE. At that time, we had about 25 people in our group that would meet every Friday. And we would play around with a whole bunch of ideas and this was the reason for most of the success of TEX and METAFONT.

Another program I wrote at this time was called *Blaise*, because it was a preprocessor to Pascal. [laughter]

Petr Olšák: I have two questions.

What is your opinion of LATEX, as an extension of TEX at the macro level? I think that TEX was made for the plain TEX philosophy, which means that the user has read the *The TEXbook* ... [laughter] while LATEX is done with macros, and takes plain TEX as its base. And the second question: Why is TEX not widely implemented and used in commercial places. They use only mouse and WYSIWYG-oriented programs.

DEK: The first question was, what do I think about LATEX?

I always wanted to have many different macro packages oriented to different classes of users, and LATEX is certainly the finest example of these macro packages. There were many others in the early days. But Leslie Lamport had the greatest vision as to how to do this. There's also AMS-TEX, and the mathematicians used Max Diaz's macros — I think it might have been called MaxTEX or something — in the early days before we had LATEX. Mike Spivak and Leslie Lamport provided very important feedback to me on how I could improve TEX to support such packages. I didn't want to ... I like the idea of a macro system that can adapt to special applications. I myself don't use LATEX because I don't have time to read the manual. [laughter] LATEX has more features than I need myself, in the way I do things. Also, of course, I understand TEX well enough that it's easier for me not to use high-level constructions beyond my control.

But for many people it's a simpler system, and it automates many of the things that people feel naturally ought to be automated. For me, the things that it automates are largely things that I consider are a small percentage of my total work. It doesn't bother me that I hand tune my bibliography, but it bothers other people a lot. I can understand why a lot of people prefer their way of working.

Also, when you're writing in a system like LATEX you can more easily follow a discipline that makes it possible for other programs to find the structure of your document. If you work in plain TEX, you can be completely unstructured in your approach and you can defeat any possible process that would try to automatically extract bibliographic entries and such things from your document. If you restrict yourself to some kind of a basic structure, then other processes become possible. So that's quite valuable. It allows translation into other structures, languages and so on.

But I use TEX for so many different purposes where it would be much harder to provide canned routines. LATEX is at a higher level; it's not easy to bend it to brand-new applications. Very often I find that, for the kind of things that I want to do, I wake up in the morning and I think of a project ... or my wife comes to me and says, "Don, can you make the following for me?" So I create ten lines of TEX macros and all of a sudden I have a new language specifically for that kind of a document. A lot of my electronic documents don't look like they have any markup whatsoever.

Now, your second question, why isn't TEX used more in commercial publication? In fact, I was quite pleasantly surprised to see how many commercial publishers in the Czech Republic are using TEX. Thursday night, I saw three or four Czech-English dictionaries that were done with TEX, and you know it's being used for the new Czech encyclopedia. And Petr Sojka showed me an avant garde novel that had been typeset with TEX with some nice tricks of its own very innovative page layout. In America, it's used heavily in legal publications, and behind the scenes in lots of large projects.

I never intended to have a system that would be universal and used by everybody. I always wanted to write a system that would be used for just the finest books. [laughter] Just the ones where the people had a more difficult than ordinary task, or they wanted to go the extra mile to have excellent typography. I never expected that it would compete with systems that are for the masses.

I'm not a competitive person, in fact. It made me very happy to think that I was making a system that would be primarily for mathematics. As far as I knew, there wasn't anybody in the world who would feel offended if I made it easier to typeset mathematics. Printers considered this to be "penalty copy", and it was something that they did only grudgingly. They charged a penalty for doing this extra horrible work, to do mathematics. I never expected that I would be replacing systems that are used in a newspaper office or anything like that. It turned out that after I got going, we found we could make improvements ... in one experiment we re-typeset two pages of *Time* magazine, to show how much better it would be if they had a good line-breaking algorithm. But I never expected when I began that such magazines would ever use what I was doing because, well, it was a billion-dollar industry and I didn't want to put anyone out of work or anything.

So it was very disturbing to me in the early 80s when I found there was one man who was very unhappy that I invented TEX, because he had worked hard to develop a mathematical typesetting system that he was selling to people, and he was losing customers. So he wrote to the National Science Foundation in America, saying, "I'm a taxpayer and you're using my tax money to put me out of business." This made me very unhappy. I thought everything I was doing was for everybody's good. And here was a person I'd obviously hurt. But I also thought that I still should make TEX available to everyone, even though it had been developed with some help from the government. I don't think the government should give money only to things that are purely academic and not useful.

Yes?

?: I have a question about the usage of your typographic programs in commercial institutions like DTP studios and so on. I'd like to ask about using parts of the TEX source. You made clear that the programmers were free to incorporate parts of the TEX source into their own programs. There are some remarkable examples of this, do you know.

DEK: That question came up also last summer when I had a question and answer session at the TUG meeting in Florida.[5] I thought it would be fairly common to have special versions of TEX. I designed TEX so that it has many hooks inside; you can write extensions and then have a much more powerful TEX system readily adapted.

I guess I was thinking that every publishing house using TEX would have an in-house programmer who would develop a special version of TEX if they wanted to do an edition of the Bible, if they wanted to do an Arabic-to-Chinese dictionary or something. If they were doing an encyclopedia, they could have their own version of TEX that would be used for this application.

A macro language is Turing-complete — it can do anything — but it's certainly silly to try to do everything in a high-level language when it's so easy to do it at the lower level. Therefore I built in hooks to TEX and I implemented parts of TEX as demonstrations of these hooks, so that a person who read the code could see how to extend TEX to other things. We anticipated certain kinds of things for chemistry or for making changebars that would be done in the machine language for special applications.

Certainly, if I were a publishing house, if I were in the publishing business myself, I would have probably had ten different versions of TEX by now for ten different complicated projects that had come in. They would all look almost the same as TEX, but no one else would have this program — they wouldn't need it, they're not doing exactly the book that my publishing house was doing.

That was what I thought would occur. And certainly, there was a point in the middle 80s when there were more than a thousand people in the world that knew the TEX program, that knew the intricacies of the TEX program quite well. They had read it, and they would have been able to make any of these extensions if they wanted. Now I would say that the number of people with a working knowledge of TEX's innards is probably less than a thousand, more than a hundred. It hasn't developed to the extent that I expected.

One of the most extensive such revisions is what I saw earlier this week in Brno — a student

[5] *TUGboat* **17**(1) (1996), pages 7–22.

whose name is Thanh,[6] I think, who has a system almost done that outputs PDF format instead of dvi format. If you specify a certain flag saying \PDFon, then the output actually comes out as a file that an Acrobat reader can immediately read. I also expected that people would go directly to PostScript; that hasn't happened yet as far as I know.

No one has done a special edition of the Bible using TeX in the way I expected. There were some extensions in Iceland; I don't remember if they did it at the higher level — I think they did it mostly at the macro level, or maybe entirely.

Anyway, I made it possible to do very complicated things. When you have a special application, I was always expecting that you would want to have a specially tuned program there because that's where it's easiest to do these powerful things.

?: I want to ask which features of TeX were in the first version — for example, line-breaking, hyphenation, and macro processing — if all these things were in the first version?

DEK: The very first version was designed in April 1977. I did have macros and the algorithm for line-breaking. It wasn't as well developed; I didn't have all the bells and whistles like \parshape at that time, but from the very beginning, from 1977 on, I knew I would treat the paragraph as a whole, not just line by line. The hyphenation algorithm I had in those days was not the one that we use now; it was based on removing prefixes and suffixes — it was a very peculiar method, but it seemed to catch about 80% of the hyphens. I worked on that just by looking at the dictionary: I would say, the word starts with "anti", then put a hyphen after the "i"; and similarly at the end of the word. Or if you have a certain combination of letters in between, in the middle, there were natural breaks. I liked this better than the *troff* method, which had been published. The hyphenation algorithm is described in the old TeX manual, which you can find in libraries.[7]

Now, you said the line-breaking, hyphenation, macros, ... I developed the macro language in the following way. I took a look at Volume 2 of *The Art of Computer Programming* and I chose representative parts of it. I made a mock-up of about five pages of that book, and said, "How would I like that to look in a computer file?" And that was the whole source of the design.

[6] Han The Thanh; see Petr Sojka, Han The Thanh and Jiří Zlatuška, "The Joy of TeX2PDF — Acrobatics with an alternative to DVI format", *TUGboat* **17**(2) (1996).

[7] *TeX and METAFONT: New Directions in Typesetting* (cited in footnote 1).

I stayed up late one night and created TeX. I went through Volume 2 and fantasized about natural-looking instructions — here I'll say "backslash algorithm", and then I'll say "algorithm i", and then I'll say "algstep", you know. This gave me a little file that represented the way I wanted the input to look for *The Art of Computer Programming*. The file also included some mathematical formulas. It was based on the idea of *eqn*; the *troff* language had shown me a way to represent mathematics that secretaries could learn easily. And that was the design. Then I had to implement a macro language to support those features.

The macro language developed during 1978, primarily with the influence of Terry Winograd. Terry was writing a book on linguistics, a book on English grammar. He wanted to push macros much harder than I did, and so I added \xdef and fancier parameters for him.

The hyphenation algorithm we have now was Frank Liang's Ph.D. research. He worked with me on the original hyphenation method, and his experience led him to discover a much better way, which can adapt to all languages — I mean, to all western languages, which are the ones that use hyphens.

As far as the spacing in mathematics is concerned, I chose three standards of excellence of mathematical typesetting. One was Addison-Wesley books, in particular *The Art of Computer Programming*. The people at Addison-Wesley, especially Hans Wolf, their main compositor, had developed a style that I had always liked best in my textbooks in college. Secondly, I took *Acta Mathematica*, from 1910 approximately; this was a journal in Sweden ... Mittag-Leffler was the editor, and his wife was very rich, and they had the highest budget for making quality mathematics printing. So the typography was especially good in *Acta Mathematica.* And the third source was a copy of *Indagationes*, the Dutch journal. There's a long fine tradition of quality printing in the Netherlands, and I selected an issue from 1950 or thereabouts, where again I thought that the mathematics was particularly well done.

I took these three sources of excellence and I looked at all the mathematics formulas closely. I measured them, using the TV cameras at Stanford, to find out how far they dropped the subscripts and raised the superscripts, what styles of type they used, how they balanced fractions, and everything. I made detailed measurements, and I asked myself, "What is the smallest number of rules that I need to do what they were doing?" I learned that I could boil it down into a recursive construction that uses only seven types of objects in the formulas.

I'm glad to say that three years ago, *Acta Mathematica* adopted TeX. And so the circle has closed. Addison-Wesley has certainly adopted TeX, and I'm not sure about the Dutch yet — I'm going to visit them next week. [laughter] But anyway, I hope to continue the good old traditions of quality.

I have to call on people who haven't spoken. George...

Jiří Veselý: I have a question. You are asked every time carefully regarding all suggestions and things like that for improvements. Once I was asked about the possibility to make a list of all hyphenated words in the book. I was not able to find in your book a way to do this. I would like to know something about your philosophy what to include and what not to include. What would be in that special package, and what would be in TeX?

DEK: The question is, what is the philosophy that I use to try to say what should be a basic part of TeX and what should be harder to do or special, or something like that. Of course, these decisions are all arbitrary. I think it was important, though, that the decisions were all made by one person, even though I'm not ... I certainly make a lot of mistakes. I tried the best to get input from many sources, but finally I take central responsibility to keep some unity. Whenever you have a committee of people designing a system, everyone in the committee has to feel proud that they have contributed something to the final language. But then you have a much less unified result because it reflects certain things that were there to please each person. I wanted to please as many people as I could but keep unity. So for many years we had a weekly meeting for about two hours every Friday noon, and we had visitors from all over the world who would drop in. We would hear their comments and then we would try to incorporate the ideas that we heard during that time.

Now you ask specifically about why don't we have an easy way to list all the hyphenations that were made in the document. It sounds like a very nice suggestion, which I don't recall anyone raising during those weekly meetings. The words that actually get hyphenated, the decision to do that is made during the *hpack* routine, which is part of the line-breaking algorithm. But the fact that a hyphenation is performed by *hpack* doesn't mean that it's going to appear in the final document, because you could discard the box in which this hyphenation was done.

It's very easy in TeX to typeset something several times and then choose only one of those for the actual output. So, to get a definitive representative of the hyphenation, you'd have to catch it in the output routine, where the discretionary had appeared. This would be easy to do now in a module specially written for TeX. I would say that right now, in fact, you could get almost exactly what you want by writing a filter that says to TeX "Turn on all of the tracing options that cause it to print, to give the page contents." Then a little filter program would take the trace information through a UNIX pipe and it would give you the hyphenated words. It would take an afternoon to write this program, maybe two afternoons ... and a morning. [laughter] You could get that now, but it was not something that I can recall I ever debated whether or not I should do at the time we were having these weekly discussions on TeX.

My paper on "The errors of TeX" has the complete record of all the changes that were made since 1979, with dates, and with references to the code, exactly where each change appears. And so you can see the way the evolution was taking place. Often the changes would occur as I was writing *The TeXbook* and realizing that some things were very hard for me to explain. I would change the language so it would be easier to explain how to use it. This was when we were having our most extensive meeting with users and other people in the group as sources of ideas; the part of the language I was writing about was the part that was changing at the moment.

During 1978, I myself was typesetting Volume 2, and this led naturally to improvements as I was doing the keyboarding. In fact, improvements occurred almost at a steady rate for about 500 pages: Every four pages I would get another idea how to make TeX a little better. But the number of ways to improve any complicated system is endless, and it's axiomatic that you never have a system that cannot be improved. So finally, I knew that the best thing I could do would be to make no more improvements — this would be better than a system that was improving all the time.

In fact, let me explain. As I was first developing TeX at the Stanford Artificial Intelligence Laboratory, we had an operating system called WAITS, which I think is the best that the world has ever seen. Four system programmers were working full time making improvements to this operating system. And every day that operating system was getting better and better. And every day it was breaking down and unusable.

In fact I wrote the first draft of *The TEXbook* entirely during downtime. I would take my tablet of paper to the Artificial Intelligence Laboratory in the morning and I would compute as long as I could. Then the machine would crash, and I would write another chapter. Then the machine would come up and I could type a little bit and get a little more done. Then, another hang up; time to write another chapter. Our operating system was always getting better, but I couldn't get much computing done.

Then the money ran out; three of the programmers went to Lawrence Livermore Laboratory and worked on a new operating system there. We had only one man left to maintain the system, not to make any more improvements. And it was wonderful! [laughter] That year, I could be about as productive as anyone in the world.

So I knew that eventually I would have to get to the point where TEX would not anymore improve. It would be steady and reliable, and people would understand the warts it had ... the things that it doesn't do.

I still believe it's best to have a system that is not a moving target. After a certain point, we need something that is stable, not changing at all. Of course, if there's some catastrophic scenario that we don't want ever to happen, I still change TEX to avoid potential disasters. But I don't put in nice ideas any more.

Of course, there are other people working on extensions to TEX that will be useful for another generation. And they will also be well advised at a certain point to say "Now we will stop, and not change our system any more." Then there will be a chance for another group later.

Karel Horák: I'd like to ask about the idea of the italic font in mathematics. I never saw other textbooks that use different fonts for italics in text and in mathematics, so I'm asking if it's your own idea or if it comes also from these three sources?

DEK: That's right. I didn't find in any of the other books the idea of having a text italic and a math italic. I wanted the math italic to look as beautiful as possible, and I started with that. But then I found that the text italic was not as good, so I had METAFONT and it was easy to get text italic that would look better. If I made the text italic good, then the math would not position the subscripts and the superscripts as well.

It's partly because of what I explained before — TEX has only four numbers to go with every character. Printers, in fact, in the old days, had only three numbers; they didn't have the italic correction. So they couldn't do even that much automatically; the better printers adjusted mathematical spacing by hand. But italic now, the italic fonts of today by all the font designers are much better than they used to be. We've seen a great improvement in italic typography during the last ten, fifteen years. In fact, if you read older books you'll sometimes say, "How could anybody read this italic?", or "Why did they accept such peculiar spacing?", but it was based on the constraints of metal type. The whole idea of italic correction was not in any other book, but it was necessary for me to get the spacing that I wanted.

When I showed type designers mathematical formulas, they could never understand why mathematicians want italic type in their formulas. It seems you're combining a roman 2 with an italic x. And they said, "Wouldn't the positioning be so much simpler if you had a regular, non-sloped font in mathematics?" And I think it was Jan van Krimpen, who worked with a Nobel Prize physicist in the Netherlands, in Haarlem — what was his name?[8] I think he was the second person to receive the Nobel Prize in physics; he died in the 20s — anyway he and van Krimpen were going to develop a new font for mathematics in the Netherlands, and it wasn't going to have italics for mathematics. It was going to be unified between the Greek letters and other symbols that mathematicians wanted. But the project stopped because the physicist died; van Krimpen finished only the Greek, which became fairly well used.

Several other font designers have visited Stanford. When they looked at mathematics, they said, "Well, why don't you use a non-sloping font?" Hermann Zapf made a proposal to the American Mathematical Society that we would create a new typeface for mathematics which would include the Fraktur alphabet, and Greek, and script, and special characters, as well as ordinary letters. One key idea was that it would not have sloped characters, so that x would be somehow straight up and down. Then it should be easier to do the positioning, the balancing. Hermann created a series of designs, and we had a large committee of mathematicians studying the designs and commenting on them and tuning them.

This font, however, proved to be too radical a change for mathematicians. I've seen mathematicians actually writing their documents where they will write an x slanted twice as much — I mean, they make it look very italic; it looks like a mathematical

[8] It was H.A. Lorentz. See John Dreyfus, *The Work of Jan van Krimpen* (London: Sylvan Press, Museum House, 1952), page 28.

letter to them. So after 300 years of seeing italic math in print, it's something that we feel is right. There are maybe two dozen books printed, well, maybe more, maybe a hundred, printed with the AMS Euler font, but most mathematicians think it's too different.

I find now that the Euler Fraktur font is used by almost everyone. In Brno, I saw Euler Roman used as a text font for a beautiful book, a translation of Durer's *Apocalyse* in Czech. I also saw it a few days ago in some class notes. Once, when I was in Norway, I noticed that everyone's workstation was labeled with the workstation's name in AMS Euler, because people liked it. It's a beautiful font, but it hasn't been used as the typeface for mathematics in a large number of books.

Karel Horák: If there are no other questions, I would thank Professor Knuth very much for this session. [wide prolonged applause]

DEK: Thank you all for excellent questions.

Karel Horák: [Closing comments in Czech.]

Survey

Computer Typesetting or Electronic Publishing? New trends in scientific publication*

Philip Taylor

Abstract

W ciągu ostatnich 15 lat skład tekstu wspomagany komputerowo całkowicie wyeliminował stosowanie tradycyjnych technik drukarskich. Obecnie stoimy przed jeszcze bardziej radykalną rewolucją technologiczną, jaką stanowi pojawienie się wydawnictw elektronicznych (electronic publishing – w skrócie e.p.).

W przeciwieństwie do składu komputerowego i tradycyjnego, publikacje elektroniczne w ogóle nie wykorzystują papieru, nośnikem informacji staje się ekran komputera.

Potencjalne korzyści płynące z zastosowania tej nowej metody publikacji wydają się oczywiste: znaczna redukcja kosztów i niemal natychmiastowa dystrybucja do zainteresowanych odbiorców. Nie można jednak pominąć i wad, takich jak: możliwość bezprawnego kopiowania, łatwość plagiatów i nielegalnej redystrybucji.

Obie wspomniane metody publikacji różnią się w sposób zasadniczy: o ile systemy składu tekstu z góry narzucają postać końcową strony, to systemy publikacji elektronicznych stwarzają jej bardziej niezależną reprezentację, w której ostateczna forma zależy raczej od programu prezentującego dokument (tzw. viewera, czy po polsku – przeglądarki).

W niniejszej pracy omówione są najnowsze osiągnięcia zarówno w dziedzinie składu komputerowego jak i publikacji elektronicznych. Wskazuje się również na różnice tych dwóch metod rozpowszechniania.

* This paper was first presented as an invited talk at a combined meeting of the Polish Mathematical Society and EmNet Project (Euromath Trust), Torun, Poland, 1995. It has subsequently been published in English in GUST (Grupa Użytkowników Systemu TEX); Biuletyn Zeszyt 6, pp. 12-27, and in Polish translation in *Pro Dialog*, August 1996. It is republished with the permission of the author, the editors, and the commissioning organization.

A Brief History of TEX

Until about fifteen years ago, typesetting was virtually ignored by the vast majority of mathematicians, scientists, and scholars in general: manuscripts were prepared using a typewriter, the more esoteric symbols (which meant almost all symbols for mathematicians) were laboriously inserted by hand, and the whole was then simply dispatched to the publisher. Some time later galleys would be returned, emendations noted in the margin, and once again the whole would be sent to the publisher. A similar but shorter cycle was probably repeated for the page proofs, and finally the author's intentions appeared in final form in the finished book. At no point did the author and the typesetter communicate directly, and indeed the former was almost certainly virtually unaware of the latter's existence.

The typesetter, however, was only too aware of the author: mathematical copy is traditionally referred to as 'penalty copy' in the printing trade, since it is notoriously difficult to set correctly. In the time that his colleague could set ten pages of straight text, the mathematical typesetter was barely able to accomplish a single page, and even when set he knew that there was every possibility that it would have to be re-set more than once, since mathematicians are only too keen to invent new symbols of their own when no existing symbol seems entirely appropriate. And since the typesetter would never have encountered such a symbol before, he would (quite reasonably) assume that it was simply a badly drawn version of a symbol with which he *was* familiar, and substitute the latter...

Needless to say, some of the more aware authors began experimenting with computer technology as soon as it became generally accessible, and for a while the academic world seemed convinced that if it were possible to get just a couple more symbols onto the daisy-wheel of a Diablo printer, all would become possible: there were even specialist companies who would re-mould a daisy-wheel, replacing an apparently unwanted glyph with one which its owner deemed indispensable. Of course, the approach was doomed to failure: one can no more set mathematics with a fixed set of 144 glyphs than can one with a set of 128, and despite the best efforts of all concerned, the daisy-wheel printer was soon consigned to the scrap bin.

In parallel with this, the dot-matrix printer manufacturers first began to have a significant impact. With a 7×5 dot matrix, there are potentially

$$\sum_{i=0}^{35} \binom{35}{i} = 2^{35}$$

different characters (a very large number indeed!), but unfortunately a number of these are virtually indistinguishable: a single dot at co-ordinates $(4, 3)$ looks astonishingly like another single dot at at co-ordinates $(4, 4)$ to even the most astute reader (I believe that there are 33 034 338 305 *distinct* characters, as opposed to a total of 34 359 738 368 characters, where a character is regarded as *distinct* if it's not simply the result of sliding another character horizontally, vertically, or both: this figure is based on an analysis by Dr Warren Dicks of the Autonomous University of Barcelona). Furthermore, the print quality of a 7×5 dot matrix printer is so appallingly bad that no attempt should *ever* be made to set a book using one – unfortunately this well-meant advice was seldom heeded at the time.

Of course, in order to exploit these technological revolutions, suitable software had to be written, and the Unix world in particular decided to standardise on ROFF and its derivatives: NROFF, TROFF and finally DITROFF all made their mark. Unfortunately none of the ROFF derivatives ever directly supported the typesetting of mathematics, and so adjunct programs such as EQN and TBL had to be used to add mathematical functionality. There were also commercial systems, used to set publications such as the *Transactions of the American Mathematical Society*, but these were both expensive and arcane, using a rather non-mnemonic syntax to represent the possible mathematical constructions.

Fortunately (as is absolutely clear in retrospect), at least one eminent mathematician believed that something better not only could, but *should*, be created; and being not only a mathematician but a computer scientist, he decided to create it. His name was Knuth, and his creation was TEX.

Yet had it not been for a happy co-incidence, TEX might never have been born. At the time, Knuth was working on his *opus magnum*, a seven-book series entitled *The Art of Computer Programming*, and by 1977 the popularity of the early volumes of this series had proved so great that Volume 2 had already run to a second edition. Unfortunately the timing of this was such that whilst the first edition had been set using traditional hot-lead technology, the second edition was produced using one of the first phototypesetters [an aside to readers: throughout this paper I use the term *typesetter* to mean both the *person* performing the task of setting type, and the *equipment* used to achieve that end: I hope that it is always clear from the context which of these two meanings is to be inferred, since there is no other word which could easily and felicitously be substituted for either of these usages]. And whilst the new phototypesetter was more than capable *in theory* of achieving results as good as, if not better than, the traditional hot lead device used previously, the results in practice left a great deal to be desired. Knuth, as mathematician and computer scientist, was convinced that the fault lay not in the technology but in the software used to drive it, and he decided that rather than see his life's work appear in second-rate format, he would devote a short portion of his professional life to developing a suite of software which *would* exploit the full potential of the phototypesetter. Little did he know when he took this brave decision that it was to take not the anticipated one year but at least ten, although he most certainly had a demonstrably working version within his anticipated time-frame.

The first published reference to TEX is probably *Mathematical Typography*, published as report *STAN-CS-78-648* by the Computer Science Department of Stanford University; in the bibliography to this, Knuth gives the definitive reference as being *Tau Epsilon Chi, a system for technical text* which was at the time "in preparation" and is now sadly out of print. For those interested in the subject, the former paper makes fascinating reading, and the bibliography alone makes it a more than worthwhile acquisition; it was reproduced in the *Bulletin of the American Mathematical Society*, in which form it should still be available.

TEX was both typical and atypical of programs of its era: it was typical in that it was completely script-oriented, pre-dating as it did any widely-used graphical user interface; it was atypical in that it was a completely programmable *macro* programming language, in which there were no reserved words, and in which even individual characters could change their semantics on the fly. Thus a TEX document consisted both of the text to be typeset and the commands to accomplish that typesetting, and only TEX itself could unambiguously determine whether any particular element of the document was to be interpreted as 'program' or 'data'.

Despite being created primarily in order to accomplish one particular end – the typesetting of Volume 2 of *The Art of Computer Programming* – TEX rapidly took on a life of its own, and soon became the *de facto* standard for typesetting within much of Stanford University. Before long its fame had spread, and by 1980 the TEX Users Group had sprung into existence, with members of the Steering Committee drawn from far beyond the restricted domain of Stanford faculty. The American Mathematical Society were represented on that Committee, and liaison between the AMS and Knuth was very close: Knuth assigned the TEX logo to the AMS who then applied for trademark protection to prevent it being used to describe any unauthorised modification of TEX – unfortunately this application was rejected because of a prior registration of TEX (sic) by Honeywell, but despite this lack of formal registration, Knuth's high profile and high standing ensure that the TEX logo (or its non-typeset equivalent, `TeX`) is universally recognised and respected.

Within a couple of years, it became clear that the initial implementation of TEX left something to be desired, both in terms of functionality and in terms of portability, and Knuth set out to redress both by re-implementing TEX from scratch. This time he decided to eschew SAIL ('Stanford Artificial Intelligence Language') as the language of implementation, and instead to adopt the far more widely available programming language Pascal. To further increase its portability, he adopted only a strict subset of Pascal, encompassing only those features which he was confident could be found (or easily emulated) on all Pascal implementations; but he also decided to take this opportunity to render the program in a form which he termed 'literate': that is, he wanted people to be able to *read* the source of TEX in the same way that they might read a book, and to therefore be able to benefit by being exposed to a major piece of software engineering presented in a highly literate manner. Once again Knuth decided that there were no adequate tools available for this, and once again he digressed from the main project by breaking off to design and implement the concept of a WEB program, together with its two adjunct programs TANGLE and WEAVE.

A WEB program consists of a highly stylised dialect of Pascal, interspersed by lengthy comments describing the purpose and function of every element and module of the program (I suspect that Knuth would deny this, and say that a WEB program consists of a highly elaborate description of the workings of the program, interspersed by occasional fragments of Pascal which implement that functionality: and I suspect that he would almost certainly be right!). By permitting the elements of a Pascal program to be presented in arbitrary order (as opposed to the strict order of presentation required by the Pascal standard), WEB allows the programmer the opportunity to present the elements of a program in a natural and logical order, as opposed to the artificial order imposed by the Pascal design criterion of 'efficient compilability': it is then the task of TANGLE to paste together these fragments in the order required by Pascal, and the task of WEAVE to bring together both the program fragments and the comment fragments into a form which can immediately be typeset by TEX.

Thus for the first time TEX became self-referential: in order to be able to produce the Pascal code from the WEB source, one needed a working version of TANGLE; to be able to produce a literate listing of the WEB source, one needed a working copy of WEAVE; but both TANGLE and WEAVE are themselves written in WEB, so to produce a working TANGLE one needs a working TANGLE, and so *ad infinitum*. Of course 'bootstrapping' (as the technique is generally termed) is well understood in the Computer Science world, and it was estimated that the task of 'hand compiling' TANGLE from the WEB source was well within the competence of 'the average implementor': however, I remember only too clearly the trauma through which a colleague went when he attempted this bootstrapping for himself...

During the re-implementation, Knuth re-wrote almost the complete TEX program: he had learned much about its limitations during the first couple of years of use, and by 1982 a completely re-written TeX had emerged. This version of TeX (often referred to as TeX 82, to differentiate it from the earlier version which analogously became known as TeX 78) was rapidly ported to a wide range of machines, and is quite possibly the most widely available program in the world today, being available on every class of system from the smallest PC to the largest super-computer. Its almost universal acceptance as *the* standard package for computer typesetting is almost certainly the result of a large set of very positive attributes: the source of the program, and the vast majority of implementations, are available either free of charge or at a modest cost which covers no more than the media on which they are supplied; the program is virtually bug-free, a claim which Knuth backed up until very recently by offering a cheque for every bug found, the value of the cheque doubling each year since the scheme's inception (he still offers a cheque, but the value no longer doubles, since he estimated that before too long it might exceed the total Federal reserves...); the program is highly stable (there were virtually no major changes during the period 1982–90, and similarly there have been virtually no changes at all since 1990, nor will there be at any point in the future); and there are an enormous number of users throughout the world, most of whom are only too keen to pass on their expertise to any who need it, so any real problems resulting from a lack of experience with TeX can be rapidly resolved by a message to any one of a number of TeX-related mailing lists and news groups (even those without network access are not cut off, as the TUG (TeX Users Group) office offers telephone support from 03:00 in the morning until late in the evening – a service which is *not* restricted to members of TUG).

So, during the 1980's, TeX emerged as *the* standard package for computer typesetting: it was available on almost every conceivable system, device drivers were written for everything from dot-matrix printers to 2400 dpi phototypesetters (but *not* daisy wheel printers!), and an ever-increasing number of publications appeared which were either typeset using TeX, or were about TeX, or both. Many scientific journals adopted it (or one of its derivatives such as LaTeX, which may be thought of as a somewhat restrictive but more user-friendly 'front end' to TeX) as the standard format in which papers were to be prepared. Since an author could very easily proof a paper using a local implementation of TeX, and since TeX was *guaranteed* to produce identical results no matter on which system it was run, the number of iterations between author and publisher was reduced to the bare minimum, and all benefitted. And since TeX has been designed by a mathematician, and since a part of its *raison d'être* had been to allow mathematics to be typeset almost as easily as running text, its take-up by the mathematical community was if anything even faster than its take-up by the scientific and academic communities in general.

To give a simple example of why TeX is ideally suited to the typesetting of mathematics, consider the following set of equations:

$$\begin{aligned}\left(\int_{-\infty}^{\infty} e^{-x^2}\,dx\right)^2 &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-(x^2+y^2)}\,dx\,dy\\ &= \int_{0}^{2\pi}\int_{0}^{\infty} e^{-r^2} r\,dr\,d\theta\\ &= \int_{0}^{2\pi}\left(-\frac{e^{-r^2}}{2}\bigg|_{r=0}^{r=\infty}\right)d\theta\\ &= \pi. \end{aligned} \tag{11}$$

A mathematician writing this by hand would almost certainly start with the left-most element of the first line, proceed from left to right, and alternate between baseline, subscript and superscript elements as logic dictated; a pure WYSIWYG ('What you see is what you get') word processor, on the other hand, would require the typist to analyse each row of the equations into horizontal strata (thus the top stratum might contain only ∞, 2, ∞ and ∞, for example) and to enter these stratum by stratum; since, in general, WYSIWYG systems do not automatically displace preceding or following lines of text horizontally when an intervening line is shortened or lengthened, the correction of such equations is tedious and error-prone in the extreme. More recent, WYSIWYG-like, systems require a different approach in which the author has to enter the formula in the order dictated by its parse-tree; needless to say, this approach too demands more of the author than should reasonably be expected.

TeX allows the mathematician to enter the formulæ in the most natural manner, starting at the left and finishing at the right; alignment is automatically maintained if insertions or deletions are made, and even the horizontal alignment of the four primary = signs is performed automatically, virtually regardless of the length of individual left-

or right elements. To clarify this, here is the exact TeX source which was used to set the table:

```
$$
\eqalignno
  {\biggl(
      \int_{-\infty}^\infty e^{-x^2}\,dx
   \biggr)^2
  &=\int_{-\infty}^\infty
    \int_{-\infty}^\infty
         e^{-(x^2+y^2)}\,dx\,dy \cr
  &=\int_0^{2\pi}\int_0^\infty
         e^{-r^2}r\,dr\,d\theta \cr
  &=\int_0^{2\pi}
    \biggl(
      -{e^{-r^2}\over 2}
           \bigg \vert_{r=0}^{r=\infty}\,
    \biggr)
    \,d\theta \cr
  &=\pi.&(11)\cr
}
$$
```

It is worth noting that TeX completely ignores any spaces in mathematical text, since the rules for typesetting mathematics are complex, and cannot be expected to be understood by mere mathematicians! Thus the layout of the equations above is simply for the convenience of the author, and is completely ignored by TeX, which is far more concerned by special characters such as dollars, backslashes, braces, underscores, carets and ampersands. And whilst each of these characters has a distinct meaning to TeX (a dollar symbol, for example, both introduces and terminates a stretch of mathematical text), that meaning may at any time be overridden, and either assigned to a different character or, if not needed, turned off completely. So, for example, if some particular computer lacked a backslash key, it would be trivial to assign the semantics of backslash to some other key (say, yen, if a Japanese keyboard were to be used).

Furthermore, it can be seen that TeX is highly mnemonic in its choice of control sequences ('commands', preceded by a backslash); to pick out just a few examples, `\int` represents an integral sign, `\infty` an infinity, `\exp` the exp operator (representing the exponential e) and so on. Compound subscripts and superscripts are presented in logical order, rather than in order of their appearance vertically on the page; and facilities are provided for the author to give TeX hints about the logical structure of the expression, so that (for example), `\,` is used to set off differentials such as $d\theta$ from the preceding text by a little extra white space, thereby improving both the appearance and the legibility of the expression.

Thus the attraction of TeX for mathematicians is clear: a highly logical markup language, capable of being entered from any keyboard; access to a very wide range of mathematical symbols; professional standards of layout; widespread acceptability by journals; and the ability to proof on anything from a dot-matrix printer to a 600 dpi laser printer. Add to this the now universal ability to preview the document on the computer screen (something the early advocates of TeX could only dream of), and it is hard to explain why any mathematician with access to a computer would *not* typeset his papers using TeX!

However, use of TeX is restricted neither to mathematicians nor to North Americans, and at the TeX User Group conference in 1989, an influential and voluble group of European TeX users ganged up on Knuth and succeeded in convincing him that, despite his assertion on the previous day of the conference that the development of TeX was finished, there were features missing from the current implementation which made TeX entirely useless to the majority of the world, since whilst it behaved perfectly in unaccented languages, it was grossly deficient for typesetting any language which made more than occasional use of diacritics. And Knuth, recognising the validity of this argument, agreed that something had to be done.

The result of all this was TeX 3: TeX 82 became known simply as TeX 2, and TeX 3 became the One True TeX. In practice, this just didn't happen: those who had no need for the extended diacritic support offered by TeX 3 simply continued to use TeX 2, and for quite a while TeX macro writers had to write very defensive code which first checked the environment before making any assumptions about (for example) the number of distinct characters with which TeX could internally deal (this was 128 prior to TeX 3, and 256 thereafter). With the release of TeX 3, Knuth made it *absolutely* clear that this really did represent the end of the TeX evolutionary line: he had better things to do with his time, and TeX was now frozen (modulo any essential bug fixes, which he undertook to continue to make if and only if it could be shewn that their fixing was essential). Furthermore he made it equally plain that TeX could *not* be further evolved by anyone else: he wished to leave for his children, and for his children's children, and for all perpetuity, TeX as his creation, and not as his-creation-as-modified-by-someone-else.

In general, the TEX world took this in good part: Knuth is enormously highly respected by those who use TEX, and there were very few who advocated ignoring his wishes and who were prepared to suggest modifying TEX. But there were also a quite significant number of TEX users, among them the present author, who felt that if TEX did *not* evolve, then it would simply die. Not because of any fundamental deficiencies in TEX – it is generally accepted that there are very few – but because the world had moved on since 1978, and whilst a script-driven language might have been state-of-the-art then, it most certainly was not state-of-the-art now. Furthermore, despite increasing the number of distinct internal characters from 128 to 256, Knuth had done little if anything to enhance TEX to deal with Asian languages, in which the number of distinct characters may be measured in thousands if not in tens of thousands. And finally, there were those who felt that there were *some* areas in which a very significant increase in functionality could be gained (particularly from the perspective of the macro programmer, who is also known as a 'format writer' when the suite of macros provides a complete functional system in its own right) with relatively little investment in terms of modifying TEX.

The implementation of these ideas probably represents the leading edge of TEX technology today: companies such as Blue Sky have produced instantaneous/incremental TEX interpreters, which are capable of displaying the effects of a change to the source code of a TEX document in real time; Advent Publishing have produced 3B2, which allows both a graphical and a textual specification of a layout, automatically updating one to reflect changes in the other; John Plaice and Yannis Haralambous have implemented a 64-bit version of TEX which uses Unicode internally; and the group with which I am most closely associated (the NTS group, where NTS stands for 'New Typesetting System') have produced a completely compatible successor to TEX, called e-TEX, which adds functionality without compromising compatibility (the NTS group also wish to re-implement TEX from scratch, using a modern rapid-prototyping language such as Prolog or CLOS, the idea being to allow rapid experimentation with alternative typesetting algorithms or paradigms). Whether or not any of these ideas will catch on remains to be seen, although among Apple Macintosh *aficionados* Classic Textures (the Blue Sky product referred to above) is already highly thought of. One fundamental question is that of stability: since one of the great strengths of TEX is its stability, how will the world feel about systems which encompass TEX but which are specifically intended to remain evolutionary and responsive, rather than fossilised and unyielding? Only time will tell.

What is perhaps worth noting is that all of these projects have ensured that Knuth's wishes are honoured not only in the letter but in the spirit: none seeks to call itself TEX (indeed, that of John Plaice and Yannis Haralambous is called Omega, which could never be confused with TEX), yet all acknowledge the debt which they owe to Knuth and to TEX: without them, none of these other projects would ever have seen the light of day.

Parallel Developments

Of course, while TEX was evolving, the rest of the world did not stand still: computer science continued to develop, and computer networking moved from the laboratory to the military and the Universities and ultimately to the whole world. Line-oriented editors fell by the wayside, and were replaced by full-screen editors (except in the rather time-warped world of MS/DOS, which continued to offer only EDLIN until comparatively recently). Script-oriented markup languages such as the ROFF family referred to earlier were challenged by increasingly sophisticated word-processors, and WYSIWYG ('What You See is What You Get'), GUI ('Graphical User Interface'), and WIMP ('Windows, Icons, Menus and Pull-down lists') became the order of the day.

At about the same time that Knuth was starting work on TEX, John Warnock and Martin Newell re-implemented an earlier language ('the Design System') as 'JaM' ('John and Martin'!) whilst working at Xerox PARC, and from this cloistered beginning ultimately emerged both the Interpress (Xerox printing protocol) and PostScript languages. Whilst Interpress remained relatively unfamiliar, Adobe PostScript took the computing world by storm: for the first time there was a *de facto* page description language, which allowed complex pages to be described algorithmically (and thus very efficiently). Although Hewlett Packard's Printer Control Language (PCL) continued (and continues) to be both widely supported and widely emulated, PostScript rapidly established itself as *the* standard for high-level printers (by which I mean laser printers and better), and fairly quickly printer manufacturers sought to provide either PostScript interpreters or PostScript emulators for their high-end products. Unfortunately (for the emulator writers) PostScript is a complex language, and many of the earlier emulations were deficient in one or more respects; Adobe, of course, as designers of the language had far fewer problems in this respect, although even they released improved versions of their interpreter as time went by.

For a long while parts of PostScript remained a closely guarded secret: the mysterious `eexec` operator was undocumented, and whilst the PostScript manual gave information on the format of so-called 'Type 3' fonts, the equally mysterious 'Type 1' fonts remained undocumented. Of course, reverse engineering is a well-understood tool, and finally the barriers were broken: descriptions of `eexec` started to appear in the press, and ultimately Adobe themselves relented and gave full documentation of both `eexec` and their Type 1 fonts.

Before long, Type 1 fonts established themselves as as much a standard for fonts as PostScript was a standard for page-description languages; companies such as Corel started to release Type 1 fonts of their own, closely modelled on industry-standard fonts but sufficiently different (at least in name) to avoid accusations of font piracy (although this latter problem continues to worry top font designers such as Hermann Zapf to this day). All the major font foundries started to offer their fonts in Type 1 format, and many gave a commitment to have *all* of their fonts in Type 1 format within the foreseeable future. The so-called 'font magic' which enabled early Adobe fonts to render well even on relatively low resolution devices such as 300 dpi laser printers was renamed 'font hinting', and this too was eventually documented by Adobe. New features continued to be added to the PostScript language, and in 1990, Adobe announced a completely new version of the PostScript language, 'PostScript Level 2'. This new version unified all previous additions to the language, and added many new features as well, such as the ability to have compact (binary) representations of a PostScript document as well as the earlier (ASCII) representation; new colour models were introduced, and support was added for composite fonts.

PostScript was originally conceived as an embedded language for printers, but before long it became clear that a version of PostScript which could drive a computer screen would be extremely useful. Adobe created their own version of this called 'Display PostScript', but in the meantime L. Peter Deutsch had started work on a PostScript interpreter of his own, called 'Ghostscript', and fundamental to its functionality was the ability to drive the screen of any computer on which it was used (it also contained drivers for a wide-range of non-PostScript printers, as well as pseudo-drivers for some of the more popular graphics interchange formats). During 1995 Peter finally announced Ghostscript version 3, which provided almost a complete Level 2 emulation, and whilst the official Adobe interpreter remained a licensed (and relatively expensive) product, Ghostscript was and remains free of charge to those who do not use it for profit-making purposes; a very significant debt of gratitude is owed by the computer world to L. Peter Deutsch, both to his skill in writing Ghostscript and to his generosity in making it so freely available, and also to the many individuals who have donated their own drivers and/or enhancements to the Ghostscript project (PS-View, from Bogusław Jackowski and Piotr Pianowski warrants special mention).

From the ARPAnet to the Web

A few years before Knuth started work on TEX, the American military as personified by [D]ARPA (the [Defence] Advanced Research Projects Agency), had initiated a pilot project to link computers over very wide distances; whilst local computer links were not uncommon, links across thousands of miles were unheard of, but [D]ARPA realised the potential military importance of such links and therefore initiated a whole series of research projects aimed at making this a reality. Whilst these projects initially started in isolation, as soon as the pilot network was available the project gained a momentum – indeed, a very existence – of its own, and the whole development strategy henceforth was established by discussion *across*, as well as *about*, the network. This network, known as the ARPAnet for obvious reasons, evolved a mechanism for distributed discussion and voting known as the 'Request for Comments' ('RFC'), and any new idea for anything from a protocol to a picnic was likely to find itself the subject of an RFC. From these RFCs emerged some of the most important *de facto* standards on which we still to this day: TCP ('Transmission Control Protocol'), IP ('Internet Protocol'), SMTP ('Simple Mail Transfer Protocol') and so on were all enshrined in the published versions of the RFCs, and each was allocated a unique number: electronic mail, for example, was addressed by and specified in, RFC 822.

Although the American military launched the networking initiative, it was the American universities which were actually the primary contributors to its success, and once the network was well established it ceased to be 'the ARPAnet' and became instead 'the Internet', the name by which it is still known today. Strictly speaking, the Internet is not a network *per se*, but a network-of-networks; however, the distinction is of little significance, and most now regard the Internet simply as *the* international computer network. From its military origins, where permission-to-connect almost required a personal interview with a five-star general, the Internet has now become the network to which even the most humble private citizen may aspire to gain access: Internet service providers have sprung up across much of the Western world, and connecting to the Internet today requires little more formality than a letter (and a fairly modest cheque!) to an Internet service provider, together with the purchase of a equally modest personal computer and a modem: at the time of writing, there are Internet connections from something like 150 countries throughout the world (the number of actual Internet nodes is *far* harder to gauge, but it is already estimated to lie between five and ten *million*).

Initially the protocols used, and services provided, on the Internet were very primitive: FTP ('File Transfer Protocol'), TELNET (remote terminal access), PING (check if a remote node is alive), and SMTP were probably the most common, with FINGER (check if a remote user is logged in) coming not far behind. But whilst the end-user protocols were fairly simple, the underlying mechanisms were not, and the DNS ('Domain Name Service') provided a quite sophisticated mechanism for a distributed node lookup protocol. As more experience was gained, the range of protocols and services increased, and things such as Usenet News (a distributed bulletin board) and NFS ('Network File System', providing remote access to a complete file system) were added. Then the information explosion really took off, and tools for information retrieval and display began to proliferate: GOPHER and WAIS ('Go for', and 'Wide Area Information Service, respectively) were early candidates, shortly followed by WWW (the 'World Wide Web', now usually shortened to 'the Web'). It should be emphasised that there is *no* connection between WEB programs and the World Wide Web; within this document at least, the former is consistently shewn in upper case, whilst the latter is consistently shewn in mixed case.

With the advent of the Web came one major breakthrough: whereas previously each protocol had specified its own unique method of identifying a remote resource, WWW brought with it the concept of the URL (the 'Universal Resource Locator'), so that from within a single program (the 'browser'), almost *any* Internet resource could be specified. For example, a remote FTP resource would commence `ftp://`, a remote GOPHER resource would be `gopher://`, and the Web's native resource, HTTP ('HyperText Transfer Protocol') would commence `http://` (aware readers may appreciate that this is a slight over-simplification, but the deviations from reality are essentially very small).

With the Web and URLs came unified browsers: tools such as MOSAIC which allowed access to a wide range of Internet resources from a single graphical front end. Even if a resource had no unique URL, it was still possible to associate with it an adjunct renderer which would display it correctly: thus, for example, although there is no unique URL for an MPEG file ('Motion Picture Expert Group': a compact standard for encoding and storing full-motion video), a correctly configured browser such as MOSAIC could identify an MPEG resource from its file type (the portion of the file name which follows the period), and on down-loading such a resource would then spawn off an instantiation of the appropriate renderer, so that down-loading and viewing were essentially indivisible entities.

Native-mode documents for access over the Web are coded in a language called HTML ('HyperText Markup Language'): this is a direct derivative of an earlier (but still current) specification for a generalised markup language called SGML ('Standard Generalised Markup Language'), and a conformant HTML document is also normally a conformant SGML document, although as is often the case the converse does not necessarily obtain. Both HTML and (typical but not all) SGML documents are characterised by the frequent occurrence of tags which are enclosed in angle-brackets: they therefore resemble the 'metalinguistic variables' of a much earlier standard – the BNF (or Backus Naur/Normal Form) of the original Algol-60 report – although they do not perform the same function. In an HTML document, each tag specifies the *nature* of the entity to which it refers: whilst this can be augmented by a specification of how the entity should appear, in the purest form only the nature of the entity is specified, and it is left to the browser to determine how the entity should appear. This represents a very significant philosophical breakthrough: no longer need a document be formatted by its author, the reader then requiring the technology to resolve that format; instead, using HTML, a document is simply tagged using high-level content-oriented markup, and the reader may then display that document using whatever technology is available. For example, most Internet systems are capable of running a browser called LYNX: this is a purely textual browser, and so it makes no attempt to represent subtleties of the document; it simply takes advantage of whatever text-mode functionality is available to it (for example, emboldening or underlining) to display the document to the best of its ability. Images which would normally require a graphics mode browser to resolve are simply displayed as the word IMAGE. On more sophisticated systems, graphics mode browsers such as MOSAIC (previously referred to), or the now ubiquitous NETSCAPE, can be used: these will exploit the graphics capability of their platforms to the full, and are capable of displaying full-colour, and even motion-picture-insertions, either using inherent functionality or through the medium of adjunct software which has been used to customise the browser.

But an HTML document is far more than just a passive entity: elements of it can be designated as 'hot spots', and if a hot spot is selected (using the mouse on a graphical system, or the tab and/or cursor keys on a line-mode system), a further document may be downloaded and displayed entirely automatically: the document containing the hot spot and the document referred to by the hot spot do not need to originate from, or be stored at, the same site: a document stored at (say) the University of Western Ontario can reference, through a hot spot, another document stored at (say) the University of Queensland. Furthermore, although the discussion so far has been concerned with 'documents', hot spots can in fact be linked to *any* Internet resource, provided only that the resource is specifiable via a URL. Thus a document which was fetched using HTTP can reference another document that can be fetched using only GOPHER; that document could specify a third document which is accessible only via FTP: that could contain a reference to a Usenet Newsgroup; and so on.

Yet even this does not represent the limits of a Web document: such documents can also be *forms*, with fields which must be completed by the reader; when the form is completed, a further hot spot can transfer it to a remote site, where it will be interpreted and acted upon. In this way, the original ARCHIE protocol (ARCHIE is an Internet tool for locating files available via anonymous FTP) has been extended from its traditional usage in which it is launched from a command line invocation specifying the file of interest and some constraints on the manner of search; with the HTML version (a.k.a. 'Archiplex'), the Archie user invokes his preferred Web browser to fetch an Archie form from a convenient server; he then completes the form, and uses a hot spot to return it to the Archie server; the latter then locates the file of interest, and returns a list of places at which it can be found, where the list of places is possibly constrained by options selected on the form by the user (for example, he may say that he's only interested in copies of the file that can be found within his own domain). The list of hits is then displayed by the browser, and once again using the mouse, tab key and/or cursor keys, the user selects one instance of the file of interest; the file has associated with it a hot spot, so the instant he selects the file from the list, a request is issued to retrieve the file; assuming that there are no hiccups, the file is fetched entirely automatically and displayed on the originating screen. If the file is not displayable for some reason (perhaps it is an executable image, or something else for which the concept of 'display' is ill-defined), the browser will inform the user and ask if he wishes to save it to a local disk.

Whilst previous introductions such as GOPHER and WAIS had a relatively modest impact on overall use of the Internet, which in general continued to be used mainly by academics and hackers, the introduction of HTML and the concept of the Web brought about a revolution in Internet usage: commercial companies clamoured to get on-line, governments put up their own Web pages, and every man and his dog suddenly appeared to be beset by the need to create a unique and highly individualistic 'home page' (Web documents are often regarded as being divided into pages, by analogy with a paper document, and a 'home page' is a (usually brief) document giving information about the individual who owns it; many institutions provide facilities whereby each user can create his or her own home page without formal approval). The reason for this sudden change in usage patterns is not hard to explain: whereas the more traditional Internet tools such as FTP required a modicum of expertise before they could be successfully used, the various Web browsers were intentionally designed to be 'user friendly' from the very outset, and this user friendliness together with the ability to seamlessly down-load and display documents in an astonishing variety of formats without any expertise whatsoever resulted in an unprecedented rate of take-up and an almost universal acceptance. There can be little doubt that the current near-exponential rise in Internet registrations and usage results almost entirely from the concept and ease of use of the Web.

The Web and Publishing: Unlikely Bedfellows?

Whilst it might initially seem that the two themes of this paper represent quite distinct branches of the evolutionary tree, it did not take long for those involved in publishing to realise the untapped potential of the Web: even prior to the establishment of the Web there had been some experimental use of the Internet for electronic publishing. In particular, the so-called e-journal *EJournal* (subtitled Electronic Journal for Humanists) was a direct electronic analogue of a more traditional journal, containing scholarly essays as well as shorter "letters to the editor". However, *EJournal* uses simply ASCII text as the communications medium, whilst the Web potentially allows even greater richness of medium than any traditionally produced journal, since unlike a paper journal a Web journal could contain not only text and static graphics but full motion video and sound as well.

In July 1994 the American Mathematical Society launched a project entitled "New Media", chaired by Frank Quinn of Virginia Tech., to investigate the possibility of developing a multimedium, interactive, hypertextual version of TEX: the brief of the sub-committee established to investigate this was to "co[-]ordinate the development of a technical authoring tool which will integrate text, graphics video, non[-]linear documents, hypertext links, and interactive computation. [The] tool should share the characteristics of the TEX typesetting system which have made it so remarkably useful: open file structures, open and portable source code, a stable standard core, and an uncompromising commitment to the highest quality. [It] is expected to be an extension of TEX.".

The rationale behind this proposal is also interesting: "Educational communities need interactive texts. Technical communities need hypertext and non[-]linear document types to tie together complex or cumulative efforts. Users of computation need better ways to document and illustrate programs. All these capabilities are available now in primitive forms, and authors are pushing ahead. Some are writing interactive texts using computer mathematics programs. Others are experimenting with hypertext extensions of TEX, [WWW] documents, etc. Commercial publishers are experimenting with hypertext, CD ROM publication, and linked databases. In a few years we can expect powerful tools for constructing interactive multimedia documents. But they may be 'accessible' in the same sense that typesetting was accessible before TEX: publishers will use expensive proprietary systems with closed file formats, and authors will use a multitude of free or inexpensive systems which require professional resetting to get professional results. Our experience with TEX shows that this fragmentation is undesirable and unnecessary. The HyperMath Project is being organized to avoid it.".

The introductory document for the project then went on to explain: "The HyperMath Project is primarily a framework to co[-]ordinate work already in progress. Several groups have already incorporated simple hypertext links into versions of TEX. The NTS (New Typesetting System) group is exploring improvements to traditional paper-and-ink typesetting. Most implementations of TEX have methods for incorporating graphic material, and there are publicly available packages which do this. The 'Interactive Mathematics Text' project and many groups in the calculus reform movement are using Mathematica, Maple, MatLab, and other programs to write interactive texts. These and similar initiatives can be brought together in the development of a general tool. But the opportunity is limited. As development proceeds, the costs of switching to a common standard increase, and the benefits become less obvious. We should not let this opportunity pass. The Project will sponsor working groups and conferences. The working groups will develop standards and goals, and work on prototypes. Communications among working groups will be maintained to ensure coherence and uniformity. And contacts will be developed between developers and end-users to ensure that real needs are being addressed. Working groups are planned in the following areas: traditional text; non[-]linear documents (including hypertext); inclusions (graphics, video, and sound); interactivity; and users. The first HyperMath conference is planned for the San Francisco area in conjunction with the combined math society meeting early in January, 1995, contingent on funding. The 'New Media' subcommittee of the Publications Policy Committee of the American Mathematical Society will serve as the advisory board for the HyperMath Project."

This was heady stuff: sadly by September of the same year it had been abandoned as being "too ambitious", and replaced by a more incremental approach, now entitled "non-traditional forms of publication". Whereas the earlier project had been predicated on the development and adoption of enhanced TEX, the new project proposed that "the AMS should adopt the Adobe portable document format 2.0 as the standard (output) format for electronic publication of documents". It then went on to explain that "This does not mean giving up TEX, nor does it solve all TEX problems. It is a proposed replacement for DVI as output, not the use of TEX source in authoring." What did this mean?

The first thing to realise is that by now all three threads of this paper have finally come together: TEX, Adobe, and the Web. Whilst Adobe had been very successful in developing PostScript as a page-description language, and marketing embedded PostScript interpreters for incorporation in laser printers and the like, it had been somewhat less successful in ensuring that Display PostScript became established as another *de facto* standard. Indeed, with the advent of Ghostscript, a significant future for Display PostScript was by no means certain, and the proliferation of Web-based browsers (MOSAIC, NETSCAPE and the like) which could slave Ghostscript was a further challenge to Adobe's position in the marketplace. Unlike Adobe's PostScript interpreters and Display PostScript systems, Web browsers were (and remain) freely available: that is, they are literally available *free of charge*, even when they are as sophisticated as NETSCAPE (which is developed and supplied by a commercial organisation). Whilst Adobe could maintain its niche as a supplier of PostScript interpreters, it was becoming clear this was a limited, and possibly even diminishing, market: if Web-based publication rather than paper-based publication ever became the norm, the rôle of PostScript printers and image setters might be seriously challenged as more and more documents were read from a computer screen rather than from paper. It was therefore no great surprise when Adobe finally announced (there had been clues previously, such as their work on so-called 'multiple master fonts' and Carousel) their alternative to a Web browser as a universal document rendering engine: Adobe Acrobat. Just as with the Web browsers, Adobe Acrobat is available free of charge (indeed, they send complete CD ROMs containing a full multilingual installation kit at the slightest provocation). And, rather like NETSCAPE, who seek to recover the costs of developing their browser by selling their Web server, Adobe will endeavour to recover the cost of the development and production of their Acrobat reader by selling the technology which is required to produce an Acrobat document in the first place.

And what is an Acrobat document? The very same thing that the AMS are investigating as a possible standard for their mathematical publications: something written in Adobe Portable Document Format (PDF). And although in theory one can develop applications of one's own which will write PDF, in practice many will elect simply to purchase Adobe Acrobat (which acts as a pseudo-printer driver for MicroSoft Windows, Apple Macintosh or Unix systems), or Adobe Acrobat Pro[fessional] (which also includes Adobe's "Distiller" to convert PostScript documents into PDF documents), or Adobe Acrobat Capture (which uses the TWAIN protocol for scanners to generate PDF documents directly from a scanner). Thus despite their apparent generosity in giving away Acrobat free of charge, Adobe are (of course) really seeking to increase their market share by encouraging the purchase of other Adobe products.

HTML or PDF?

With HTML and PDF emerging as *the* two portable hypertext exchange standards, organisations (and, to a lesser extent, individuals) are going to be forced to make a choice. It may well be that for some applications the choice will be clear-cut, but for others there may seem little to choose between the two. It is therefore worth exploring the basic differences between HTML and PDF, in order to better allow an informed choice to be made.

HTML, being SGML, is essentially a very high level, content-oriented, markup language: its *forte* is the specification of the content of a document, and its weakness is the relatively little control that an HTML author has over the final appearance of the document. Because it is so high level, it is not possible using the current received wisdom of computer science to automatically generate HTML from an arbitrary document: if a word-processor, for example, is used to prepare a document, and if that document has been created *ex nihilo* without consideration for its logical structure, so that only the final appearance of the document has been considered, then it is almost certainly impossible to reverse-engineer the document to ascertain its logical structure: in these circumstances HTML would have little option but to represent it as an indivisible bit-map, thereby effectively wasting almost all of HTML's functionality. Despite this restriction, HTML has much to offer, for two main reasons: (1) the tools needed to generate it are already in the public domain, although the interface between those tools and pre-existing software such as word-processors is unlikely to be available (it is far more likely that word-processor packages will start to be shipped with HTML drivers, but their use may require a major re-think by the user concerning the the way in which a document is created); and (2) high-level markup is increasingly recognised as being *the* way in which to mark up a document: as experience of the use of typesetting systems such as TEX/LATEX is gained, it becomes ever more clear that low-level, form-oriented, markup is simply a dead-end and should rapidly be expunged from the practices of responsible authors.

PDF, on the other hand, consists essentially of a strict subset of PostScript with the added functionality of hypertext: PDF documents can reference other PDF documents using hot spots, rather like HTML. According to the PDF blurb (this paper is written before my copy of Adobe Acrobat Pro has arrived, so what follows must be taken as speculative at the moment):

- Create electronic documents as easily as printing from existing applications with PDF Writer.
- Protect files with passwords; control access, printing, changing the document, adding and changing notes, copying text and graphics.
- Find exactly what is needed across multiple PDF files by searching on keywords, author, title, subject synonyms, etc.
- Re-use information easily by extracting, copying, reordering and replacing pages among PDF files – with bookmarks, links and notes preserved.
- Create custom views into information.
- Add value, set priorities and maintain a dynamic information network with links, bookmarks, notes and connections to external applications and documents.
- Take advantage of third-party plug-ins to add new features to Acrobat.
- Integrate Acrobat with desktop applications with Acrobat's support for OLE automation, Notes F/X, AppleEvents, and more.

Although perhaps it is too soon to compare HTML and PDF with any real accuracy, it would seem that at the moment they are intended for, and best suited for, rather different applications: HTML documents can either be created *ex nihilo* (for those who have no better way, simply cloning and modifying an existing HTML document is an excellent way to get started), or by using an HTML editor (of which there are already several in the public domain), or by using a package or packages (for example, a suitable word-processor) for which an HTML driver already exists. PDF documents may be created using one of the Adobe tools – Acrobat Writer, Distiller or Capture – depending on whether or not the source documents pre-exist. As HTML allows only a degree of control in the formatting and placement of entities, it is not really suitable for the presentation of anything other than simple mathematics, although HTML 3 demonstrates that the designers of HTML are aware of many of the limitations of the previous version, and are working towards a specification which may ultimately allow arbitrarily complex formulæ to be displayed. [A comment in the HTML 3 discussion document reads "Including support for equations and formulæ in HTML 3 adds relatively little complexity to a browser. The proposed format is strongly influenced by TEX."]. Of course, since HTML allows reference to be made to non-HTML documents, many of these difficulties can be overcome: an HTML browser such as NETSCAPE can be configured to invoke an external renderer if no internal renderer is suitable for the entity referenced, and in that way both DVI (from TEX) and PostScript documents can be referenced from, and displayed from within, an HTML document. Since both DVI and PostScript are equally suited to the accurate representation of mathematical material, there is no real reason why a mathematical document should not be displayed from within an HTML framework by an HTML browser configured with a suitable external renderer. PDF, on the other hand, has no need for external renderers, since its native mode of operation uses a strict subset of PostScript; indeed, Adobe Acrobat is intended to be configurable *as* an external renderer for HTML browsers such as NETSCAPE! By using Adobe's 'multiple master' font technology, Acrobat can generate a reasonable substitute for any font specified in a PDF document, even if that font is not available within the system on which the document is being displayed. It is by no means unlikely that before very long a DVI-to-PDF driver will emerge, and in the true tradition of TEX it is also extremely likely that such a driver will be placed in the public domain; DVI-to-HTML is an unlikely eventuality, however, since by the time a TEX document has been converted into DVI, too much information has been lost to allow the high-level structure of the document to be re-created.

On the other hand, we can certainly envision a format being created for TEX which embeds `\specials` in the DVI file to convey information about the high-level structure of the source document: since the user interface would be completely unaffected by the presence of these specials, such a format could appear to the user exactly like any of the present formats or format variants which support appropriate high-level markup (AMS-TEX, LATEX, AMS-LATEX, LAMS-TEX, etc.). Such specials could then be directly mapped into HTML constructs, and thus a TEX-to-HTML route is neither impossible nor unlikely; indeed, it is surprising that no such extended format has yet been announced (at least, to my knowledge). Finally it is worth remembering that HTML is essentially a *distributed* markup language; it is primarily intended for documents which need to reference other documents which may be anywhere on the Internet; PDF, on the other hand, is essentially Internet-unaware, and whilst it can transparently reference other documents that are visible through (say) NFS (or, using ALEX, anonymous FTP), it makes no assumptions that documents might be anywhere other than the local filestore or on a Microsoft-compatible network. [This last sentence is somewhat tentative: in the absence of the definitive PDF specification, it is somewhat difficult to accurately interpret the claim that Adobe Acrobat allows one to "Add value, set priorities and maintain a dynamic information network with links, bookmarks, notes and connections to external applications and documents', but I suspect that the 'dynamic information network' does not allow the transparent referencing of arbitrary Internet resources, although this may well come in time.]

Computer Typesetting or Electronic Publishing: Pros and Cons

Computer typeset material, particularly that typeset using TEX or a functionally equivalent system, represents the finest in typeset quality that can be easily accomplished today; where such computer typesetting software is unavailable, comparable results can only be accomplished by a skilled professional using either old-fashioned technology (e.g. hot lead) or a modern but proprietary system. This is not to say that the use of a system such as TEX *guarantees* professional quality results: there are far too many counter-examples in existence which demonstrate that in completely unskilled hands, TEX and comparable systems are capable of generating absolutely appalling results. None the less, in reasonably skilled hands, and/or using a format package which prevents the author from making design decisions, TEX is capable of generating results which meet the highest professional standards, particularly in the field of mathematics where TEX essentially performs as an 'expert system'. The disadvantage of such a system is that in its intermediate form (DVI), a TEX document is not fully portable: a DVI file contains references to, but no instances of, fonts; at the point where the DVI file is converted into its final form (usually paper, but on-screen preview is now also ubiquitous), the same fonts which were used to create the document must be available in order to render it correctly; in their absence, only a poor approximation of the intended document is possible. [It is worth noting that the creator and the viewer/printer of a TEX document need a common set of fonts, but each needs a quite different representation of those fonts: the creator needs only the *font metrics*, which specify the height, depth and width of each glyph, and kerning and ligaturing information for the glyph set; the viewer/printer of the document can normally get by without the metrics, but instead needs the actual glyph set, either as bitmaps or as outlines.]

Electronic publishing, on the other hand, and particularly e.p. accomplished through the medium of HTML, does not place any emphasis on the quality of the end product: indeed, HTML voluntarily cedes control over the appearance of the final document to the browser used to render it, although there are some placement options which allow the author a little control over the final appearance (and there are considerably more such options in HTML 3). Within an HTML document there is no font information *per se* (again, this is true only of current HTML: HTML 3 adds the concept of style sheets, which will "[...] eventually lead to smart layout under the author's control, with rich magazine style layouts for full screen viewing, switching to simpler layouts when the window is shrunk"); instead the document consists of a set of high-level markup tags, which are mapped by the browser to a particular font or font variant. Whereas a DVI file is a monolithic entity, and makes no reference to any external resources other than fonts, an HTML file is frequently little more than a container for other HTML files, and may make reference to an extremely wide range of resources (further HTML files, images, AFS files, Usenet newsgroups, e-mail addresses, FTP-accessible files, etc.) which may be anywhere on the network, and which may themselves contain further references and so *ad infinitum*.

PDF is essentially a reasonable compromise between the two: the creator of the document specifies its appearance, and the PDF reader then displays that document to the best of its ability: if the fonts needed to display it properly are embedded, or if they are resident on the target system, then the document will be displayed exactly as the author intended; if the fonts are not accessible, then Adobe's proprietary 'multiple master' technology will be used to interpolate a substitute for the missing font(s) which allows the original line-breaks, leading, etc., to be retained. A PDF document may reference further PDF documents, but these are assumed to be available on the local filestore; there is no apparent support for the automated fetching of remote Internet documents, although the absence of the PDF documentation at the time of writing makes analysis of this feature rather more of an informed guess than a definitive statement.

All three formats discussed allow searching to be conducted; within a DVI file there is no intrinsic support for indexing, but it would not be at all difficult for a DVI viewer to create a dynamic index to the document being viewed. Both HTML and PDF allow fully indexed documents to be referenced.

Publication in the Twenty-First Century

It is no longer possible to assume, as countless previous generations of scientists have done, that "publication" involves printing on sheets of paper which are ultimately distributed as a part of a journal or as a book: increasingly both economic and environmental pressures will dictate that only essential information be committed to paper, and anything even slightly ephemeral will be restricted to electronic distribution. At least two *de facto* standards have already emerged for electronic publication: HTML, which originates in the distributed and anarchic world of the Internet; and PDF, which originates in the commercial world. At the time of writing, HTML is the better established, and two freeware browsers are widely used (MOSAIC and NETSCAPE), with a third (ARENA), being developed specifically to support HTML 3; for PDF, there is only one reader currently available (Acrobat), and that too is classified as freeware. HTML devolves to the browser most of the decisions concerning the actual appearance of a document; PDF allows the author to make most of those decisions, but reserves the right to substitute interpolated fonts if the genuine article are not available at the point of rendering. HTML is essentially a distributed protocol, and will allow bibliographies to reference cited texts no matter where they are in the world (so long as they are on-line), thereby adding truly incalculable value to the bibliography of a document; PDF, it would appear, is essentially a local protocol; whilst bibliographies could still cite full-text sources, those sources would need to be available to the system on which the bibliography is being read.

Many issues remain to be resolved before the world can truly move to electronic publishing as the mainstream form: Internet access in every home, office, library, vehicle, and restaurant will be just a start. There remains the very contentious issue of copyright: whilst there are usually economic costs associated with the photocopying of a printed document, the costs of copying an electronic document are virtually nil, and therefore the enforcement of copyright for electronic publications is a major concern. It is highly likely that some form of encryption and licensing will emerge to prevent the unauthorised copying and/or re-distribution of electronic texts. From the psychological and physiological point of view, displays will need to become significantly better (in many senses: weight, resolution, glare, portability, etc.) before the electronic book completely replaces the printed equivalent: few of us, going on holiday today, would choose to take a notebook computer with a CD ROM containing the complete works of Shakespeare in preference to a couple of (disposable) paperbacks... Although originally developed as front ends for the generation of printed material, typesetting systems such as TEX will almost certainly have a major rôle to play as front ends for electronic publishing, since (for example) the linear representation of mathematical formulæ is equally convenient and applicable whether one's mathematics are eventually to appear on paper or on a computer screen. Within ten years, HTML and PDF will appear *passé*: new standards emerge faster than most of us can keep up, and today's technology is tomorrow's door-prop. But the future of the book (or even the newspaper) as the normal means of communication is surely as doomed as that of the petrol-driven car as the normal means of conveyance; the first to guess exactly what form the replacement will take may become as rich as today's newspaper magnates and publishing house principals; or perhaps the converse will occur, and the Internet will finally cause the collapse of the publishing empires, as academics and authors suddenly realise that they are no longer beholden to the few. Self-publishing may become the norm, or peer review may take on an entirely different form; perhaps a two-tier hierarchy of electronic publishing will emerge, with unrefereed papers being available via each academic's home page whilst those that have survived the refereeing process will be available from prestigious and highly accredited archives. What is certain is that almost all of the readers of this paper will find out for themselves what the future holds, at least as regards computer typesetting and electronic publishing: the future is just around the corner, and approaching at an ever increasing speed.

Addendum

Within the last forty-eight hours, I have learned of two new facts which significantly impinge on the material above: NETSCAPE have licensed the use of PDF technology from Adobe, which will allow them to incorporate a PDF renderer within their HTML browser, and Michel Goossens & Sebastian Rahtz have demonstrated the feasibility of using Adobe's 'multiple master' fonts from with TEX; further details of the latter, including very useful information on multiple master fonts, are given in the *Baskerville* issue cited in the Bibliography.

Acknowledgements.

I would like to thank Professor Adam Jakubowski and Jerzy Ludwichowski for making it possible for me to present this paper, to Elżbieta Kuczyńska and Bogumiła Rykaczewska-Wiorogórska (University of Warszawa) for kindly providing two alternative translations of the abstract into Polish, and to Professors Adam Jakubowski and Andrzej Jonscher for providing reverse translations into English to enable me to check the accuracy of the initial translation. I would like to thank Dr Warren Dicks of the Autonomous University of Barcelona for his analysis of the problem of the number of visually distinguishable configurations of an $m \times n$ matrix, as used to establish the number of distinct characters which can be generated by a 7×5 dot matrix printer. I would like to thank Dr Frank Quinn of Virginia Tech. and the American Mathematical Society for granting me permission to reproduce extracts from their documents on "The New Media" and "Non-traditional forms of publication", and finally I would like to thank Barbara Beeton of that same Society for allowing herself to be persuaded to review the paper before publication and for her many helpful comments; needless to say, any errors which remain are solely my responsibility.

Bibliography

I have not given formal references in the text, since I feel that they are inappropriate in a paper of this nature; however, the following short list of publications may be of interest to those who wish to pursue further the topics discussed here.

Mathematical Typography by Donald E. Knuth, *Bulletin of the American Mathematical Society* (new series) 1 (March 1979), 337–372. [Reprinted as part 1 of *TEX and METAFONT: New Directions in Typesetting* (Providence, R.I: American Mathematical Society, and Bedford, Mass: Digital Press, 1979).]

Tau Epsilon Chi, a system for technical text by Donald E. Knuth, Stanford Computer Science Report 675 (Stanford, California, September 1978), 198 pp. [Reprinted as part 2 of *TEX and METAFONT*, the book cited above.]

The `WEB` *system of structured documentation* by Donald E. Knuth, Stanford Computer Science Report 980 (Stanford, California, September 1983), 206 pp.

Literate programming by Donald E. Knuth, *The Computer Journal* 27 (1984), 97–111.

Using Adobe Type 1 Multiple Master fonts with TEX by Michel Goossens and Sebastian Rahtz, *Baskerville* Vol. 5, No. 3 (UK TEX Users' Group, June 1995, ISSN 1354-5930), 4–8.

HTTP: A Protocol for Networked Information `http://www.w3.org/WWW/Protocols/HTTP/HTTP2.html`

A Quick Review of HTML 3.0 `http://www.w3.org/hypertext/WWW/Arena/tour/start.html`

HyperText Markup Language Specification Version 3.0 `http://www.hpl.hp.co.uk/people/dsr/html3/CoverPage/html`

Document Type Definition for the HyperText Markup Language (HTML DTD) `http://www.w3.org/hypertext/WWW/MarkUp/html3-dtd.txt`

Adobe Acrobat `http://www.adobe.com/Acrobat/Acrobat0.html`

◇ Philip Taylor
Royal Holloway & Bedford New College, University of London
`P.Taylor@Alpha1.Rhbnc.Ac.Uk`

Software & Tools

Making Indices for VINITI's "Mathematics" Abstract Journal

S. A. Strelkov and G. R. Epshtein

Abstract

The process of automatically making the indices for the abstract journal *Mathematics* is considered. The basis for creation of the indices is the database of the bibliography description sources and the text abstracts, which are typeset in the russification LaTeX 2.09.

An IBM PC is applied for working with a database and printing camera-ready copies. *EmTeX*, *MakeIndex* and Perl scripts are used. Program *DviSpell* is applied for generating the Sort Keys.

Introduction

The principal task of the Printing and Publishing Plant of the All-Russia Institute of Scientific and Technical Information (VINITI) is to complete processing of rough copies, and to prepare the originals for typesetting, for the issue of scientific information publications.

Camera-ready copies for abstract journals are produced with the help of TEX. The abstract journal *Mathematics* is typeset in LATEX 2.09 with NFSS release 1. The russification format file `rlplain.fmt`, permitting the application of Russian letters in TEX-commands, is used. XCM* fonts, the basis of Russian font forms, was developed in IHEP (Protvino) [1]. Typesetting, compilation of tex files, and output of `.dvi` files to a printer or to the screen take place on an IBM PC, thus emTEX programs are used.

We introduce our method for the accumulation of databases of the abstracts along with preparation of a camera-ready copy of the journal, *Mathematics*. During this process the abstracts and bibliographical descriptions of sources are typeset in the fields with the mnemonic entries.

```
{{{}}}\НомерРеферата 5
{{{}}}\СИД 6019838
{{{}}}\ЗаглавиеОсновное Mathematik kommt vor
{{{}}}\Автор Schmidt G\"unter
{{{}}}\Объем 61--74
{{{}}}\ЯзыкТекста Нем.
{{{}}}\ВидДокумента 1
{{{}}}\РубрШифр 51.009
{{{}}}\Процент 51:37
{{{}}}\ПОД АКК изучение геометрии
{{{}}}\Процент АКК геометрия окон
{{{}}}\Процент АКК декартова схема
{{{}}}\РусПеревод Математика выходит вперед.
{{{}}}\Реферат Обсуждается вопрос о прицbackground
```

The operators type in several abstracts in one text file, for example, the file `4n131.tex` contains the abstracts since number 131. The files that pertain to one issue of a journal are assembled in one directory, for example, `d:\4n\4n*.tex`. These files can then be used to collect the current number of a journal, to bring information into a database, and/or to construct the various indices.

The developed software is intended to formulate a collection of current, cumulative indices for an issue of *Mathematics*:

1. Short Authors Index;
2. Full Authors Index;
3. Classified Index;
4. Subject Index;
5. Index of Periodicals;
6. Index of Scientific Forums;
7. Index of Proceedings;
8. Index of Persons.

We shall consider the process of formation of the cumulative short authors index. The following procedures are usually used for a construction of an index [2]:

1. Construction of a raw index (`.idx` file) with the help of LATEX.
2. Running the program *MakeIndex* for building an alphabetized index (.ind file).
3. Reading of the index (.ind file) by LATEX to give the final typeset result.

In our case, the index construction process is carried out automatically with the help of additional converter programs written in the Perl 4.036 language. The program *DviSpell* [3] and appropriate Perl scripts are used for generating Sort Keys.

The operators, working in Norton Comander on an IBM PC, shift by the pointer to the file with the `.idx` extension (for example, author.idx) and press `<Enter>`.

Batch file mkidx.bat is started and all the necessary components of an alphabetized index are produced. In a couple of minutes, the author.tex file, with an index completely ready for inclusion in the main file of a number of the abstract journal, will be formed in the current directory.

Generating and Transforming the Raw Index

At first, the raw index can be created in the usual manner through LaTeXwith use of the command `\makeidx`. Here it is necessary to note that in an abstract journal a number of the abstracts in the given issue (short number) for the references in an index are used instead of a numbers of pages. An issue number (1-12) and an issue code (А, Б, В, Г, .93.) are specified in a semi-annual index also. A complete number of the abstract looks like: `11Б129` or `4.93.1297`. In the abstract journal style file `rjmat.sty` the respective alterations are made for an insertion of short numbers of the abstracts in a raw index, which has a special aspect:

```
Lewis Ted   0410001щщщщ
Schmidt G\"unter   0410005щщщщ
Протасова~Л.~А.    0410012щщщщ
Горбачук~М.~Л.    0410013щщщщ
Ішлінський~О.~Ю.    0410013щщщщ
```

Here the raw index, with complete numbers of the abstracts (file `author_f.idx`), is shown, which is extracted from files-portions assembled in the directory of issues.

The program *Mkidxmat.pl*, in the Perl language, was written for this purpose. This program is called as follows:

```
perl.exe mkidxmat.pl file_dir file_mode -d
```

The first parameter is the name of the file, containing names of directories subject to handling.

Example of the `file_dir`:

```
d:\1a
d:\2a
d:\3a
d:\4a
```

The second parameter is the name of the file, containing names of conditions of handling and names of output files adequate for these conditions. Numbers of the abstracts must also be specified as short or complete.

Format of a line:

```
<mode name>:<full or short>:<output file name>
```

Example of the file `file_mode`:

```
author:short:author_s.idx
author:full:author_f.idx
```

Apart from files described as outputs, the file `mkidxmat.log` is created with work protocols. The non-blank third parameter generates output of additional debug information in the protocol file.

Use DviSpell for Generating the Sort Keys

A feature of the indices for abstract journals is the usage of both the Russian and Latin alphabets, with almost all possible accents. The accents are mainly used in surnames of the authors. For a set of some letters, a mathematical mode is used.

When typesetting a large volume of journals, a tool for an automatic construction of sort keys is required. Because the operators do not know the English language, standard tools for construction of keys, available in the program *MakeIndex* 2.11 [6, 7] and 3.08 [8], do not provide a satisfactory outcome in such situations. This circumstance has induced the authors to use E. Mattes' program, *DviSpell* [3], for automatic construction of sort keys.

DviSpell is the table-controlled program intended for transformation of `.dvi` files into readable text files. This program is an effective, flexible tool for discernment and transformation of symbols with accents. The tables controlling the work of *DviSpell* are concentrated in files with the extension `.dsi`. The program *DviPrep* translates `.dsi` files into a binary file of parameters with the extension `.dsb`.

The `.dvi` file to be transformed with the help of *DviSpell* is a result of compilation of the file `dvimake.tex`, which looks like this:

```
\documentstyle[russian]{article}
\begin{document}
\parindent=0mm
Lewis Ted   0410001щщщщ
Schmidt G\"unter   0410005щщщщ
Протасова~Л.~А.    0410012щщщщ
Горбачук~М.~Л.    0410013щщщщ
Ішлінський~О.~Ю.    0410013щщщщ
A   \vspace{8dd} 0\par
......................
Z   \vspace{8dd} 0\par
А   \vspace{8dd} 0\par
......................
Я   \vspace{8dd} 0\par
\end{document}
```

This file is produced from the file `author.idx` by simple transformations, adding at the end of the file lines such as

```
Ю   \vspace{8dd} 0\par.
```

These lines become first on the letter in the sorting file.

Note: In the examples of files the long lines have been broken into several short lines.

DviSpell converts the characters found in the `.dvi` file into symbolic names using the layout table and the font table. The resulting sequence of symbols is converted into another sequence of symbols, using the active conversion table. The result is printed using the active output table.

The main idea of the application *Dvispell* for a construction of sort keys is for use on two letter code by the lower case latin letters in a table of output. If the order of the characters, letters with accents, and other symbols to be included in the sorting is known, a table of output of the symbols exceeding 256 is created. A fragment of such a table is represented below.

```
(output makeindx
(WORDSPACE ' )
(NEWLINE   "0a)
(NEWPAGE   ' )
(Z#A     ('a 'a)) % capital A
.................................
(Z#Ishrt ('a 'l)) % capital short I
.................................
(Z#YU    ('b 'g)) % capital Yu
(Z#YA    ('b 'h)) % capital Ya
.................................
)
```

In the tables of fonts and the tables of transformations are added descriptions of Russian fonts of sets LH* [5] and XCM* [1]. In a table of the letters the Russian letters and the additional accented characters (a-cedilla, d-cedilla, e-cedilla, e#dieresis, E#dieresis, i-caron, and I-caron) are added. The names of the Russian letters in the tables begin with Z#. This is to prevent incorrect operation of the substitution algorithm used in *DviSpell*.

As a result, the binary file of parameters – russianz.dsb – taking into account a given order of accented letters and unaccented characters, is obtained. Namely, the accented letter directly follows the letter in a lexicographic order of names of accents.

For a production of indices in technological process the simplified scheme of a construction of sort keys, realized in the file russianm.dsb is used. Letters are not case-sensitive and the Russian letters go before Latin. The accents do not influence sort keys, i.e., the conversion table of accents is empty. In the output table makeidx these will not be transformed. In a table of the letters the Russian letters are added, and all the accented letters are eliminated.

DviSpell is called as follows:

```
dvispell.exe -y 0.8 -o makeidx -d russian.dsb
             -v dvimake.dvi author.eee
```

The value of the parameter -y must equal 0.8, to prevent lines in the output file author.eee from running together.

Running the MakeIndex Program

With the help of programs in Perl we shall transform the files author.idx and author.eee to an input file author.jjj for the sorting program *MakeIndex*:

```
\indexentry{clcecwcics ctcecd@
Lewis Ted
\markboth{LEW}{LEW}}{0410001}
\indexentry{cscchcmcicdct cgcucnctcecr@
Schmidt G\"unter
\markboth{SCH}{SCH}}{0410005}
\indexentry{arasaqauaaataqacaa an aa@
Протасова~Л.~А.
\markboth{ПРО}{ПРО}}{0410012}
\indexentry{adaqasabaaazavam ao an@
Горбачук~М.~Л.
\markboth{ГОР}{ГОР}}{0410013}
\indexentry{cibaanciapatbeamakal aq bg@
Ішлінський~О.~Ю.
\markboth{ІШЛ}{ІШЛ}}{0410013}
\indexentry{ca@A \vspace{8dd}}{0}
................................
\indexentry{cz@Z \vspace{8dd}}{0}
\indexentry{aa@А \vspace{8dd}}{0}
................................
\indexentry{be@Я \vspace{8dd}}{0}
```

The sort keys are located in the left part of the command \indexentry{, in the right part (after @) is the surname of the author extracted from author.eee. In the fields of the command \markboth the first three decoded letters from the sort key are located. This is necessary for construction of the page headers of a cumulative index.

For the programs *MakeIndex* 3.08 or 2.11, the *EMX* 0.9b compiler is used. For *MakeIndex* 3.08, enlarged values of parameters are set at:

```
#define LONG_MAX      1024
#define STRING_MAX    256
#define ARABIC_MAX    7
```

The command line to start *MakeIndex* looks like this:

```
makeidx32.exe -s mkind.ist
              -o author.ind author.jjj
```

The sorted file author.ind will be transformed with the help of the programs in Perl to the file author.tex, ready for insertion in the main file:

```
\chapter{
}{АВТОРСКИЙ УКАЗАТЕЛЬ}
{А\,в\,т\,о\,р\,с\,к\,и\,й%
\,у\,к\,а\,з\,а\,т\,е\,л\,ь{}
\dsection{}{}{Авторский указатель{}
\begin{multicols}{5}
\raggedright
\leftskip=8dd  \parindent=-8dd
\par
\vspace{0dd}
```

```
{\bf А}блялимов~С.~Б.%
\markboth{АБЛ}{АБЛ} 4А94\par
Абрамов~С.~А.%
\markboth{АБР}{АБР} 4А275\par
............................
\vspace{8dd}
{\bf Ю}рьев~Д.~В.%
\markboth{ЮРЬ}{ЮРЬ} 4186\par
{\bf Я}нчевский~В.~И.%
\markboth{ЯНЧ}{ЯНЧ} 4А273 \par
\end{multicols}
\rule{0dd}{3mm}
\begin{multicols}{5}
\raggedright
\leftskip=8dd
\parindent=-8dd
\vspace{0dd}
{\bf A}bdesselam~B.%
\markboth{ABD}{ABD} 4A364\par
Aberbach Ian~M.%
\markboth{ABE}{ABE} 4A341\par
............................
\vspace{8dd}
{\bf Z}\'adori L\'aszl\'o%
\markboth{ZAD}{ZAD} 4A204\par
ZsilinszkyL\'aszl\'o%
\markboth{ZSI}{ZSI} 4A87\par
............................
\.Zukowski Tomasz%
\markboth{ZUK}{ZUK} 4A454\par
Zulli Louis%
\markboth{ZUL}{ZUL} 4A431\par
\endinput
```

Conclusions

In this paper we have only touched on the main ideas of construction of an index file. The actual process involves a great many more intermediate steps, including transformations of files, filtration and checks. When the typesetting is done with essentially no errors, the process of construction of indices is carried out automatically.

References

[1] Н.Л. Глонти, С.В. Клименко, В.К. Малышев, А.В. Самарин, Б.Б. Филимонов : Метапроект кирилловского алфавита для печатающих устройтв с высоким разрешением: Препринт ИФВЭ 90-96, – Протвино, 1990.

[2] M. Goossens, F. Mittelbach, and A. Samarin : The LaTeX Companion, Addison-Wesley, Reading, MA, 1994. : Chapter 12. Index Generation. pp. 345-370.

[3] Eberhard Mattes: file dvispell.doc 1.0b 16-Aug-1995, file dvisprep.doc 1.0a 04-Jun-1995.

[4] Eberhard Mattes : file emxdoc.doc 0.9b 10-Dec-1995.

[5] O. Lapko : MAKEFONT as part of *CyrTUG*-emTeX package, Proceedings of the Eighth European TeX Conference, Gdańsk, pp. 110–114, 1994.

[6] Pehong Chen and Michael A. Harisson : Index preparation and processing. Software-Practice and Experience, 19(9): 897-915, September 1988.

[7] Pehong Chen. : MakeIndex manual page.

[8] Joachim Schrod and Gabor Herr : MakeIndex Version 3.0. Reference, Draft version, August 1991.

◇ S. A. Strelkov
The Keldysh Institute of Applied Mathematics RAS
Miusskaya Sq., 4
Moscow A-47, Russia, 125047
`strelka@applmat.msk.su`

◇ G. R. Epshtein
The Printing and Publishing Plant of the All-Russia Institute of Scientific and Technical Information
October Street, 403
Lyubertsy, Moscow region, Russia, 140010

Philology

TEX and Russian Traditions of Typesetting

Mikhail Ivanovich Grinchuk

Abstract

It is well-known that the Russian style of mathematical typesetting requires one to repeat the signs of operation and relation when a line-break occurs in a formula. Standard TEX styles have no such capability. In attempting to solve this problem, a TEX program was created such that we can use, practically, standard typesetting and generate automatic repetition of all signs. Several difficulties which arose during the realization of this program are discussed and solutions proposed.

Problems

It is well-known that the remarkable system TeX was created in the tradition of English typesetting, which is different from that accepted in Russia, in particular, as concerns the typesetting of mathematical formulae. We shall specify some, more-or-less essential, differences between these two systems. For brevity we shall refer to the systems as E (English) and R (Russian).

1. The "proclaim" of theorems in the E system nowadays are typeset by a slanted font (`\sl`), whereas in the R system the standard is italic (`\it`).
2. Even in the text, typed by slanted or italic font, the R system requires punctuation marks and digits to be in the "orthogonal" shape (among punctuation marks the distinction is essential for braces, brackets, parentheses, and the marks ':', ';', '?', '!'). This rule is not the case for fictional literature.
3. The R system often requires the use of l e t t e r - s p a c e d typesetting.
4. The E system has two kinds of quotation marks: "xxxx" and 'xxxx'. The R system has two kinds too, but these are different: «xxxx» and „xxxx“.
5. Use of dashes. It is possible to illustrate four distinct situations:
   a) ...we shall take 2—3 litres of water...
   b) ...we shall take two—three litres of water...
   c) ...the Newton—Leibnitz formula...
   d) ...and other—they do not want...

   The E system distinguishes between short (–) and long (—) dashes; the first one is used for cases (a) and (b). In case (c), the hyphen char (-) is usually used, though it leads to confusion between two surnames and one double surname. In all four cases, the dash is used without spaces at either end, and a possible line break may occur following the dash.

   In contrast with the E system, the R system has only the long dash, surrounded by spaces (half the width of a usual space). [Though it may appear rather strange, in my opinion, it would be more regular to act in the following manner: in case (a) to use a short dash without spaces; in case (b)—the long dash without spaces; in case (c)—the long dash with half-spaces; and in case (d)—the long dash with normal spaces.] The possible line-break in all cases would occur after a dash.
6. The R system does not recommend increasing space between sentences (i.e., we use "`\frenchspacing`").
7. There are some differences in punctuation of headings and captions: the E system accepts "1.1 Introduction" and "Fig. 1.", but the R system uses "1.1. Introduction" and "Fig. 1".
8. There is a short list of non-equal "mathematical words": `tan/tg`, `cosec/csc`, `sinh/sh`, etc.
9. The R system uses `\varphi`, `\varkappa`, `\varepsilon`, `\varnothing`, `\leqslant`, and `\geqslant` instead of `\phi`, `\kappa`, `\epsilon`, `\emptyset`, `\le`, and `\ge`.
10. With regard to ellipses, the R system uses `\cdots` only in products of the form $x_1 \cdots x_n$. All other cases requires `\ldots`. The R system does not use 4-dot ellipses, and in combination with '?' or '!' uses "?.." and "!.." (i.e., two dots).
11. And now the most essential: the E system permits one to break a formula after a binary operation or binary relation, but the R system also requires repetition of the mark at the beginning of the second line. This is true also for `\dots`.

Some Solutions

For almost all of the listed differences, it is very easy to add to or modify TeX. Corresponding to the previously listed items:

1. If the fonts are switched "manually", there is no problem; otherwise, it is necessary to make slight corrections to the style file.
2. The optimum solution is to have slanted and italic fonts with non-slanted punctuation and digits. If we have no such fonts, then when necessary, it is possible either to use something like `{\rm(}` or `$($`, or to define a command like `\def\({{\rm(}}`.
3. If you require much letterspaced text, it is possible to insert spaces manually (but perhaps, it is better not to use spaces, but to use the `\~`). Here we need to increase interword spacing (up to `\quad`), and where appropriate to insert discretionary hyphens `\-`.

   It is not very difficult to write a command for the call "`\spaced{Proof}`" to generate "P r o o f", but there are possible difficulties in a situations like
   ```
   \spaced{Lemma 9$'$ \cite{17} (the main)}
   ```
4. Either you have «such» quotes, or not. Russian TeX manuals try to standardize commands (written by Cyrillic letters) — `\lk` and `\pk` (or

\l and \p) — as shortcuts for “Lyevaya Kavychka” and “Pravaya Kavychka” (Left Quote and Right Quote). But it is more natural to use the ligatures « and » (there will be no difficulties associated with “eating” the following spaces).

The „ and “ can be made of existing signs; they are the same as used in `german.sty` commands `\glqq` and `\grqq`. However, it is more natural to type “`,,`” and “`‘‘`”.

5. If we do not use “half-spaces”, it is enough to use a combination “`~---`” with a space added at the end. The half-space is produced as follows:

```
\newskip \thehalfspace
\def \halfspace{\unskip
  \thehalfspace=
     \the \fontdimen2 \the \font
     plus \the \fontdimen3 \the \font
     minus \the \fontdimen4 \the \font
   \divide\thehalfspace by 2
   \hskip \thehalfspace
   \ignorespaces }
```

6. For this item, we use `\frenchspacing`.
7. Requires either manual correction at each point, or it is necessary to correct the style files (for LaTeX, such work was done by Lwowsky).
8. This is also relatively simple — find how `\tan` is defined, and define the `\tg` in the same way.
9. AMS-TeX has all necessary marks. In other cases you need to load AMS-TeX fonts `msam` and `msbm`.
10. The easiest solution is to use only the commands `\cdots` and `\ldots`.
11. See the next section.

The Choices

In one approach, we can make, for example, a breakable “+” sign like this:

```
\def\plus{\discretionary{+}{}{}+}
```

or

```
\def\plus{+\discretionary{}{+}{}}
```

(the more natural variant

```
\def\plus{\discretionary{+}{+}{+}}
```

is impossible, because in mathematics mode, the `\discretionary` third field should be empty). The second variant provides a more correct spacing.

Naturally, if you are using such commands, it is necessary to forbid “built-in” math breaks, making the parameters `\relpenalty` and `\binoppenalty` more than 10000.

The offered choices are not too successful for two reasons — first, the `\discretionary` mark will always be large (`\scriptstyle` is ignored); and secondly, the object inside `\discretionary` is textual, and the transition here to mathematics is impossible.

A more reliable solution is a pair of commands

```
\def\brokenrel#1{%
    \mathrel{#1}\selector{#1}}
\def\brokenbin#1{%
     \mathbin{#1}\selector{#1}}
```

(which means that to have a breakable command like “`\ne`” the user must type `\brokenrel\ne`, and for the breakable operation “`\times`” typesetting is `\brokenbin\times`). Here the command `\selector` is defined as:

```
\def\selector#1{\mathchoice
   {\discretionary {}%
     {\hbox{$\displaystyle#1$}}{}}%
   {\discretionary {}%
     {\hbox{$\textstyle#1$}}{}}%
   {\discretionary {}%
     {\hbox{$\scriptstyle#1$}}{}}%
   {\discretionary {}%
     {\hbox{$\scriptscriptstyle#1$}}{}}%
              }
```

It is more convenient, however, rather than input the commands `\brokenrel` and `\brokenbin` manually, to redefine the “breakable” operation as:

```
\let\originaltimes=\times
\def\times{\brokenbin\originaltimes}

\let\originalleqslant=\leqslant
\def\leqslant{\brokenrel
         \originalleqslant}
```

One must not overlook the command `\not`:

```
\let\originalnot=\not
\def\not#1{\brokenrel{\originalnot#1}}
```

In a similar way it is possible to make breakable marks “+”, “−”, “<”, “>”, and “=” (as well as “*”):

```
\mathcode`\+="8000
{\catcode`\+=\active
 \gdef+{\brokenbin{\mathchar"202B}}}

\mathcode`\>="8000
{\catcode`\>=\active
  \gdef>{\brokenrel{\mathchar"313E}}}
```

(codes `"202B` and `"313E`, etc., are used in the beginning of format files like `plain.tex` or `lplain.tex`, where the `\mathcode` table is set).

But the methodical realization of all such redefinitions may lead to some unexpectedly strange breaks:

...... system of $\geq$
$\geq n$ vectors......
...... has the value of $-$
-1 or -2 ......
...... $x = -$
$-y$......
...... $x = (-$

$-y+z)\ \ldots\ldots$

Fortunately, these situations are simple to correct. The idea is to provide, at the beginning of each formula and after each "redefined" operation, some large additional penalty (for example, 13131). Then each potentially breakable mark should at first check the previous penalty and should not be broken if it is 13131. A possible realization:

```
\relpenalty=13131
\binoppenalty=14141
\everymath={\penalty\relpenalty}

\def\brokenbin#1{%
 \ifnum\lastpenalty=\relpenalty
   \mathbin\{#1} \else
   \mathbin\{#1} \selector{#1} \fi
     \penalty\binoppenalty}

\def\brokenrel#1{%
  \ifnum\lastpenalty=\relpenalty
     \mathrel{#1}\else
     \mathrel{#1}\selector{#1}\fi
     \penalty\relpenalty}
```

It is thus possible to forbid breaks after `\bigl`-signs (`\Bigl`, `\biggl`, etc.):

```
\def\bigl#1{\mathopen\big#1%
  \penalty\relpenalty}
```

(The `\left`...`\right`-constructs give no problems because breaks are prohibited).

To forbid breaks after left parentheses, braces, and brackets, etc., we can redefine their signs in math. For example,

```
\mathcode`\(="8000
{\catcode`\(=\active
 \gdef({\delimiter"0283000
        \penalty\relpenalty}}
```

(the magic number is the `\delcode` of "(").

Another possible difficulty may occur. After such total redefinitions, expressions with indices and expressions of the type `\mathord\rightarrow` (e.g., this appears in the standard definition of the command `\arrowfill`) can "deteriorate". A solution here is to specify the index in braces (e.g., to replace `\mathord\rightarrow` with `\mathord{\rightarrow}`). If you do *not* want to place indices in braces, it is possible to redefine the signs for indices, defining them as active symbols with parameters:

```
\let\upperindex=^
\let\lowerindex=_
\catcode`\^=\active
   \def^#1{\upperindex{#1}}
\catcode`\_=\active
   \def_#1{\lowerindex{#1}}
```

Acknowledgements

This work was supported in part by Russian Foundation for Basic Research, grant no. 96–7–89406.

◇ Mikhail Ivanovich Grinchuk
Department of Mechanics and Mathematics, Moscow State University, Moscow, Russia

Humanities

TEX and the Humanities

Christina Thiele

Introduction

Well, as you can see, this isn't a thematic issue about TEX and the Humanities. Pierre MacKay and I were entrusted with putting such an issue together last spring, but the silly truth is — we've both been far busier than either of us had anticipated, doing just that: typesetting humanities materials with TEX. Trust us — TEX is alive and kicking wildly all over the humanities playground.

But back to that *nostra culpa* bit ... It must also be said that we haven't exactly been inundated with offers of articles or ideas — most suggestions in fact have been in the form of messages from the `comp.text.tex` newsgroup forwarded by the *TUGboat* editor. It's Pierre's view that perhaps people who are using TEX, but not for maths and sciences, somehow feel their work isn't of sufficient complexity or of broad-enough interest to write about. Rubbish, we say! This column should help dispel such misperceptions, still around almost 20 years after TEX first came on the scene.

The Humanities column will become a regular feature in *TUGboat*, providing an on-going dialogue about this very large area of activity. It will include reports and updates on new projects, information on resources, be they style files or articles, along with titles just published (especially handy for those times when you feel at a loss to cite new and diverse examples of TEX-without-math), and so on.

We are hoping that this first overview will not only provide a quick glimpse of some of the important hidden areas of TEX activity, but will also serve as an incentive to people working in these and other related areas, bringing us information about their projects, their particular hurdles (all surmounted, of course!), and their unexpected discoveries.

— — * — —

The section headings are in alphabetic order, to make it easier to find your particular field(s) of interest. Some headings refer to format more than content (Collected works, Critical editions, Dictionaries, Journals, for example) while others are more directly related to specific fields of study and research (Linguistics, in this issue; Economics and Music in the next). The "Other projects" section is of course for things we haven't already stuck into a pre-conceived box, or for items where there's only one representative to date. Please send new headings you'd like to see treated.

Following a (fairly) brief description of the more salient characteristics of each category, expect to find information and news on current projects, any resources (style files, documentation, etc.) that are available, and then a list of publications in that field.

It is still difficult to explain to someone (a publisher, for example) that not only is it logical and normal to consider TEX as much as any other typesetting option for non-maths/sciences, but that it has been going on for quite some time. Indeed, in the hunt for journal titles, it would seem that 1983 was the starting point. But a list of publications, while arid in itself, is often the best way, simply by its length, to silence the doubters who still exist — and to encourage those who really do want to use the best possible tool for putting their research to paper.

— — * — —

This edition of the column is necessarily incomplete: restrictions of space and time mean we've focussed on fields where information is readily at hand. And so, it's true, the initial listing of titles includes many which I've worked on — well, I had to fill these sections with *something*!

There are of course other areas outside of the humanities, and of the maths/sciences: from magazines (Don Hosek's *Serif* being an excellent example) to phone books (I recall someone talking once about such a project, perhaps in Ireland). This column won't become the grab-bag of anything not in maths or the sciences — I shall leave that area of proseletyzing to someone else!

One final word. To try and keep the present column to a reasonable length, only publications dated from 1994 onwards are included (with a few exceptions). We're hoping to eventually have these publications listed on the Web, which would allow not only more complete entries and information about each book's TEX-ing particulars, but also would make it possible to have an archival place of record for everything that's gone before.

Collected works

What's intended with this category is projects which focus on the entire opus or a facet of a single person or a small discrete group — and are just plain huge! Not just a single book or even a two-volume affair, not a periodical that comes out with a new issue every few months, but a project which anticipates many volumes, and a work period spanning many years, possibly with changes in personnel and hardware along the way. The sheer mass of material, the need for stability in the software over the long term, combined with the need for flexibility as the work progresses — these are some of the core factors which have led these projects to TEX. And we're glad they did!

Following a lead provided by Paul Neubauer at Ball State University, we've learned of a project that's been going on for some 20 years! — an edition of the letters of Harriet Beecher Stowe[1] — under the direction of E. Bruce Kirkham (Dept. of English, Ball State). Paul's suggestion to consider LATEX has allowed the project team to write and insert both textual notes (describing emendations to the manuscript), and informational notes (identifying all the persons, places, and events Mrs. Stowe mentions in her letters). To date, some 2,200 of the estimated 2,500 letters have been dealt with.

We'd like to hear about more such projects, and any volumes that have already been published.

Credits: E. Bruce Kirkham, Paul Neubauer

Critical editions

In the general field of literary studies, critical editions are amongst the most complex in terms of structure. An "old" text, something typically dated three or four hundred years ago, is reproduced, with great care taken to identify variations in actual copies, not many of which may still exist in various libraries around the world. One or more of these copies is identified as being the most stable, or the most unchanged, or the best (by whatever criteria may be appropriate), and then variations from that main text are noted in the critical apparatus — footnote-like elements at the bottom of each page. Normally line numbers run down either the left- or right-hand margin, perhaps at every 5th or 10th

[1] American author (1811–1896) of *Uncle Tom's Cabin* (the book that helped precipitate the American Civil War), and 36 other works.

line, in order to quickly locate the word or phrase in question.

Glosses ("meanings") of words no longer easily understood often appear at page bottom, or at the end of the text, where extensive explanations and commentary are placed. Cross-referencing by line and page numbers is also prevalent: line numbers are often incorporated into the apparatus at page-bottom, while line and page numbers are often found in the commentary.

In short, highly elaborated and formalised structures are to be found in critical editions. And as can be imagined, line- and page-numbering (always subject to change as corrections are input), variable space required for the critical apparatus at page-bottom, and the necessary cross-referencing can make this an extremely expensive typesetting activity for publishers to undertake (not to mention the many years of research needed before the typesetting stage even begins to loom on the horizon). All of which make critical editions sometimes appear to be a rather non-lucrative type of publication.

Of course, one would like to imagine that the author or typesetter using TEX makes this type of complexity a mere cakewalk, via macros (and of course, skilled tagging!), and thus re-opens an area of research unhindered by typographic (and economic) constraints. If production costs are too high, projects either don't get off the ground, or have to be scaled back. But with the opportunities which `edmac` (and some early clones) afford, it is possible for scholars to consider undertaking research which can be presented with the detail and the quality that they would wish.

Existing resources At the moment, there is one major package devoted to critical editions — `edmac`, from John Lavagnino and Dominik Wujastyk.[2] While only for plain TEX, `edmac` — in existence now for almost a decade, but only finalised and completed with documentation a few years ago — provides tremendous possibilities for not merely the typesetting of critical editions, but perhaps more importantly, for expanding the scope of complexity in such works. The `edmac` package can be found on CTAN in `tex-archive/macros/plain/contrib/edmac/`. And I've just wandered into a tidy description of the package via the TUG home page, under "Other TEX resources on the Web", then "Packages and programs".

[2] *Critical Edition Typesetting: The EDMAC format for plain TEX.* San Francisco and Birmingham: TEX Users Group and UKTEX Users Group, 1996, 110pp.

The books listed for Dovehouse Editions in Ottawa are all done by me, using an early `edmac` forerunner (provided in February of 1989 by a fellow called Mac Pigman, to whom I shall forever remain indebted), which was then heavily reworked by Andrew Dobrowolski. I suspect that the features we built in are now covered by what eventually became the final version of `edmac`.

We hope to have an article on Andrea de Leeuw van Weenen's project [1] in a future *TUGboat* issue, along with something more on `edmac`.

What we don't know — in addition to the dozens of published works which surely have been produced over the years [please send us details!] — is if there are resources such as newsgroups or discussion lists for people working on critical editions, and if other style files exist which are either not well-known, or are in-house only.

Publications

[1] de Leeuw van Weenen, Andrea, ed. *The Icelandic Homily Book.* Perg. 15 4° in the Royal Library, Stockholm. Reykjavík: Stofnun Árna Magnússonar á Íslandi, 1993, 640pp. [For a brief article about this project, which took some 19 years to complete, see *MAPs* 14 (95.1), pages 31–34.]

[2] Raspa, Anthony, ed. *Pseudo-Martyr* by John Donne [1610]. Kingston and Montreal: McGill-Queens, 1993. lxxxix + 427pp.

[3] *Barnabe Riche: His Farewell to Military Profession*, ed. Donald Beecher. The Barnabe Riche Series 1. Ottawa: Dovehouse Editions, 1992, 336pp.

[4] *The Bachelor's Banquet*, ed. Faith Gildenhuys. The Barnabe Riche Series 2. Ottawa: Dovehouse Editions, 1993, 153pp.

[5] Bishop Francis Godwin's *The Man in the Moon*, ed. John Anthony Butler. The Barnabe Riche Series 3. Ottawa: Dovehouse Editions, 1995, 118pp.

[6] *The Dialogue of Solomon and Marcolphus*, ed. Donald Beecher. The Barnabe Riche Series 4. Ottawa: Dovehouse Editions, 1995, 241pp.

[7] Robert Greene's *Menaphon*, ed. Brenda Cantar. The Barnabe Riche Series 5. Ottawa: Dovehouse Editions, 1996, 198pp.

[8] *A Gathering of Griseldas*, ed. Faith Gildenhuys. The Barnabe Riche Series 6. Ottawa: Dovehouse Editions, 1996, 220pp.

[9] Lord Herbert of Chirbury, *Pagan Religion* (*De religione gentilium*), ed. John Butler. Ottawa: Dovehouse Editions, 1996, 380pp.

[10] Thomas Lodge's *Rosalind*, ed. Donald Beecher. The Barnabe Riche Series 7. Ottawa: Dovehouse Editions, 1997, 266pp.

[11] de Vroom, Theresia, ed. *Ten Netherlandic Plays*, Carleton Renaissance Plays in Translation series. Ottawa: Dovehouse Editions, 1997, 242pp.

Credits: Andrea de Leeuw van Weenen

Dictionaries

Everyone knows what these look like! The general structural elements include a page layout of two or more columns (depending on page size), with running heads indicating the first word and last word on each page, and a tremendously varied assortment of font styles and sizes, along with very specific punctuation sequences, all of which imply highly-tagged input files. These documents can be unilingual, bilingual, or multilingual, and not all use the Latin alphabet.

Via the `ling-tex` list, I've heard from Harold F. Schiffman about an English-Tamil dictionary in the works. To date they've produced over 400 pages of output, done in LaTeX (English and Tamil fonts), and expect the "final product will be 550–600 [pages], probably totally photo-composed in LaTeX, unless some publisher has other ideas." The Tamil font `WnTamil` was designed by Tom Ridgeway of the University of Washington, with input from Schiffman.

And while I wanted to list books only of the past two years, to keep things short ... well, dictionaries take so long that I'll bend a bit. The Newman dictionary listed below is interesting in that it had special fonts created for the Hausa text; thanks to Jörg Knappen for passing on the information (he adds that this of course predates the FC fonts for African languages).

Publications

[1] Jungmann, Paul, and J.J.S. Weitenberg. *A Reverse Analytical Dictionary of Classical Armenian.* Trends in Linguistics, Documentation 9. Berlin: Mouton de Gruyter, 1993, 842pp.

[2] Newman, Roxana Ma. *An English-Hausa Dictionary.* New Haven, Yale UP, 1990, 327pp.

[3] Weitenberg, J.J.S., and A. de Leeuw van Weenen. *Lemmatized Index of the Armenian Version of Deuteronomy.* Septuagint and Cognate Studies Series 32. Atlanta, GA: Scholars Press, 1990, 108pp.

Credits: Harold F. Schiffman; Jörg Knappen, Andrea de Leeuw van Weenen, Emma Pease

Journals

The principal feature of a journal is its repetitiveness. Once a style is established, it becomes an easy matter to deal with the document's general structure and thus its coding. Two other features one could highlight are: (a) each issue contains materials from many sources (different software and hardware and skill levels of the contributors); and, (b) each issue has more than one component: articles, book reviews, short notes, perhaps lists of recent publications, occasionally advertisements, and then the cover material. Whatever the frequency (annual, twice or three times a year, quarterly), regular turnaround must be maintained.

There are three entries in the list below for which I have no information beyond their titles. If anyone can provide details, please get in touch.

Existing resources To date, my impression is that most humanities journals use TeX and co. as an in-house tool, the bulk of their submissions coming from word-processing programs such as WordPerfect, MS Word, and so on. We would therefore like to use this space as an ongoing source of information on style files made available to authors by various publishers or editors for submissions to their humanities journals.

In 1993, I set out to find out what humanities journals were being produced in TeX for a poster display on its use in the Humanities for the annual meeting of the Society for Scholarly Publishing. I sent out a form I'd devised, and the results are mainly to be found below, with some additions over the years from Pierre MacKay and Michael Covington. Please send updates and corrections for anything erroneous. And if anyone knows of other humanities journals which use TeX, or which once did, please let me know and I'll forward the form to them.

There is also a very useful article by Gabriel Valiente Feruglio[3] which includes a long listing of titles (a fascinating and impressive collection). There are a few which I'd like to hear more about, since they sound like humanities candidates, as it were: *Computers and the Humanities*, *Man and World*, and *Philosophical Studies* (all from Kluwer Acadmic).

Publications

[1] *Canadian Journal of Linguistics* (plain TeX from 1984 to 1992; LaTeX since 1992)

[2] *Canadian Review of Social Policy* (from 1991; plain TeX)

[3] *Carleton Papers in Applied Language Studies* (1984–1992; plain TeX)

[4] *Classical Antiquity* (from 1994; plain TeX)

[5] *Computational Linguistics* (no details)

[6] *Eighteenth-Century Fiction* (from 1988; plain TeX)

[7] *English Studies in Canada* (plain TeX from 1989 to June 1995; LaTeX from Sept. 1995)

[8] *ESQ, A Journal of the American Renaissance* (no details)

[3] "Do journals honor LaTeX submissions?" *TUGboat* 17,2 (1996), pages 191–199.

[9] *Florilegium* (plain TeX from 1987 to 1995; LaTeX since 1996)

[10] *Hesperia* (from 1992; plain TeX)

[11] *Journal of Jewish Studies* (no details)

[12] *Journal of Natural Language Processing* (since 1994; LaTeX)

[13] *Journal of Southeast Asian Language Teaching* (1990–1995; LaTeX)

[14] *Middle East Studies Bulletin* (1984–92; plain TeX)

[15] *Papers of the Algonquian Conference* (1983–1994; plain TeX for all but 1994)

[16] *Phoenix* (from 1989; plain TeX)

[17] *Raven: A Journal of Vexillology*[4] (from 1993; LaTeX)

[18] *Recherches sémiotiques/Semiotic Inquiry* (1986–1988; plain TeX)

[19] *Revista Canadiense de Estudios Hispánicos* (1987–1989; plain TeX)

[20] *Scandinavian-Canadian Studies* (1986–1991; plain TeX)

[21] *Wind Row* (no details)

Credits: Michael Covington; Paul Neubauer

Linguistics

In addition to those interesting characters in phonetic fonts, linguistics material also requires certain highly formalised structures: tree diagrams (diagonal lines, not squared), matrices, examples with glosses lined up under the foreign language word(s), and several other components. Other forms of linguistic research, such as semiotics, the study of signs, rely heavily upon mathematical notations and symbols. Most linguistics material is presented in transliterated form; that is, in the Latin alphabet.

Existing resources In 1995, an inventory of currently known style files was published in *TUGboat* 16,1 (1995), pages 42–44, entitled "TeX and Linguistics". This was based on information gleaned from the discussion list `ling-tex`. The inventory can be found on CTAN in `tex-archive/info/ling-mac.tex`.

To date, we are unaware of any publishers who have house style files either for in-house production or for their authors, or of any journals which require submissions in TeX and provide authors with style files. For linguistics journals and other periodicals produced with TeX, see the **Journals** section.

Publications

[1] Belletti, Adriana, and Luigi Rizzi, *Parameters and Functional Heads.* New York: Oxford University Press, 1996, viii + 300.

[2] Boutkan, Dirk, en Maarten Kossman. *Het stadsdialect van Tilburg, klank- en vormleer.* [The Tilburg dialect: phonetics and grammar]. Cahiers van het P.J. Meertensinstituut Nr. 7. Amsterdam, 1996, vii + 102 pp.

[3] Rosetta, I.M.T. *Compositional Translation.* Dordrecht: Kluwer Academic, 1994, xviii + 478 pages.

[4] Tucker, A.N. *A Grammar of Kenya Luo (Dholuo).* C.A. Creider, ed. Cologne: Rüdiger Köppe Verlag, 1994, 626pp.

[5] Weber, David. *Ortografía: Lecciones del quechua.* Serie Lingüística Peruana 32. Lima: Instituto Lingüístico de Verano, 1994, 183pp.

[6] Weber, David. *Relativización y cláusulas nominalizadas en el quechua huallaguino.* Serie Lingüística Peruana 33. Lima: Instituto Lingüístico de Verano, 1994, 150pp.

[7] Weber, David. *Una gramática del quechua del Huallaga (Huánuco).* Serie Lingüística Peruana No. 40. Lima: Instituto Lingüístico de Verano, 1996, 641pp.

Credits: Chet Creider, Michael Doob, Andrea de Leeuw van Weenen, Jan Odijk, David Weber

Other projects

This is for projects that don't fall into my neat little boxes listed above, or for which no clear heading comes to mind. The Degenaar entry is in the field of the history of psychology/philosophy. The remaining entries are variously from English literary studies, general interest titles, and historical accounts (non-fiction).

So ... it seems we do end up with a grab bag collection, after all. Suggestions on how to deal with such titles — do we include them here, are they interesting but not "humanities – other"? What should we do with these "orphaned others"?

Publications

[1] Arima, E. *Blackfeet and Pale Faces: The Pikani and Rocky Mountain House. Commemorative History from the Upper Saskatchewan and Missouri Fur Trade.* Ottawa: Golden Dog Press, 1995, 256pp.

[2] Campbell, Ian. *Murder at the Abbaye: The Story of Twenty Canadian Soldiers Murdered at the Abbaye d'Ardenne.* Ottawa: Golden Dog Press, 1996, 200pp.

[3] Degenaar, Marjolein. *Molyneux's Problem: Three Centuries of Discussion on the Perception of Forms.* Dordrecht: Kluwer Academic Publishers, 1996. 156pp.

[4] Kelly, William, ed. *The Mounties As They Saw Themselves.* Ottawa: Golden Dog Press, 1996, 354pp.

[5] Lovejoy, R.B. *The Golden Dog Book of Fairy Tales and Animal Stories.* Ottawa: Golden Dog Press, 1996, 132pp.

[4] Vexillology (dating from 1970): the study of flags.

[6] McDougall, Robert L. *Narrative of War: From the Beaches of Sicily to the Hitler Line with the Seaforth Highlanders of Canada, 10 July 1943–8 June 1944.* Ottawa: Golden Dog Press, 1996, 218pp.

[7] Mozersky, Judy. *Locked In: A Young Woman's Battle with Stroke.* Ottawa: Golden Dog Press, 1996, 138pp.

Credits: Dr. G. Lokhorst

— — * — —

And so ends the first edition of a new column. We hope you've found something new here, something interesting, something intriguing. The next issue will continue with the survey of topics (including Music, Economics, and — following a suggestion from David Weber — Bibles); by then, we expect to have updates and probably some corrections to the information below. Suggestions on how to make this a lively place to turn to when your *TUGboat* issue arrives are always welcome and needed. Even a more catchy title to this column might be nice!

At this stage, we're just hungry for information, news, book details, and so on. Offers of help with the bibliographic work would definitely be appreciated: keeping track of what's been submitted, making sure entries are as complete as possible, and eventually getting them into shape for the Web; send mail to `cthiele@ccs.carleton.ca`, and use the subject line `TB-Humanities`, if possible.

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

Book review: *The Printing Revolution in Early Modern Europe*

Jonathan Fine

Elizabeth L. Eisenstein, *The Printing Revolution in Early Modern Europe*. ISBN 0 521 44770 4 (paperback) £7.95, $11.95, 300 + xiv pages, Cambridge University Press (Canto imprint).

This book is an abridgement for the lay reader of a full-scale work written for scholars, the result of a decade of study. It is that rare and valuable thing, a scholarly work accessible and interesting to the educated general reader.[1]

[1] Originally published in *Baskerville* vol. 5, no. 5 (1995, pages 15–17. Republished with permission.

The author "had long been dissatisfied with the prevailing explanations for the intellectual revolutions of early modern times." She decided to investigate:

> What were some of the most important consequences of the shift from script to print? Anticipating a strenuous effort to master a large literature, I began to investigate what had been written on this obviously important subject. To my surprise, I did not find even a small literature available for consultation. No one had yet attempted to survey the consequences of the fifteenth century communications shift.

In this century there has been a communications shift, which is still continuing, and whose consequences are likely to be as important and durable as that of the fifteenth century (from script to print). The telegraph was followed by the telephone, radio and then television. Electronic computers (built out of radio valves, telephone exchange parts and the like) were the next development. Magnetic recording technology (developed for voice and music) was adapted for use by computers, moving from tapes to cards, drums and discs. Telephone lines were used to transfer data and thus to link geographically remote computers. The latest in the technological revolution is the Internet or World Wide Web, CD-ROM, and the beginnings of what are called portable documents.

History is more than a sequence of Kings and Queens, or in our case a list of technological Aces and Jokers. This technology came from somewhere. Its creation appears to be influenced by political, economic and social forces. For example, universities support science. In our case, fundamental contributions of Faraday, Maxwell, Planck, the Curies, Einstein, Rutherford, Heisenberg, Schrödinger, Bohr, Dirac and many others have laid the scientific basis for the present shift from paper to digital or electronic media.

From where then did science arise? Was Newton chance? What effect did science and religion have on each other? How did what we may call early modern Europe emerge from late medieval society? Was it due to the Renaissance? Eisenstein's book is a carefully considered and well-written discussion of the emergence of a human culture based on printed communication, that began in the late fifteenth century, and its influence on the humanities, on religion and on science. This print culture was something new, just as the film and television culture of today is new. Each has an effect on the human mind.

Before describing the text of this book, some comments about illustrations. They have been well chosen. There are about sixty illustrations, mostly pages from rare books, but sadly they are not indexed. They have reproduced well, and add greatly to the book. For Bibles, there is a page from the Gutenberg Bible compared to a contemporary hand-copied Bible, the so-called wicked Bible of 1631 (which misses the 'not' from the adultery commandment), and the frontispiece and beginning of Genesis from Christopher Plantin's great Antwerp Polyglot Bible of 1571. There is the title page of Galileo's final treatise (printed by the still flourishing Elsevier), and its appearance on the (printed) Vatican index of prohibited books. There are many other useful and illuminating illustrations, including an extract from a pattern book for tailors and dressmakers (Seville, 1588) which made the Spanish fashion "visible through the far-flung Hapsburg empire".

The quotations and incidental facts are also well chosen. Here are two such gems. The hazards of having a famous face are not new to this century. Eisenstein writes:

> The difference between the older repeatable image which was stamped on coins and the newer by-product of print is suggested by one of the more celebrated episodes of the French Revolution. The individual features of emperors and kings were not sufficiently detailed when stamped on coins for their faces to be recognized when they travelled incognito. But a portrait engraved on paper money enabled an alert Frenchman to recognize and halt Louis XVI at Varennes.

while to open the Preface she quotes Joseph Ames (1749) who wrote:

> I do ingenuously confess that in attempting this history of Printing I have undertaken a task much too great for my abilities the extent of which I did not so well perceive at first.

The book is divided into two parts. The first, "The emergence of print culture in the West", can be thought of as an essay on the differences between the earlier scribal culture and the new print culture. Before print each book was a unique item, and each new copy would have new copyists' errors. It rarely makes sense to talk of an edition of a hand-written book. Printed books come in editions, of hundreds or thousands or beyond. Each copy will be identical, from one to the next, except that printers too can make errors.

Even without the commissioning of new works, the wider circulation of scribal texts is a significant change. For example

> as a student at Cracow in the 1480s, the young Copernicus probably found it hard to get a look at a single copy of Ptolemy's *Almagest* — even in a corrupted medieval Latin form. Before he died, he had three different editions at hand. As a fourteen-year-old in Copenhagen in 1560, the young Tycho Brahe could purchase all of Ptolemy's work, including an improved translation of the full *Almagest* made from the Greek.

Perhaps the most important consequence of the shift was the emergence of an expanding republic of letters. The reading public, their booksellers and printers, their authors and illustrators.

> As the key figure around whom all arrangements revolved, the master printer himself bridged many worlds. He was responsible for obtaining money, supplies, and labor, while developing complex production schedules, coping with strikes, trying to estimate book markets and lining up learned assistants. He had to keep on good terms with officials, while cultivating talented authors and artists who might bring his firm profit or prestige. In those places where his enterprise prospered and he achieved a position of influence with fellow townsmen, his workshop became a veritable cultural centre attracting local literati and celebrated foreigners, providing both a meeting place and message centre for an expanding cosmopolitan Commonwealth of Learning.

Christopher Plantin, whose activities are briefly discussed, is an outstanding example of a master printer: "After settling in Antwerp and establishing ties with Leiden, Plantin decided to learn Dutch. Never one for wasted effort, he 'placed in piles and in alphabetical order' each new word that he learned. Thus was launched a collaborative venture which resulted in … the 'first Dutch dictionary worth its name.'" For more on Plantin's remarkable life and legacy, consult Colin Clair's biography.

The second part of the book is entitled "Interaction with other developments". In it "possible relationships and connections are explored with the aim of providing a basis for some tentative conclusions concerning the effects of the communications shift upon three movements which seem strategic in the shaping of the modern mind."

The first is the Renaissance, which has long been something of a challenge for historians to understand. Eisenstein suggests that it would be productive for historians to "direct attention to something that really did happen, that was obviously of crucial importance, that occurred in the second half of the fifteenth century and at no other time in the history of the West". This is of course the shift to print culture. Later she writes "early humanists, from Petrarch to Valla, owe their still vital reputation as culture heroes to the prosaic print-made knowledge industry. ... Earlier scholars had been less fortunate."

The second is "Western Christendom disrupted". Protestantism was "the first movement of any kind, religious or secular, to use the new presses for overt propaganda and agitation against an established institution. By pamphleteering directed at arousing popular support and aimed at readers who were unversed in Latin, the reformers unwittingly pioneered as revolutionaries and rabble rousers." Between 1517 and 1520, we are told, Luther's thirty publications sold well over 300,000 copies. Even by modern standards this is a considerable achievement.

This book was originally written as a contribution to historical debate. Its main thesis is that the effect of print as an agent of social change has been ignored or underestimated. "One of the mysteries of Reformation history [is] how this proposal for academic disputation [Luther's theses], written in Latin, could have kindled such enthusiastic support and thereby have such far-reaching impact", wrote one modern scholar. To dispel this mystery, Eisenstein suggests that we should

> instead of jumping directly from church door to public clamor, move more cautiously, a step at a time, looking at the activities of printers, translators, and distributors who acted as agents of change. Probably we ought to pause with particular care over the interval in December 1517 when three separate editions were printed almost simultaneously by printers located in three separate towns.

"On the whole," the author concludes, "it seems safe to conclude that all the problems associated with the disruption of Western Christendom will become less baffling if we approach them by respecting the order of events and put the advent of printing ahead of the Protestant Revolt."

The third movement is "The book of nature transformed: printing and the rise of modern science". As with the Reformation, the author argues that "the advent of printing ought to be featured more prominently by historians." This theme is developed mainly with regard to maps and astronomy, the trial of Galileo and the publishing activities of the Royal Society. Copernicus has already been mentioned. Newton was encouraged to publish. Galileo discouraged.

One of the illustrations is of a grandly titled "A description of the whole world" (1606), and another is a printed challenge from Blaeu (1622) to find any errors in his printed sea charts. He wrote, "Whatsoever there is yet resting to be corrected or made better, is as easie to be corrected in the Cardes [charts] that are printed, as in them that are written, which we also are readie to doe at our charge, if any man can by good proofe shewe us any thing, that is to be corrected in the Cardes that are printed by us." This is an early example of errors being, if not rewarded, at least corrected without charge.

The author gives many useful examples of the opportunities for (self-)promotion that print provides. We know much about the early history of print from the documents they printed about themselves: "Indeed, their use of title pages entailed a significant reversal of scribal procedures; they put themselves first. Scribal colophons had come last."

The author cogently argues the importance of print for the creation of accurate maps: "But this kind of checking could not occur until voyagers were provided with uniform maps and encouraged to exchange information with map publishers." Before print, maps, like books, were usually held in fixed and safe locations. They were much too rare and valuable to be subjected to the hazards of a voyage to foreign parts. Information from merchants was sent back to trained cartographers, but without print this information could not then be sent back out for the use (and checking) by others. The development of maps (and globes) contributed to the European discovery of the rest of the world.

In the conclusion the author writes

> This book has stopped short in the age of the wooden handpress. It has barely touched on the industrialization of paper making and the harnessing of iron presses to steam. Nothing has been said about the railway tracks and telegraph wires which linked European capitals in the mid-nineteenth century, or about the Linotype and Monotype machines which went together with mass literacy and tabloid journalism. The typewriter, the telephone, and a vast variety of more recent media have been entirely ignored. Too much territory has been traversed too rapidly as it is. Because

> contrary views have been expressed, however, it seems necessary to point out that there are irreversible aspects to the early modern printing revolution. Cumulative processes were set in motion in the mid-fifteenth century, and they have not ceased to gather momentum in the age of the computer printout and the television guide.

And so we are back again in the present, with its own communications shift from paper to electronic media, from print to computer. I have read this book through several times and on each occasion (I am not well educated in history) I get a richer deeper understanding of the present time. I strongly recommend it to anyone who wishes to develop for themself an understanding of the human and social consequences of the growing move towards electronic publishing and the information superhighway.

◇ Jonathan Fine
203 Coldhams Lane
Cambridge CB1 3HY, U.K.
`J.Fine@pmms.cam.ac.uk`

Queries

Editor's note: When answering a query, please send a copy of your answer to the TUGboat editor as well as to the author of the query. Answers will be published in the next issue following their receipt.

— — ∗ — —

Suggestions wanted for new edition of *Math into LaTeX*

We are planning a new edition of my book:

Math into LaTeX
An Introduction to LaTeX and AMS-LaTeX
Birkhauser Boston
ISBN 0-8176-3805-9

(Note that Part I of the book: A Short Course, is on the CTAN in the /tex-archive/info/mil directory.)

I would like to get suggestions from the TeX user community what changes/additions are required. Specifically, it would be useful for me if you could:

1. Send me corrections to the book.
2. Point out where the exposition is not clear enough or where more examples are required.

What additional topics would you like to see discussed? For instance, a new appendix on Babel?

The "Formula gallery" (Section I.3) seems to be very popular. Should there be a similar "Newcommand and newenvironment gallery" in Chapter IV? (Can you suggest a better title for such a section?) If you think so, please send me candidates for it.

Here is an example (using the amsthm package):

```
\newtheorem*{named}{\myname}
\newcommand{\myname}{foo}
\newenvironment{namedtheorem}[1]%
   {\renewcommand{\myname}{#1}
       \begin{named}}
   {\end{named}}
```

This is used to produce a theorem named by the user. Example use:

```
\begin{namedtheorem}{Name of the theorem}
 Body of theorem.
\end{namedtheorem}
```

which produces
Name of the theorem. Body of theorem.
in the appropriate style.

Please reply to this query by email.

◇ George Grätzer
University of Manitoba
Winnipeg MB Canada
George_Gratzer@umanitoba.ca

— — ∗ — —

PPCHTEX molecule templates

I just tried the PPCHTeX package to typeset chemical equations and I am very impressed. In the meantime I have written several times to the author and got a lot of help, and he added some new features I mentioned. Because he was so nice I want to do him a favour and compile a package of ready-made molecules that users have made and want to contribute. The package then will be distributed with PPCHTeX. So please, if you want to make this package the very best ;-), just send me your molecules, along with comments about their name and structure.

[Editor's note: The PPCHTeX package, by J. Hagen and A. F. Otten, was the subject of an article in *TUGboat* **17**#1 (1996), pages 54–66.]

◇ Dirk Kuypers
Jülicher Strasse 206–208
52070 Aachen, Germany
dk@comnets.rwth-aachen.de

Abstracts

Les Cahiers GUTenberg Contents of Recent Issues

Numéro 21 — juin 1995 Congrès 95 GUTenberg: Une nouvelle vision des documents La Grande Motte, 1 et 2 juin 1995

Editor's note: Thematic issue entitled "A new vision of documents", the proceedings of GUTenberg '95, a meeting held June 1–2, 1995 at La Grande Motte (just outside Montpellier) by the French-speaking TEX Users Group. The issue contains 166 pages in all, with 11 articles.

Alain Cousquer, Éditorial : un congrès, pour quoi faire? [Editorial: Why have a conference?]; pp. 3–4

Cousquer outlines the pros and cons of TEX meetings, starting with reasons for the 4-year gap since the last major GUTenberg meeting: difficulties in getting things organised, the costs, finding a theme that would attract a wide range of people. And while focused, thematic day-long meetings have proven quite successful — the interest, number of participants, and the high calibre of the discussions — there's nothing quite like a longer meeting for people to meet and exchange views and experiences over a few days' time.

The theme for the 1995 meeting — a new vision for documents — was introduced with a paper by Fernand Baudin (whose presentation is not in the issue) and then carried further by the subsequent eleven presentations.

Michel Goossens and Michèle Jouhet, Les commandes graphiques en LATEX 2ε [Graphics commands in LATEX 2ε]; pp. 5–29

Authors' abstract: "LATEX 2ε provides a generalized driver-independent interface for the inclusion of external graphic material, as well as for scaling and rotation of LATEX boxes. These features are not in the LATEX 2ε kernel, but are implemented in the graphics or graphicx extension packages. These packages rely on features that are not in TEX itself, but which must be supplied by the "driver" used to print the dvi file.[1] Inside a `picture` environment LATEX 2ε also offer[s] better support to draw Bezier curves with the `\qbezier` command."

[1] Unfortunately not all drivers support the same features, and even the internal method of accessing these extensions varies between drivers. Consequently all these packages take options such as "`dvips`" to specify which driver is being used. [footnote in original]

Michel Goossens and Michèle Jouhet, Utiliser la couleur avec LATEX 2ε [Using colour with LATEX 2ε]; pp. 30–52

Authors' abstract: "The first part of this article reviews some basic principles underlying the use of colour in publishing. We explain how to use colour in a functional way to help focus attention, explain relationships, guide the viewer/reader through the presented information. Simple rules for optimizing communication using colour codes are discussed. The colour interface of LATEX 2ε is the subject of the second part of the article, where we describe commands to write text in colour and to generate coloured boxes or backgrounds, and the various colour models to define names of available colours. As with the graphics package it [is] the DVI drivers that are responsible for actually generating the colours on paper or on screen." [includes 9 figures in colour]

Sarra Ben Lagha and Mohamed Ben Ahmed, Intégration de graphiques dans du texte [Integrating graphics with text]; pp. 53–67

Publishing technical documents is a complex operation, requiring a variety of skills and tools. For document exchange in a diverse environment to take place, the documents have to be standardized. Their various components (text, graphics, etc.) must each conform to one or more standards, while the subsequent integration of these components must then also meet certain standards. Here we're talking about documents that address multiple standards, on multiple platforms. This article reviews the basic ideas behind structured SGML documents and the CGM graphics standard, before presenting two methods for integrating graphics into structured SGML documents.

Keywords: document, graphics, integration, standards, SDIF, SGML, DTD, CGM, GKS.

[from the Résumé]

Jean-Jacques Girardot and Bernard Amade, Une expérience de production automatisée de documents : approche, problèmes et solutions [An experiment in automated document production: approach, problems and solutions]; pp. 68–74

This article discusses two experiments in production automation of documents linked to financial applications.

The main difficulty was to find a way to combine a "logical" vision of documents — and the constraints that existing applications impose on the production of such an abstract form — with both the manual addition of information by users and the final form of the documents. [from the Résumé]

Jacques André, Jean-Daniel Fekete and Hélène Richy, Traitement mixte image/texte de documents anciens [Combined image/text handling of old documents]; pp. 75–85

We are interested in "texts with graphics" which are documents (for example, old manuscripts) studied for both their content and their form. Putting them into electronic form requires manipulating them both as textual objects and as images. The idea of a "hotspot" [*zone sensible*] for images is a first step but proves inadequate for the electronic manipulation of old documents in a professional manner. [from the Résumé]

Denis Mégevand, Personnaliser ses lettres avec LaTeX 2_ε [Customising letters with LaTeX 2_ε]; pp. 86–95

The `lettre` style [an adaptation and extension of `letter`] can be used for letters and personal faxes, and can be adapted for the European style and format of envelopes with windows. Various languages can be used for the text; once selected, various default values (format for the date, and even the correct words for the various elements of a letter — enclosure, salutation, closing, etc.) are invoked. The style file also allows the user to customize a letterhead and the other various "standard" fields found in letters and faxes. [from the Résumé]

Daniel Taupin, Faire un ouvrage portable en LaTeX 2_ε [Making a large job portable in LaTeX 2_ε]; pp. 96–106

Author's abstract: "Writing a big book in LaTeX 2_ε with many figures raises several problems of 'TeX capacity' and also some portability problems between the various kinds of printers and drivers. We solved that challenge by inserting figures in PCX code, and using several well-known and more [reliable] MS-DOS tools which enable enlarging and shrinking these figures, [while] minimising distortions and information loss."

Daniel Taupin, Ross Mitchell, and Andreas Egler, MusiXTeX : L'écriture de la musique polyphonique ou instrumentale avec TeX [MusiXTeX: Writing polyphonic or instrumental music with TeX]; pp. 107–113

Authors' abstract: "MusiXTeX is a new music typesetting package derived from MusicTeX, but it provides more beautiful scores than MusicTeX did. While MusicTeX was a single-pass package, MusiXTeX is a three-pass system: the first pass performs a rough TeXing which reports the spacings of each music section, the second pass is a computation of the best note spacings, and the third one is the final TeXing process.

The quality of single notes is the same as in MusicTeX, but slurs are much more beautiful, and notes are regularly spaced instead of being irregularly spaced with glue."

Michel Beigbeder and Jean-Jacques Girardot, La conversion de midifiles en MusicTeX [Converting midifiles into MusicTeX]; pp. 114–126

The typesetting of musical scores is quite different from that of text, and thus requires sufficiently sophisticated programs — and even TeX in its regular form is not up to the task. However, LaTeX style files now provide the functionality specific to musicologists' needs, by providing a bridge between an already widely used format for notation — *midifiles*, a de facto standard of the Computer Music Association — and MusicTeX.

Keywords: midifile, MusicTeX, musical parts

Hélène Richy, Chrystèle Hérault and Jacques André, Notion de "feuille de style" (résumé étendu) [The idea of a "style sheet" (expanded summary)]; pp. 127–134

The structured document concept makes it possible to distinguish between the content of a document and its logical structure (DTD in SGML). The idea of the graphical structure of a document can now be added. As we show in this article, the specifications of such a graphical structure are far from standard: various proposals based on HTML in particular exist. Beyond some apparent deviations, the idea of style sheet is beginning to be accepted and should make document exchange easier.

Keywords: style sheet, structured document, formatting, HTML, Grif

[from the Résumé]

R.D. Hersch and V. Ostromoukhov, Introduction à la génération d'images en demi-tons [Introducing the generation of half-tone images]; pp. 135–165

A detailed technical discussion of half-tone images, after a brief introduction, the 31-page article moves through eight sections on the various methods and results available to the user, and is completed with a short bibliography. Each section includes schematic diagrams, equations for various calculations, and sample output for each method, each one a further refinement of the last. Not an article for the general reader, but likely very interesting and helpful to those dealing with halftone images in their documents.

Numéro 22 — septembre 1995

Ligatures & caractères contextuels

Editor's note: Thematic issue entitled "Ligatures and context-dependent characters". It is a compilation of papers originating from several sources in 1992: a typography course given by the Didot project[2] at the École supérieure Estienne (in Paris), and several meetings held in Lurs (Upper Provence) by the "Rencontres de Lure", an international typographic association founded in 1952.

The 135-page issue contains 8 articles.

JACQUES ANDRÉ and JEAN-LOUIS ESTÈVE, Éditorial : de Didot à GUTenberg [Editorial: From Didot to GUTenberg]; pp. iii–vi

Along with his co-editor from the École Estienne, Jacques André introduces the word play on "Didot" [see footnote] and then elaborates on the Didot project, which tries to bring together the highly aesthetic world of typography and the highly technical world of the computer. About a dozen Didot sessions took place between 1989 and 1993, resulting in two books and a teaching program on computer-based typography. This *Cahier* issue suffered production delays since it seemed only appropriate to ensure the highest typographic quality in a publication about high-quality typography; and this was after lengthy editorial work had been completed. The issue was prepared using LaTeX 2ε and Adobe-Garamond as the main font; the resulting PostScript files were sent via the Internet to the Louis-Jean printing house for offset printing on Arcole paper. An electronic version is available at `http://www.univ-rennes1.fr/pub/GUTenberg/publications/publis.html`.

JACQUES ANDRÉ, Introduction : vous avez dit "ligature"? [Introduction: you said "ligature"?]; pp. 1–4

While Jérôme Peignot's article "Petit traité de la ligature"[3] serves as the base text on the subject — until photocomposition, that is — this issue of the *Cahiers* brings the reader up to date on new, or revitalized and re-tooled ligatures. The author outlines a number of types of ligatures, which are not confined to usual character combinations (e.g. "fi" or ffi"), but include also special characters (e.g. & and @) or logotypes. In short, "ffi" is just the tip of the iceberg.

ADOLF WILD, La typographie de la *Bible* de Gutenberg [The typography of the Gutenberg *Bible*]; pp. 5–16

Author's abstract: "The author examines Gutenberg's *Bible* from a typographical point of view, looking both at page structure (in the light of Rosarivò's grid constants) and, especially, at line structure, where it can be seen that in contrast with present-day methods of justification, based on variable inter-word spacing, Gutenberg balanced his lines using a variety of letter combinations: logotypes, ligatures, abbreviations, etc."

RENÉ PONOT, Le *Didot* a-t-il besoin de ligatures ? [Does *Didot* need ligatures?]; pp. 17–42

Author's abstract: "Having first reviewed the history of ligatures, the author then describes the origins of the Didot typeface and its principal features. He concludes that this typeface has no need for ligatures."

GÉRARD BLANCHARD, Nœuds & esperluettes : Actualité et pérennité d'un signe [Logotypes and ampersands: A symbol for today and forever]; pp. 43–59

In May 1992 a Didot seminar on "logotypes, ligatures and context-dependent symbols" was held in Paris at the École Estienne. In the course of this meeting, Gérard Blanchard presented a paper with an unexpected topic: the ampersand, because, to his eyes, this symbol — or rather, this "type" — was the touchstone for the sense of calligraphy in typography. Starting from a Tschichold text,[4] this article gives the background to his ideas. [from the opening summary]

JACQUES ANDRÉ, Ligatures & informatique [Ligatures & computers]; pp. 61–85

Author's abstract: "In digital typography a ligature consists in replacing n glyphs by a single glyph. This raises problems at the keyboard level: how can two letters such as *c* and *t* be specified to be replaced by the *ct* ligature? How does the user know if the font in current use contains the necessary ligature? Is it possible to design a system in which the user does not have to deal with such problems? The present paper looks at such problems from the point of view of coding and font metrics.

[2] "**DI**dacticiel de **D**essin assisté par **O**rdinateur de caractères **T**ypographiques", or "**DI**gitization and **D**esign **O**f **T**ypefaces".

[3] *Communication et langages*, 73.3 (1987), pp. 20–35.

[4] *Formen Wandlungen der Etzeichen*; translation into French by René Grasset forthcoming, from the École Estienne.

The final section provides a brief introduction to script ligatures and the related problems of handwriting simulation and recognition."

Yannis Haralambous, Tour du monde des ligatures [A world tour of ligatures]; pp. 87–99

The term "ligature" signifies a number of different things and is often the source of confusion. In the TEX world, the situation is exacerbated by the fact that D.E. Knuth has given this name to a portion of the font metrics information.

To clarify this very intriguing subject, we are going to try to classify the different types of ligatures that one finds in natural languages and to relate these findings to TEX. The three types of ligatures are: linguistic, esthetic, and contextual. [from the introduction]

Thierry Gouttenègre, Ligatures & bâtardes bourguignonnes : Du xv^{e} au xx^{e} siècle [Ligatures and the Burgundian Schwabacher fonts: from the 15th to the 20th century]; pp. 101–106

Author's abstract: "The author outlines some ligature-related problems which he had to deal with when designing a typeface based on a fifteenth-century book hand."

François Boltana, Ligatures & calligraphie assistée par ordinateur [Ligatures and computer-assisted calligraphy]; pp. 107–124

Author's abstract: "Three calligraphic typefaces (Champion, Messager and Aurore) provide their designer with the basis of a discussion of the possibilities created by the use of computer tools [to] recreate ligatures. Such a development would not only bring us closer to the rich calligraphic texture of [18th century] written forms, it would also give new life to the concept of contextual typography created by Gutenberg. For though Gutenberg's technique is no longer appropriate to modern conditions, the spirit in which he copied calligraphic forms is more than ever topical."

Gérard Blanchard, Postface ; pp. 125–132

Blanchard discusses the wherefores and whys of ligatures themselves, calling them *cosa mentale*— something like a "state of mind". From flourishes to minimalist strokes, the ordering of elements, and thus of ligatures survives. Ligatures tie things up neatly — they join two words or phrases into a single entity; logotypes, another variation on ligatures, are like a company logo — the final seal on a package for delivery. Both pass from one language to the next, understood and untranslated.

The author then discusses each of the 8 articles in the issue, bringing in much additional information on other papers and presentations, providing a very useful and informed view of what has been detailed by each of the other authors.

Cahiers GUTenberg, Sommaire des derniers numéros parus [Summary of recent issues]; pp. 133–135

A listing of the contents of issues 16 through 22 of the *Cahiers*.

Numéro 23 — avril 1996

FAQ : questions souvent posées

Editor's note: Thematic issue entitled "FAQ : frequently asked questions". This is a French translation (by various members of GUTenberg's board). The issue has six preliminary pages (including the editorial) and 120 pages of contents (including a listing of contents of previous *Cahier* issues, and information about joining GUTenberg).

GUTenberg Board Members, Editorial ; pp. iii–vi

The general notion behind a FAQ is given— that while there are many sources of information available to users, there is often much repetition of certain questions and their answers over time. However, it's not enough to simply compile the frequently asked questions; one also has to ensure that the answers are indeed complete and correct. As well, while the main source of both questions and FAQs is the Internet, having a paper copy available for the wider audience is also important — and a big job. Bobby Bodenheimer is well known as the originator of the (LA)TEX FAQ, a large portion of which was published in *Cahier* 13 (1992). The UKTUG group also published the FAQ, and their revised and updated edition[5] is the source of the current French version, which augments the original text with issues pertinent to French-language texts, additional questions deemed important, and French sources and/or translations of information. As well, an index has been added.

The editorial closes with a translation of the *Baskerville* editorial by Robin Fairbairns, which mentions that translations in French (voilà!) as well as Czech and Russian, are in progress, and that Elsevier Science had graciously underwritten publication of copies for attendees at the TUG'95 meeting in Florida.

In short — the FAQ is being viewed by many groups as being of great assistance and significance to all users.

[5] *Baskerville* 5(6) (1996), pp. 1–29.

UK TEX USERS GROUP, Questions souvent posées sur (LA)TEX [(LA)TEX Frequently Asked Questions]; pp. 1–114

The entire issue is given over to some 129 questions, divided up into sections A through S, and is completed with a 7-page index. The issue can be viewed simply as a French translation of the FAQ, with additions to directly address French-language issues; it can also be viewed as an exceptional resource for TEX- and computer-related terminology in French. While the GUTenberg team occasionally maintained terms such as "bitmap" (they question whether anyone would know what to make of "plan de bits"!), they made every effort to provide official/correct French terminology. Thus, not only are the "contents" a mine of information but also the "form" of the contents are equally valuable. And for a linguist, form and content are always at the heart of the issue `;-)`

Cahiers GUTenberg, Sommaire des derniers numéros parus [Summary of recent issues]; pp. 115–117

A listing of the contents of issues 16 through 23 of the *Cahiers*.

Numéro 24 — juin 1996

TEI : Text Encoding Initiative

Editor's note: This thematic issue contains 8 articles.

FRANÇOIS ROLE, Éditorial : TEI — Text Encoding Initiative ; pp. 1–3

While the *Cahiers* have often addressed the use of TEX in the mathematical sciences, articles on such topics as multilingualism, literature, and the social sciences have also appeared in its pages. GUTenberg and its *Cahiers* are thus open to discussions on any and all electronic document codings, and not uniquely those of TEX.

The TEI project grew out of a conference held in Poughkeepsie, NY, in 1987. The purpose: to facilitate the electronic exchange of documents within the scientific community; in 1994, the "Guidelines for Electronic Text Encoding and Interchange" were published (and now contain some 1300 pages). The TEI recommendations rely on SGML, and in fact, they contain several DTDs, along with detailed commentary on the use of tags and attributes.

This issue of the *Cahiers* begins with an introduction to the subject, and then essentially contains a translation of a large and comprehensive document called "TEI Lite", written by two of TEI's founders, Lou Burnard and C.M. Sperberg-McQueen. The subsequent articles provide short introductions to various applications of the TEI principles to a variety of research projects, all currently underway in France.

The editorial concludes with a listing of Web sites where TEI tools can be obtained:

`http://www.sil.org/sgml/publicSW.html#parsers`
`http://aix1.uottawa.ca/~dmeggins/SGMLSpm/sgmlspm.html`
`http://ftp.ifi.uio.no/pub/SGML/demo/`
`http://www.oclc.org:5046/oclc/research/panorama/panorama.html`
`http://www.sq.com/products/panorama/pan-free.html`
`http:/ftp.lri.fr/LRI/soft/ihm/tei2latex-0.1.tar.gz`

NANCY IDE and JEAN VÉRONIS, Présentation de la TEI : *Text Encoding Initiative* [Introducing TEI: The Text Encoding Initiative]; pp. 4–10

This is a short overview of the TEI project, its history and development, the purposes and principles behind the project. It includes an outline of the core TEI DTD, as well as a list of project groups which have adopted the TEI recommendations. The piece concludes with information on where to get more complete documentation, some of it being the English originals of what appears in this issue of the *Cahiers*. There is a bibliography and then a translation of the Table of Contents of the TEI Guidelines document.

A Web site with the TEI documentation is: `http://www-tei.uic.edu/orgs/tei`. As well, electronic copies can be ftp'd from: `ftp-tei.uic.edu` (pub/tei), `info.ex.ac.uk` (pub/SGML/TEI), `TEI.IPC.Chiba-u.ac.jp` (TEI/P3), or `ftp.ifi.uio.no` (pub/SGML/TEI). And finally, paper copy is also available by ordering from `http://www-tei.uic.edu/orgs/tei/info/p3order.html`. [Recall: it's 1,300 pages long, hence a cost of $75.00 or £50.]

JACQUES ANDRÉ, Balises, structures et TEI [Tags, structures and the TEI]; pp. 11–22

By way of introducing the Text Encoding Initiative (TEI), a summary is provided of this type of coding, which features the use of parenthetical tags in the text, thereby making it possible to mark one or more eventual structural outputs. The difference between such "internal" coding and "external" coding (as formatted by a text editor for screen display), and those [codings] found in commercial products such as Word must particularly be emphasized. This structured tagging is meaningless unless one uses tools designed to interpret it, making it possible to revise the coding as well as editing the documents. [from the Resumé]

Lou Burnard and C.M. Sperberg-McQueen, La TEI simplifiée : une introduction au codage des textes électroniques en vue de leur échange [TEI made easy: an introduction to electronic documents for the purposes of interchange]; pp. 23–151

Authors' abstract: "This document is a French translation [by François Role] of the English document: 'TEI Lite: An Introduction to Text Encoding for Interchange'. These Text Encoding Initiative (TEI) Guidelines are addressed to anyone who wants to interchange information stored in an electronic form. They emphasize the interchange of textual information, but other forms of information such as images and sounds are also addressed. References of the genuine English documents are listed below."

François Role, Le codage informatique des apparats critiques : évaluation des recommandations de la *Text Encoding Initiative* [Computer coding of the critical apparatus: An evaluation of the TEI recommendations]; pp. 153–165

Following a quick definition of the notion of critical apparatus and a detailed presentation of the TEI recommendations for this type of document [critical editions], we will examine some possible extensions to the TEI to better address the particular characteristics of certain types of texts (modern manuscripts, texts whose physical characteristics are of interest). [from the Resumé]

Nancy Ide and Jean Véronis, Une application de la TEI aux industries de la langue : le *Corpus Encoding Standard* [A TEI application in the field of languages: The *Corpus Encoding Standard*]; pp. 166–169

The Text Encoding Initiative covers an extremely wide range of documents (prose, poetry, theater, dictionaries, lexicons, and so on), each intended for a variety of final uses (electronic publication, literary and historical analysis, hypertext, etc.). The tags available, as well as the Document Type Definition (DTD), are therefore both too rich and too poor for any one application. The tags and the DTD, however, can be made modular.

The Corpus Encoding Standard (CES) is one attempt at extending and customizing tags and the DTD to address the encoding of linguistic corpora, which are becoming larger and larger as current research projects (e.g. MULTEXT, EAGLES) expand their areas of interest and depth of focus. This brief introduction to the CES outlines the main principles being considered, and then provides a list of Web sites for further information.

Nancy Ide and Jean Véronis, Codage TEI des dictionnaires électroniques [TEI Coding of dictionaries]; pp. 170–176

The purpose of the Dictionary Working Group is to establish a set of TEI standards for coding dictionary entries; other components, such as headers and footers, preliminary matter, and so on, are the same as for other types of documents. Preliminary guidelines have been distributed to research groups around the world, to test and improve them. From an initial test base of modern western dictionaries, it is hoped that the work can expand to include massive works such as the *Oxford English Dictionary* and the *Trésor de la Langue Française*. The basic TEI principles appear to be robust enough to deal with the specific case of dictionaries, although the possible need for new tags and attributes should not be discounted.

Patrice Bonhomme, Florence Bruneseaux and Laurent Romary, Codage, documentation et diffusion de ressources textuelles [Coding, documentation and distribution of textual sources]; pp. 177–180

In the social sciences community, including computational linguistics, electronic text sources often form the basis of research. The diversity of such sources, their content and structure, complicates processing of their data. This article briefly presents some solutions which TEI (an SGML application) can offer; as well, some examples produced by Project Dialogue [the authors are part of this project] show the results of processing TEI-encoded data.

Keywords: corpus, textual sources, server, alignment, SGML

[from the Resumé]

Stéphane Harié, Élisabeth Murisasco, Jacques Le Maitre and Jean Véronis, SgmlQL : un langage de requêtes pour la manipulation de documents SGML [SgmlQL: A query language for manipulating SGML documents]; pp. 181–184

SgmlQL is a programming language developed by the European MULTEXT project, which allows the manipulation of SGML documents, and in particular, of TEI documents. Based on SQL, it makes possible complex manipulation of documents: extracting portions of an SGML document according to specified criteria; tests, counts, and other calculations on SGML elements; constructions of new elements, based on query results.

The article then provides some examples, discusses some current implementations, and refers the reader to Web sites for further information:
`http://www.lpl.univ-aix.fr/projects/multext/MtSgmlQL`
`http://www.lpl.univ-aix.fr/projects/multext/`

Sommaire des derniers *Cahiers GUTenberg* parus [Summary of recent issues]; pp. 185–187

A listing of the contents of issues 16 through 23 of the *Cahiers*.

[Compiled by Christina Thiele]

Articles from *Cahiers* issues can be found in PostScript format at the following site:
`http://www.univ-rennes1.fr/pub/GUTenberg/publicationsPS`

Calendar

1996

Dec 8 – 12	SGML '96, Toronto, Canada. For information contact the Graphic Communications Association, Fax: +1 703-548-2867.
Dec 21	Introductory course on "LaTeX and 4allTeX, Universiteit Utrecht, The Netherlands. Presented by Wietse Dol and Erik Frambach. For information, contact `jvannes@pi.net`.

1997

Jan 9	DANTE TeX–Stammtisch at the Universität Bremen, Germany. For information, contact Martin Schröder (`MS@Dream.HB.North.de`; telephone +49-421-2239425). *Regular schedule*: First Thursday (if not a holiday), 18:00, Universität Bremen MZH, 28359 Bremen, 4th floor, across from the elevator.
Jan 9	DANTE TeX–Stammtisch at the Universität Karlsruhe, Germany. For information, contact Klaus Braune (`braune@rz.uni-karlsruhe.de`; telephone 0721/608-4031). *Regular schedule*: First Thursday (if not a holiday), 19:30, Rechenzentrum der Universität Karlsruhe, Zirkel 2, 3.0G Raum 316.
Jan 28	DANTE TeX–Stammtisch, Köln, Germany. For information, contact Uwe Münch (`muench@ph-cip.uni-koeln.de`) or visit `http://www.uni-koeln.de/themen/texmf/lokal/termine.html#stammtisch`. Fourth Tuesday, 20:00, ZPR Seminarraum (in basement), Weyertal 80, 50931 Köln.
Jan 29	DANTE TeX–Stammtisch, Hamburg, Germany. For information, contact Volker Hüttenrauch (`Volker_Huettenrauch@hh.maus.de`). Last Wednesday, 18:00, details to be posted to the `tex-d-l` list.
Jan 30	DANTE TeX–Stammtisch, Berlin, Germany. For information, contact Rolf Niepraschk (`niepraschk@ptb.de`). Last Thursday, 19:00, Gaststätte "Bärenschänke" Friedrichstr. 124 near the U-Bahnhof "Oranienburger Tor".
Feb 6	DANTE TeX–Stammtisch at the Universität Bremen, Germany. (For details, see Jan 9.)
Feb 6	DANTE TeX–Stammtisch at the Universität Karlsruhe, Germany. (For details, see Jan 9.)

Status as of 20 December 1996

Feb 15	First Official LaTeX2HTML meeting, Technical University of Darmstadt, Germany. For information, contact `latex2html@cdc.informatik.th-darmstadt.de`.
Feb 26 – 28	DANTE'97 and 16th Meeting of DANTE e.V., Fachhochschule, München, Germany. For information, contact `dante97@fitug.de` or visit `http://www.dante.de/dante/Tagungen.html` or `http://www.fitug.de/dante97`.
Feb 25	DANTE TeX–Stammtisch, Köln, Germany. (For details, see Jan 28.)
Feb 27	DANTE TeX–Stammtisch, Berlin, Germany. (For details, see Jan 30.)
Mar 6	DANTE TeX–Stammtisch at the Universität Bremen, Germany. (For details, see Jan 9.)
Mar 6	DANTE TeX–Stammtisch at the Universität Karlsruhe, Germany. (For details, see Jan 9.)
Mar 10 – 12	Tenth International Unicode Conference, Europe, Software + the Internet: Going Global with Unicode, Mainz, Germany. For more information, visit `http://www.reuters.com/unicode/iuc10` or `http://www.unicode.org`.
Mar 14	*Revised* **TUG Election, Nomination deadline**
Mar 14	*Revised Deadline* for submission of abstracts for TUG'97.
Mar 21	*Revised Date of Notification* of acceptance to TUG'97 presenters.
Mar 25	DANTE TeX–Stammtisch, Köln, Germany. (For details, see Jan 28.)
Mar 27	DANTE TeX–Stammtisch, Berlin, Germany. (For details, see Jan 30.)
Apr 2 – 4	LAMPE 97: Lausanne — Atelier sur les Modèles de Page Électronique/ Workshop on Electric Page Models, EPFL, Lausanne, Switzerland. For more information, visit `http://www.irisa.fr/lampe97`.
Apr 3	DANTE TeX–Stammtisch at the Universität Bremen, Germany. (For details, see Jan 9.)
Apr 3	DANTE TeX–Stammtisch at the Universität Karlsruhe, Germany. (For details, see Jan 9.)
Apr 6 – 11	Hypertext '97 — 8th ACM Conference on Hypertext, University of Southampton, Southampton, U.K. For information, contact Wendy Hall (`ht97-info@ecs.soton.ac.uk`).
Apr 24	DANTE TeX–Stammtisch, Berlin, Germany. (For details, see Jan 30.)
Apr 29	DANTE TeX–Stammtisch, Köln, Germany. (For details, see Jan 28.)
May	NTG 19th Meeting, Technische Universiteit Delft, The Netherlands. For information, contact `ntg@nic.surfnet.nl` or visit `http://www.ntg.nl/`.
May 2	Preliminary papers due for TUG'97.
May 1 – 3	BachoTeX '97: GUST 5th Annual Meeting, Bachotek, Poland. For information, contact Jola Szelatyńska (`mjsz@cc.uni.torun.pl`) or visit `http://www.GUST.org.pl`.
May 10 – 15	IASP (International Association of Scholarly Publishers), 6th International conference, "From Scholar to Scholar", Vancouver, Canada. For information, contact Christopher J. Hudson, J. Paul Getty Museum Publications, 16982 Pacific Coast Highway, Malibu, CA 90265-5799 USA.
May 13	**TUG Election, Ballot postmark deadline**
May 15	DANTE TeX–Stammtisch at the Universität Bremen, Germany. (For details, see Jan 9.)
May 16	Preprint deadline for TUG'97.
May 27	**TUG Election, Ballot receipt deadline**
May 27	DANTE TeX–Stammtisch, Köln, Germany. (For details, see Jan 28.)
May 29 – Jun 1	Conference on Electronic Communication in Mathematics, The Geometry Center at the University of Minnesota, Minneapolis, MN For information, visit `http://math.albany.edu:8800/hm/emj/`.
Jun 3 – 7	Joint International Conference ACH-ALLC'97 (Association for Computers and the Humanities, and Association for Literary and Linguistic Computing), Queen's University, Kingston, Ontario, Canada. For information, visit `http://www.qucis.queensu.ca/achallc97`. Deadline for proposals, 20 Nov 1996.
Jun 4 – 6	SSP 19th Annual Meeting, Society for Scholarly Publishing, JW Marriott Hotel, Washington, DC. For information, visit `http://www.edoc.com/ssp/` or write to `ssp@resourcenter.com`.
Jun 20	Camera Ready Copy Deadline for TUG'97.

Jul 4 – 7	SHARP Conference (Society for the History of Authorship, Reading and Publishing), University of Cambridge, U.K. For information, contact James Raven, SHARP Conference Programme Committee, 51 Sherlock Close, Cambridge CB3 0HP, U.K.
Jul 28 – Aug 1	**TUG'97** — The 18th annual meeting of the TEX Users Group, "*TEX Comes Home*", held at the Lone Mountain Conference Center in San Francisco, CA, USA.
Sep 25 – 26	Fourth international conference, Hypertexts and Hypermedia: Products, Tools, Methods, Université de Paris VIII, Laboratoire Paragraphe, Paris, France. For information, contact Imad Saleh or Alain Lelu, (`conf97@labart.univ-paris8.fr` or `lelu@cnam.fr`). Submissions due by 2 Apr 1997.

1998

Apr 1 – 3	EP 98: Seventh International Conference on Electronic Publishing, Document Manipulation and Typography, St. Malo, France. For information, visit `http://www.irisa.fr/ep98`. or contact Jacques André (`jandre@irisa.fr`). Submissions (full papers) due by 15 Jul 1997.
Jun 3 – 5	SSP 20th Annual Meeting, Society for Scholarly Publishing, San Diego, California. For information, visit `http://www.edoc.com/ssp/` or write to `ssp@resourcenter.com`.

For additional information on TUG-sponsored events listed above, contact the TUG office (415-982-8449, fax: 415-982-8559, e-mail: `tug@tug.org`). For events sponsored by other organizations, please use the contact address provided.

Late-Breaking News

Production Notes

Mimi Burbank

This issue is more than two months late — this winter has been absolutely awful in terms of influenza and illness. The team wishes to apologize to our readers and inform you that we *do* plan to have the March issue out on time.

Two articles from the TUG'96 conference are included in this issue — those articles by S.A. Strelkov and G.R. Epshtein (page 382) and M.I. Grinchuk (page 385).

All files were received electronically. 90% of the articles received were in LaTeX 2_ε, with only two articles being received in plain TeX. All articles were received as fully tagged for *TUGboat*, using either `plain`-based or the new LaTeX 2_ε *TUGboat* conventions described in the Authors' Guide (see *TUGboat* 17, no. 3, pages 282–288).

Output The final camera copy was prepared at SCRI on a DEC Alphastation 2100/500 running OSF1 v3.2, using Karl Berry's *Web2c* TeX implementation version 6.1, in the form of Thomas Esser's teTeX package, loaded from TUG's *TeX Live* CD-ROM.

Output was printed on a QMS 680 print system at 600 dpi.

Future Issues

We have made some progress in getting some of the TTN material into this issue of *TUGboat*, and hopefully new editors will be found and new columns will become the norm for 1997. The coming issue of *TUGboat* will contain an article by Yannis Haralambous entitled, "The Traditional Arabic Typecase, Unicode, TeX and METAFONT", demonstrating the use of Ω and Al-Amal, which is part of the ScholarTeX package.

We have a new regular column on TeX and the Humanities, and invite our readers to submit material to Christina Thiele at `cthiele@ccs.carleton.ca`, and use the subject line `TB-Humanities`.

This is *your* publication — what would *you* like to see in these pages? Comments and queries may be sent to `tub-prod@scri.fsu.edu`.

◇ Mimi Burbank
SCRI, Florida State University,
Tallahassee, FL 32306 – 4052
`mimi@scri.fsu.edu`

TUG Business

Facts and Figures

Mimi Burbank
TUG Treasurer

The following pages are presented as a summary of information for our membership. Since my appointment as Treasurer of TUG, I have been gathering material from various sources to make up a *book* with membership data, financial data and Minutes. I thought it would be useful for our members to see comparative lists of both financial material and membership figures. The material is taken from offical reports to the Board of Directors, as well as membership lists, and much of it was provided by Christina Thiele and Barbara Beeton, both long-standing members of TUG and of its Board. These figures represent consistent and significant categories rather than a verbatim copy of the financial reports, and I do ask the reader to peruse the material with the understanding that these figures are presented *as is*, and are useful for demonstrating "trends" *only*.

Membership Figures

The following figures are taken from many sources, and represent information provided to the Board at different times of the year. I have endeavored to identify the month the figures are presented, but in some cases I do not know. Some breakdowns are not available, and the institutional membership count is not obvious from the records provided.

Year	Indiv	Inst	Attend Ann. Mtg.
1980			
Dec	130		50
1981			
Dec	565		92
1982			
Dec	731		136
1983			
Sep	694	44	
Dec	835		135
1984			
Apr	705	52	
Sep	953	69	153
Dec	1,033	76	
1985			
—	1,344	96	
Oct	1,404	95	139
1986			
May	1,728	103	
—	2,012	113	145
Sep	2,125	113	
1987			
Jul	2,418	136	167
Oct	2,758	129	
1988			
Jun	2,886	147	
Oct	3,319		
1989			
Mar	3,033	137	
Jun	3,268	151	
Oct	3,730	152	
1990			
Mar	3,999	141	— (Cork)
Oct	3,789	145	167 (Texas)
1991			
—	3,298	111	240/1st day then ~200
1992			
—	2,815	115	145
Oct	3,172		
1993			
Jul	2,651		168
1994			
Jul	2,117		
1995			
Jul	1,751		83
1996			
Mar	~1,200		
May	1,336		
—	1,404		77 (Dubna)
Oct	1,531		

Financial Records

All financial reports (except for the 1991 report, which was prepared by the Treasurer, Allen Dyer) were prepared by the TUG accountants Michael D. Aaronson of Providence, RI, or Michael S. Nichols of Santa Barbara, CA, and limited to representing the financial statements information that was provided by the TUG management. These statements were unaudited and unreviewed and, accordingly, no opinion or assurance was expressed by the accounting firm regarding the information provided.

The following financial material does *not* reflect every item listed in each financial report, simply because the reports are not parallel in content; hence, there are no "totals".

Table 1: Ten-year Income/Expense Summary

ITEM	1995	1994	1993	1992	1991	1990	1989	1988	1987	1986
REVENUE INCOME										
Dues	$ 95,462	$ 123,234	$ 159,084	$ 205,195	$ 190,410	$ 170,968	$ 167,001	$ 127,013	$ 161,596	$ 94,420
Mtg/Courses	39,738	64,219	58,044	134,791	117,702	206,438	222,148	162,674	207,395	204,831
Sales	37,761	74,785	44,628	114,693	132,563	190,353	220,738	205,129	169,366	167,690
Contrib	32,371	17,735	8,080	15,530	10,694	9,157	8,222	9,233	12,423	8,807
Advertising	–0–	2,942	655	38,077	34,690	32,980	36,935	21,985	16,345	7,999
Royalties	2,219	3,658	5,300	⟵ See the line listed as Contrib – Knuth royalties were included there ⟶						
Interest	2654	2,498	2,421	3,733	6,915	13,175	21,578	16,162	11,699	5,298
EXPENSES										
Salaries	$ 73,844	$ 87,515	$ 72,875	$ 127,550	$ 122,485	$ 154,878	$ 141,217	$ 129,638	$ 147,054	$ 134,881
Taxes/Benefits	6,068	12,494	22,563	38,381	24,256	38,437	37,164	39,442	(incl. in salaries)	
Publications	33,125	23,042	34,888	45,127	79,044	94,225	114,849	60,238	125,538	150,732
Mtg/Courses	29,909	59,302	37,509	88,649	60,214	147,694	118,458	88,955	137,850	89,953
Cost Goods Sold	14,293	30,373	19,442	56,311	66,362	101,858	133,630	125,130	not itemized	
Rent	11,815	14,581	15,076	17,000	12,000	10,900	10,809	9,025	"	
Communic/Utils	4,017	4,608	5,451	6,530	5,804	6,885	4,685	4,429	These expenses	
Bank Charges	2,077	3,106	5,615	6,492	1,414	172	3,172	2,703	were lumped under	
Postage/Mailing	2,027	7,297	13,992	20,351	14,008	13,014	17,954	16,764	"Administrative	
Print/Copy	823	3,444	8,333	9,459	310	1,629	12,803	11,453	Costs" and	
Legal/Accting	3,123	4,148	10,402	6,494	10,018	3,883	3,144	12,276	were included under	
Insurance	4,541	2,651	2,334	1,004	1,462	3,616	4,020	2,369	"Salaries",	
Consultants	1,854	2,721	8,890	139	8,208	8,482	5,686	–0–	above	
Equip Maint	1,164	910	2,280	2,175	3,634	3,997	1,500	–0–	"	
Prop Tax	811	575	453	–0–	–0–	–0–	–0–	–0–	"	
Bad Debts	5,053	5,983	18,501	–0–	–0–	1,797	1,263	2,456	"	
Committee Exp	–0–	–0–	–0–	9,764	6,671	8,266	10,266	11,307	"	
CASH FLOW										
Cash Beginning of Year[1]	$ 142,594	$ 113,588	$ 167,573	$ 181,972	$ —?—	$ —?—	$ 247,072	$ 254,279	$ 143,503	$ 44,584
Cash End of Year	149,291	121,428	111,326	167,573	181,982	—?—	243,541	147,072	311,835	143,503
Excess/(Deficiency) Revenue Over Expenses	9,413	7,840	(15,951)	3,404	3,448	(92,860)	(3,531)	(7,207)	168,322	98,919

[1] Any discrepancies in end-of-year and subsequent beginning-of-year figures may indicate the time the figures were prepared and prior to any end-of-year deposits.

Explanatory notes accompanying financial statements by accountants

The following represent the list of categories taken from the 1992 financial statement, and are indicative of explanations of the different categories covered in the tables above.

1. **Description of organization and summary of significant accounting policies**
   a) Description of organization:

   The TEX Users Group (TUG) is a Rhode Island nonprofit corporation exempt from federal and state income taxes as a trade association under Section 501(c)(6) of the Internal Revenue Code. The TEX Users Group provides information and technical assistance to the users of TEX, a sophisticated typesetting computer application through the publication of a journal, sale of software and reference material, and the administration of conferences, meetings and TEX courses.

   In December 1992, TUG transferred its operations to Santa Barbara, California. It closed the Providence, Rhode Island office as of December 31, 1992.

   b) Summary of significant accounting policies:
   - Accounting method: TUG used the cash basis of accounting through December 31, 1987 and effective January 1, 1988, the organization switched to the accrual basis of accounting and prepares its financial statements in accordance with the principles of fund accounting.
   - Property and equipment: Property and equipment is recorded at cost. Depreciation is provided for via the straight-line method.
   - Cash equivalents: For the purposes of presentation of cash on the statements of cash flows, cash is considered to be currency on hand, cash in banks and certificates of deposit.

2. **Net accounts receivable**

   TUG uses the specific identification method in identifying potential bad debts.

   The balance in trade accounts may include an accrual for advertising income relating to the last issues of *TUGboat* for the previous year which are printed in the current year.

3. **Inventory**

   Inventory consists of books, past issues of *TUGboat*, software and other materials both purchased and created in-house. Inventory is valued at the lower of average cost or market value based on the FIFO method of valuation.

4. **Property and equipment**

   Changes in property and equipment (office furniture and equipment, as well as computer software and equipment) accounts are explained with breakdown of additions and deletions, less accumulated depreciation.

5. **Other payables**

   Other payables are explained for the current year (if any).

6. **Deferred income**

   Deferred income consists of cash collected in the current year applicable to membership dues, advertising commitments or course registrations for the subsequent year.

7. **Commitments**
   a) Operating leases (space rental, storage rental).
   b) Grants payable: i.e., commitments for funding the TEXHAX program from 1993–1996 (paid for in 1993).

8. **Pension plan**

   TUG has a non-contributory profit-sharing plan initiated effective January 1, 1988. Participants are 100% vested in TUG contributions as contribution levels are determined each year. Pension expense is listed by year.

9. **Contributions**

   Balance in contributions for the year, including amounts of royalty fees assigned to TUG by authors. Some of this money is "restricted funds" (see below) and does not represent acro-TUG money.

10. **Restricted funds**

    Restricted funds were set up by TUG retroactive to 1991 to allow members or others to contribute to the cost of running specific programs. Restricted funds were set up to support the development of a new version of LATEX (LATEX3) and the development of TEXHAX. TUG's membership forms (in the past) have allowed members to designate contributions to these two restricted funds.

Institutional Members

The Aerospace Corporation,
El Segundo, California

American Mathematical Society,
Providence, Rhode Island

CNRS - IDRIS,
Orsay, France

CERN, *Geneva, Switzerland*

College of William & Mary,
Department of Computer Science,
Williamsburg, Virginia

Communications
Security Establishment,
Department of National Defence,
Ottawa, Ontario, Canada

CSTUG, *Praha, Czech Republic*

Elsevier Science Publishers B.V.,
Amsterdam, The Netherlands

Florida State University,
Supercomputer Computations
Research, *Tallahassee, Florida*

Grinnell College,
Noyce Computer Center,
Grinnell, Iowa

Hong Kong University of
Science and Technology,
Department of Computer Science,
Hong Kong

Institute for Advanced Study,
Princeton, New Jersey

Institute for Defense Analyses,
Communications Research
Division, *Princeton, New Jersey*

Iowa State University,
Ames, Iowa

Los Alamos National Laboratory,
University of California,
Los Alamos, New Mexico

Marquette University,
Department of Mathematics,
Statistics and Computer Science,
Milwaukee, Wisconsin

Mathematical Reviews,
American Mathematical Society,
Ann Arbor, Michigan

New York University,
Academic Computing Facility,
New York, New York

Nippon Telegraph &
Telephone Corporation,
Basic Research Laboratories,
Tokyo, Japan

Princeton University,
Princeton, New Jersey

Smithsonian Astrophysical
Observatory, *Cambridge,*
Massachusetts

Space Telescope Science Institute,
Baltimore, Maryland

Springer-Verlag,
Heidelberg, Germany

Stanford University,
Computer Science Department,
Stanford, California

Texas A & M University,
Department of Computer Science,
College Station, Texas

United States Naval Observatory,
Washington DC

University of California, Berkeley,
Center for EUV Astrophysics,
Berkeley, California

University of California, Irvine,
Information & Computer Science,
Irvine, California

University of Canterbury,
Christchurch, New Zealand

University College,
Cork, Ireland

University of Delaware,
Newark, Delaware

University of Groningen,
Groningen, The Netherlands

Universität Koblenz–Landau,
Koblenz, Germany

University of Manitoba,
Winnipeg, Manitoba

University of Oslo,
Institute of Informatics,
Blindern, Oslo, Norway

University of Stockholm,
Department of Mathematics,
Stockholm, Sweden

University of Texas at Austin,
Austin, Texas

Università degli Studi di Trieste,
Trieste, Italy

Uppsala University,
Uppsala, Sweden

Vrije Universiteit,
Amsterdam, The Netherlands

Wolters Kluwer,
Dordrecht, The Netherlands

Yale University,
Department of Computer Science,
New Haven, Connecticut

TEX Consulting & Production Services

Information about these services can be obtained from:

TEX Users Group
1850 Union Street, #1637
San Francisco, CA 94123, U.S.A.
Phone: +1 415 982-8449
Fax: +1 415 982-8559
Email: tug@tug.org

North America

Anagnostopoulos, Paul C.
Windfall Software,
433 Rutland Street, Carlisle, MA 01741;
(508) 371-2316; greek@windfall.com
We have been typesetting and composing high-quality books and technical Publications since 1989. Most of the books are produced with our own public-domain macro package, ZzTEX, but we consult on all aspects of TEX and book production. We can convert almost any electronic manuscript to TEX. We also develop book and electronic publishing software for DOS and Windows. I am a computer analyst with a Computer Science degree.

Cowan, Dr. Ray F.
141 Del Medio Ave. #134, Mountain View, CA 94040;
(415) 949-4911; rfc@netcom.com
Twelve Years of TEX and Related Software Consulting: Books, Documentation, Journals, and Newsletters
TEX & LATEX macropackages, graphics; PostScript language applications; device drivers; fonts; systems.

Hoenig, Alan
17 Bay Avenue, Huntington, NY 11743; (516) 385-0736
TEX typesetting services including complete book production; macro writing; individual and group TEX instruction.

NAR Associates
817 Holly Drive E. Rt. 10, Annapolis, MD 21401;
(410) 757-5724
Extensive long term experience in TEX book publishing with major publishers, working with authors or publishers to turn electronic copy into attractive books. We offer complete free lance production services, including design, copy editing, art sizing and layout, typesetting and repro production. We specialize in engineering, science, computers, computer graphics, aviation and medicine.

Ogawa, Arthur
40453 Cherokee Oaks Drive,
Three Rivers, CA 93271-9743;
(209) 561-4585
Experienced in book production, macro packages, programming, and consultation. Complete book production from computer-readable copy to camera-ready copy.

Quixote Digital Typography, Don Hosek
555 Guilford, Claremont, CA 91711;
(909) 621-1291; Fax: (909) 625-1342;
dhosek@quixote.com
Complete line of TEX, LATEX, and METAFONT services including custom LATEX style files, complete book production from manuscript to camera-ready copy; custom font and logo design; installation of customized TEX environments; phone consulting service; database applications and more. Call for a free estimate.

Richert, Norman
1614 Loch Lake Drive, El Lago, TX 77586;
(713) 326-2583
TEX macro consulting.

Southern California PrintCorp
(800) 899-7267 x801
SAVE MONEY on 1-day Linotronic output for journals. Special *TUGboat* offer. Call now for more information.

Southern California PrintCorp
1915 Midwick Drive, Suite B, Altadena, CA 91001
(800) 899-7267 x888, Fax (818) 399-3565,
BBS (818) 398-3567
We have a ten year history providing 24-hour turn-around imagesetting of PostScript files. Call for FREE information on how *TUGboat*-ers can obtain low-cost, fastest available Linotronic publication production services in the U.S.

Type 2000
16 Madrona Avenue, Mill Valley, CA 94941;
(415) 388-8873; Fax: (415) 388-8865
pti@crl.com
$2.50 per page for 2000 DPI TEX and PostScript camera ready output! We provide high quality and fast turnaround to dozens of publishers, journals, authors and consultants who use TEX. Computer Modern, PostScript and METAFONT fonts available. We accept DVI and PostScript files only and output on RC paper. $2.25 per page for 100+ pages, $2.00 per page for 500+ pages; add $.50 per page for PostScript.

Outside North America

TypoTEX Ltd.
Electronical Publishing, Battyány u. 14. Budapest,
Hungary H-1015; (036) 11152 337
Editing and typesetting technical journals and books with TEX from manuscript to camera ready copy. Macro writing, font designing, TEX consulting and teaching.